Guia do usuário

Amazon EKS



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

Amazon EKS: Guia do usuário

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

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

Table of Contents

..... xxvi
..... xxvii
O que é o Amazon EKS? 1
Amazon EKS: gerenciamento simplificado do Kubernetes 1
Desenvolvimento e escalabilidade com o Kubernetes: funcionalidades do Amazon EKS 2
Recursos do Amazon EKS 3
Serviços relacionados 4
Preços do Amazon EKS 5
Casos de uso comuns 6
Arquitetura 8
Ambiente de gerenciamento 8
Computação 8
Funcionalidades do EKS 10
Conceitos do Kubernetes 11
Por que usar o Kubernetes? 12
Clusters 17
Workloads 22
Próximas etapas 28
Opções de implantação 29
Introdução às opções de implantação do Amazon EKS 29
Amazon EKS na nuvem 29
Amazon EKS no data center ou em ambientes de borda 30
Amazon EKS Anywhere para ambientes isolados 31
Ferramentas do Amazon EKS 32
Configuração 33
Próximas etapas 33
Configurar a AWS CLI 33
Para criar uma chave de acesso 34
Para configurar a CLI do AWS 34
Para obter um token de segurança 35
Para verificar a identidade do usuário 35
Próximas etapas 36
Configuração do kubectl e eksctl 36
Instalar ou atualizar o `kubectl` 36

Etapa 1: verifique se o kubectl está instalado ... 36
Etapa 2: instalar ou atualizar kubectl ... 37
Instalar o eksctl ... 51
Próximas etapas ... 51
Início rápido ... 52
Neste tutorial ... 52
Pré-requisitos ... 52
Configurar o cluster ... 53
Criar IngressClass ... 53
Implantar a aplicação de exemplo do jogo 2048 ... 54
Persistir dados usando o Modo Automático do Amazon EKS ... 58
Limpeza ... 60
Conhecer o Amazon EKS ... 61
Visão geral ... 61
Workshop do Amazon EKS ... 62
Tutoriais práticos de configuração de clusters do Amazon EKS ... 63
Amostras do Amazon EKS ... 64
Tutoriais do AWS ... 65
Workshop para desenvolvedores ... 65
Workshop do Terraform ... 65
Treinamento do AWS Amazon EKS ... 65
Conceitos básicos ... 67
Criar cluster (Modo Automático do EKS) ... 67
Criar cluster (eksctl) ... 68
Pré-requisitos ... 69
Etapa 1: Criar o cluster e nós do Amazon EKS ... 69
Etapa 2: exibir recursos do Kubernetes ... 70
Etapa 3: excluir clusters e nós ... 72
Próximas etapas ... 72
Criar cluster (console e CLI) ... 73
Pré-requisitos ... 73
Etapa 1: Criar o cluster do Amazon EKS ... 74
Etapa 2: Configurar o computador para se comunicar com seu cluster ... 77
Etapa 3: criar nós ... 78
Etapa 4: visualizar recursos ... 80
Etapa 5: excluir recursos ... 81

Próximas etapas 81
Modo Automático do EKS 83
Atributos 83
Componentes automatizados 84
Configuração 86
Modelo de responsabilidade compartilhada 87
Criar cluster 88
CLI do eksctl 89
AWS CLI 91
Console de Gerenciamento 99
Habilitar clusters existentes 101
Referência de migração 102
Migração de volumes do EBS 103
Migração de balanceadores de carga 104
Habilitar no cluster 104
Migrar do Karpenter 108
Migrar de MNGs 112
Migrar do Fargate 113
Executar workloads 118
Implantar workload de ampliação 119
Implantar balanceador de carga 122
Implantar workload com estado 128
Implementar workload acelerada 133
Configure 140
Criar classe de nó 144
Criar grupo de nós 152
Grupos de nós de capacidade estática 159
Criar classe de entrada 165
Create service (Criar serviço) 174
Criar StorageClass 180
Desabilitar o Modo Automático do EKS 189
Atualizar a versão do Kubernetes 191
Analisar grupos de nós integrados 192
Controlar a implantação 194
Executar complementos críticos 195
Usar políticas de rede 197

Marcar sub-redes 209
Gerar o relatório do CIS 211
Criptografar volumes raiz 214
Atualizar os controles de AMI 218
Controle de reservas de capacidade 221
Uso das zonas locais 226
Como funciona 231
Instâncias gerenciadas 231
Identidade e acesso 236
Redes 242
Solução de problemas 246
Agente de monitoramento de nós 247
Obter a saída do console de uma instância gerenciada do EC2 usando a CLI do AWS EC2 247
Obter logs de nós usando contêineres de depuração e a CLI do `kubectl` 248
Visualizar recursos associados ao Modo Automático do EKS no Console da AWS 249
Visualizar erros do IAM na conta da AWS 250
Solucionar problemas de pods com falhas ao agendar o nó do Modo Automático 250
Solucionar problemas de nós que não estão se associando ao cluster 250
Compartilhar volumes entre pods 253
Exibir eventos do Karpenter nos logs do ambiente de gerenciamento 254
Solucionar problemas de controladores incluídos no Modo Automático 255
Recursos relacionados 214
Notas da versão 256
19 de novembro de 2025 256
19 de novembro de 2025 256
23 de outubro de 2025 256
1.º de outubro de 2025 257
30 de setembro de 2025 257
29 de setembro de 2025 257
10 de setembro de 2025 257
24 de agosto de 2025 257
15 de agosto de 2025 257
6 de agosto de 2025 257
30 de junho de 2025 257
20 de junho de 2025 258

13 de junho de 2025 258
30 de abril de 2025 258
18 de abril de 2025 258
11 de abril de 2025 258
04 de abril de 2025 259
31 de março de 2025 259
21 de março de 2025 259
14 de março de 2025 259
Configuração de clusters 260
Criar um cluster automático 260
Pré-requisitos 52
Criar cluster - AWS console 262
Crar cluster: AWS CLI 267
Próximas etapas 99
Criar um cluster 273
Pré-requisitos 52
Etapa 1: Criar perfil IAM de cluster 274
Etapa 2: Criar cluster 275
Etapa 3: atualizar o kubeconfig 285
Etapa 4: configuração do cluster 286
Próximas etapas 99
Criação de um ambiente de gerenciamento provisionado 287
Visão geral 97
Casos de uso 151
Níveis de escalabilidade do ambiente de gerenciamento 290
Considerações 104
Conceitos básicos do ambiente de gerenciamento provisionado 294
Insights de cluster 302
Tipos de insights de cluster 302
Considerações 302
Casos de uso 303
Conceitos básicos 304
Visualizar insights de clusters 304
Atualizar a versão do Kubernetes 311
Considerações sobre o Modo automático do Amazon EKS 312
Resumo 312

Etapa 1: preparar-se para o upgrade ... 313
Etapa 2: analisar as considerações sobre o upgrade ... 314
Etapa 3: atualizar o ambiente de gerenciamento do cluster ... 315
Etapa 4: atualizar componentes do cluster ... 317
Fazer o downgrade da versão do Kubernetes de um cluster do Amazon EKS ... 319
Excluir um cluster ... 319
Considerações ... 104
Excluir cluster (eksctl) ... 320
Excluir cluster (console do AWS) ... 321
Excluir cluster (AWS CLI) ... 322
Proteção dos clusters do EKS contra exclusão acidental ... 323
Acesso ao endpoint do cluster ... 324
Formato de endpoint de cluster IPv6 ... 325
Formato de endpoint de cluster IPv4 ... 325
Endpoint privado do cluster ... 326
Modificar o acesso ao endpoint do cluster ... 327
Acessar um servidor de API somente privado ... 330
Configurar o acesso ao endpoint ... 332
Habilitar o suporte ao Windows ... 335
Considerações ... 104
Pré-requisitos ... 52
Habilitar o suporte ao Windows ... 337
Implantar pods do Windows ... 339
Compatibilidade com uma maior densidade de pods em nós do Windows ... 339
Desativar o suporte ao Windows ... 340
Clusters privados ... 341
Requisitos de arquitetura do cluster ... 341
Requisitos do nó ... 341
Requisitos do Pod ... 343
Ajuste de escala automático ... 346
Modo Automático do EKS ... 346
Soluções adicionais ... 346
Saiba mais sobre a mudança de zona ... 347
Noções básicas sobre o fluxo de tráfego de rede leste-oeste entre pods ... 348
Requisitos de mudança de zona do EKS ... 352
Perguntas frequentes ... 361

Recursos adicionais ... 364
Habilitar mudança de zona ... 364
Considerações ... 365
O que é o Amazon Application Recovery Controller? ... 365
O que é mudança de zona? ... 365
O que é mudança de zona automática? ... 366
O que o EKS faz durante uma mudança automática? ... 366
Registre o cluster EKS com o Amazon Application Recovery Controller (ARC) (console do AWS) ... 366
Próximas etapas ... 99
Gerenciar acesso ... 368
Tarefas comuns ... 368
Contexto ... 209
Considerações para o Modo Automático do EKS ... 319
Acesso à API do Kubernetes ... 370
Associe identidades do IAM a permissões do Kubernetes ... 371
Definir modo de autenticação do cluster ... 372
Entradas de acesso ... 373
ConfigMap aws-auth ... 433
Vincular um provedor OIDC ... 443
Desvincular provedor OIDC ... 448
Acessar recursos do cluster ... 449
Permissões obrigatórias ... 450
Acessar meu cluster com o kubectl ... 456
Criar arquivo `kubeconfig` automaticamente ... 457
Acesso da workload à AWS ... 458
Tokens de contas de serviço ... 458
Complementos de cluster ... 460
Concessão de permissões do AWS Identity and Access Management para workloads em clusters do Amazon Elastic Kubernetes Service ... 460
Credenciais com IRSA ... 465
Identidade de Pods ... 489
Gerenciar computação ... 524
Comparar opções de computação ... 524
Grupos de nós gerenciados ... 531
Conceitos dos grupos de nós gerenciados ... 532

Tipos de capacidade do grupo de nós gerenciados 535
Criar 539
Atualizar 550
Atualizar detalhes de comportamento 556
Modelos de execução 560
Excluir 577
Nós autogerenciados 579
Amazon Linux 581
Bottlerocket 590
Windows 594
Ubuntu Linux 603
Métodos de atualização 607
AWS Fargate 620
Considerações sobre o AWS Fargate 620
Tabela de comparação do Fargate 623
Conceitos básicos 626
Definir perfis 632
Excluir perfis 637
Detalhes da configuração do pod 638
Eventos de aplicação de patches de sistema operacional 641
Coletar métricas 644
Registro em log 647
Tipos de instância do Amazon EC2 658
O máximo de pods do Amazon EKS recomendado para cada tipo de instância do Amazon EC2 660
Considerações para o Modo Automático do EKS 319
AMIs predefinidas e otimizadas 662
Descontinuação da AMI do AL2 663
Amazon Linux 669
Bottlerocket 679
Ubuntu Linux 686
Windows 686
Integridade de nós 749
Agente de monitoramento de nós 750
Reparo automático de nós 750
Problemas de integridade de nós 752

Visualizar a integridade do nó 764
Obter logs de nós 766
Nós híbridos 769
Atributos 770
Limites 771
Considerações 771
Recursos adicionais 771
Pré-requisitos 772
Executar nós híbridos 842
Configurar 864
Como funciona 951
Nós híbridos nodeadm 992
Solução de problemas 1010
Armazenamento de dados de aplicações 1032
Amazon EBS 1032
Considerações 1033
Pré-requisitos 1033
Etapa 1: criar um perfil do IAM 1034
Etapa 2: obter o driver de CSI do Amazon EBS 1042
Etapa 3: implantar uma aplicação de exemplo 1043
Amazon EFS 1043
Considerações 1043
Pré-requisitos 1044
Etapa 1: criar um perfil do IAM 1045
Etapa 2: obter o driver de CSI do Amazon EFS 1051
Etapa 3: criar um sistema de arquivos do Amazon EFS 1051
Etapa 4: implantar uma aplicação de exemplo 1052
Amazon FSx para Lustre 1052
Implantar o driver 1052
Otimizar (EFA) 1060
Otimizar (não EFA) 1076
Amazon FSx para NetApp ONTAP 1084
Amazon FSx para OpenZFS 1085
Amazon File Cache 1085
Mountpoint para Amazon S3 1085
Considerações 1086

Implantar o driver 1086
Remover o driver 1097
Controlador de snapshots da CSI 1098
Configurar redes 1100
Adicionar uma sub-rede da VPC existente a um cluster do Amazon EKS no console de gerenciamento 1100
Requisitos para sub-rede e VPC 1100
Requisitos e considerações para VPCs 1101
Requisitos e considerações para sub-redes 1103
Requisitos e considerações de sub-redes 1109
Crie uma VPC 1110
Pré-requisitos 52
Sub-redes públicas e privadas 1111
Somente sub-redes públicas 1114
Somente sub-redes privadas 1115
Requisitos para grupos de segurança 1117
Grupo de segurança do cluster padrão 1117
Restringir o tráfego do cluster 1119
Grupos de segurança compartilhados 1120
Gerenciar os complementos de redes 1121
Complementos integrados 1121
Complementos opcionais de rede da AWS 1122
CNI da Amazon VPC 1123
Plug-ins CNI alternados 1247
Multus 1248
AWS Load Balancer Controller 1249
CoreDNS 1268
kube-proxy 1287
Workloads 1293
Implantação de exemplo (Linux) 1293
Pré-requisitos 52
Crie um namespace 1294
Crie uma implantação do Kubernetes 1294
Criar um serviço 1296
Revisar os recursos criados 1297
Executar um shell em um pod 1299

Próximas etapas 1301
Implantação de exemplo (Windows) 1301
Pré-requisitos 52
Crie um namespace 1294
Crie uma implantação do Kubernetes 1294
Criar um serviço 1296
Revisar os recursos criados 1304
Executar um shell em um pod 1299
Próximas etapas 1308
Vertical Pod Autoscaler 1308
Implantar o Vertical Pod Autoscaler 1309
Testar a instalação do Vertical Pod Autoscaler 1310
Horizontal Pod Autoscaler 1314
Execute uma aplicação de teste do Horizontal Pod Autoscaler 1315
Balanceamento de carga da rede 1318
Pré-requisitos 52
Considerações 104
Criar um balanceador de carga da rede 1322
(Opcional) Implantar uma aplicação de exemplo 1324
Balanceador de carga da aplicação 1328
Pré-requisitos 52
Reutilizar ALBs com grupos de entrada 1331
(Opcional) Implantar uma aplicação de exemplo 1333
Restringir IPs externos de serviços 1336
Copiar uma imagem para um repositório 1338
Visualizar registros de imagens da Amazon 1342
Complementos do Amazon EKS 1345
Considerações 1346
Namespace personalizado para complementos 1347
Considerações sobre o Modo automático do Amazon EKS 1348
Suporte 1349
Complementos da AWS 1351
Complementos da comunidade 1373
Complementos do Marketplace 1379
Criação de um complemento 1396
Atualização de um complemento 1410

Verificação da compatibilidade 1418
Remover um complemento 1419
Perfis do IAM 1423
Campos que você pode personalizar 1430
Verificar imagens de contêiner 1434
Funcionalidades do EKS 1435
Capacidades disponíveis 1436
AWS Controlllers for Kubernetes (ACK) 1436
Argo CD 1436
kro (Kube Resource Orchestrator) 1437
Benefícios das funcionalidades do EKS 1437
Preços 1438
Funcionamento das funcionalidades do EKS 1438
Casos de uso comuns 1439
Como trabalhar com funcionalidades 1441
Recursos de funcionalidade do EKS 1441
Noções básicas sobre o status da funcionalidade 1442
Criação de funcionalidades 1443
Listagem de funcionalidades 1443
Descrição de uma funcionalidade 1445
Atualização da configuração de uma funcionalidade 1446
Exclusão de uma funcionalidade 1447
Próximas etapas 99
Recursos Kubernetes 1448
Recursos do Argo CD 1449
Recursos do kro 1450
Recursos do ACK 1451
Limites de recurso 1452
Próximas etapas 99
Considerações 1453
Perfis do IAM para a funcionalidade e controle de acesso RBAC do Kubernetes 1453
Padrões de arquitetura de vários clusters 1454
Comparação entre funcionalidades do EKS e soluções autogerenciadas 1456
AWS Controllers for Kubernetes 1459
Funcionamento do ACK 1459
Benefícios do ACK 1460

Serviços da AWS compatíveis 1461
Integração com outras funcionalidades gerenciadas do EKS 1462
Conceitos básicos do ACK 1462
Criação da funcionalidade do ACK 1462
Conceitos do ACK 1473
Configurar permissões 1481
Considerações 1487
Solução de problemas 1494
Comparação em relação à ferramenta autogerenciada 1499
Argo CD 1500
Funcionamento do Argo CD 1501
Benefícios do Argo CD 1502
Integração com o Centro de Identidade da AWS 1503
Integração com outras funcionalidades gerenciadas do EKS 1462
Conceitos básicos do Argo CD 1503
Criar o recurso Argo CD 1503
Conceitos do Argo CD 1515
Configurar permissões 1522
Como trabalhar com o Argo CD 1530
Considerações sobre o Argo CD 1569
Solução de problemas 1578
Comparação em relação à ferramenta autogerenciada 1583
kro 1588
Funcionamento do kro 1588
Benefícios do kro 1589
Integração com outras funcionalidades gerenciadas do EKS 1462
Conceitos básicos do kro 1590
Criação da funcionalidade do kro 1591
Conceitos do kro 1602
Configurar permissões 1609
Considerações 1614
Solução de problemas 1620
Comparação em relação à ferramenta autogerenciada 1627
Solução de problemas das funcionalidades 1629
Abordagem geral para a solução de problemas 1629
Verificação da integridade da funcionalidade 1629

Status comuns das funcionalidades 1630
Verificação do status dos recursos do Kubernetes 1631
Análise das permissões do IAM e do acesso ao cluster 1631
Solução de problemas específicas por funcionalidade 1633
Problemas comuns em todas as funcionalidades 1634
Próximas etapas 99
Gerenciamento de clusters 1636
Monitoramento de custos 1636
Visualizar custos por pod 1637
Instalar o Kubecost 1638
Acessar o painel do Kubecost 1639
Saiba mais sobre o Kubecost 1640
Servidor de métricas 1650
Considerações 104
Implantação como um complemento da comunidade com os complementos do Amazon EKS 1651
Implantação com manifesto 1652
Implantar aplicações com o Helm 1653
Marcando seus Recursos 1654
Conceitos Básicos de Tags 1655
Marcando seus Recursos 1655
Restrições de tags 1656
Marcar recursos para faturamento 1657
Trabalhando com tags usando o console 1658
Trabalhar com etiquetas usando a CLI, a API ou o `eksctl` 1659
Cotas de serviço 1661
Visualizar cotas de serviço do EKS com o Console de gerenciamento da AWS 1661
Visualizar service quotas do EKS com a CLI AWS 1661
Cotas de serviço do Amazon EKS 1662
Cotas de serviço do AWS Fargate 1663
Segurança 1664
Práticas recomendadas 1665
Analisar vulnerabilidades 1665
Referência do Center for Internet Security (CIS) para Amazon EKS 1666
Versões da plataforma do Amazon EKS 1666
Lista de vulnerabilidades do sistema operacional 1667

Detecção de nós com o Amazon Inspector 1667
Detecção de clusters e nós com o Amazon GuardDuty 1667
Validar conformidade 1667
Considerações para o EKS 1669
Segurança da infraestrutura 1669
Resiliência 1674
Prevenção contra o problema de confused deputy entre serviços 1674
Considerações para o Kubernetes 1676
Assinatura de certificado 1677
Funções e usuários padrão do 1679
Habilitar a criptografia de segredos 1684
AWS Secrets Manager 1689
Criptografia envelopada padrão para todos os dados de API do Kubernetes 1690
Considerações sobre o Modo Automático do EKS 1699
Segurança e autenticação da API 1699
Segurança de rede 1699
Segurança da instância gerenciada do EC2 1700
Automatizar o gerenciamento de recursos. 1701
Práticas recomendadas de segurança 1702
Considerações para funcionalidades do EKS 1702
Perfil do IAM para a funcionalidade 1703
Entradas de acesso do EKS 1703
Permissões adicionais do Kubernetes 1705
Permissões do IAM exigidas pela funcionalidade 1707
Práticas recomendadas de segurança 1707
Próximas etapas 99
Referência do IAM 1711
Público 1711
Autenticação com identidades 1712
Gerenciando acesso usando políticas 1715
Amazon EKS e IAM 1718
Políticas baseadas em identidade 1722
Perfis vinculados a serviço 1730
Função do IAM de execução de pod 1747
Função do IAM do conector 1752
Políticas gerenciadas pela AWS 1756

Solução de problemas 1786
Função do IAM do cluster 1789
Perfil do IAM do nó 1793
Perfil do IAM para o cluster do modo automático 1798
Perfil do IAM para o nó do modo automático 1802
Perfil do IAM para a funcionalidade 1805
Monitorar clusters 1811
Monitoramento e registro em log no Amazon EKS 1811
Ferramentas do Amazon EKS de monitoramento e registro em log 1813
Painel de observabilidade 1817
Resumo 1817
Integridade do cluster 1817
Monitoramento do ambiente de gerenciamento 1818
Insights de um cluster 1820
Problemas de integridade de nós 1821
Funcionalidades do EKS 1821
Observabilidade da rede de contêineres 1821
Casos de uso 151
Recursos 83
Conceitos básicos 304
Pré-requisitos e observações importantes 1824
Uso da AWS CLI, da API do EKS e da API do NFM 1825
Uso da infraestrutura como código (IaC) 1827
Como funciona? 201
Considerações e limitações 1837
Métricas do Prometheus 1838
Etapa 1: ativar as métricas do Prometheus 1839
Etapa 2: usar as métricas do Prometheus 1841
Etapa 3: gerenciar extratores do Prometheus 1841
Implementar usando o Helm 1842
Ambiente de gerenciamento 1845
Amazon CloudWatch 1853
Métricas básicas no Amazon CloudWatch 1853
Amazon CloudWatch Observability Operator 1863
Logs do ambiente de gerenciamento 1863
Habilitar ou desabilitar os logs do ambiente de gerenciamento 1865

Visualizar os logs do ambiente de gerenciamento do cluster ... 1867
AWS CloudTrail ... 1869
Referências ... 1869
Entradas do arquivo de log ... 1870
Métricas do grupo do Auto Scaling ... 1873
ADOT Operator ... 1874
Painel do Amazon EKS ... 1874
O que é o painel do Amazon EKS? ... 1874
Como o painel usa o AWS Organizations? ... 1875
Habilitar o painel do EKS usando o Console da AWS ... 1876
Visualizar o painel do EKS ... 1877
Configurar o painel ... 1877
Desabilitar o painel do EKS usando o Console da AWS ... 1879
Solucionar problemas do painel do EKS ... 1879
Configurar organizações ... 1880
Como trabalhar com outros serviços ... 1886
AWS Backup ... 1886
AWS CloudFormation ... 1887
Modelos do Amazon EKS e do AWS CloudFormation ... 1887
Saiba mais sobre o AWS CloudFormation ... 1888
AWS CodeConnections ... 1888
Uso do AWS CodeConnections com o Argo CD ... 1888
Considerações para usar o CodeConnections com o Argo CD ... 1889
Amazon Detective ... 1890
Usar o Amazon Detective com o Amazon EKS ... 1890
Amazon GuardDuty ... 1891
Zonas locais da AWS ... 1892
AWS Resilience Hub ... 1893
AWS Secrets Manager ... 1893
Uso do AWS Secrets Manager com o Argo CD ... 1894
Amazon Security Lake ... 1896
Como usar o Security Lake com o Amazon EKS ... 1896
Habilitar o Security Lake para Amazon EKS ... 1897
Analisar logs do EKS no Security Lake ... 1897
Amazon VPC Lattice ... 1897
Amazon EKS Connector ... 1899

Considerações sobre o Amazon EKS Connector ... 1899
Funções do IAM necessárias para o Amazon EKS Connector ... 1900
Conectar um cluster ... 1900
Considerações ... 1901
Pré-requisitos ... 1901
Etapa 1: Registrar o cluster ... 1901
Etapa 2: instalar o agente do `eks-connector` ... 1904
Próximas etapas ... 1906
Conceder acesso aos clusters ... 1906
Pré-requisitos ... 1906
Cancelar o registro de um cluster ... 1908
Cancelar o registro de clusters do Kubernetes ... 1908
Limpar os recursos no cluster do Kubernetes ... 1909
Solucionar problemas do EKS Connector ... 1910
Solução básica de problemas ... 1910
Problema do helm: 403 proibido ... 1912
Erro do console: o cluster está paralisado no estado Pending (Pendente) ... 1912
Erro no console: O usuário system:serviceaccount:eks-connector:eks-connector não pode representar usuários de recursos no grupo API no escopo do cluster ... 1912
Erro no console: [...] é proibido: O usuário [...] não pode listar o recurso [...] no grupo API no escopo do cluster ... 1913
Erro no console: o Amazon EKS não pode se comunicar com o servidor de API do cluster do Kubernetes. O cluster deve estar em um estado ATIVO para estabelecer uma conexão bem-sucedida. Tente novamente em alguns minutos. ... 1914
Os pods do Amazon EKS Connector estão apresentando loops de falhas ... 1914
Falha ao iniciar eks-connector: InvalidActivation ... 1915
O nó do cluster está sem conectividade de saída ... 1916
Os pods do Amazon EKS Connector estão no estado `ImagePullBackOff` ... 1916
Perguntas frequentes ... 1917
Considerações sobre segurança ... 1918
Responsabilidades da AWS ... 1918
Responsabilidades do cliente ... 1918
Amazon EKS no AWS Outposts ... 1920
Quando usar cada opção de implantação ... 1920
Comparar as opções de implantação ... 1921
Executar clusters locais ... 1924

Regiões do AWS compatíveis 1924
Implantar um cluster local 1924
Versões da plataforma do EKS 1935
Criar uma VPC e sub-redes 1955
Preparar para desconexões 1959
Considerações sobre a capacidade 1964
Solução de problemas de clusters 1967
Nós 1980
`eksctl` 1982
Console de gerenciamento da AWS 1983
IA/ML no EKS 1990
Por que escolher o EKS para IA e ML? 1990
Casos de uso principais 1991
Estudos de caso 1992
Comece a usar machine learning no EKS 1993
Inferência em tempo real 1993
Criação de cluster 1993
Inferência de LLM usando vLLM 2031
Configuração do cluster 2049
Uso de AMIs do Linux com suporte a GPU para o EKS 2049
Instalação do plug-in de dispositivo para GPUs 2055
Configurar AMIs de GPU do Windows 2061
Configuração de treinamento de clusters com o EFA 2067
Configureação de clusters de inferência com o Inferentia 2078
Gerenciamento de capacidade 2085
Reserva de GPUs para grupos de nós gerenciados 2085
Reserva de GPUs para nós autogerenciados 2088
Uso do P6e-GB200 com o EKS 2092
Fórmulas 2103
Evita que pods sejam agendados em nós específicos 2104
Recursos 2106
Workshops 2106
Melhores práticas 2106
Arquiteturas de referência 2108
Tutoriais 2109
Ferramentas 2111

Protocolo de contexto para modelos (MCP) 2111
Visão geral 97
Exemplos de integração 2113
Conceitos básicos 304
Conceitos básicos 2114
Ferramentas 2130
Amazon Q 2142
Como funciona 507
Uso do Amazon Q para solução de problemas 2143
Considerações 104
Saiba mais 2144
Versionamento 2145
Versões do Kubernetes 2145
Versões disponíveis no suporte padrão 2146
Versões disponíveis com suporte estendido 2146
Calendário de lançamento do Amazon EKS Kubernetes 2146
Obter informações sobre a versão com a AWS CLI 2147
Perguntas frequentes da versão do Amazon EKS 2149
Perguntas frequentes sobre o suporte estendido do Amazon EKS 2151
Versões de suporte padrão 2153
Versões estendidas de suporte 2159
Visualizar período de suporte 2162
Visualizar política de upgrade 2162
Habilitar suporte estendido 2165
Desabilitar suporte estendido 2166
Versões da plataforma 2167
Versão do Kubernetes 1.34 2169
Versão do Kubernetes 1.33 2170
Versão 1.32 do Kubernetes 2174
Versão do Kubernetes 1.31 2179
Versão do Kubernetes 1.30 2185
Versão do Kubernetes 1.29 2193
Obter a versão atual da plataforma 2202
Alterar versão da plataforma 2203
Solução de problemas 2204
Capacidade insuficiente 2204

Falha nos nós ao ingressar no cluster ... 2204
Acesso negado ou não autorizado (kubectl) ... 2206
hostname doesn't match ... 2207
getsockopt: no route to host ... 2207
Instances failed to join the Kubernetes cluster ... 2208
Códigos de erro para o grupo de nós gerenciados ... 2208
Not authorized for images ... 2213
O nó está no estado NotReady ... 2213
Coletor de logs do EKS ... 2214
A rede de runtime do contêiner não está pronta ... 2215
Tempo limite de handshake TLS ... 2216
InvalidClientTokenId ... 2217
Os grupos de nós devem corresponder à versão do Kubernetes antes da atualização do ambiente de gerenciamento ... 2217
Ao iniciar muitos nós, ocorrem erros Too Many Requests ... 2218
Resposta de erro HTTP 401 - Não autorizado em solicitações do servidor de API do Kubernetes ... 2218
A versão da plataforma Amazon EKS está mais de duas versões atrás da versão atual da plataforma ... 2219
Perguntas frequentes sobre a integridade do cluster e códigos de erro com caminhos de resolução ... 2222
Projetos relacionados ao Amazon EKS ... 2229
Suporte para software implantado no EKS ... 2229
Ferramentas de gerenciamento ... 2230
eksctl ... 2230
AWS Controllers for Kubernetes ... 2230
kro (Kube Resource Orchestrator) ... 2231
Argo CD ... 2231
Flux CD ... 2231
CDK para Kubernetes ... 2231
Redes ... 2232
Plug-in CNI da Amazon VPC para Kubernetes ... 2232
AWS Load Balancer Controller para Kubernetes ... 2232
ExternalDNS ... 2232
Machine learning ... 2233
Kubeflow ... 2233

ajuste de escala automático 2233
Cluster autoscaler 2233
Karpenter 2233
Escalator 2233
Monitoramento 2234
Prometheus 2234
Integração contínua/implantação contínua 2234
Jenkins X 2234
Novos recursos e roteiro 2235
Histórico do documento 2236
Contribuir 2299
Edição de uma única página 2299
Pré-requisitos 52
Procedimento 107
Visão geral da solicitação pull 2301
Edição de arquivos com o GitHub 2302
Pré-requisitos 52
Procedimento 107
Visualização dos comentários de estilo 2304
Instalação da Vale 2305
Instalação da extensão da Vale para o VS Code 2305
Sincronização da Vale 2305
Visualize os comentários de estilo no VS Code 2305
Criar página 2306
Criar página 2306
Adicionar página à navegação 2306
Inserir link 2307
Criar um link para uma página ou seção no Guia do usuário do EKS 2307
Link para outro guia na documentação da AWS 2307
Link para uma página da web externa 2307
Criação com o Amazon Q 2308
Instalação do Amazon Q com o VS Code 2309
Login no Amazon Q 2309
Uso do Amazon Q para criar conteúdo 2309
Dicas 2310
Pré-visualizar a solicitação pull 2310

Pré-visualizar 2311
Limitações da pré-visualização 2311
Sintaxe do AsciiDoc 2312
Nova página 2312
Títulos 2312
Formatação de texto 2312
Listas 2313
Links 2313
Exemplos de código 2313
Imagens 2314
Tabelas 2314
Chamadas 2314
Incluir outros arquivos 2315
Atributos (similares a entidades) 2315
Procedimentos 2315

Ajudar a melhorar esta página

Para contribuir com este guia de usuário, escolha o link Editar esta página no GitHub, disponível no painel direito de cada página.

O que é o Amazon EKS?

Amazon EKS: gerenciamento simplificado do Kubernetes

O Amazon Elastic Kubernetes Service (EKS) fornece um serviço de Kubernetes totalmente gerenciado que elimina a complexidade da operação de clusters do Kubernetes. O EKS permite:

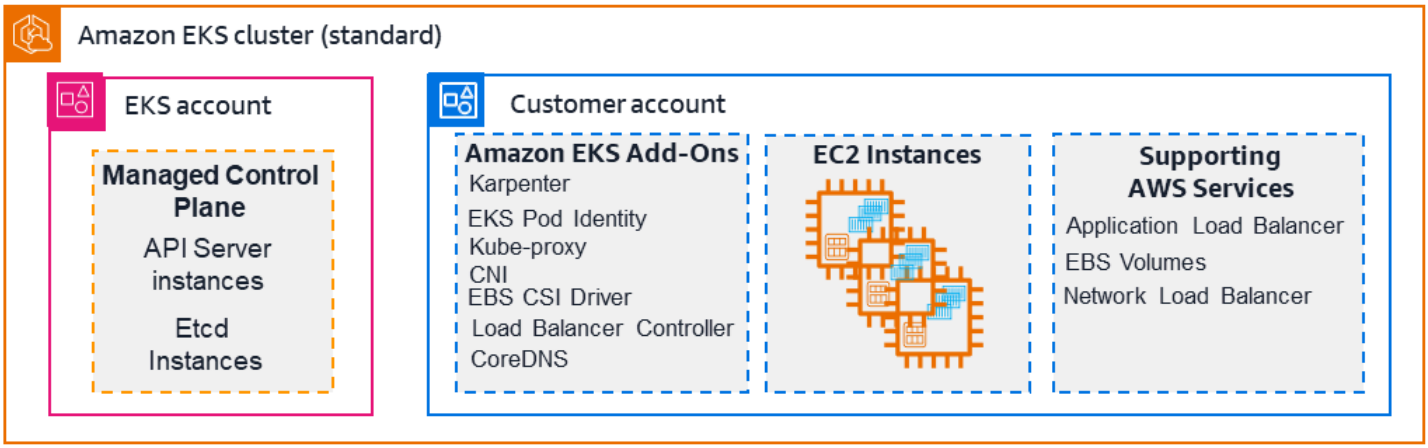
- Implantar aplicações mais rapidamente com menos sobrecarga operacional
- Escalar para atender às mudanças nas demandas de workloads
- Melhorar a segurança por meio da integração com a AWS e atualizações automatizadas
- Escolher entre o EKS padrão ou o EKS Auto Mode totalmente automatizado

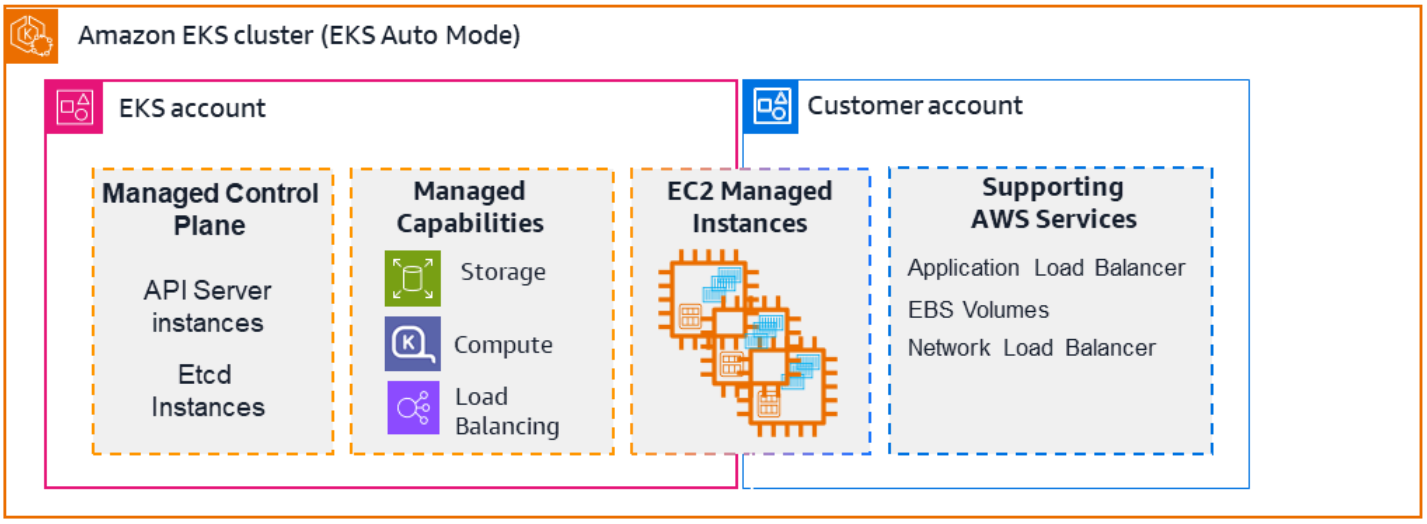
O Amazon Elastic Kubernetes Service (Amazon EKS) é a principal plataforma para executar clusters do Kubernetes, tanto na nuvem Amazon Web Services (AWS) quanto em seus próprios data centers (EKS Anywhere e Amazon EKS Hybrid Nodes).

O Amazon EKS simplifica a criação, a proteção e a manutenção de clusters do Kubernetes. Pode ser mais econômico fornecer recursos suficientes para atender aos picos de demanda do que manter seus próprios data centers. Duas das principais abordagens para usar o Amazon EKS são as seguintes:

- Padrão do EKS: a AWS gerencia o ambiente de gerenciamento do Kubernetes quando você cria um cluster com o EKS. Os componentes que gerenciam os nós, programam workloads, se integram à Nuvem AWS e armazenam e escalam as informações do ambiente de gerenciamento para manter os clusters em execução são gerenciados para você automaticamente.
- Modo automático do EKS: usando o recurso Modo automático do EKS, o EKS estende seu controle para gerenciar também os nós (plano de dados do Kubernetes). Ele simplifica o gerenciamento do Kubernetes provisionando automaticamente a infraestrutura, selecionando as instâncias de computação ideais, escalando dinamicamente os recursos, otimizando continuamente os custos, aplicando patches aos sistemas operacionais e integrando-se aos serviços de segurança da AWS.

O diagrama a seguir ilustra como o Amazon EKS integra os clusters do Kubernetes com a Nuvem AWS, dependendo do método de criação de cluster escolhido:





Com o Amazon EKS, você reduz obstáculos e agiliza a entrada em produção, melhora a performance, a disponibilidade e a resiliência, além de aumentar a segurança dos sistemas. Para obter mais informações, consulte Amazon Elastic Kubernetes Service.

Desenvolvimento e escalabilidade com o Kubernetes: funcionalidades do Amazon EKS

O Amazon EKS vai além da criação e do gerenciamento de clusters; portanto, ele auxilia no desenvolvimento e na escalabilidade de sistemas de aplicações utilizando o Kubernetes. As funcionalidades do Amazon EKS são serviços de cluster totalmente gerenciados que ampliam a funcionalidade do cluster com ferramentas nativas do Kubernetes de fácil manutenção, incluindo:

- Argo CD: o Argo CD oferece implantação contínua declarativa, baseada em GitOps, para as workloads, recursos da AWS e infraestrutura de nuvem.

- AWS Controllers for Kubernetes (ACK): o ACK permite a criação e o gerenciamento do ciclo de vida de recursos da AWS de forma nativa no Kubernetes, unificando a orquestração de workloads e os fluxos de trabalho de infraestrutura como código.
- kro (Kube Resource Orchestrator): o kro amplia as funcionalidades nativas do Kubernetes para simplificar a criação, a orquestração e a composição de recursos personalizados, oferecendo as ferramentas de que você precisa para desenvolver seus próprios componentes de infraestrutura em nuvem sob medida.

As funcionalidades do EKS são recursos de nuvem que minimizam a carga operacional de instalar, manter e garantir a escalabilidade desses componentes fundamentais para a plataforma em seus clusters, permitindo que você se concentre no desenvolvimento de software em vez de focar nas operações da plataforma de cluster.

Para saber mais, consulte Funcionalidades do EKS.

Recursos do Amazon EKS

O Amazon EKS fornece os seguintes recursos de alto nível:

Interfaces de gerenciamento

O EKS oferece uma variedade de interfaces para provisionar, gerenciar e manter clusters, incluindo o Console de gerenciamento da AWS, API e SDKs do Amazon EKS, CDK, AWS CLI, CLI do eksctl, AWS CloudFormation e Terraform. Para obter mais informações, consulte Conceitos básicos e Configuração de clusters.

Acessar ferramentas de controle

O EKS conta com os recursos Kubernetes e AWS Identity and Access Management (AWS IAM) para gerenciar o acesso de usuários e workloads. Para obter mais informações, consulte the section called “Acesso à API do Kubernetes” e the section called “Acesso da workload à AWS ”.

Recursos de computação

Para recursos de computação, o EKS aceita a gama completa de tipos de instâncias do Amazon EC2 e as inovações da AWS, como o Nitro e Graviton com o Amazon EKS para otimizar a computação das workloads. Para obter mais informações, consulte Gerenciar computação.

Armazenamento

O Modo Automático do EKS cria automaticamente classes de armazenamento usando volumes do EBS. Usando os drivers da interface de armazenamento de contêineres (CSI), você também pode usar o Amazon S3, Amazon EFS, Amazon FSX e Amazon File Cache para as necessidades de armazenamento da aplicação. Para obter mais informações, consulte Armazenamento de dados de aplicações.

Segurança do

O modelo de responsabilidade compartilhada é aplicado no que se refere à segurança no Amazon EKS. Para obter mais informações, consulte Práticas recomendadas de segurança, Segurança da infraestrutura e Segurança para o Kubernetes.

Ferramentas de monitoramento

Use o painel de observabilidade para monitorar os clusters do Amazon EKS. As ferramentas de monitoramento incluem o Prometheus, CloudWatch, Cloudtrail e ADOT Operator. Para obter mais informações sobre painéis, servidores de métricas e outras ferramentas, consulte Custos de clusters do EKS e Servidor de métricas do Kubernetes.

Funcionalidades do cluster

O EKS oferece funcionalidades de cluster gerenciadas para implantação contínua, gerenciamento de recursos de nuvem e composição de recursos com base em inovações de código aberto. O EKS instala APIs do Kubernetes em seus clusters, mas os controladores e outros componentes são executados no EKS e são totalmente gerenciados, oferecendo aplicação de patches, escalabilidade e monitoramento automatizados. Para obter mais informações, consulte Funcionalidades do EKS.

Compatibilidade e suporte do Kubernetes

O Amazon EKS é certificado como compatível com o Kubernetes para que você possa implantar aplicações compatíveis com o Kubernetes sem refatorar e usar ferramentas e plug-ins da comunidade do Kubernetes. O EKS oferece o suporte padrão e o suporte estendido para o Kubernetes. Para obter mais informações, consulte the section called “Versões do Kubernetes”.

Serviços relacionados

Serviços para uso com o Amazon EKS

É possível usar outros serviços da AWS com os clusters que você implanta usando o Amazon EKS:

Amazon EC2

Obtenha capacidade computacional escalável e sob demanda com o Amazon EC2.

Amazon EBS

Anexe recursos de armazenamento em bloco escaláveis e de alta performance com o Amazon EBS.

Amazon ECR

Armazene imagens de contêineres com segurança com o Amazon ECR.

Amazon CloudWatch

Monitore os recursos e as aplicações da AWS em tempo real com o Amazon CloudWatch.

Amazon Prometheus

Acompanhe métricas de aplicações em contêineres com o Amazon Managed Service for Prometheus.

Elastic Load Balancing

Distribua o tráfego de entrada entre vários destinos com o Elastic Load Balancing.

Amazon GuardDuty

Detectar ameaças aos clusters do EKS com o Amazon GuardDuty.

AWS Hub de Resiliência da

Avalie a resiliência de clusters do EKS com o Hub de Resiliência da AWS.

Preços do Amazon EKS

O Amazon EKS tem preços por cluster baseados no suporte à versão dos clusters do Kubernetes, preços para o Modo Automático do Amazon EKS e preços por vCPU para o Amazon EKS Hybrid Nodes.

Ao usar o Amazon EKS, você paga separadamente pelos recursos da AWS que usa para executar as aplicações nos nós de processamento do Kubernetes. Por exemplo, se você estiver executando nós de processamento do Kubernetes como instâncias do Amazon EC2 com volumes do Amazon

EBS e endereços IPv4 públicos, você será cobrado pela capacidade da instância por meio do Amazon EC2, pela capacidade do volume por meio do Amazon EBS e pelo endereço IPv4 por meio da Amazon VPC.

Visite as respectivas páginas de preços dos serviços da AWS que você está usando com as aplicações do Kubernetes para obter informações detalhadas sobre os preços.

- Para obter informações sobre os preços do cluster do Amazon EKS, do modo automático do Amazon EKS, das funcionalidades do Amazon EKS e do Amazon EKS Hybrid Nodes, consulte Preços do Amazon EKS.
- Para conferir os preços do Amazon EC2, consulte Preço sob demanda do Amazon EC2 e Preços spot do Amazon EC2.
- Para conferir os preços do AWS Fargate, consulte Definição de preço do AWS Fargate.
- Você pode usar os Savings Plans para computação usada em clusters do Amazon EKS. Para obter mais informações, consulte Definição de preço com Savings Plans.

Casos de uso comuns no Amazon EKS

O Amazon EKS oferece serviços robustos gerenciados do Kubernetes na AWS, destinados a otimizar aplicações conteinerizadas. A seguir, alguns dos casos de uso mais comuns do Amazon EKS para ajudar você a aproveitar os pontos fortes do EKS para atender as suas necessidades específicas.

Implantação de aplicações de alta disponibilidade

: Usando o balanceamento de carga elástico, você pode garantir que as aplicações estejam altamente disponíveis em várias zonas de disponibilidade.

Desenvoler arquiteturas de microsserviços

: Use os recursos de descoberta de serviços do Kubernetes com o AWS Cloud Map ou o Amazon VPC Lattice para criar sistemas resilientes.

Automação dos processos de lançamento de software

: Gerencie pipelines de integração contínua e implantação contínua (CI/CD) que simplificam o processo de desenvolvimento, teste e implantação automatizado de aplicações. Para obter mais informações sobre a implantação contínua declarativa, consulte Implantação contínua com o Argo CD.

Executar aplicações de tecnologia sem servidor

Use o AWS Fargate com o Amazon EKS para executar aplicações sem servidor. Isso significa que você pode se concentrar exclusivamente no desenvolvimento de aplicações, enquanto o Amazon EKS e o Fargate lidam com a infraestrutura subjacente.

Executar workloads de machine learning

O Amazon EKS é compatível com estruturas populares de machine learning, como TensorFlow, MXNet e PyTorch. Graças à compatibilidade com GPU, você pode lidar até com tarefas complexas de machine learning eficazmente.

Implantação consistente on-premises e na nuvem

Para simplificar a execução do Kubernetes em ambientes on-premises, você pode usar os mesmos clusters, recursos e ferramentas do Amazon EKS para executar nós autogerenciados no AWS Outposts, ou pode usar o Amazon EKS Hybrid Nodes com sua própria infraestrutura. Para ambientes independentes e isolados, você pode usar o Amazon EKS Anywhere para automatizar o gerenciamento do ciclo de vida do cluster do Kubernetes em sua própria infraestrutura.

Executar processamento em lote e workloads de big data com eficiência de custos

Utilize as instâncias Spot para executar seu processamento em lote e workloads de big data, como Apache Hadoop e Spark, por uma fração do custo. Isso permite que você aproveite a capacidade não utilizada do Amazon EC2 por preços reduzidos.

Gerenciamento de recursos da AWS com base no Kubernetes

Use o AWS Controllers for Kubernetes (ACK) para realizar a criação e o gerenciamento de recursos da AWS diretamente do cluster do Kubernetes, por meio de APIs nativas do Kubernetes.

Criação de abstrações para a engenharia de plataforma

Crie APIs personalizadas do Kubernetes que agrupam vários recursos em abstrações de alto nível usando o kro (Kube Resource Orchestrator).

Proteção de aplicações e garantia de conformidade

Implemente práticas de segurança robustas e mantenha a conformidade com o Amazon EKS, que se integra aos serviços de segurança da AWS, como o AWS Identity and Access Management (IAM), a Amazon Virtual Private Cloud (Amazon VPC) e o AWS Key Management Service (AWS KMS). Isso garante privacidade e proteção de dados conformes com os padrões do setor.

Arquitetura do Amazon EKS

O Amazon EKS está alinhado com a arquitetura geral de cluster do Kubernetes. Para obter mais informações, consulte Componentes do Kubernetes na documentação do Kubernetes. As seções a seguir resumem alguns detalhes adicionais da arquitetura do Amazon EKS.

Ambiente de gerenciamento

O Amazon EKS garante que todo cluster tenha seu próprio ambiente de gerenciamento do Kubernetes exclusivo. Esse design mantém a infraestrutura de cada cluster separada, sem sobreposições entre clusters ou contas da AWS. A configuração inclui:

Componentes distribuídos

: O ambiente de gerenciamento posiciona pelo menos duas instâncias de servidor de API e três instâncias de etcd em três zonas de disponibilidade AWS em uma região AWS.

Performance ideal

: O Amazon EKS monitora e ajusta ativamente as instâncias do ambiente de gerenciamento para manter o máximo de performance.

Resiliência

: Se uma instância do ambiente de gerenciamento apresentar falhas, o Amazon EKS a substitui rapidamente, utilizando uma zona de disponibilidade diferente, se necessário.

Tempo de atividade consistente

: Executando os clusters em várias zonas de disponibilidade, um Acordo de Serviço (SLA) de disponibilidade de endpoints do servidor de API confiável é alcançado.

O Amazon EKS usa o Amazon Virtual Private Cloud (Amazon VPC) para limitar o tráfego entre os componentes do ambiente de gerenciamento dentro de um único cluster. Os componentes de um cluster não podem visualizar nem receber comunicações de outros clusters ou contas da AWS, exceto quando autorizado pelas políticas de controle de acesso baseado em perfil (RBAC) do Kubernetes.

Computação

Além do ambiente de gerenciamento, um cluster do Amazon EKS tem um conjunto de máquinas de trabalho denominadas nós. Selecionar o tipo de nó de cluster apropriado do Amazon EKS é crucial

para atender aos seus requisitos específicos e otimizar a utilização dos recursos. O Amazon EKS oferece os seguintes tipos de nós primários:

Modo Automático do EKS

O Modo Automático do EKS estende o gerenciamento da AWS além do ambiente de gerenciamento para incluir o plano de dados, automatizando o gerenciamento da infraestrutura de clusters. Ele integra os principais recursos do Kubernetes como componentes integrados, incluindo ajuste de escala automático de computação, rede, balanceamento de carga, DNS, armazenamento e suporte a GPU. O Modo Automático do EKS gerencia dinamicamente os nós com base nas demandas das workloads, usando AMIs imutáveis com recursos de segurança aprimorados. Ele automatiza atualizações e upgrades, respeitando os orçamentos de interrupção de pods, e inclui componentes gerenciados que, de outra forma, exigiriam o gerenciamento de complementos. Essa opção é ideal para usuários que desejam aproveitar a experiência da AWS nas operações diárias, minimizar a sobrecarga operacional e se concentrar no desenvolvimento de aplicações em vez de no gerenciamento da infraestrutura.

AWS Fargate

Fargate é um mecanismo de computação de tecnologia sem servidor para contêineres que elimina a necessidade de gerenciar as instâncias subjacentes. Com o Fargate, você especifica as necessidades de recursos da aplicação e o AWS provisiona, escala e mantém a infraestrutura automaticamente. Essa opção é ideal para usuários que priorizam a facilidade de uso e desejam se concentrar no desenvolvimento e na implantação de aplicações, em vez de no gerenciamento da infraestrutura.

Karpenter

O Karpenter é um autoscaler de clusters do Kubernetes flexível e de alta performance que ajuda a melhorar a disponibilidade das aplicações e a eficiência do cluster. O Karpenter executa recursos computacionais do tamanho certo em resposta à mudança na carga da aplicação. Essa opção pode provisionar recursos computacionais just-in-time que atendam aos requisitos de sua workload.

Grupos de nós gerenciados

Grupos de nós gerenciados são uma combinação de automação e personalização para gerenciar um conjunto de instâncias do Amazon EC2 em um cluster do Amazon EKS. O AWS cuida de tarefas como aplicação de patches, atualização e escalação de nós, facilitando os aspectos operacionais. Em paralelo, argumentos do `kubelet` personalizados são compatíveis, ampliando as possibilidades para a aplicação de políticas avançadas de gerenciamento de CPU e memória.

Além disso, eles aprimoram a segurança por meio das perfis do AWS Identity and Access Management (IAM) para contas de serviço e, ao mesmo tempo, reduzem a necessidade de permissões separadas por cluster.

Nós autogerenciados

Nodos autogerenciados oferecem controle total das instâncias do Amazon EC2 dentro de um cluster do Amazon EKS. Sendo o responsável por gerenciar, escalar e manter os nós, você tem controle total sobre a infraestrutura subjacente. Essa opção é adequada para os usuários que precisam de controle granular e personalização dos nós, e estão prontos para investir tempo no gerenciamento e na manutenção da infraestrutura.

Amazon EKS Hybrid Nodes

Com o Amazon EKS Hybrid Nodes, você pode usar a infraestrutura on-premises e de borda como nós nos clusters do Amazon EKS. O Amazon EKS Hybrid Nodes unifica o gerenciamento do Kubernetes em todos os ambientes e transfere a gestão do ambiente de gerenciamento do Kubernetes para a AWS das aplicações on-premises e de borda.

Funcionalidades do EKS

O Amazon EKS fornece funcionalidades de cluster totalmente gerenciadas por meio da instalação e do gerenciamento de APIs do Kubernetes no cluster, incluindo as definições de recursos personalizados do Kubernetes. Ao mesmo tempo, ele opera controladores e demais componentes na infraestrutura da AWS, de forma isolada do cluster. O EKS disponibiliza aplicação de patches, escalabilidade e monitoramento automatizados dessas funcionalidades, gerenciando totalmente o ciclo de vida para reduzir o esforço operacional de serviços internos ao cluster que são destinados à orquestração de workloads, ao gerenciamento de recursos da AWS e muito mais.

O EKS fornece os seguintes tipos de funcionalidades:

AWS Controlllers for Kubernetes (ACK) da)

Com o AWS Controllers for Kubernetes (ACK), é possível gerenciar recursos da AWS por meio de APIs do Kubernetes, o que permite definir buckets do S3, bancos de dados do RDS, perfis do IAM e demais recursos da AWS como recursos personalizados do Kubernetes. É possível gerenciar recursos da AWS juntamente com as workloads do Kubernetes usando as mesmas ferramentas e fluxos de trabalho, com suporte para mais de 50 serviços da AWS, incluindo o S3, o RDS, o DynamoDB e o Lambda.

Argo CD

O Argo CD executa a implantação contínua baseada em GitOps para as workloads de aplicações, recursos da AWS e configuração do cluster, usando repositórios do Git como fonte única de verdade. O Argo CD sincroniza automaticamente os clusters com os repositórios do Git e detecta desvios, realizando a reconciliação contínua para garantir que as aplicações e os recursos implantados correspondam ao estado desejado no controle de versão. Você pode usar o Argo CD para gerenciar aplicações em um cluster específico ou implantar e gerenciar aplicações em vários clusters de um único recurso do Argo CD, com a implantação automatizada de repositórios do Git sempre que alterações forem confirmadas.

kro (Kube Resource Orchestrator)

O kro (Kube Resource Orchestrator) possibilita a criação de APIs personalizadas do Kubernetes que compõem vários recursos em abstrações de nível superior, permitindo que as equipes responsáveis pela plataforma definam padrões reutilizáveis para combinações comuns de recursos. Isso permite que as equipes responsáveis pela plataforma forneçam funcionalidades de autoatendimento com as barreiras de proteção adequadas, permitindo que os desenvolvedores provisionem infraestruturas complexas por meio de APIs simples e personalizadas, mantendo os padrões e as práticas recomendadas da organização.

Conceitos sobre Kubernetes

O Amazon Elastic Kubernetes Service (Amazon EKS) é um serviço gerenciado AWS baseado no projeto Kubernetes de código aberto. Embora seja necessário saber como o serviço Amazon EKS se integra à Nuvem AWS (especialmente quando você cria um cluster do Amazon EKS pela primeira vez), uma vez instalado e em execução, o cluster do Amazon EKS poderá ser usado da mesma forma que qualquer outro cluster do Kubernetes. Portanto, para começar a gerenciar clusters do Kubernetes e implantar workloads, é necessário ter pelo menos uma compreensão básica dos conceitos do Kubernetes.

Esta página divide os conceitos do Kubernetes em três seções: the section called “Por que usar o Kubernetes?”, the section called “Clusters” e the section called “Workloads”. A primeira seção descreve o valor da execução de um serviço do Kubernetes, em particular como um serviço gerenciado como o Amazon EKS. A seção Workloads aborda como as aplicações do Kubernetes são criadas, armazenadas, executadas e gerenciadas. A seção Clusters apresenta os diferentes componentes que compõem os clusters do Kubernetes e quais são suas responsabilidades pela criação e manutenção dos clusters do Kubernetes.

Tópicos

- Por que usar o Kubernetes?
- Clusters
- Workloads
- Próximas etapas

À medida que você avançar nesse conteúdo, os links levarão a descrições adicionais dos conceitos do Kubernetes tanto na documentação do Amazon EKS e quanto na do Kubernetes, caso deseje se aprofundar em algum dos tópicos abordados aqui. Para obter detalhes sobre como o Amazon EKS implementa o ambiente de gerenciamento e os recursos computacionais do Kubernetes, consulte the section called “Arquitetura”.

Por que usar o Kubernetes?

O Kubernetes foi projetado para aprimorar a disponibilidade e a escalabilidade ao executar aplicações em contêineres essenciais e com qualidade de produção. Em vez de apenas executar o Kubernetes em uma única máquina (embora isso seja possível), o Kubernetes alcança esses objetivos permitindo que você execute aplicações em conjuntos de computadores que podem se expandir ou se contrair para atender à demanda. O Kubernetes inclui recursos que facilitam para você:

- Implantar aplicações em várias máquinas (usando contêineres implantados em pods)
- Monitorar a integridade de contêineres e reiniciar contêineres que falharam
- Aumentar ou reduzir a escala dos contêineres verticalmente dependendo da carga
- Atualizar os contêineres com versões novas
- Alocar recursos entre contêineres
- Equilibrar o tráfego entre máquinas

A automatização desses tipos de tarefas complexas com o Kubernetes permite que os desenvolvedores de aplicações se concentrem em criar e melhorar suas workloads de aplicações, em vez de se preocupar com a infraestrutura. O desenvolvedor normalmente cria arquivos de configuração formatados como arquivos YAML que descrevem o estado desejado da aplicação. Isso pode incluir quais contêineres executar, limites de recursos, número de réplicas de pod, alocação de CPU/memória, regras de afinidade e muito mais.

Atributos do Kubernetes

Para alcançar seus objetivos, o Kubernetes oferece os seguintes atributos:

- Containerized : é uma ferramenta de orquestração de contêineres do Kubernetes. Para usar o Kubernetes, suas aplicações deverão ser conteinerizadas primeiro. Dependendo do tipo de aplicação, isso pode ocorrer na forma de um conjunto de microsserviços, como trabalhos em lotes ou em outras formas. Então, suas aplicações podem usar um fluxo de trabalho do Kubernetes que engloba um enorme ecossistema de ferramentas, em que os contêineres podem ser armazenados como imagens em um registro de contêineres, implantados em um cluster do Kubernetes e executados em um nó disponível. É possível criar e testar contêineres individuais em seu computador local com o Docker ou outro runtime de contêiner antes de implantá-los no cluster do Kubernetes.
- Escalonável: se a demanda por suas aplicações exceder a capacidade das instâncias em execução dessas aplicações, o Kubernetes poderá ter a escala aumentada verticalmente. Conforme necessário, o Kubernetes pode dizer se as aplicações precisam de mais CPU ou memória e responder expandindo automaticamente a capacidade disponível ou usando mais da capacidade existente. O aumento da escala pode ser feito no nível do pod, se houver computação suficiente disponível para executar mais instâncias da aplicação (escalação automática horizontal do pod), ou no nível do nó, se mais nós precisarem ser criados para lidar com o aumento da capacidade (Cluster Autoscaler ou Karpenter). Como a capacidade não é mais necessária, esses serviços podem excluir pods desnecessários e desligar nós desnecessários.
- Disponível: se uma aplicação ou nó se tornar não íntegro ou indisponível, o Kubernetes poderá mover as workloads em execução para outro nó disponível. Você pode forçar o problema simplesmente excluindo uma instância em execução de uma workload ou um nó que esteja executando suas workloads. O ponto principal aqui é que as workloads poderão ser levadas para outros locais se não for mais possível executá-las onde estão.
- Declarativo: o Kubernetes usa a reconciliação ativa para verificar constantemente se o estado que você declara para o cluster corresponde ao estado real. Ao aplicar objetos do Kubernetes a um cluster, normalmente por meio de arquivos de configuração formatados em YAML, você pode, por exemplo, solicitar a inicialização das workloads que deseja executar no cluster. Posteriormente, é possível alterar as configurações para fazer algo como usar uma versão mais recente de um contêiner ou alocar mais memória. O Kubernetes fará o que for necessário para estabelecer o estado desejado. Isso pode incluir ativar ou desativar nós, interromper e reiniciar workloads ou obter contêineres atualizados.

- Compossível- como uma aplicação geralmente consiste em vários componentes, você quer poder gerenciar um conjunto desses componentes (geralmente representados por vários contêineres) juntos. Embora o Docker Compose ofereça uma maneira de fazer isso diretamente com o Docker, o comando Kompose do Kubernetes pode ajudar você a fazer isso com o Kubernetes. Consulte Traduzir um arquivo Docker Compose para recursos do Kubernetes para obter um exemplo de como fazer isso.
- Extensível: ao contrário do software proprietário, o projeto de código aberto do Kubernetes foi projetado para ser aberto para você, que pode estender o Kubernetes da maneira que desejar para atender às suas necessidades. As APIs e os arquivos de configuração são abertos para modificações diretas. Incentivamos os terceiros a criar seus próprios controladores para estender a infraestrutura e os recursos do usuário final do Kubernetes. Os webhooks permitem que você configure regras de cluster para aplicar políticas e se adaptar às mudanças nas condições. Para obter mais ideias sobre como estender os clusters do Kubernetes, consulte Como estender Kubernetes.
- Portátil: muitas organizações padronizaram suas operações no Kubernetes porque ele permite gerenciar todas as necessidades de aplicações da mesma forma. Os desenvolvedores podem usar os mesmos pipelines para criar e armazenar aplicações conteinerizadas. Essas aplicações podem então ser implantadas em clusters do Kubernetes executados on-premises, em nuvens, em terminais de ponto de venda em restaurantes ou em dispositivos de IoT distribuídos pelos locais remotos da empresa. Sua natureza de código aberto permite que as pessoas desenvolvam essas distribuições especiais do Kubernetes, junto com as ferramentas necessárias para gerenciá-las.

Gerenciar o Kubernetes

O código-fonte do Kubernetes está disponível gratuitamente, portanto, com seu próprio equipamento, você pode instalar e gerenciar o Kubernetes por conta própria. No entanto, o autogerenciamento do Kubernetes requer profundo conhecimento operacional e exige tempo e esforços para ser mantido. Por esses motivos, a maioria das pessoas que implantam workloads de produção escolhe um provedor de nuvem (como o Amazon EKS) ou um provedor on-premises (como o Amazon EKS Anywhere) com sua própria distribuição do Kubernetes testada e o suporte de especialistas no Kubernetes. Isso permite a você descarregar grande parte do trabalho pesado indiferenciado necessário para manter seus clusters, incluindo:

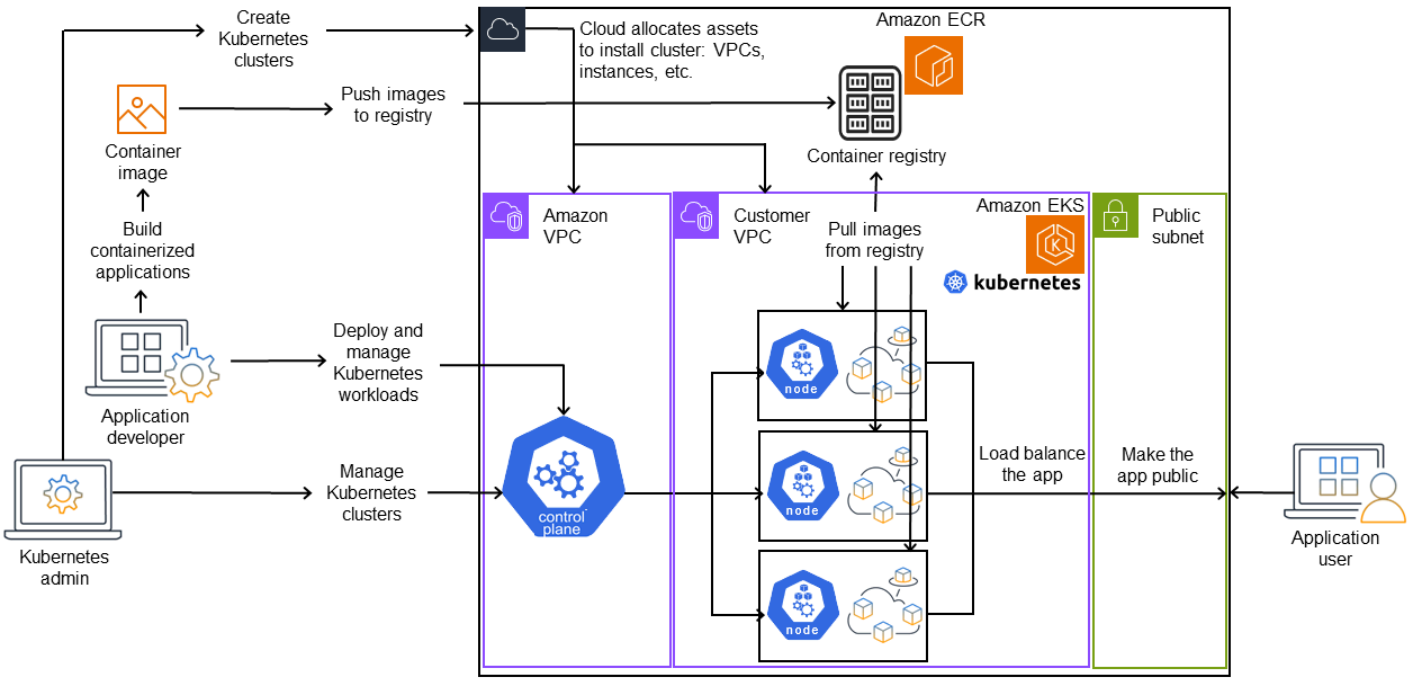
- Hardware: se você não tiver hardware disponível para executar o Kubernetes de acordo com seus requisitos, um provedor de nuvem, como o AWS Amazon EKS, pode reduzir os custos iniciais. Com o Amazon EKS, isso significa que é possível consumir os melhores recursos de nuvem

oferecidos pela AWS, incluindo instâncias de computador (Amazon Elastic Compute Cloud), seu próprio ambiente privado (Amazon VPC), gerenciamento central de identidade e permissões (IAM) e armazenamento (Amazon EBS). A AWS gerencia os computadores, as redes, os data centers e todos os outros componentes físicos necessários para executar o Kubernetes. Da mesma forma, você não precisa planejar seu data center para lidar com a capacidade máxima nos dias de maior demanda. Para o Amazon EKS Anywhere ou outros clusters do Kubernetes on-premises, você é responsável por gerenciar a infraestrutura usada em suas implantações do Kubernetes, mas ainda pode contar com a AWS para manter o Kubernetes atualizado.

- Administrar o ambiente de gerenciamento: o Amazon EKS gerencia a segurança e a disponibilidade do ambiente de gerenciamento do Kubernetes hospedado pela AWS, que é responsável por agendar contêineres, gerenciar a disponibilidade de aplicações e outras tarefas importantes para que você possa se concentrar em suas workloads de aplicações. Se seu cluster falhar, é esperado que a AWS tenha os meios necessários para restaurá-lo para um estado de execução. Para o Amazon EKS Anywhere, você mesmo gerenciaria o ambiente de gerenciamento.
- Atualizações testadas: ao atualizar seus clusters, você pode contar com o Amazon EKS ou o Amazon EKS Anywhere para fornecer versões testadas de suas distribuições do Kubernetes.
- Complementos: há centenas de projetos criados para estender e trabalhar com o Kubernetes que podem ser adicionados à infraestrutura do cluster ou ser usados para ajudar na execução das workloads. Em vez de desenvolver e gerenciar esses complementos por conta própria, a AWS fornece the section called “Complementos do Amazon EKS” que você pode usar com seus clusters. O Amazon EKS Anywhere fornece pacotes selecionados que incluem compilações de muitos projetos populares de código aberto. Assim, você não precisa criar o software sozinho ou gerenciar patches críticos de segurança, correções de bugs ou atualizações. Da mesma forma, se os padrões atenderem às suas necessidades, é normal que esses complementos exijam muito pouca configuração. Consulte the section called “Cluster estendido” para obter detalhes sobre como estender seu cluster com complementos.

Kubernetes em ação

O diagrama a seguir mostra as principais atividades que você realizará como administrador ou desenvolvedor de aplicações do Kubernetes para criar e usar um cluster do Kubernetes. No processo, ele ilustra como os componentes do Kubernetes interagem entre si usando a Nuvem AWS como exemplo do provedor de nuvem subjacente.



Um administrador do Kubernetes cria o cluster do Kubernetes usando uma ferramenta específica para o tipo de provedor no qual o cluster será criado. Este exemplo usa a Nuvem AWS como provedor, que oferece o serviço gerenciado do Kubernetes denominado Amazon EKS. O serviço gerenciado realiza automaticamente a alocação dos recursos necessários para criar o cluster, o que inclui a criação de duas novas nuvens privadas virtuais (Amazon VPCs) para o cluster, a configuração de redes e o mapeamento de permissões do Kubernetes diretamente nas novas VPCs para o gerenciamento de ativos na nuvem. O serviço gerenciado também garante que os serviços do ambiente de gerenciamento tenham locais para serem executados, e aloca zero ou mais instâncias do Amazon EC2 como nós do Kubernetes para a execução de workloads. A AWS gerencia uma Amazon VPC para o ambiente de gerenciamento, enquanto a outra Amazon VPC contém os nós do cliente que executam as workloads.

Muitas das futuras tarefas do administrador do Kubernetes serão realizadas usando ferramentas do Kubernetes como o `kubectl`. Essa ferramenta faz solicitações de serviços diretamente no ambiente de gerenciamento do cluster. As maneiras como as consultas e alterações são feitas no cluster são então muito semelhantes às formas como você as faria em qualquer cluster do Kubernetes.

Um desenvolvedor de aplicações que deseja implantar workloads nesse cluster pode realizar várias tarefas. O desenvolvedor precisa criar a aplicação em uma ou mais imagens de contêiner e, em

seguida, enviar essas imagens para um registro de contêineres acessível pelo cluster do Kubernetes. A AWS oferece o Amazon Elastic Container Registry (Amazon ECR) para esse fim.

Para executar a aplicação, o desenvolvedor pode criar arquivos de configuração no formato YAML que informam ao cluster como executar a aplicação, incluindo quais contêineres extrair do registro e como agrupá-los em pods. O ambiente de gerenciamento (agendador) agenda os contêineres em um ou mais nós e o runtime do contêiner em cada nó realmente extrai e executa os contêineres necessários. O desenvolvedor também pode configurar um application load balancer para equilibrar o tráfego para os contêineres disponíveis em execução em cada nó e expor a aplicação para que ela se torne disponível em uma rede pública para o mundo externo. Com tudo isso feito, alguém que queira usar a aplicação poderá se conectar ao endpoint da aplicação para acessá-la.

As seções a seguir analisam os detalhes de cada um desses recursos, da perspectiva de clusters e workloads do Kubernetes.

Clusters

Se seu trabalho é iniciar e gerenciar clusters do Kubernetes, tenha em mente como os clusters do Kubernetes são criados, aprimorados, gerenciados e excluídos. Você também deverá saber quais são os componentes que compõem um cluster e o que precisa ser feito para mantê-los.

As ferramentas para gerenciar clusters lidam com a sobreposição entre os serviços do Kubernetes e o provedor de hardware subjacente. Por esse motivo, a automação dessas tarefas tende a ser feita pelo provedor do Kubernetes (como o Amazon EKS ou Amazon EKS Anywhere) usando ferramentas específicas para o provedor. Por exemplo, para iniciar um cluster do Amazon EKS, é possível usar `eksctl create cluster`, enquanto que para o Amazon EKS Anywhere é possível usar `eksctl anywhere create cluster`. Observe que, embora esses comandos criem um cluster do Kubernetes, eles são específicos do provedor e não fazem parte do projeto do Kubernetes em si.

Ferramentas de criação e gerenciamento de clusters

O projeto do Kubernetes oferece ferramentas para criar um cluster do Kubernetes manualmente. Portanto, se você deseja instalar o Kubernetes em uma única máquina, ou executar o ambiente de gerenciamento em uma máquina e adicionar nós manualmente, pode usar ferramentas da CLI como kind, minikube ou kubeadm, que estão listadas em Instalar ferramentas do Kubernetes. Para simplificar e automatizar todo o ciclo de vida da criação e do gerenciamento de clusters, é muito mais fácil usar ferramentas com suporte de um provedor do Kubernetes estabelecido, como o Amazon EKS ou o Amazon EKS Anywhere.

Na Nuvem AWS, é possível criar clusters do Amazon EKS usando ferramentas de CLI, como eksctl, ou ferramentas mais declarativas, como o Terraform (consulte Amazon EKS Blueprints for Terraform). Você também pode criar um cluster no Console de gerenciamento da AWS. Consulte Recursos do Amazon EKS para ver uma lista do que você obtém com o Amazon EKS. As responsabilidades do Kubernetes que o Amazon EKS assume por você incluem:

- Plano de controle gerenciado- o site AWS garante que o cluster do Amazon EKS esteja disponível e seja dimensionável, pois gerencia o ambiente de gerenciamento para você e o disponibiliza em AWS Availability Zones.
- Gerenciamento de nós: em vez de adicionar nós manualmente, você pode fazer com que o Amazon EKS crie nós automaticamente conforme necessário, usando os grupos de nós gerenciados (consulte the section called “Grupos de nós gerenciados”) ou o Karpenter. Os grupos de nós gerenciados têm integrações com o ajuste de escala automático de clusters do Kubernetes. Usando as ferramentas de gerenciamento de nós, você pode se beneficiar da economia de custos graças a recursos como instâncias spot, consolidação e disponibilidade de nós e utilização de recursos de programação para definir como as workloads são implantadas e os nós são selecionados.
- Redes do cluster: usando modelos do CloudFormation, o `eksctl`configura a rede entre os componentes do ambiente de gerenciamento e do plano de dados (nó) no cluster do Kubernetes. Ele também configura endpoints por meio dos quais as comunicações internas e externas podem ocorrer. Consulte Desmistificar redes de cluster para nós de processamento do Amazon EKS para obter detalhes. As comunicações entre os Pods no Amazon EKS são realizadas por meio das Identidades de Pods do Amazon EKS (consulte the section called “Identidade de Pods”), que fornecem uma maneira de permitir que os Pods utilizem métodos da AWS para gerenciar credenciais e permissões em nuvem.
- Complementos: o Amazon EKS evita que você tenha que criar e adicionar componentes de software que são comumente usados para dar suporte a clusters do Kubernetes. Por exemplo, quando você cria um cluster do Amazon EKS usando o Console de gerenciamento da AWS, ele adiciona automaticamente os complementos kube-proxy do Amazon EKS (the section called “kube-proxy”), plug-in da CNI da Amazon VPC para Kubernetes (the section called “CNI da Amazon VPC”) e CoreDNS (the section called “CoreDNS”). Consulte the section called “Complementos do Amazon EKS” para obter mais informações sobre esses complementos, incluindo uma lista dos que estão disponíveis.

Para executar seus clusters em seus próprios computadores e redes on-premises, a Amazon oferece o Amazon EKS Anywhere. Em vez de a Nuvem AWS atuar como o provedor, existe a opção

de executar o Amazon EKS Anywhere nas plataformas VMware vSphere, bare metal (provedor Tinkerbell), Snow, CloudStack ou Nutanix usando seu próprio equipamento.

O Amazon EKS Anywhere é baseado no mesmo software Amazon EKS Distro usado pelo Amazon EKS. No entanto, o Amazon EKS Anywhere depende de diferentes implementações da interface da API de cluster do Kubernetes (CAPI) para gerenciar o ciclo de vida completo das máquinas em um cluster do Amazon EKS Anywhere (como CAPV para vSphere e CAPC para CloudStack). Como o cluster inteiro está sendo executado em seu equipamento, você assume a responsabilidade adicional de gerenciar o ambiente de gerenciamento e fazer backup dos seus dados (consulte `etcd` mais adiante neste documento).

Componentes do cluster

Os componentes do cluster do Kubernetes são divididos em duas áreas principais: ambiente de gerenciamento e nós de processamento. Os componentes do ambiente de gerenciamento gerenciam o cluster e fornecem acesso às suas APIs. Os nós de processamento (às vezes chamados apenas de nós) fornecem os locais onde as workloads reais são executadas. Os componentes do nó consistem em serviços que são executados em cada nó para se comunicar com o ambiente de gerenciamento e executar contêineres. O conjunto de nós de processamento do seu cluster é chamado de plano de dados.

Ambiente de gerenciamento

O ambiente de gerenciamento consiste em um conjunto de serviços que gerenciam o cluster. Todos esses serviços podem ser executados em um único computador ou estar espalhados por vários computadores. Internamente, elas são chamadas de instâncias do ambiente de gerenciamento (CPIs). A forma como as CPIs são executadas depende do tamanho do cluster e dos requisitos de alta disponibilidade. Conforme a demanda aumenta no cluster, um serviço de ambiente de gerenciamento pode ser escalado para fornecer mais instâncias desse serviço, com as solicitações sendo balanceadas entre as instâncias.

As tarefas realizadas pelos componentes do ambiente de gerenciamento do Kubernetes incluem:

- Comunicação com os componentes do cluster (servidor de API): o servidor de API (kube-apiserver) expõe a API do Kubernetes para que as solicitações ao cluster possam ser feitas de dentro e de fora do cluster. Em outras palavras, as solicitações para adicionar ou alterar os objetos de um cluster (pods, serviços, nós etc.) podem vir de comandos externos, como solicitações de `kubectl` para executar um pod. Da mesma forma, as solicitações podem ser feitas do servidor

da API para componentes dentro do cluster, como uma consulta ao serviço `kubelet` para saber o status de um pod.

- Armazenamento de dados sobre o cluster (armazenamento do valor da chave **`etcd`**): o serviço `etcd` exerce o papel fundamental de registrar o estado atual do cluster. Se o serviço `etcd` se tornasse inacessível, você não conseguiria atualizar ou consultar o status do cluster, embora as workloads continuassem sendo executadas por algum tempo. Por esse motivo, os clusters críticos geralmente têm várias instâncias do serviço `etcd` com balanceamento de carga em execução ao mesmo tempo e fazem backups periódicos do armazenamento dos valores de chave do `etcd` para o caso de perda ou corrompimento de dados. Lembre-se de que, no Amazon EKS, tudo isso é gerenciado automaticamente para você por padrão. O Amazon EKS Anywhere fornece instruções sobre backup e restauração de etcd. Consulte o etcd Data Model para aprender como o `etcd` gerencia os dados.
- Agendar pods para nós (agendador): as solicitações para iniciar ou parar um pod no Kubernetes são direcionadas para o Agendador do Kubernetes (kube-scheduler). Como um cluster pode ter vários nós capazes de executar o pod, cabe ao agendador escolher em qual nó (ou nós, no caso de réplicas) o pod deve ser executado. Se não houver capacidade disponível suficiente para executar o pod solicitado em um nó existente, a solicitação falhará, a menos que você tenha feito outras provisões. Essas provisões podem incluir a habilitação de serviços, como grupos de nós gerenciados (the section called “Grupos de nós gerenciados”) ou Karpenter, que são capazes de iniciar automaticamente novos nós para gerenciar as workloads.
- Manter os componentes no estado desejado (Controller Manager): o Controller Manager do Kubernetes é executado como um processo daemon (kube-controller-manager) para observar o estado do cluster e fazer alterações no cluster para restabelecer os estados esperados. Em particular, existem vários controladores que observam diferentes objetos do Kubernetes, incluindo um `statefulset-controller`, `endpoint-controller`, `cronjob-controller`, `node-controller` e outros.
- Gerenciar recursos de nuvem (Cloud Controller Manager): as interações entre o Kubernetes e o provedor de nuvem que realiza solicitações para os recursos subjacentes do data center são tratadas pelo Cloud Controller Manager (cloud-controller-manager). Os controladores gerenciados pelo Cloud Controller Manager podem incluir um controlador de rota (para configurar rotas de rede na nuvem), controlador de serviço (para usar serviços de balanceamento de carga na nuvem) e controlador de ciclo de vida de nós (para manter os nós sincronizados com o Kubernetes durante os ciclos de vida).

Nós de processamento (plano de dados)

Para um cluster do Kubernetes de nó único, as workloads são executadas na mesma máquina que o ambiente de gerenciamento. No entanto, uma configuração mais padrão é ter um ou mais sistemas de computador separados (nós) dedicados à execução de workloads do Kubernetes.

Quando você cria um cluster do Kubernetes pela primeira vez, algumas ferramentas de criação de clusters permitem configurar um determinado número de nós a serem adicionados ao cluster (seja identificando sistemas de computador existentes ou fazendo com que o provedor crie novos sistemas). Antes que qualquer workload seja adicionada a esses sistemas, serviços são adicionados a cada nó para implementar esses recursos:

- Gerenciar cada nó (**`kubelet`**): o servidor de API se comunica com o serviço kubelet em execução em cada nó para garantir que o nó esteja corretamente registrado e que os Pods solicitados pelo Agendador estejam em execução. O kubelet pode ler os manifestos do Pod e configurar volumes de armazenamento ou outros recursos necessários aos Pods no sistema local. Ele também pode verificar a integridade dos contêineres executados localmente.
- Executar contêineres em um nó (runtime do contêiner)- o runtime do contêiner em cada nó gerencia os contêineres solicitados para cada pod atribuído ao nó. Isso significa que ele pode extrair imagens do contêiner do registro apropriado, executar o contêiner, interrompê-lo e responder às consultas sobre o contêiner. O runtime de contêiner padrão é containerd. A partir da versão 1.24 do Kubernetes, a integração especial do Docker (`dockershim`) que poderia ser usada como runtime de contêiner foi removida do Kubernetes. Embora ainda seja possível usar o Docker para testar e executar contêineres em seu sistema local, para usar o Docker com o Kubernetes, agora é necessário Instalar o mecanismo do Docker em cada nó para usá-lo com o Kubernetes.
- Gerenciar a rede entre contêineres (**`kube-proxy`**): para poder oferecer suporte à comunicação entre pods, o Kubernetes usa um recurso conhecido como Service para configurar redes de pods que rastreiam endereços IP e portas associados a esses pods. O serviço kube-proxy é executado em cada um dos nós para permitir que a comunicação entre os pods ocorra.

Cluster estendido

Há alguns serviços que podem ser adicionados ao Kubernetes para oferecer suporte ao cluster, mas eles não são executados no ambiente de gerenciamento. Esses serviços geralmente são executados diretamente em nós no namespace kube-system ou em seu próprio namespace (como geralmente é feito com provedores de serviços terceirizados). Um exemplo comum é o serviço

CoreDNS, que fornece serviços de DNS ao cluster. Consulte Descobrindo serviços integrados para obter informações sobre como ver quais serviços de cluster estão sendo executados no kube-system em seu cluster.

Há diferentes tipos de complementos que podem ser adicionados aos seus clusters. Para garantir a integridade dos seus clusters, é possível adicionar recursos de observabilidade (consulte Monitorar clusters) que possibilitam atividades como registro de log, auditoria e métricas. Com essas informações, você pode solucionar problemas que ocorrem, geralmente por meio das mesmas interfaces de observabilidade. Exemplos desses tipos de serviços incluem o Amazon GuardDuty, o CloudWatch (consulte the section called “Amazon CloudWatch”), o AWS Distro para OpenTelemetry, o plug-in CNI da Amazon VPC para Kubernetes (consulte the section called “CNI da Amazon VPC”) e o Monitoramento do Kubernetes no Grafana. Para armazenamento (consulte Armazenamento de dados de aplicações), os complementos do Amazon EKS incluem o driver da CSI para o Amazon Elastic Block Store (consulte the section called “Amazon EBS”), o driver da CSI para o Amazon Elastic File System (consulte the section called “Amazon EFS”), e diversos complementos de armazenamento de entidades externas, como o driver da CSI para o Amazon FSx para NetApp ONTAP (consulte the section called “Amazon FSx para NetApp ONTAP”).

Para obter uma lista mais completa dos complementos do Amazon EKS disponíveis, consulte the section called “Complementos do Amazon EKS”.

Workloads

O Kubernetes define uma workload como “uma aplicação em execução no Kubernetes”. Essa aplicação pode consistir em um conjunto de microsserviços executados como contêineres em pods ou pode ser executada como um trabalho em lote ou outro tipo de aplicações. O trabalho do Kubernetes é garantir que as solicitações feitas para que esses objetos sejam configurados ou implantados sejam executadas. Como alguém que implanta aplicações, você deve aprender sobre como os contêineres são criados, como os pods são definidos e quais métodos você pode usar para implantá-los.

Contêineres

O elemento mais básico de uma workload de aplicação que você implementa e gerencia no Kubernetes é um Pod . Um pod representa uma forma de sustentar os componentes de uma aplicação, bem como de definir especificações que descrevem os atributos do pod. Compare isso com algo como um pacote RPM ou Deb, que agrupa software para um sistema Linux, mas não é executado como uma entidade.

Como o pod é a menor unidade implantável, ele normalmente contém um único contêiner. No entanto, vários contêineres podem existir em um pod nos casos em que os contêineres estão fortemente acoplados. Por exemplo, um contêiner de servidor Web pode ser empacotado em um pod com um tipo de contêiner auxiliar que pode fornecer registro em log, monitoramento ou outro serviço intimamente vinculado ao contêiner do servidor Web. Nesse caso, estar no mesmo pod garante que, para cada instância em execução do pod, os dois contêineres sempre sejam executados no mesmo nó. Da mesma forma, todos os contêineres em um pod compartilham o mesmo ambiente, com os contêineres em um pod funcionando como se estivessem no mesmo host isolado. O efeito disso é que os contêineres compartilham um único endereço IP que fornece acesso ao pod e podem se comunicar entre si como se estivessem sendo executados em seu próprio host local.

As especificações do pod (PodSpec) definem o estado desejado do pod. É possível implantar um pod individual ou vários pods usando recursos de workload para gerenciar modelos de pod. Os recursos de workload incluem Implantações (para gerenciar várias réplicas de pod), StatefulSets (para implantar pods que precisam ser exclusivos, como pods de banco de dados) e DaemonSets (em que um pod precisa ser executado continuamente em cada nó). Falaremos mais sobre isso mais tarde.

Enquanto um pod é a menor unidade que você pode implantar, um contêiner é a menor unidade que você cria e gerencia.

Criar contêineres

O pod é, na verdade, apenas uma estrutura em torno de um ou mais contêineres, onde cada contêiner contém o sistema de arquivos, executáveis, arquivos de configuração, bibliotecas e outros componentes para realmente executar a aplicação. Pelo fato de uma empresa chamada Docker Inc. ter popularizado os contêineres pela primeira vez, algumas pessoas se referem aos contêineres como contêineres do Docker. No entanto, desde então, a Open Container Initiative definiu tempos de execução, imagens e métodos de distribuição de contêineres para o setor. Adicione a isso o fato de que os contêineres foram criados com base em muitos recursos existentes do Linux, outros geralmente se referem aos contêineres como contêineres OCI, contêineres Linux ou apenas contêineres.

Quando você cria um contêiner, normalmente começa com um Dockerfile (literalmente chamado assim). Dentro desse Dockerfile, é possível identificar:

- Uma imagem de base- Uma imagem de contêiner de base é um contêiner normalmente criado a partir de uma versão mínima do sistema de arquivos de um sistema operacional (como o Red Hat

Enterprise Linux ou o Ubuntu) ou de um sistema mínimo aprimorado para fornecer software para executar tipos específicos de aplicações (como aplicações nodejs ou python).

- Software aplicação- Você pode adicionar o software aplicação ao contêiner da mesma forma que o adicionaria a um sistema Linux. Por exemplo, em seu Dockerfile, é possível executar `npm` e `yarn` para instalar uma aplicação Java ou `yum` e `dnf` para instalar pacotes RPM. Em outras palavras, com um comando RUN em um Dockerfile, é possível executar qualquer comando que esteja disponível no sistema de arquivos da sua imagem base para instalar ou configurar o software dentro da imagem de contêiner resultante.
- Instruções- A referência do Dockerfile descreve as instruções que você pode adicionar a um Dockerfile ao configurá-lo. Isso inclui instruções usadas para criar o que está no próprio contêiner (`ADD` ou `COPY` arquivos do sistema local), identificar comandos para executar quando o contêiner for executado (`CMD` ou `ENTRYPOINT`) e conectar o contêiner ao sistema em que ele é executado (identificando o `USER` de execução, um `VOLUME` local para montagem ou as portas para `EXPOSE`).

Embora o comando `docker` e o serviço tenham sido tradicionalmente usados para criar containers (`docker build`), outras ferramentas disponíveis para criar imagens de contêineres incluem podman e nerdctl. Consulte Criação de melhores imagens de contêineres ou Visão geral da compilação do Docker para saber mais sobre a criação de contêineres.

Armazenamento de contêineres

Após criar sua imagem de contêiner, você poderá armazená-la em um registro de distribuição de contêineres em sua estação de trabalho ou em um registro de contêiner público. A execução de um registro de contêiner privado em sua estação de trabalho permite armazenar imagens de contêiner localmente, tornando-as prontamente disponíveis para você.

Para armazenar imagens de contêiner de forma mais pública, é possível enviá-las para um registro de contêiner público. Os registros de contêineres públicos fornecem um local central para armazenar e distribuir imagens de contêineres. Exemplos de registros de contêineres públicos incluem o Amazon Elastic Container Registry, o Red Hat Quay Registry e o Docker Hub Registry.

Ao executar workloads em contêineres no Amazon Elastic Kubernetes Service (Amazon EKS), recomendamos obter cópias de imagens oficiais do Docker armazenadas no Amazon Elastic Container Registry. O Amazon ECR armazena essas imagens desde 2021. Você pode pesquisar imagens de contêineres populares na Galeria Pública do Amazon ECR e, especificamente para as imagens do Docker Hub, pode pesquisar na Galeria do Docker do Amazon ECR.

Depure contêineres em execução

Como os contêineres são criados em um formato padrão, um contêiner pode ser executado em qualquer máquina capaz de executar um runtime de contêiner (como o Docker) e cujo conteúdo corresponda à arquitetura da máquina local (como `x86_64` ou `arm`). Para testar um contêiner ou simplesmente executá-lo em seu desktop local, você pode usar nossos comandos `docker run` ou `podman run` para iniciar um contêiner no host local. Para o Kubernetes, no entanto, cada nó de processamento tem um runtime de contêiner implantado, e cabe ao Kubernetes solicitar que um nó execute um contêiner.

Depois que um contêiner é designado para ser executado em um nó, o nó verifica se a versão solicitada da imagem do contêiner já existe nesse nó. Do contrário, o Kubernetes solicita ao runtime do contêiner que extraia esse contêiner do registro de contêiner apropriado e, em seguida, execute esse contêiner localmente. Lembre-se de que uma imagem de contêiner refere-se ao pacote de software que é movido entre seu laptop, o registro de contêiner e os nós do Kubernetes. Um contêiner se refere a uma instância em execução dessa imagem.

Pods

Quando seus contêineres estiverem prontos, trabalhar com os pods inclui configurar, implantar e tornar os pods acessíveis.

Configuração de pods

Ao definir um pod, você atribui um conjunto de atributos a ele. Esses atributos devem incluir pelo menos o nome do pod e a imagem de contêiner a ser executada. No entanto, há muitas outras coisas que podem ser configuradas com as definições do pod (consulte a página PodSpec para obter detalhes sobre o que pode ser incluído em um pod). Isso inclui:

- Armazenamento: quando um contêiner em execução é interrompido e excluído, o armazenamento de dados nesse contêiner desaparece, a menos que você configure um armazenamento mais permanente. O Kubernetes é compatível com muitos tipos diferentes de armazenamento e os abstrai sob o conceito de Volumes. Os tipos de armazenamento incluem CephFS, NFS, iSCSI e outros. Você pode até mesmo usar um dispositivo de blocos local conectado ao computador local. Com um desses tipos de armazenamento disponíveis em seu cluster, é possível montar o volume de armazenamento em um ponto de montagem selecionado no sistema de arquivos do seu contêiner. Um volume persistente é aquele que continua existindo depois que o pod é excluído, enquanto um volume efêmero é removido quando o pod é excluído. Se o administrador do cluster tiver criado diferentes StorageClasses para o cluster, talvez você tenha a opção de

escolher os recursos do armazenamento que você usa como, por exemplo, se o volume é excluído ou recuperado após o uso, se ele se expandirá se for necessário mais espaço e até mesmo se ele atende a determinados requisitos de performance.

- Segredos- ao disponibilizar Segredos para os contêineres nas especificações do Pod, você pode fornecer as permissões de que esses contêineres precisam para acessar sistemas de arquivos, bancos de dados ou outros ativos protegidos. Chaves, senhas e tokens estão entre os itens que podem ser armazenados como segredos. O uso de segredos faz com que você não precise armazenar essas informações em imagens de contêineres, mas apenas disponibilizá-los para contêineres em execução. Semelhantes aos segredos são os ConfigMaps. Um `ConfigMap` tende a conter informações menos críticas, como pares de chave-valor para configurar um serviço.
- Recursos de contêineres- os objetos para configuração adicional de contêineres podem assumir a forma de configuração de recursos. Para cada contêiner, é possível solicitar a quantidade de memória e CPU que ele pode usar, bem como impor limites para a quantidade total desses recursos que o contêiner pode usar. Consulte Gerenciamento de recursos para pods e contêineres para obter exemplos.
- Interrupções- os pods podem ser interrompidos involuntariamente (um nó fica inoperante) ou voluntariamente (uma atualização é desejada). Ao configurar um orçamento de interrupção do pod, é possível exercer algum controle sobre a disponibilidade da sua aplicação quando interrupções ocorrem. Consulte Especificação de um orçamento de interrupção para sua aplicação para obter exemplos.
- Namespaces: o Kubernetes oferece diferentes maneiras de isolar os componentes e as workloads do Kubernetes uns dos outros. Executar todos os pods de uma aplicação específica no mesmo Namespace é uma forma comum de proteger e gerenciar esses pods juntos. Você pode criar seus próprios namespaces para usar ou optar por não indicar um namespace (o que faz com que o Kubernetes use o namespace `default`). Os componentes do ambiente de gerenciamento do Kubernetes geralmente são executados no namespace kube-system.

A configuração descrita acima geralmente está reunida em um arquivo YAML para ser aplicada ao cluster do Kubernetes. Para clusters do Kubernetes pessoais, basta armazenar esses arquivos YAML em seu sistema local. No entanto, com clusters e workloads mais críticos, o GitOps é uma forma bastante popular de automatizar o armazenamento e as atualizações dos recursos de workload e infraestrutura do Kubernetes.

Os objetos usados para reunir e implantar as informações do pod são definidos por um dos seguintes métodos de implantação.

Implantar pods

O método que você escolheria para implantar pods depende do tipo de aplicação que você planeja executar com esses pods. Algumas das opções disponíveis são:

- Aplicações stateless: uma aplicação stateless não salva os dados da sessão de um cliente, de modo que outra sessão não precisa consultar o que aconteceu em uma sessão anterior. Isso torna mais fácil simplesmente substituir os pods por novos caso eles percam a integridade ou movê-los sem salvar o estado. Se você estiver executando uma aplicação sem estado (como um servidor Web), poderá usar uma implantação para implantar pods e ReplicaSets. Um ReplicaSet define quantas instâncias de um pod você deseja executar simultaneamente. Embora seja possível executar um ReplicaSet diretamente, é comum executar réplicas diretamente em uma implantação para definir quantas réplicas de um pod devem ser executadas por vez.
- Aplicações stateful: uma aplicação stateful é aquela em que a identidade do pod e a ordem em que os pods são iniciados são importantes. Essas aplicações precisam de armazenamento persistente que seja estável e que precise ser implantado e escalado de forma consistente. Para implantar uma aplicação com estado no Kubernetes, é possível usar StatefulSets. Um exemplo de aplicação que normalmente é executada como StatefulSet é um banco de dados. Em um StatefulSet, é possível definir réplicas, o pod e seus contêineres, volumes de armazenamento a serem montados e locais no contêiner em que os dados são armazenados. Consulte Executar uma aplicação com estado replicada para obter um exemplo de banco de dados sendo implantado como um ReplicaSet.
- Aplicações por nó: há ocasiões em que você deseja executar uma aplicação em todos os nós do cluster do Kubernetes. Por exemplo, seu data center pode exigir que cada computador execute uma aplicação de monitoramento ou um serviço de acesso remoto específico. Para o Kubernetes, é possível usar um DaemonSet para garantir que a aplicação selecionada seja executada em todos os nós do seu cluster.
- Aplicações executadas até a conclusão: há algumas aplicações que você deseja executar para concluir uma determinada tarefa. Isso pode incluir uma que gere relatórios de status mensais ou limpe dados antigos. Um objeto Job pode ser usado para configurar uma aplicação para inicializar e executar e, em seguida, sair quando a tarefa for concluída. Um objeto CronJob permite configurar uma aplicação para ser executado em uma hora, minuto, dia do mês, mês ou dia da semana específicos usando uma estrutura definida pelo formato crontab do Linux.

Como tornar as aplicações acessíveis via rede

Com as aplicações frequentemente implantadas como um conjunto de microsserviços que eram movidos para diferentes lugares, o Kubernetes precisava de uma forma de fazer com que esses microsserviços se encontrassem. Além disso, para que outras pessoas pudessem acessar uma aplicação fora do cluster do Kubernetes, ele precisava de uma forma para expor essa aplicação em endereços e portas externos. Esses recursos relacionados à rede são implementados com objetos Service e Ingress, respectivamente:

- Serviços: como um pod pode se deslocar para diferentes nós e endereços, outro pod que precise se comunicar com o primeiro pode ter dificuldade para localizá-lo. Para resolver esse problema, o Kubernetes permite representar uma aplicação como um Service. Com um Service, é possível identificar um pod ou conjunto de pods com um nome específico e, em seguida, indicar qual porta expõe o serviço dessa aplicação a partir do pod e quais portas outra aplicação poderia usar para entrar em contato com esse serviço. Outro pod em um cluster pode simplesmente solicitar um serviço pelo nome, que o Kubernetes direcionará essa solicitação para a porta adequada para uma instância do pod executando esse serviço.
- Ingress: Ingress é o que pode tornar as aplicações representadas pelos serviços do Kubernetes disponíveis para clientes que estão fora do cluster. Os recursos básicos de Ingress incluem um balanceador de carga (gerenciado pelo Ingress), o controlador do Ingress e regras para rotear solicitações do controlador para o Service. Existem vários controladores do Ingress que você pode escolher com o Kubernetes.

Próximas etapas

Compreender os conceitos básicos do Kubernetes e como eles se relacionam com o Amazon EKS ajudará você a navegar na documentação do Amazon EKS e na documentação do Kubernetes para encontrar as informações necessárias para gerenciar clusters do Amazon EKS e implantar workloads nesses clusters. Para começar a usar o Amazon EKS, escolha uma das seguintes opções:

- the section called “Criar cluster (eksctl)”
- the section called “Criar um cluster”
- the section called “Implantação de exemplo (Linux)”
- Gerenciamento de clusters

Implementar clusters do Amazon EKS em ambientes locais e on-premises

Introdução às opções de implantação do Amazon EKS

O Amazon Elastic Kubernetes Service (Amazon EKS) é um serviço do Kubernetes totalmente gerenciado que permite que você execute o Kubernetes facilmente na nuvem e nos ambientes on-premises.

Na nuvem, o Amazon EKS automatiza o gerenciamento da infraestrutura de clusters do Kubernetes para o ambiente de gerenciamento e os nós do Kubernetes. Isso é essencial para programar contêineres, gerenciar a disponibilidade de aplicações, escalar recursos dinamicamente, otimizar a computação, armazenar dados de cluster e realizar outras funções críticas. Com o Amazon EKS, você obtém performance, escalabilidade, confiabilidade e disponibilidade robustos da infraestrutura da AWS, além de integrações nativas com serviços de rede, segurança, armazenamento e observabilidade da AWS.

Para simplificar a execução do Kubernetes nos ambientes on-premises, você pode usar os mesmos clusters, recursos e ferramentas do Amazon EKS para the section called “Nós” ou o Amazon EKS Hybrid Nodes em sua própria infraestrutura, ou você pode usar o Amazon EKS Anywhere para ambientes isolados e autônomos.

Amazon EKS na nuvem

Você pode usar o Amazon EKS com computação em regiões da AWS, zonas locais da AWS e zonas do AWS Wavelength. Com o Amazon EKS na nuvem, a segurança, a escalabilidade e a disponibilidade do ambiente de gerenciamento do Kubernetes são totalmente gerenciadas pela AWS na região da AWS. Ao executar aplicações com computação nas regiões da AWS, você obtém toda a variedade de recursos da AWS e do Amazon EKS, incluindo o Modo Automático do Amazon EKS, que automatiza totalmente o gerenciamento da infraestrutura de clusters do Kubernetes para computação, armazenamento e rede na AWS com um único clique. Ao executar aplicações com computação em zonas locais da AWS e zonas do AWS Wavelength, você pode usar os nós autogerenciados do Amazon EKS para conectar instâncias do Amazon EC2 à computação em cluster, e pode usar os outros serviços da AWS disponíveis nas zonas locais da AWS e zonas do AWS Wavelength. Para obter mais informações, consulte Recursos de zonas locais da AWS e Recursos de zonas do AWS Wavelength.

	Amazon EKS nas regiões da AWS	Amazon EKS em zonas locais e do Wavelength
Gestão do ambiente de gerenciamento do Kubernetes	Gerenciado pela AWS	Gerenciado pela AWS
Localização do ambiente de gerenciamento do Kubernetes	Regiões da AWS	Regiões da AWS
Plano de dados do Kubernetes	• Modo Automático do Amazon EKS • Grupos de nós gerenciados pelo Amazon EKS • Nós autogerenciados do Amazon EC2 • AWS Fargate	• Grupos de nós gerenciados pelo Amazon EKS (somente em zonas locais) • Nós autogerenciados do Amazon EC2
Local do plano de dados do Kubernetes	Regiões da AWS	Zonas do Wavelength ou locais da AWS

Amazon EKS no data center ou em ambientes de borda

Caso precise executar aplicações nos próprios data centers ou em ambientes de borda, você pode usar o Amazon EKS no AWS Outposts ou o Amazon EKS Hybrid Nodes. Você pode usar nós autogerenciados com instâncias do Amazon EC2 no AWS Outposts para a computação em cluster, ou você pode usar o Amazon EKS Hybrid Nodes com sua própria infraestrutura on-premises ou de borda para a computação em cluster. O AWS Outposts é uma infraestrutura gerenciada pela AWS que você executa nos data centers ou nas instalações de colocalização, enquanto o Amazon EKS Hybrid Nodes é executado nas máquinas físicas ou virtuais que você gerencia nos ambientes on-premises ou de borda. O Amazon EKS no AWS Outposts e o Amazon EKS Hybrid Nodes exigem uma conexão confiável dos ambientes on-premises com uma região da AWS, e você pode usar os mesmos clusters, recursos e ferramentas do Amazon EKS que você usa para executar aplicações na nuvem. Ao executar no AWS Outposts, você também pode implantar todo o cluster do Kubernetes no AWS Outposts com clusters locais do Amazon EKS no AWS Outposts.

	Amazon EKS Hybrid Nodes	Amazon EKS no AWS Outposts
Gestão do ambiente de gerenciamento do Kubernetes	Gerenciado pela AWS	Gerenciado pela AWS
Localização do ambiente de gerenciamento do Kubernetes	Regiões da AWS	Regiões da AWS ou AWS Outposts
Plano de dados do Kubernetes	Máquinas físicas ou virtuais gerenciadas pelo cliente	Nós autogerenciados do Amazon EC2
Local do plano de dados do Kubernetes	Data center do cliente ou ambiente de borda	Data center do cliente ou ambiente de borda

Amazon EKS Anywhere para ambientes isolados

O Amazon EKS Anywhere simplifica o gerenciamento de clusters do Kubernetes por meio da automação de trabalhos pesados indiferenciados, como configuração da infraestrutura e operações do ciclo de vida de clusters do Kubernetes em ambientes on-premises e de borda. Diferentemente do Amazon EKS, o Amazon EKS Anywhere é um produto gerenciado pelo cliente, e os clientes são responsáveis pelas operações do ciclo de vida do cluster e pela manutenção dos clusters do Amazon EKS Anywhere. O Amazon EKS Anywhere é baseado no subprojeto Cluster API (CAPI) do Kubernetes e é compatível com uma variedade de infraestruturas, incluindo VMware vSphere, bare metal, Nutanix, Apache CloudStack e AWS Snow. O Amazon EKS Anywhere pode ser executado em ambientes isolados e oferece integrações opcionais com serviços regionais da AWS para observabilidade e gerenciamento de identidade. Para receber suporte para o Amazon EKS Anywhere e acesso aos complementos do Kubernetes fornecidos pela AWS, você pode adquirir Assinaturas Enterprise do Amazon EKS Anywhere.

	Amazon EKS Anywhere
Gestão do ambiente de gerenciamento do Kubernetes	Gerenciadas pelo cliente
Localização do ambiente de gerenciamento do Kubernetes	Data center do cliente ou ambiente de borda

	Amazon EKS Anywhere
Plano de dados do Kubernetes	Máquinas físicas ou virtuais gerenciadas pelo cliente
Local do plano de dados do Kubernetes	Data center do cliente ou ambiente de borda

Ferramentas do Amazon EKS

Você pode usar o Amazon EKS Connector para registrar e conectar qualquer cluster do Kubernetes compatível à AWS e visualizá-lo no console do Amazon EKS. Quando um cluster é conectado, você pode ver o status, a configuração e as workloads desse cluster no console do Amazon EKS. Você pode usar esse recurso para visualizar clusters conectados no console do Amazon EKS, mas o Amazon EKS Connector não permite operações de gerenciamento ou mutação para os clusters conectados por meio do console do Amazon EKS.

O Amazon EKS Distro é a distribuição da AWS dos componentes subjacentes do Kubernetes que impulsionam todas as ofertas do Amazon EKS. Ele inclui os componentes principais necessários para um cluster funcional do Kubernetes, como componentes do ambiente de gerenciamento do Kubernetes (etcd, kube-apiserver, kube-scheduler, kube-controller-manager) e componentes de rede (CoreDNS, kube-proxy, plug-ins da CNI). O Amazon EKS Distro pode ser usado para autogerenciar clusters do Kubernetes com as ferramentas de sua escolha. As implantações do Amazon EKS Distro não são cobertas pelos planos do AWS Support.

Configurar para usar o Amazon EKS

Para se preparar para o gerenciamento da linha de comando dos clusters do Amazon EKS, é necessário instalar várias ferramentas. Use o seguinte para configurar credenciais, criar e modificar clusters e trabalhar com clusters quando estiverem em execução:

- Configurar a AWS CLI: obtenha a AWS CLI para configurar e gerenciar os serviços necessários para trabalhar com clusters do Amazon EKS. Em particular, você precisa do AWS CLI para configurar as credenciais, mas também precisa dele com outros serviços do AWS.
- Configurar o kubectl e o eksctl: a `eksctl` CLI interage com o AWS para criar, modificar e excluir clusters do Amazon EKS. Depois que um cluster estiver ativo, use o comando `kubectl` de código aberto para gerenciar objetos do Kubernetes nos clusters do Amazon EKS.
- Configurar um ambiente de desenvolvimento (opcional): considere adicionar as seguintes ferramentas:
  - Ferramenta de implantação local: se você for novo no Kubernetes, considere a possibilidade de instalar uma ferramenta de implantação local, como o minikube ou o kind. Essas ferramentas permitem que você tenha um cluster do Amazon EKS em sua máquina local para testar aplicações.
  - Gerenciador de pacotes: o Helm é um gerenciador de pacotes popular para Kubernetes que simplifica a instalação e o gerenciamento de pacotes complexos. Com o Helm, fica mais fácil instalar e gerenciar pacotes como o AWS Load Balancer Controller no cluster do Amazon EKS.

Próximas etapas

- Configurar a AWS CLI
- Configuração do kubectl e eksctl
- Início rápido: implantar uma aplicação Web e armazenar dados

Configurar a AWS CLI

A AWS CLI é uma ferramenta de linha de comando para trabalhar com os serviços da AWS, incluindo o Amazon EKS. Ela também é usada para autenticar usuários ou funções do IAM para acessar o cluster do Amazon EKS e outros recursos da AWS da sua máquina local. Para provisionar recursos na AWS com base na linha de comando, você precisará obter um ID de chave de acesso

da AWS e uma chave secreta para usar na linha de comando. Em seguida, você precisa configurar essas credenciais na AWS CLI. Se você ainda não instalou a AWS CLI, consulte Instalar ou atualizar a versão mais recente da AWS CLI, no Guia do usuário da AWS Command Line Interface.

Para criar uma chave de acesso

1. Faça login no Console de gerenciamento da AWS.
2. Para contas de usuário único ou de vários usuários:
   - Conta de usuário único -:: No canto superior direito, escolha seu nome de usuário AWS para abrir o menu de navegação. Para este exemplo, selecione **`webadmin`** .
   - Conta de vários usuários -:: Selecione IAM na lista de serviços. No painel do IAM, selecione Usuários e escolha o nome do usuário.
3. Selecione Security credentials (Credenciais de segurança).
4. Em Chaves de acesso, escolha Criar chave de acesso.
5. Escolha Interface de linha de comandos (CLI) e, em seguida, escolha Próximo.
6. Selecione Create access key (Criar chave de acesso).
7. Selecione Download do arquivo .csv.

Para configurar a CLI do AWS

Depois de instalar a AWS CLI, execute as etapas a seguir para configurá-la. Para obter mais informações, consulte Configurar a AWS CLI, no Guia do usuário da AWS Command Line Interface.

1. Em uma janela de terminal, insira o seguinte comando:

   ```
   aws configure
   ```

   De forma alternativa, é possível configurar um perfil nomeado, como `--profile cluster-admin`. Se você configurar um perfil nomeado na AWS CLI, deverá sempre passar esse sinalizador nos comandos subsequentes.

2. Insira credenciais de usuário da AWS. Por exemplo:

   ```
   Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
   Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   Default region name [None]: region-code
   ```

```
Default output format [None]: json
```

Para obter um token de segurança

Se necessário, execute o seguinte comando para obter um novo token de segurança para a AWS CLI. Para obter mais informações, consulte get-session-token em AWS CLI Command Reference.

Por padrão, o token é válido por 15 minutos. Para alterar o tempo limite padrão da sessão, passe o sinalizador `--duration-seconds`. Por exemplo:

```
aws sts get-session-token --duration-seconds 3600
```

Esse comando retorna as credenciais de segurança temporárias para uma sessão CLI AWS. Você verá a seguinte saída de resposta:

```
{
    "Credentials": {
        "AccessKeyId": "ASIA5FTRU3LOEXAMPLE",
        "SecretAccessKey": "JnKgvwfqUD9mNsPoi9IbxAYEXAMPLE",
        "SessionToken": "VERYLONGSESSIONTOKENSTRING",
        "Expiration": "2023-02-17T03:14:24+00:00"
    }
}
```

Para verificar a identidade do usuário

Se necessário, execute o seguinte comando para verificar as credenciais AWS da sua identidade de usuário IAM (como *`ClusterAdmin`*) para a sessão de terminal.

```
aws sts get-caller-identity
```

Esse comando retorna o Amazon Resource Name (ARN) da entidade do IAM que está configurada para a AWS CLI. Você verá o seguinte exemplo de saída de resposta:

```
{
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "01234567890",
    "Arn": "arn:aws:iam::01234567890:user/ClusterAdmin"
```

```
}
```

Próximas etapas

- Configuração do kubectl e eksctl
- Início rápido: implantar uma aplicação Web e armazenar dados

Configurar o **kubectl** e o **eksctl**

Após a AWS CLI ser instalada, há duas outras ferramentas que você deve instalar para criar e gerenciar seus clusters do Kubernetes:

- `kubectl`: a ferramenta de linha de comando `kubectl` é a principal ferramenta que você usará para gerenciar recursos em seu cluster do Kubernetes. Esta página descreve como baixar e configurar o binário `kubectl` que corresponde à versão do seu cluster do Kubernetes. Consulte Instalar ou atualizar o kubectl.
- `eksctl`: a ferramenta de linha de comando `eksctl` foi criada para criar clusters do EKS na Nuvem AWS ou on-premises (com o EKS Anywhere), bem como para modificar e excluir esses clusters. Consulte Instalar eksctl.

Instalar ou atualizar o **kubectl**

Este tópico ajuda a baixar e instalar ou atualizar o binário `kubectl` no dispositivo. O binário é idêntico às versões da comunidade upstream. O binário não é exclusivo do Amazon EKS ou da AWS. Use as etapas abaixo para obter a versão específica de `kubectl` necessária, embora muitos construtores simplesmente executem `brew install kubectl` para instalá-la.

> **Note**
>
> Você deve usar uma versão do `kubectl` que esteja em uma versão secundária de diferença do ambiente de gerenciamento do cluster do Amazon EKS. Por exemplo, um cliente `1.32` `kubectl` funciona com clusters do Kubernetes `1.31`, `1.32` e `1.33`.

Etapa 1: verifique se o **kubectl** está instalado

Determine se você já tem o `kubectl` instalado no dispositivo.

```
kubectl version --client
```

Se você tiver `kubectl` instalado no caminho do dispositivo, o resultado de exemplo inclui informações semelhantes às seguintes. Se quiser atualizar a versão instalada atualmente para uma versão posterior, conclua a próxima etapa, certificando-se de instalar a nova versão no mesmo local em que a versão atual está.

```
Client Version: v1.31.X-eks-1234567
```

Se não receber nenhum resultado, você não tem o `kubectl` instalado ou ele não está instalado em um local que esteja no caminho do seu dispositivo.

Etapa 2: instalar ou atualizar **`kubectl`**

Instale ou atualize o `kubectl` em um dos seguintes sistemas operacionais:

- the section called “macOS”
- the section called “Linux (amd64)”
- the section called “Linux (arm64)”
- the section called “Windows”

> **Note**
>
> Se os downloads estiverem lentos para a sua região da AWS das regiões da AWS usadas nesta seção, considere configurar o CloudFront para trazer o conteúdo para frente. Para obter mais informações, consulte Começar a usar uma distribuição básica do CloudFront.

macOS

Siga as etapas abaixo para instalar o `kubectl` no macOS. As etapas incluem:

- Escolher e baixar o binário para a versão do Kubernetes desejada.
- Opcionalmente, verificar a soma de verificação do binário.
- Adicionar execução às permissões do binário.
- Copiar o binário em uma pasta em sua variável PATH.

- Opcionalmente, adicionar o diretório do binário a sua variável PATH.

Procedimento:

1. Baixe o binário para a versão do Kubernetes do cluster do Amazon S3.
   - Kubernetes `1.34`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.29`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.28`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2025-01-10/bin/
darwin/amd64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
darwin/amd64/kubectl
```

2. (Opcional) Verifique o binário baixado com a soma de verificação do `SHA-256` para o seu binário.
   a. Baixe a soma de verificação do `SHA-256` para a versão do Kubernetes do cluster.
      - Kubernetes `1.34`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/
darwin/amd64/kubectl.sha256
```

      - Kubernetes `1.33`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/
darwin/amd64/kubectl.sha256
```

      - Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/
darwin/amd64/kubectl.sha256
```

      - Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/
darwin/amd64/kubectl.sha256
```

      - Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/
darwin/amd64/kubectl.sha256
```

      - Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2025-01-10/bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/darwin/amd64/kubectl.sha256
```

b. Verifique a soma de verificação `SHA-256` para o binário baixado.

```
openssl sha1 -sha256 kubectl
```

c. Verifique se a soma de verificação gerada no resultado corresponde à soma de verificação no arquivo `kubectl.sha256` baixado.

3. Aplique permissões de execução ao binário.

```
chmod +x ./kubectl
```

4. Copie o binário em uma pasta em seu `PATH`. Se você já tiver instalado uma versão do `kubectl`, recomendamos criar um `$HOME/bin/kubectl` e garantir que `$HOME/bin` venha primeiro em seu `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Opcional) Adicione o caminho `$HOME/bin` ao arquivo de inicialização do shell para que ele esteja configurado quando você abrir um shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bash_profile
```

Linux (amd64)

Siga as etapas abaixo para instalar o `kubectl` no Linux (amd64). As etapas incluem:

- Escolher e baixar o binário para a versão do Kubernetes desejada.
- Opcionalmente, verificar a soma de verificação do binário.
- Adicionar execução às permissões do binário.
- Copiar o binário em uma pasta em sua variável PATH.
- Opcionalmente, adicionar o diretório do binário a sua variável PATH.

Procedimento:

1. Baixe o binário `kubectl` para a versão do Kubernetes do cluster do Amazon S3.
   - Kubernetes `1.34`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.29`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/linux/amd64/kubectl
     ```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/linux/amd64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/amd64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/amd64/kubectl
```

2. (Opcional) Verifique o binário baixado com a soma de verificação do `SHA-256` para o seu binário.

   a. Baixe a soma de verificação do `SHA-256` para a versão do Kubernetes do cluster do Amazon S3 usando o comando para a plataforma de hardware do seu dispositivo.

   - Kubernetes `1.34`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/linux/amd64/kubectl.sha256
```

   - Kubernetes `1.33`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/linux/amd64/kubectl.sha256
```

   - Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/linux/amd64/kubectl.sha256
```

   - Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/linux/amd64/kubectl.sha256
```

   - Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
linux/amd64/kubectl.sha256
```

b. Verifique a soma de verificação `SHA-256` do binário baixado com um dos comandos a seguir.

```
sha256sum -c kubectl.sha256
```

ou

```
openssl sha1 -sha256 kubectl
```

c. No primeiro, você deve ver `kubectl: OK`; no segundo, confirme se a soma de verificação gerada no resultado corresponde à soma de verificação no arquivo `kubectl.sha256` baixado.

3. Aplique permissões de execução ao binário.

```
chmod +x ./kubectl
```

4. Copie o binário em uma pasta em seu `PATH`. Se você já tiver instalado uma versão do `kubectl`, recomendamos criar um `$HOME/bin/kubectl` e garantir que `$HOME/bin` venha primeiro em seu `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Opcional) Adicione o caminho `$HOME/bin` ao arquivo de inicialização do shell para que ele esteja configurado quando você abrir um shell.

> **Note**
>
> Essa etapa pressupõe que você esteja usando o shell Bash. Se você estiver usando outro shell, altere o comando para usar o arquivo de inicialização do shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Linux (arm64)

Siga as etapas abaixo para instalar o `kubectl` no Linux (arm64). As etapas incluem:

- Escolher e baixar o binário para a versão do Kubernetes desejada.
- Opcionalmente, verificar a soma de verificação do binário.
- Adicionar execução às permissões do binário.
- Copiar o binário em uma pasta em sua variável PATH.
- Opcionalmente, adicionar o diretório do binário a sua variável PATH.

Procedimento:

1. Baixe o binário `kubectl` para a versão do Kubernetes do cluster do Amazon S3.
   - Kubernetes `1.34`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/linux/arm64/kubectl
     ```

- Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/linux/arm64/kubectl
```

- Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/linux/arm64/kubectl
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/linux/arm64/kubectl
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/linux/arm64/kubectl
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/linux/arm64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/arm64/kubectl
```

2. (Opcional) Verifique o binário baixado com a soma de verificação do `SHA-256` para o seu binário.
   a. Baixe a soma de verificação do `SHA-256` para a versão do Kubernetes do cluster do Amazon S3 usando o comando para a plataforma de hardware do seu dispositivo.
      - Kubernetes `1.34`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.33`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/
linux/arm64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
linux/arm64/kubectl.sha256
```

b. Verifique a soma de verificação SHA-256 do binário baixado com um dos comandos a seguir.

```
sha256sum -c kubectl.sha256
```

ou

```
openssl sha1 -sha256 kubectl
```

c. No primeiro, você deve ver `kubectl: OK`; no segundo, confirme se a soma de verificação gerada no resultado corresponde à soma de verificação no arquivo `kubectl.sha256` baixado.

3. Aplique permissões de execução ao binário.

```
chmod +x ./kubectl
```

4. Copie o binário em uma pasta em seu PATH. Se você já tiver instalado uma versão do `kubectl`, recomendamos criar um `$HOME/bin/kubectl` e garantir que `$HOME/bin` venha primeiro em seu $PATH.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Opcional) Adicione o caminho `$HOME/bin` ao arquivo de inicialização do shell para que ele esteja configurado quando você abrir um shell.

> **Note**
>
> Essa etapa pressupõe que você esteja usando o shell Bash. Se você estiver usando outro shell, altere o comando para usar o arquivo de inicialização do shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Windows

Siga as etapas abaixo para instalar o `kubectl` no Windows. As etapas incluem:

- Escolher e baixar o binário para a versão do Kubernetes desejada.
- Opcionalmente, verificar a soma de verificação do binário.
- Copiar o binário em uma pasta em sua variável PATH.
- Opcionalmente, adicionar o diretório do binário a sua variável PATH.

Procedimento:

1. Abra um terminal do PowerShell.
2. Baixe o binário `kubectl` para a versão do Kubernetes do cluster do Amazon S3.
   - Kubernetes `1.34`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.33`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.32`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.31`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.30`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.29`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

- Kubernetes `1.28`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/windows/amd64/kubectl.exe
```

- Kubernetes `1.27`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/windows/amd64/kubectl.exe
```

- Kubernetes `1.26`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/windows/amd64/kubectl.exe
```

3. (Opcional) Verifique o binário baixado com a soma de verificação do `SHA-256` para o seu binário.
   a. Baixe a soma de verificação do `SHA-256` para a versão do Kubernetes do cluster para Windows.
      - Kubernetes `1.34`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

      - Kubernetes `1.33`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

      - Kubernetes `1.32`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

      - Kubernetes `1.31`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

      - Kubernetes `1.30`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/
bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.29`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/
bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.28`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/
bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.27`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/
bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.26`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/
bin/windows/amd64/kubectl.exe.sha256
```

b. Verifique a soma de verificação `SHA-256` para o binário baixado.

```
Get-FileHash kubectl.exe
```

c. Verifique se a soma de verificação gerada no resultado corresponde à soma de verificação no arquivo `kubectl.sha256` baixado. A saída do PowerShell deve ser uma sequência de caracteres equivalente em maiúsculas.

4. Copie o binário em uma pasta em seu `PATH`. Se você tiver um diretório existente no `PATH` que é usado para utilitários de linha de comando, copie o binário para esse diretório. Caso contrário, execute as etapas a seguir.

   a. Crie um novo diretório para os binários de linha de comando, como `C:\bin`.

   b. Copie o binário `kubectl.exe` no seu novo diretório.

   c. Edite a variável de ambiente `PATH` do sistema ou usuário para adicionar o novo diretório ao `PATH`.

   d. Feche o terminal do PowerShell e abra um novo para escolher a nova variável `PATH`.

5. Depois de instalar `kubectl`, você pode verificar a versão.

```
kubectl version --client
```

6. Quando o `kubectl` é instalado pela primeira vez, ele ainda não está configurado para se comunicar com nenhum servidor. Abordaremos essa configuração conforme necessário em outros procedimentos. Se você precisar atualizar a configuração para se comunicar com um determinado cluster específico, poderá executar o comando a seguir. Substitua *`region-code`* pela região da AWS em que o cluster se encontra. Substitua *`my-cluster`* pelo nome do cluster.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

7. Considere configurar o preenchimento automático, que permite usar a tecla Tab para preencher os subcomandos do `kubectl` após digitar as primeiras letras. Consulte Kubectl autocomplete na documentação do Kubernetes para obter detalhes.

Instalar o **`eksctl`**

A CLI do `eksctl` é usada para trabalhar com clusters do EKS. Ela automatiza muitas tarefas individuais. Consulte Instalação na documentação do `eksctl` para obter instruções sobre instalação do `eksctl`. Para Linux, use as instruções aplicáveis ao UNIX.

Ao usar `eksctl`, a entidade principal de segurança do IAM que você está usando deve ter permissões para trabalhar com perfis do IAM do Amazon EKS, perfis vinculadas a serviços, AWS CloudFormation, um VPC e recursos relacionados. Para obter mais informações, consulte Ações e uso de perfis vinculadas a serviços no Guia do Usuário do IAM. Você deve concluir todas as etapas deste manual como o mesmo usuário. Execute o seguinte comando para verificar o usuário atual:

```
aws sts get-caller-identity
```

Próximas etapas

- Início rápido: implantar uma aplicação Web e armazenar dados

Início rápido: implantar uma aplicação Web e armazenar dados

Este tutorial de início rápido orienta como implantar a aplicação de amostra do jogo 2048 e manter seus dados em um cluster do modo automático do Amazon EKS, usando o eksctl.

Com o modo automático do Amazon EKS, o gerenciamento de clusters se torna mais simples, pois tarefas rotineiras como armazenamento em bloco, rede, balanceamento de carga e ajuste de escala automático de computação são automatizadas. Durante a configuração, ele gerencia a criação de nós com instâncias gerenciadas do EC2, balanceadores de carga de aplicações e volumes do EBS.

Em resumo, você implantará uma workload de amostra, incluindo as anotações personalizadas exigidas para que a integração com os serviços da AWS ocorra de forma perfeita.

Neste tutorial

Usando o modelo de cluster do `eksctl` a seguir, você criará um cluster com o Modo Automático do EKS para provisionamento automatizado de nós.

- Configuração da VPC: ao usar o modelo de cluster do eksctl apresentado a seguir, o eksctl cria automaticamente uma nuvem privada virtual (VPC) IPv4 para o cluster. Por padrão, o eksctl configura uma VPC que atende a todos os requisitos de rede, além de criar endpoints públicos e privados.
- Gerenciamento de instâncias O modo automático do EKS adiciona ou remove nós dinamicamente no cluster do EKS com base nas demandas das aplicações do Kubernetes.
- Persistência de dados use a funcionalidade de armazenamento em blocos do modo automático do EKS para garantir a persistência dos dados da aplicação, mesmo em cenários que envolvam reinicializações ou falhas de pods.
- Acesso externo à aplicação use a funcionalidade de balanceamento de carga do modo automático do EKS para provisionar dinamicamente um Application Load Balancer (ALB).

Pré-requisitos

Antes de começar, certifique-se de ter realizado as seguintes tarefas:

- Configuração do seu ambiente para o Amazon EKS
- Instalação da versão mais recente do eksctl

Configurar o cluster

Nesta seção, você criará um cluster usando o Modo Automático do EKS para o provisionamento dinâmico de nós.

Crie um arquivo `cluster-config.yaml` e cole o conteúdo a seguir nele. Substitua `region-code` por uma região válida (por exemplo, `us-east-1`).

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: web-quickstart
  region: <region-code>

autoModeConfig:
  enabled: true
```

Agora, estamos prontos para criar o cluster.

Crie o cluster EKS usando o `cluster-config.yaml`:

```
eksctl create cluster -f cluster-config.yaml
```

> **Important**
>
> Caso não use o eksctl para criar o cluster, você precisará marcar manualmente as sub-redes da VPC.

Criar IngressClass

Crie uma `IngressClass` do Kubernetes para o Modo Automático do EKS. A IngressClass define como o Modo Automático do EKS lida com os recursos de entrada. Esta etapa configura a capacidade de balanceamento de carga do Modo Automático do EKS. Quando você cria recursos

de entrada para as aplicações, o Modo Automático do EKS usa essa IngressClass para provisionar e gerenciar automaticamente balanceadores de carga, integrando as aplicações do Kubernetes aos serviços de balanceamento de carga da AWS.

Salve o seguinte arquivo yaml como `ingressclass.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb
  annotations:
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  controller: eks.amazonaws.com/alb
```

Aplique a IngressClass ao cluster:

```
kubectl apply -f ingressclass.yaml
```

Implantar a aplicação de exemplo do jogo 2048

Nesta seção, mostraremos as etapas para implantar o popular “jogo 2048” como uma aplicação de amostra no cluster. O manifesto fornecido inclui anotações personalizadas para o Application Load Balancer (ALB). Essas anotações se integram ao EKS e instruem o serviço a tratar o tráfego HTTP recebido como “exposto à internet” e direcioná-lo para o serviço apropriado no namespace `game-2048`, usando o tipo de destino “ip”.

> **Note**
>
> A imagem `docker-2048` no exemplo é uma imagem de contêiner `x86_64` e não será executada em outras arquiteturas.

1. Crie um namespace do Kubernetes denominado `game-2048` com o sinalizador `--save-config`.

   ```
   kubectl create namespace game-2048 --save-config
   ```

   Você verá a seguinte saída de resposta:

```
namespace/game-2048 created
```

2. Implante a aplicação de amostra do jogo 2048.

```
kubectl apply -n game-2048 -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.8.0/docs/examples/2048/2048_full.yaml
```

   Esse manifesto configura uma implantação, um serviço e uma entrada do Kubernetes para o namespace `game-2048`, criando os recursos necessários para implantar e expor a aplicação `game-2048` no cluster. Isso inclui a criação de um serviço denominado `service-2048` que expõe a implantação na porta `80` e um recurso de entrada denominado `ingress-2048` que define regras de roteamento para tráfego HTTP de entrada e anotações para um Application Load Balancer (ALB) voltado para a internet. Você verá a seguinte saída de resposta:

```
namespace/game-2048 configured
deployment.apps/deployment-2048 created
service/service-2048 created
ingress.networking.k8s.io/ingress-2048 created
```

3. Execute o comando a seguir para obter o recurso de entrada para o namespace `game-2048`.

```
kubectl get ingress -n game-2048
```

   Você verá a seguinte saída de resposta:

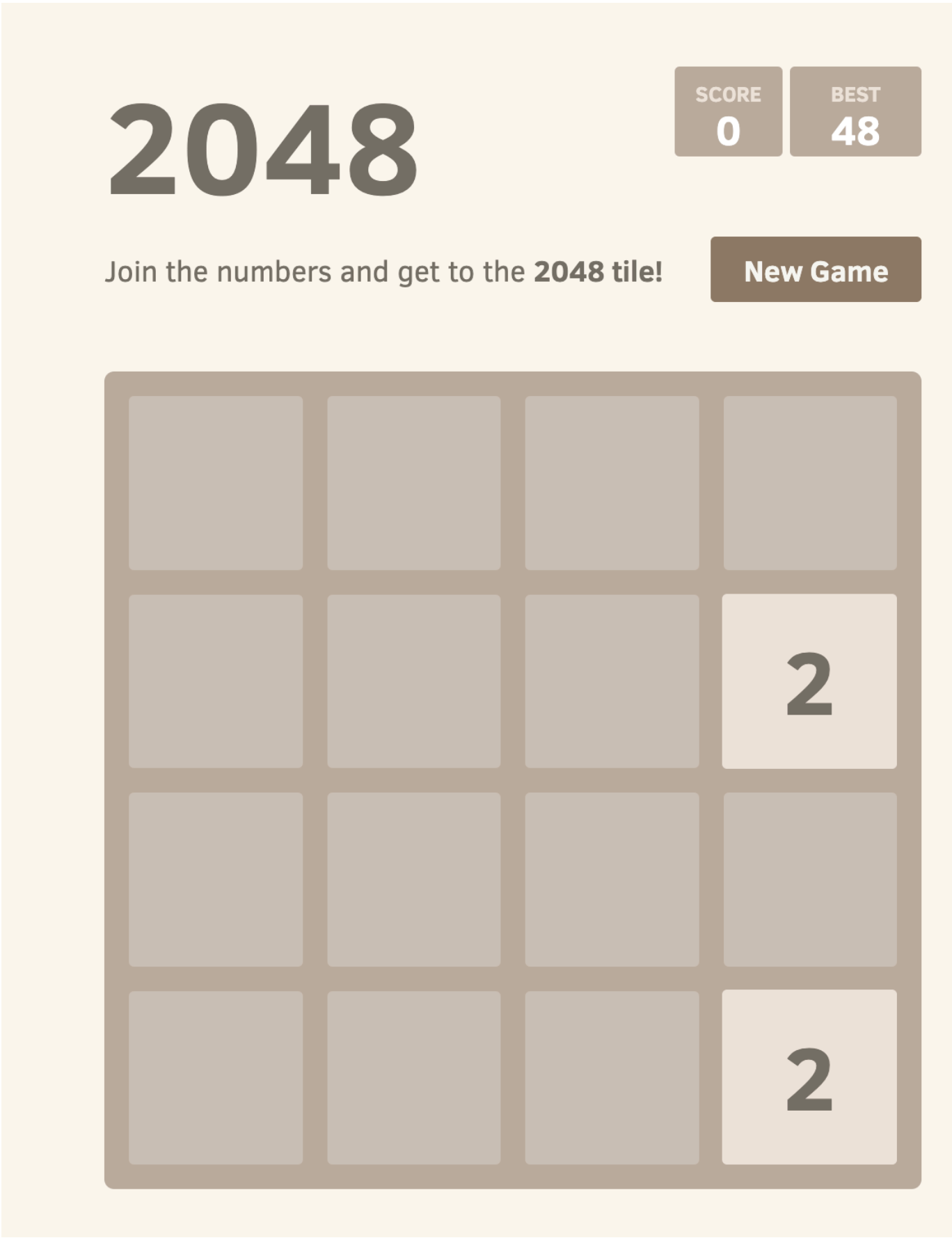
```
NAME           CLASS   HOSTS   ADDRESS
                      PORTS   AGE
ingress-2048   alb     *       k8s-game2048-ingress2-eb379a0f83-378466616.region-
code.elb.amazonaws.com   80      31s
```

   Você precisará esperar alguns minutos para que o Application Load Balancer (ALB) seja provisionado antes de começar as etapas a seguir.

4. Abra o navegador da Web e digite o `ADDRESS` da etapa anterior para acessar a aplicação Web. Por exemplo:

```
k8s-game2048-ingress2-eb379a0f83-378466616.region-code.elb.amazonaws.com
```

Você deve ver o jogo 2048 no seu navegador. Reproduza.


2048
SCORE
0
BEST
48
Join the numbers and get to the 2048 tile!
New Game
2
2

Persistir dados usando o Modo Automático do Amazon EKS

Agora que o jogo 2048 está instalado e em execução no cluster do Amazon EKS, é hora de garantir que os dados do jogo sejam persistidos com segurança usando o recurso de armazenamento em blocos do Modo Automático do Amazon EKS.

1. Crie um arquivo chamado `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. Aplique a `StorageClass`:

```
kubectl apply -f storage-class.yaml
```

3. Crie uma solicitação de volume persistente (PVC) para solicitar armazenamento para os dados do seu jogo. Crie um arquivo denominado `ebs-pvc.yaml` e adicione o seguinte conteúdo a ele:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: game-data-pvc
  namespace: game-2048
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: auto-ebs-sc
```

4. Aplique a PVC ao cluster:

```
kubectl apply -f ebs-pvc.yaml
```

Você verá a seguinte saída de resposta:

```
persistentvolumeclaim/game-data-pvc created
```

5. Você agora precisa atualizar a implantação do jogo 2048 para usar essa PVC para armazenar dados. A implantação a seguir está configurada para usar a PVC para armazenar dados do jogo. Crie um arquivo denominado `ebs-deployment.yaml` e adicione o seguinte conteúdo a ele:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  replicas: 3  # Adjust the number of replicas as needed
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - name: app-2048
          image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 80
          volumeMounts:
            - name: game-data
              mountPath: /var/lib/2048
      volumes:
        - name: game-data
          persistentVolumeClaim:
            claimName: game-data-pvc
```

6. Aplique a implantação atualizada:

```
kubectl apply -f ebs-deployment.yaml
```

Você verá a seguinte saída de resposta:

```
deployment.apps/deployment-2048 configured
```

Com essas etapas, o jogo 2048 no cluster agora está configurado para persistir os dados usando o recurso de armazenamento em blocos do Modo Automático do Amazon EKS. Isso garante que o progresso e os dados do jogo estejam seguros mesmo no caso de falhas nos pods ou nós.

Caso tenha gostado deste tutorial, deixe seu feedback para que possamos fornecer mais tutoriais de início rápido específicos para casos de uso, como este.

Limpeza

Para evitar cobranças futuras, você precisará excluir manualmente a pilha associada do CloudFormation para excluir todos os recursos criados durante este guia, incluindo a rede VPC.

Exclua a pilha do CloudFormation:

```
eksctl delete cluster -f ./cluster-config.yaml
```

Conhecer o Amazon EKS por meio de exemplos

Visão geral

Este Guia do usuário do Amazon EKS contém procedimentos de uso geral para criar seu primeiro cluster do EKS a partir da linha de comando ou do Console de gerenciamento da AWS e uma referência sólida para todos os principais componentes do Amazon EKS. No entanto, como administrador ou desenvolvedor de clusters do Amazon EKS, você pode adquirir uma compreensão mais profunda do Amazon EKS percorrendo os caminhos de aprendizado que existem em sites que não constam neste guia. Esses sites podem ajudar a:

- Configurar tipos específicos de clusters. Tipos específicos de clusters podem ser baseados em seus tipos de workloads ou requisitos de segurança. Por exemplo, você talvez queira ajustar um cluster para executar workloads em lote, de machine learning ou de computação intensiva.
- Melhorar seus clusters. Você pode adicionar recursos avançados ao seu cluster para fornecer elementos como observabilidade, armazenamento flexível, ajuste de escala ou rede de cluster especializada.
- Automatizar atualizações. Usando recursos como o GitOps, você pode configurar o provisionamento automático da infraestrutura de cluster e das workloads com base nas alterações que ocorrem nesses componentes nos seus repositórios Git.
- Usar ferramentas avançadas de configuração de clusters. Embora o `eksctl` ofereça uma maneira rápida de criar um cluster, existem outras ferramentas que podem facilitar a configuração e a atualização de clusters mais complexos. Isso inclui ferramentas como o Terraform e CloudFormation.

Para começar sua trajetória de aprendizado do Amazon EKS, recomendo que visite alguns dos sites descritos nesta página. Se você tiver problemas ao longo do caminho, também há recursos para ajudar a superá-los. Por exemplo, o Centro de Conhecimento re:Post permite pesquisar problemas de suporte relacionados ao Amazon EKS no banco de dados de suporte. Além disso, o Guia de práticas recomendadas do Amazon EKS oferece dicas sobre as melhores maneiras de configurar seus clusters do nível de produção.

Workshop do Amazon EKS

Começando com uma compreensão básica do Kubernetes e de contêineres, o Workshop do Amazon EKS é uma plataforma de aprendizado para orientar um administrador de clusters com relação a recursos importantes do Amazon EKS. Aqui estão algumas maneiras pelas quais você pode participar do workshop do Amazon EKS:

- Princípios básicos do Amazon EKS: assista ao vídeo na página de introdução para saber como o Amazon EKS implementa os recursos do Kubernetes na Nuvem AWS. Se você precisar de uma compreensão ainda mais básica do Kubernetes, assista ao vídeo What is Kubernetes.
- Configuração do Amazon EKS: se você tiver uma conta da AWS, a seção Configuração ajudará você a configurar um ambiente do CloudShell para criar um cluster. Ele oferece a opção de eksctl (uma linha de comando simples para criação de clusters) e o Terraform (uma abordagem mais baseada em infraestrutura como código para criar um cluster) para criar seu cluster do Amazon EKS.
- Conceitos básicos do Amazon EKS: experimente uma loja Web simples na seção Sample application. Você pode usá-la em todos os outros exercícios. Nesta seção, você também pode saber mais como empacotar imagens de contêiner e como os microsserviços são gerenciados usando pods do Kubernetess, implantações, serviços, StatefulSets e namespaces. Em seguida, use o Kustomize para implantar alterações nos manifestos do Kubernetes.
- Fundamentos do Amazon EKS: usando recursos da AWS, como o AWS Load Balancer Controller, o workshop mostra como expor suas aplicações para o mundo externo. Para armazenamento, o workshop mostra como usar o Amazon EBS para armazenamento em blocos, o Amazon EFS para armazenamento de sistemas de arquivos e o Amazon FSx para NetApp ONTAP para gerenciar sistemas de arquivos ONTAP na AWS. Para gerenciamento de nós, o workshop ajuda você a configurar Grupos de nós gerenciados.
- Recursos avançados do Amazon EKS: recursos mais avançados oferecidos no workshop do Amazon EKS incluem laboratórios para configurar:
  - Ajuste de escala: inclui ajuste de escala de nós (com o Autoscaler de clusters ou o Karpenter) e ajuste de escala de workloads (com o Autoscaler horizontal de pods e o Autoscaler proporcional de clusters).
  - Observabilidade: saiba mais sobre Registro em log, OpenSearch, Container Insights no Amazon EKS e Visibilidade de custos com o Kubecost em um conjunto de laboratórios de observabilidade.

- Segurança: esse conjunto de Laboratórios de segurança permite que você explore o Gerenciamento de segredos, o Amazon GuardDuty, Padrões de segurança de pods e o Gerenciamento de políticas do Kyverno.
- Rede: saiba mais sobre os recursos de rede para o Amazon EKS nos laboratórios de Rede que incluem o Amazon VPC CNI (compatível com plug-ins de rede) e o Amazon VPC Lattice (para configurar clusters em VCs e contas de usuário).
- Automação: os laboratórios de Automação orientam você pelos métodos do GitOps para gerenciar seus clusters e projetos, como Controladores da AWS para Kubernetes e Crossplane para gerenciar ambientes de gerenciamento do Amazon EKS.

Tutoriais práticos de configuração de clusters do Amazon EKS

Um conjunto de tutoriais de configuração de clusters do Amazon EKS no site da comunidade AWS pode ajudar você a criar clusters do Amazon EKS para finalidades especiais e aprimorá-los de várias maneiras. Os tutoriais são divididos em três tipos diferentes:

Criar clusters

Estes tutoriais ajudam você a criar clusters que podem ser usados para finalidades especiais. Essas finalidades especiais incluem a capacidade de executar:

- Aplicações globalmente escaláveis baseadas em IPv6
- Tarefas assíncronas em lote
- Microsserviços de alto tráfego
- Ajuste de escala com o Karpenter no Fargate
- Workloads financeiras
- Grupos de nós gerenciados pelo Windows

Aprimorar clusters

Depois de ter um cluster disponível, você pode estender e aprimorar esse cluster de forma a permitir que ele execute workloads especializadas e também aprimore os clusters. Esses tutoriais incluem maneiras de:

- Fornecer soluções de armazenamento com o EFS CSI
- Fornecer armazenamento dinâmico de banco de dados com o EBS CSI

- Expor aplicações em clusters IPv4 usando o AWS Load Balancer Controller
- Expor aplicações em clusters IPv6 usando o AWS Load Balancer Controller

Otimizar serviços da AWS

Ao usar esses tutoriais, você pode integrar melhor os clusters aos serviços da AWS. Nesses tutoriais você encontra ajuda relacionada aos seguintes tópicos:

- Gerenciamento de registros DNS para microsserviços com ExternalDNS
- Monitoramento de aplicações com o CloudWatch
- Gerenciamento de tarefas assíncronas com armazenamento do SQS e EFS
- Consumo de segredos do AWS Secrets Manager das workloads
- Configuração do mTLS com o Fargate, NGINX e ACM PCA

Amostras do Amazon EKS

O repositório Amazon EKS Samples armazena manifestos para uso com o Amazon EKS. Esses manifestos oferecem a oportunidade de experimentar diferentes tipos de aplicações no Amazon EKS ou criar tipos específicos de clusters do Amazon EKS. As amostras incluem manifestos para:

- Criar um cluster do Fargate do AWS Amazon EKS
- Criar um cluster com um perfil do IAM existente
- Adicionar um grupo de nós gerenciados do Ubuntu a um cluster
- Fazer backup e restaurare o armazenamento do pod com snapshots de volume
- Recuperar volumes do EBS montados como PVCs com várias contas
- Ativar o protocolo proxy para o controlador de entrada NGINX com Classic Load Balancers
- Configurar o registro em log no Fargate para AWS OpenSearch
- Executar o SDK para Python com um provedor de identidade federado da Web
- Implantar uma amostra de aplicação em um controlador NFS CSI
- Usar snapshots de volume para StatefulSets
- Implantar pods em nós em diferentes zonas de disponibilidade

Lembre-se de que essas amostras são apenas para fins de aprendizado e teste e não devem ser usadas na produção.

Tutoriais do AWS

O site AWS Tutoriais publica alguns tutoriais do Amazon EKS, mas também oferece uma ferramenta de pesquisa para encontrar outros tutoriais publicados em sites da AWS (como o site da comunidade AWS). Os tutoriais do Amazon EKS publicados diretamente nesse site incluem:

- Implantar uma aplicação da Web de contêiner no Amazon EKS
- Run Kubernetes clusters for less (Amazon EKS and Spot instances)
- How to cost optimize Jenkins jobs on Kubernetes

Workshop para desenvolvedores

Se você é um desenvolvedor de software e deseja criar ou refatorar aplicações para serem executados no Amazon EKS, o workshop Amazon EKS Developers é um bom ponto de partida. O workshop não só ajuda você a criar aplicações em contêineres, mas também a implantar esses contêineres em um registro de contêineres (ECR) e, depois disso, em um cluster do Amazon EKS.

Comece com o Workshop Amazon EKS Python para analisar o processo de refatoração de uma aplicação Python e, em seguida, configure seu ambiente de desenvolvimento para se preparar para a implantação da aplicação. Percorra as seções sobre contêineres, Kubernetes e o Amazon EKS para se preparar para executar suas aplicações em contêineres nesses ambientes.

Workshop do Terraform

Embora o `eksctl` seja uma ferramenta simples para criar um cluster, para tipos mais complexos de implantação de infraestrutura como código do Amazon EKS, o Terraform é uma ferramenta popular de criação e gerenciamento de clusters do Amazon EKS. O Workshop Terraform Amazon EKS ensina como usar o Terraform para criar uma AWS VPC, criar clusters do Amazon EKS e adicionar aprimoramentos opcionais ao seu cluster. Em especial, há uma seção para criar um cluster privado do Amazon EKS

Treinamento do AWS Amazon EKS

A AWS oferece treinamento formal para saber mais sobre o Amazon EKS. Um curso de treinamento de três dias, Running Containers on Amazon Elastic Kubernetes Service, ensina:

- Princípios básicos do Kubernetes e do Amazon EKS

- Como criar clusters do Amazon EKS
- Proteção do Amazon EKS com autorização RBAC do Kubernets e do AWS IAM
- Ferramentas de automação do GitOps
- Ferramentas de monitoramento
- Técnicas para melhorar o custo, a eficiência e a resiliência

Começar a usar o Amazon EKS

Antes de ler os guias de introdução, verifique se você está configurado para usar o Amazon EKS. Para ter mais informações, consulte Configuração.

Há dois manuais de introdução disponíveis para a criação de um novo cluster Kubernetes com nós no Amazon EKS:

- Introdução ao Amazon EKS – eksctl: este guia de introdução ajuda você a instalar todos os recursos necessários para começar a usar o Amazon EKS usando o `eksctl`, um utilitário de linha de comando simples para criar e gerenciar clusters do Kubernetes no Amazon EKS. No final do tutorial, você terá um cluster do Amazon EKS em execução no qual poderá implantar as aplicações. Essa é a forma mais rápida e simples para começar a usar o Amazon EKS.
- Introdução ao Amazon EKS - Console de gerenciamento da AWS e AWS CLI - Este guia de introdução ajuda você a criar todos os recursos necessários para começar a usar o Amazon EKS usando a CLI Console de gerenciamento da AWS e AWS. No final do tutorial, você terá um cluster do Amazon EKS em execução no qual poderá implantar as aplicações. Neste manual, crie manualmente cada recurso necessário para um cluster do Amazon EKS. Os procedimentos fornecem visibilidade sobre como cada recurso é criado e como eles interagem uns com os outros.

Também oferecemos as seguintes referências:

- Para obter tutoriais práticos, consulte EKS Cluster Setup no AWS Community.
- Para exemplos de código, consulte Exemplos de código para o Amazon EKS usando AWS SDKs.

Conceitos básicos do Amazon EKS | Modo Automático do EKS

Assim como outras experiências iniciais do EKS, criar seu primeiro cluster com o Modo Automático do EKS delega o gerenciamento do próprio cluster à AWS. No entanto, o Modo Automático do EKS amplia a automação do EKS ao assumir a responsabilidade de muitos serviços essenciais necessários para configurar a infraestrutura de workloads (nós, redes e vários serviços), facilitando o gerenciamento de nós e o aumento da escala verticalmente para atender às demandas das workloads.

Escolha uma das seguintes formas de criar um cluster com o Modo Automático do EKS:

- the section called “ AWS CLI”: use a interface de linha de comandos da `aws` para criar um cluster.

- the section called “Console de Gerenciamento”: use o Console de gerenciamento da AWS para criar um cluster.
- the section called “CLI do eksctl”: use a interface de linha de comandos da `eksctl` para criar um cluster.

Se você estiver comparando diferentes abordagens para criar seu primeiro cluster do EKS, saiba que o Modo Automático do EKS faz com que a AWS assuma responsabilidades adicionais de gerenciamento de clusters, que incluem a configuração de componentes para:

- Iniciar e escalar os nós à medida que a demanda da workload aumenta e diminui.
- Atualizar regularmente o próprio cluster (ambiente de gerenciamento), os sistemas operacionais dos nós e os serviços em execução nos nós.
- Escolher as configurações padrão que determinam alguns aspectos, como o tamanho e a velocidade do armazenamento de nós e a configuração da rede de pods.

Para obter detalhes sobre o que você obtém com os clusters do Modo Automático do EKS, consulte Modo Automático do EKS.

Conceitos básicos do Amazon EKS: **`eksctl`**

> **Note**
>
> Este tópico aborda como começar sem o Modo Automático do EKS.
> O Modo Automático do EKS automatiza tarefas de rotina de computação, armazenamento e redes de clusters. Saiba como começar a usar o Modo Automático do Amazon EKS.

Este manual ajuda você a criar todos os recursos necessários para começar a usar o Amazon Elastic Kubernetes Service (Amazon EKS) com o `eksctl`, um utilitário de linha de comando simples para criar e gerenciar clusters de Kubernetes no Amazon EKS. No final deste tutorial, você terá um cluster do Amazon EKS em execução, no qual poderá implantar as aplicações.

Os procedimentos neste guia criam automaticamente vários recursos que você precisa criar manualmente ao criar o cluster usando o comando Console de gerenciamento da AWS. Se preferir criar manualmente a maioria dos recursos e entender melhor como eles interagem entre

si, use o Console de gerenciamento da AWS para criar os clusters e a computação. Para ter mais informações, consulte the section called “Criar cluster (console e CLI)”.

Pré-requisitos

Antes de iniciar este tutorial, você deve instalar e configurar as ferramentas AWS CLI, kubectl e eksctl, conforme descrito em Configurar para usar o Amazon EKS.

Etapa 1: Criar o cluster e nós do Amazon EKS

> **Important**
>
> Para começar da forma mais simples e rápida possível, este tópico inclui etapas para criar um cluster e nós com configurações padrão. Antes de criar um cluster e nós para uso em produção, recomendamos que você se familiarize com todas as configurações e implante um cluster e nós com as configurações que atendam aos seus requisitos. Para ter mais informações, consulte the section called “Criar um cluster” e Gerenciar computação. Algumas configurações poderão ser habilitadas apenas durante a criação do cluster e dos nós.

Você pode criar um cluster com um dos seguintes tipos de nós. Para saber mais sobre cada tipo, consulte Gerenciar computação. Depois que o cluster for implantado, você pode adicionar outros tipos de nós.

- Fargate – Linux: selecione este tipo de nó se você quiser executar aplicações Linux no the section called “ AWS Fargate”. O Fargate é um mecanismo de computação sem servidor que permite a implantação de pods do Kubernetes sem gerenciar instâncias do Amazon EC2.
- Nós gerenciados – Linux – Selecione este tipo de nó se você quiser executar aplicações do Amazon Linux em instâncias do Amazon EC2. Embora não seja abordado neste guia, também é possível usar os nós Autogerenciado do Windows e Bottlerocket para o cluster.

Crie o cluster do Amazon EKS com o comando a seguir. Você pode substituir `my-cluster` por seu próprio valor. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster. Substitua `region-code` por qualquer região AWS com suporte ao Amazon EKS. Para obter uma lista das regiões da AWS, consulte Endpoints e cotas de serviço do Amazon EKS no guia Referência geral da AWS.

Example

Fargate - Linux

```
eksctl create cluster --name my-cluster --region region-code --fargate
```

Managed nodes - Linux

```
eksctl create cluster --name my-cluster --region region-code
```

A criação do cluster leva muitos minutos. Durante a criação, você verá várias linhas de saída. A última linha do resultado é semelhante ao exemplo de linha a seguir.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

`eksctl` criou um arquivo de configuração `kubectl` em `~/.kube/config` ou adicionou a configuração do novo cluster em um arquivo de configuração existente em `~/.kube/config` em seu computador.

Após a conclusão da criação do cluster, visualize a pilha do AWS CloudFormation chamada `eksctl-`*`my-cluster`*`-cluster` no console do AWS CloudFormation para ver todos os recursos que foram criados.

Etapa 2: exibir recursos do Kubernetes

1. Exiba os nós de cluster.

   ```
   kubectl get nodes -o wide
   ```

   Veja um exemplo de saída abaixo.

   Example

   Fargate - Linux

   ```
   NAME                                                STATUS   ROLES    AGE
    VERSION              INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
                   CONTAINER-RUNTIME
   ```

```
fargate-ip-192-0-2-0.region-code.compute.internal   Ready    <none>
 8m3s    v1.2.3-eks-1234567   192.0.2.0     <none>        Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
fargate-ip-192-0-2-1.region-code.compute.internal   Ready    <none>
 7m30s   v1.2.3-eks-1234567   192-0-2-1     <none>        Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Managed nodes - Linux

```
NAME                                          STATUS   ROLES    AGE    VERSION
        INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
   CONTAINER-RUNTIME
ip-192-0-2-0.region-code.compute.internal   Ready    <none>   6m7s
  v1.2.3-eks-1234567   192.0.2.0     192.0.2.2     Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
ip-192-0-2-1.region-code.compute.internal   Ready    <none>   6m4s
  v1.2.3-eks-1234567   192.0.2.1     192.0.2.3     Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Para obter mais informações sobre o que é visualizado na saída, consulte the section called “Acessar recursos do cluster”.

2. Exiba as workloads em execução no cluster.

```
kubectl get pods -A -o wide
```

Veja um exemplo de saída abaixo.

Example

Fargate - Linux

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE   IP
          NODE                                                NOMINATED NODE
 READINESS GATES
kube-system   coredns-1234567890-abcde   1/1     Running   0          18m
 192.0.2.0   fargate-ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          18m
 192.0.2.1   fargate-ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
```

Managed nodes - Linux

```
NAMESPACE     NAME                          READY   STATUS    RESTARTS   AGE     IP
          NODE                                                NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-12345                1/1     Running   0          7m43s
 192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>           <none>
kube-system   aws-node-67890                1/1     Running   0          7m46s
 192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>           <none>
kube-system   coredns-1234567890-abcde      1/1     Running   0          14m
 192.0.2.3   ip-192-0-2-3.region-code.compute.internal   <none>           <none>
kube-system   coredns-1234567890-12345      1/1     Running   0          14m
 192.0.2.4   ip-192-0-2-4.region-code.compute.internal   <none>           <none>
kube-system   kube-proxy-12345              1/1     Running   0          7m46s
 192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>           <none>
kube-system   kube-proxy-67890              1/1     Running   0          7m43s
 192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>           <none>
```

Para obter mais informações sobre o que é visualizado na saída, consulte the section called “Acessar recursos do cluster”.

Etapa 3: excluir clusters e nós

Após concluir o cluster e os nós criados para este tutorial, faça uma limpeza excluindo o cluster e os nós usando o comando a seguir. Se você quiser realizar outras ações com esse cluster antes de limpá-lo, consulte the section called “Próximas etapas”.

```
eksctl delete cluster --name my-cluster --region region-code
```

Próximas etapas

Os seguintes tópicos de documentação ajudam a estender a funcionalidade do seu cluster:

- Implante uma aplicação de exemplo no cluster.
- A entidade principal do IAM que criou o cluster é a única entidade que pode fazer chamadas para o servidor de API do Kubernetes com o `kubectl` ou o Console de gerenciamento da AWS. Se quiser que outras entidades principais do IAM tenham acesso ao cluster, será necessário adicioná-los. Para ter mais informações, consulte the section called “Acesso à API do Kubernetes” e the section called “Permissões obrigatórias”.

- Antes de implantar um cluster para uso em produção, recomendamos conhecer todas as configurações de clusters e nós. Algumas configurações (como habilitar o acesso SSH aos nós do Amazon EC2) devem ser feitas no momento de criação do cluster.
- Para aumentar a segurança do cluster, configure o plugin Amazon VPC Container Networking Interface para usar funções do IAM para contas de serviço.

Comece a usar o Amazon EKS - Console de gerenciamento da AWS e AWS CLI

> **Note**
>
> Este tópico aborda como começar sem o Modo Automático do EKS. Ele usa grupos de nós gerenciados para realizar a implantação dos nós.
> O Modo Automático do EKS automatiza tarefas de rotina de computação, armazenamento e redes de clusters. Saiba como começar a usar o Modo Automático do Amazon EKS. O modo automático do EKS é o método preferido para implantar nós.

Este guia ajuda você a criar todos os recursos necessários para começar a usar o Amazon Elastic Kubernetes Service (Amazon EKS) usando o Console de gerenciamento da AWS e a AWS CLI. Neste manual, você criará manualmente cada recurso. No final deste tutorial, você terá um cluster do Amazon EKS em execução, no qual poderá implantar as aplicações.

Os procedimentos deste guia fornecem visibilidade completa sobre como cada recurso é criado e como os recursos interagem entre si. Se você preferir que a maioria dos recursos seja criada automaticamente, use a CLI `eksctl` para criar os nós e o cluster. Para obter mais informações, consulte the section called “Criar cluster (eksctl)”.

Pré-requisitos

Antes de iniciar este tutorial, você deve instalar e configurar as ferramentas a seguir e os recursos necessários para criar e gerenciar um cluster do Amazon EKS.

- AWS CLI - Uma ferramenta de linha de comando para trabalhar com os serviços AWS, incluindo o Amazon EKS. Para obter mais informações, consulte Instalação no Guia do Usuário da Interface de Linha de Comando AWS. Depois de instalar a AWS CLI, recomendamos que você também a configure. Para saber mais, consulte Configuração rápida com aws configure, no Guia do usuário

da AWS Command Line Interface. Observe que o AWS CLI v2 é necessário para usar a opção update-kubeconfig mostrada nesta página.

- **`kubectl`** : uma ferramenta de linha de comando para trabalhar com clusters do Kubernetes. Para obter mais informações, consulte the section called “Configuração do kubectl e eksctl”.
- Permissões do IAM necessárias: a entidade principal de segurança do IAM que você está usando deve ter permissões para trabalhar com perfis do IAM do Amazon EKS, perfis vinculadas a serviços, AWS CloudFormation, uma VPC e recursos relacionados. Para obter mais informações, consulte Ações e uso de perfis vinculadas a serviços no Guia do Usuário do IAM. Você deve concluir todas as etapas deste manual como o mesmo usuário. Execute o seguinte comando para verificar o usuário atual:

```
aws sts get-caller-identity
```

Convém concluir as etapas neste tópico em um shell Bash. Se não estiver utilizando um shell Bash, alguns comandos de script, como caracteres de continuação de linha e a forma como as variáveis são definidas e utilizadas, exigirão o ajuste do seu shell. Além disso, as regras de citação e de escape do seu shell podem ser diferentes. Para obter mais informações, consulte Uso de aspas com cadeias de caracteres na AWS CLI, no Guia do usuário da AWS Command Line Interface.

Etapa 1: Criar o cluster do Amazon EKS

> **Important**
>
> Para começar da forma mais simples e rápida possível, este tópico inclui etapas para criar um cluster com configurações padrão. Antes de criar um cluster para uso em produção, recomendamos que você se familiarize com todas as configurações e implante um cluster com as configurações que atendam aos seus requisitos. Para obter mais informações, consulte the section called “Criar um cluster”. Algumas configurações poderão ser habilitadas apenas durante a criação do cluster.

1. Crie uma Amazon VPC com sub-redes públicas e privadas que atenda aos requisitos do Amazon EKS. Substitua *`region-code`* por qualquer região AWS com suporte ao Amazon EKS. Para obter uma lista das regiões da AWS, consulte Endpoints e cotas de serviço do Amazon EKS no guia Referência geral da AWS. É possível substituir *`my-eks-vpc-stack`* por qualquer nome de sua preferência.

```
aws cloudformation create-stack \
  --region region-code \
  --stack-name my-eks-vpc-stack \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

> **Tip**
>
> Para obter uma lista de todos os recursos criados pelo comando anterior, acesse o console do AWS CloudFormation em https://console.aws.amazon.com/cloudformation/. Escolha a pilha *my-eks-vpc-stack* e depois a guia Resources (Recursos).

2. Crie um perfil do IAM de cluster e associe a política gerenciada do Amazon EKS IAM a ele. Os clusters do Kubernetes gerenciados pelo Amazon EKS fazem chamadas a outros serviços da AWS em seu nome para gerenciar os recursos que você usa com o serviço.

   a. Copie o conteúdo a seguir em um arquivo chamado *eks-cluster-role-trust-policy.json*.

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "eks.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   b. Crie a função.

   ```
   aws iam create-role \
     --role-name myAmazonEKSClusterRole \
     --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
   ```

   c. Anexe a política de IAM gerenciada pelo Amazon EKS à função.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name myAmazonEKSClusterRole
```

3. Abra o console do Amazon EKS em eks/home#/clusters.

   Certifique-se de que a região AWS mostrada no canto superior direito do console seja a região AWS na qual você deseja criar o cluster. Se não for o caso, selecione o menu suspenso ao lado do nome da região AWS e escolha a região AWS que você deseja usar.

4. Selecione Criar cluster. Caso não visualize essa opção, escolha Clusters no painel de navegação à esquerda.

5. Na página Configure cluster (Configurar cluster), faça o seguinte:

   a. Selecione Configuração personalizada e desabilite Usar o modo automático do EKS. (Se você preferir um cluster no modo automático do EKS, consulte the section called “Console de Gerenciamento”, em vez disso.)

   b. Insira um Nome para o seu cluster, como *my-cluster*. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.

   c. Em Cluster Service Role (Função de serviço do cluster), escolha *myAmazonEKSClusterRole*.

   d. Deixe as configurações restantes com seus valores padrão e escolha Next (Próximo).

6. Na página Specify networking (Especificar redes), faça o seguinte:

   a. Escolha o ID da VPC que você criou em uma etapa anterior na lista suspensa VPC. É algo como ** | my-eks-vpc-stack-VPC*.

   b. Escolha as sub-redes criadas em uma etapa anterior na lista suspensa Sub-redes. As sub-redes serão algo como ** | my-eks-vpc-stack-**.

   c. Escolha o grupo de segurança criado em uma etapa anterior na lista suspensa Grupos de segurança adicionais. É algo como ** | my-eks-vpc-stack-ControlPlaneSecurityGroup-**.

   d. Deixe as configurações restantes com seus valores padrão e escolha Next (Próximo).

7. Na página Configurar observabilidade, escolha Próximo.

8. Na página Selecionar complementos, selecione Avançar.

Para obter mais informações sobre complementos, consulte the section called “Complementos do Amazon EKS”.

9. Na página Definir as configurações dos complementos selecionados, escolha Avançar.
10. Na página Review and create (Revisar e criar), escolha Create (Criar).

    À direita do nome do cluster, o status do cluster será Creating (Em criação) por vários minutos, até que o processo de provisionamento do cluster seja concluído. Não passe para a próxima etapa até que o status seja Active (Ativo).

> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte the section called “Capacidade insuficiente”.

Etapa 2: Configurar o computador para se comunicar com seu cluster

Nesta seção, você cria um arquivo `kubeconfig` para o cluster. As configurações neste arquivo ativam a opção CLI do `kubectl` para se comunicar com o cluster.

Antes de continuar, verifique se a criação do cluster foi concluída com êxito na Etapa 1.

1. Crie um arquivo `kubeconfig` para o cluster. Substitua *`region-code`* pela região AWS na qual você criou seu cluster. Substitua *`my-cluster`* pelo nome do cluster.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

   Por padrão, um arquivo `~/.kube` é criado em `config` ou o novo cluster é adicionado a um arquivo `config` existente no `~/.kube`.

2. Teste a configuração.

```
kubectl get svc
```

> **Note**
>
> Se você receber qualquer erro de autorização ou de tipo de recurso, consulte the section called “Acesso negado ou não autorizado (`kubectl`)” no tópico de solução de problemas.

Veja um exemplo de saída abaixo.

```
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

Etapa 3: criar nós

> **Important**
>
> Para começar a usar da maneira mais simples e rápida possível, este tópico inclui etapas para criar nós utilizando, em sua maioria, configurações padrão. Antes de criar nós para uso em produção, recomendamos que você se familiarize com todas as configurações e implante nós com as configurações que atendam aos seus requisitos. Para obter mais informações, consulte Gerenciar computação. Algumas configurações poderão ser habilitadas apenas durante a criação dos nós.

Este procedimento configura seu cluster para usar grupos de nós gerenciados para criar nós, especificando as sub-redes e perfil do IAM do nó que você criou nas etapas anteriores. Ele possibilita a execução de aplicações do Amazon Linux em instâncias do Amazon EC2.

Para aprender mais sobre os diferentes métodos de configuração de nós no EKS, consulte Gerenciar computação. Depois que o cluster for implantado, você pode adicionar outros tipos de nós. Embora não seja abordado neste guia, também é possível usar os nós Autogerenciado do Windows e Bottlerocket para o cluster.

Para criar o grupo de nós gerenciados do Linux para EC2

1. Crie uma função do IAM do nó e associe a política gerenciada do Amazon EKS IAM a ela. O daemon `kubelet` do nó do Amazon EKS chama as APIs da AWS em seu nome. Os nós recebem

permissões para essas chamadas de API por meio de um perfil de instância do IAM e políticas associadas.

a. Copie o conteúdo a seguir em um arquivo denominado `node-role-trust-policy.json`.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Crie a função do IAM para o nó.

```
aws iam create-role \
  --role-name myAmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-policy.json"
```

c. Anexe à função as políticas do IAM gerenciadas necessárias.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name myAmazonEKSNodeRole
```

d. Abra o console do Amazon EKS em eks/home#/clusters.

e. Escolha o nome do cluster que você criou na Etapa 1: Criar o cluster do Amazon EKS, como *my-cluster*.

f. Na página ***my-cluster***, faça o seguinte:

g. Escolha a guia Compute (Computação).

h. Escolha Add Node Group (Adicionar grupo de nós).

2. Na página Configure Node Group (Configurar grupo de nós), faça o seguinte:

   a. Em Nome, digite um nome exclusivo para o grupo de nós gerenciados, como *my-nodegroup*. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.

   b. Em Node IAM role name (Nome da função do IAM de nós), escolha a função *myAmazonEKSNodeRole* que você criou em uma etapa anterior. Recomendamos que cada grupo de nós use sua própria função exclusiva do IAM.

   c. Escolha Próximo.

3. Na página Set compute and scaling configuration (Definir configurações de computação e escalabilidade), aceite os valores padrão e escolha Next (Próximo).

4. Na página Specify networking (Especificar redes), aceite os valores padrão e escolha Next (Próximo).

5. Na página Review and create (Revisar e criar), reveja a configuração do grupo de nós gerenciados e escolha Create (Criar).

6. Após alguns minutos, o Status em Node Group configuration(Configuração do grupo de nós) mudará de Creating (Em criação) para Active (Ativo). Não passe para a próxima etapa até que o status seja Active (Ativo).

Etapa 4: visualizar recursos

Você pode exibir os nós e as workloads do Kubernetes.

1. No painel de navegação à esquerda, escolha Clusters. Escolha o nome do cluster criado por você, por exemplo, *my-cluster*, na lista de Clusters.

2. Na página ***my-cluster***, escolha o seguinte:

   a. Guia Computação: você verá a lista de Nós que foram implantados para o cluster. Você pode escolher o nome de um nó para consultar mais informações sobre ele.

   b. Guia Recursos: você verá todos os recursos do Kubernetes implantados por padrão em um cluster do Amazon EKS. Selecione qualquer tipo de recurso no console para obter mais informações sobre ele.

Etapa 5: excluir recursos

Após concluir o cluster e os nós criados para este tutorial, exclua os recursos que você criou. Se quiser realizar outras ações com esse cluster antes de excluir os recursos, consulte the section called “Próximas etapas”.

1. Exclua qualquer perfil de grupo de nós que você tenha criado.
   a. Abra o console do Amazon EKS em eks/home#/clusters.
   b. No painel de navegação à esquerda, escolha Clusters. Na lista de clusters, escolha *my-cluster*.
   c. Escolha a guia Compute (Computação).
   d. Se você criou um grupo de nós, escolha o grupo de nós *my-nodegroup* e, em seguida, escolha Delete (Excluir). Digite *my-nodegroup* e selecione Delete.
   e. Aguarde a exclusão dos perfis de grupos de nós antes de prosseguir.
2. Excluir o cluster.
   a. No painel de navegação à esquerda, escolha Clusters. Na lista de clusters, escolha *my-cluster*.
   b. Escolha Delete Cluster (Excluir cluster).
   c. Digite *my-cluster* e selecione Delete. Não continue até que o cluster seja excluído.
3. Exclua a pilha do VPC AWS CloudFormation que você criou.
   a. Abra o console do CloudFormation em https://console.aws.amazon.com/cloudformation/.
   b. Escolha a pilha *my-eks-vpc-stack* e, então, Delete (Excluir).
   c. Na caixa de diálogo de confirmação Excluir ***my-eks-vpc-stack***, escolha Excluir pilha.
4. Exclua as funções do IAM criadas.
   a. Abra o console do IAM em https://console.aws.amazon.com/iam/.
   b. No painel de navegação à esquerda, escolha Funções.
   c. Selecione cada perfil que você criou na lista (incluindo ***myAmazonEKSClusterRole*** e *myAmazonEKSNodeRole*). Escolha Delete (Excluir), insira o texto de confirmação solicitado e escolha Delete (Excluir).

Próximas etapas

Os seguintes tópicos de documentação ajudam a estender a funcionalidade do seu cluster:

- A entidade principal do IAM que criou o cluster é a única entidade que pode fazer chamadas para o servidor de API do Kubernetes com o `kubectl` ou o Console de gerenciamento da AWS. Se quiser que outras entidades principais do IAM tenham acesso ao cluster, será necessário adicioná-los. Para obter mais informações, consulte the section called “Acesso à API do Kubernetes” e the section called “Permissões obrigatórias”.
- Implante uma aplicação de exemplo no cluster.
- Antes de implantar um cluster para uso em produção, recomendamos conhecer todas as configurações de clusters e nós. Algumas configurações (como habilitar o acesso SSH aos nós do Amazon EC2) devem ser feitas no momento de criação do cluster.
- Para aumentar a segurança do cluster, configure o plugin Amazon VPC Container Networking Interface para usar funções do IAM para contas de serviço.

Automatizar a infraestrutura de clusters com o Modo Automático do EKS

O Modo Automático do EKS estende o gerenciamento da AWS dos clusters do Kubernetes além do próprio cluster, o que possibilita que a AWS também configure e gerencie a infraestrutura que permite o bom funcionamento das workloads. Você pode delegar as principais decisões de infraestrutura e aproveitar a experiência da AWS para as operações diárias. A infraestrutura de clusters gerenciada pela AWS inclui muitos recursos do Kubernetes como componentes principais, em vez de complementos, como ajuste de escala automático de computação, rede de pods e serviços, balanceamento de carga de aplicações, DNS de cluster, armazenamento em blocos e suporte a GPU.

Para começar, você pode implantar um novo cluster do Modo Automático do EKS ou habilitar o Modo Automático do EKS em um cluster existente. Você pode implantar, atualizar ou modificar os clusters do Modo Automático do EKS usando o eksctl, a AWS CLI, o Console de gerenciamento da AWS, as APIs do EKS ou as ferramentas de preferência de infraestrutura como código.

Com o Modo Automático do EKS, você pode continuar usando as ferramentas de preferência compatíveis com o Kubernetes. O Modo Automático do EKS se integra aos serviços da AWS, como o Amazon EC2, Amazon EBS e ELB, aproveitando os recursos da Nuvem AWS que seguem as práticas recomendadas. Esses recursos são escalados automaticamente, otimizados em termos de custo e atualizados regularmente para ajudar a minimizar os custos operacionais e as despesas gerais.

Atributos

O Modo Automático do EKS oferece os seguintes recursos de alto nível:

Simplificar o gerenciamento de clusters do Kubernetes: o Modo Automático do EKS simplifica o gerenciamento do EKS fornecendo clusters prontos para produção com o mínimo de sobrecarga operacional. Com o Modo Automático do EKS, você pode executar workloads exigentes e dinâmicas com confiança, sem precisar ter uma profunda experiência no EKS.

Disponibilidade de aplicações: o Modo Automático do EKS adiciona ou remove dinamicamente nós no cluster do EKS com base nas demandas das aplicações do Kubernetes. Isso minimiza a necessidade de planejamento manual de capacidade e garante a disponibilidade das aplicações.

Eficiência: o Modo Automático do EKS foi projetado para otimizar os custos de computação e, ao mesmo tempo, aderir à flexibilidade definida pelos requisitos do NodePool e das workloads. Ele também encerra instâncias não utilizadas e consolida workloads em outros nós para melhorar a eficiência de custos.

Segurança: o Modo Automático do EKS usa AMIs que são tratadas como imutáveis para os nós. Essas AMIs impõem software bloqueado, habilitam controles de acesso obrigatórios do SELinux e fornecem sistemas de arquivos raiz somente leitura. Além disso, os nós executados pelo Modo Automático do EKS têm uma vida útil máxima de 21 dias (que você pode reduzir), após a qual são automaticamente substituídos por novos nós. Essa abordagem aprimora sua postura de segurança ao alternar os nós regularmente, alinhando-se às práticas recomendadas já adotadas por muitos clientes.

Atualizações automatizadas: o Modo Automático do EKS mantém o cluster, os nós e componentes relacionados do Kubernetes atualizados com os patches mais recentes, respeitando os orçamentos de interrupção de pods (PDBs) e os orçamentos de interrupção de NodePool (NDBs) configurados. Até o tempo de vida máximo de 21 dias, poderá ser necessária uma intervenção se PDBs de bloqueio ou outras configurações impedirem as atualizações.

Componentes gerenciados: o Modo Automático do EKS inclui recursos do Kubernetes e da Nuvem AWS como componentes principais que, de outra forma, precisariam ser gerenciados como complementos. Isso inclui suporte integrado para atribuições de endereço IP de pods, políticas de rede de pods, serviços DNS locais, plug-ins de GPU, verificadores de integridade e armazenamento da CSI do EBS.

NodePools e NodeClasses personalizáveis: se a workload exigir alterações nas configurações de armazenamento, computação ou rede, você poderá criar NodePools e NodeClasses personalizados usando o Modo Automático do EKS. Embora não seja recomendado editar NodePools e NodeClasses padrão, é possível adicionar novos NodePools ou NodeClasses personalizados junto com as configurações padrão para atender a requisitos específicos.

Componentes automatizados

O Modo Automático do EKS simplifica a operação dos clusters do Amazon EKS automatizando os componentes-chave da infraestrutura. Habilitar o Modo Automático do EKS reduz ainda mais as tarefas para gerenciar os clusters do EKS.

Veja abaixo uma lista dos componentes do plano de dados que são automatizados:

- Computação: para muitas workloads, com o Modo Automático do EKS, você não precisa se preocupar com muitos aspectos da computação nos clusters do EKS. Isso inclui:
  - Nós: os nós do Modo Automático do EKS são projetados para serem tratados como dispositivos. O Modo Automático do EKS faz o seguinte:
    - Escolhe uma AMI apropriada que esteja configurada com muitos serviços necessários para executar as workloads sem intervenção.
    - Bloqueia o acesso a arquivos na AMI usando o modo de imposição do SELinux e um sistema de arquivos raiz somente leitura.
    - Impede o acesso direto aos nós ao desabilitar o acesso por SSH ou SSM.
    - Inclui suporte para GPU, com drivers de kernel e plug-ins separados para GPUs NVIDIA e Neuron, possibilitando workloads de alta performance.
    - Lida automaticamente com os avisos de interrupção de instâncias spot do EC2 e os eventos de integridade das instâncias do EC2
  - Ajuste de escala automático: com base no ajuste de escala automático do Karpenter, o Modo Automático do EKS monitora pods não programáveis e possibilita a implantação de novos nós para executá-los. À medida que as workloads são encerradas, o Modo Automático do EKS interrompe e encerra dinamicamente os nós quando eles não são mais necessários, otimizando o uso de recursos.
  - Atualizações: assumir o controle dos nós otimiza a capacidade do Modo Automático do EKS de fornecer patches de segurança e atualizações do sistema operacional e de componentes, conforme necessário. Essas atualizações são projetadas para proporcionar o mínimo de interrupção das workloads. O Modo Automático do EKS impõe uma vida útil máxima de 21 dias para os nós, a fim de garantir software e APIs atualizados.
- Balanceamento de carga: o Modo Automático do EKS otimiza o balanceamento de carga ao se integrar ao serviço do Elastic Load Balancing da Amazon, automatizando o provisionamento e a configuração de balanceadores de carga para serviços do Kubernetes e recursos de entrada. Ele é compatível com recursos avançados para Application e Network Load Balancers, gerencia seus ciclos de vida e os escala para atender às demandas do cluster. Essa integração fornece uma solução de balanceamento de carga pronta para produção que segue as práticas recomendadas da AWS, permitindo que você se concentre nas aplicações e não no gerenciamento da infraestrutura.
- Armazenamento: o Modo Automático do EKS configura o armazenamento efêmero para você, configurando tipos de volume, tamanhos de volume, políticas de criptografia e políticas de exclusão após o encerramento dos nós.

- Rede: o Modo Automático do EKS automatiza tarefas críticas de rede para conectividade de pods e serviços. Isso inclui suporte a IPv4 e IPv6 e o uso de blocos CIDR secundários para estender espaços de endereços IP.
- Identity and Access Management: você não precisa instalar o agente da Identidade de Pods do EKS nos clusters do Modo Automático do EKS.

Para mais informações sobre esses componentes, consulte the section called “Como funciona”.

Configuração

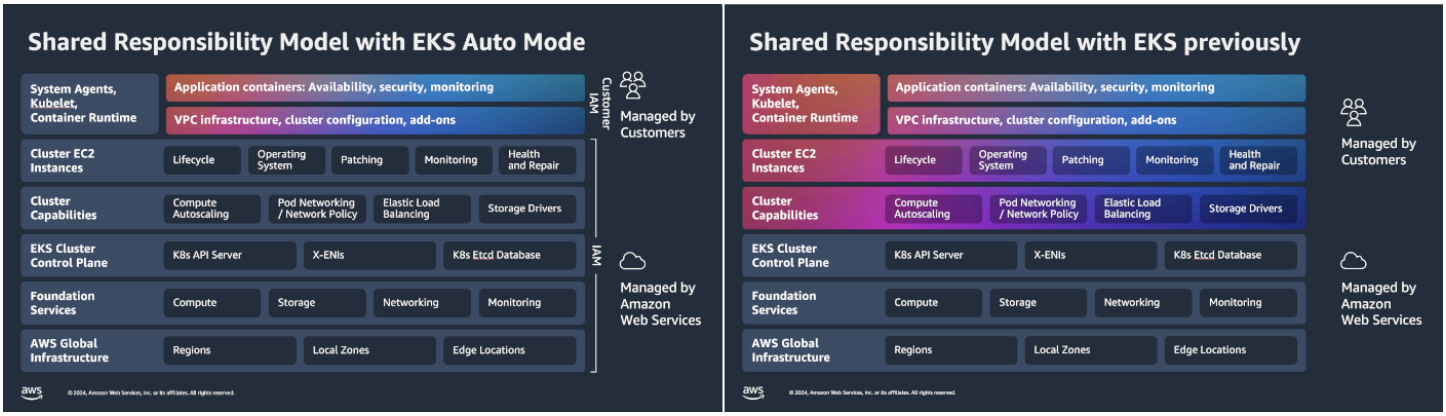
Embora o Modo Automático do EKS gerencie com eficácia a maioria dos serviços do plano de dados sem sua intervenção, poderá haver momentos em que você deseje mudar o comportamento de alguns desses serviços. Você pode modificar a configuração dos clusters do Modo Automático do EKS das seguintes maneiras:

- DaemonSets do Kubernetes: em vez de modificar os serviços instalados nos nós, você pode usar os DaemonSets do Kubernetes. Os DaemonSets são projetados para serem gerenciados pelo Kubernetes, mas executados em todos os nós no cluster. Dessa forma, você pode adicionar serviços especiais para monitorar ou supervisionar os nós.
- NodePools e NodeClasses personalizados: os NodePools e NodeClasses padrão são configurados pelo modo automático do EKS e não devem ser editados diretamente. Para personalizar o comportamento do nó, você pode criar NodePools ou NodeClasses adicionais para casos de uso como:
  - Selecionar tipos de instância específicos (por exemplo, processadores acelerados ou instâncias spot do EC2).
  - Isolar workloads para fins de segurança ou rastreamento de custos.
  - Configurar definições de armazenamento efêmero como IOPS, tamanho e throughput.
- Balanceamento de carga: alguns serviços, como balanceamento de carga, que o Modo Automático do EKS executa como objetos do Kubernetes, podem ser configurados diretamente nos clusters do Modo Automático do EKS.

Para obter mais informações sobre as opções de configuração do Modo Automático do EKS, consulte the section called “Configure”.

Modelo de responsabilidade compartilhada

O Modelo de responsabilidade compartilhada da AWS define as responsabilidades de segurança e conformidade entre a AWS e os clientes. As imagens e o texto abaixo comparam e contrastam como as responsabilidades do cliente e da AWS diferem entre o Modo Automático do EKS e o modo padrão do EKS.



O Modo Automático do EKS transfere grande parte da responsabilidade compartilhada pela infraestrutura do Kubernetes dos clientes para a AWS. Com o Modo Automático do EKS , a AWS assume mais responsabilidade pela segurança na nuvem, que antes era responsabilidade do cliente e agora é compartilhada. Agora, os clientes podem se concentrar mais em suas aplicações enquanto a AWS gerencia a infraestrutura subjacente.

Responsabilidade do cliente

No Modo Automático do EKS, os clientes continuam a manter a responsabilidade pelos contêineres de aplicações, incluindo disponibilidade, segurança e monitoramento. Eles também mantêm o controle sobre a infraestrutura da VPC e a configuração do cluster do EKS. Esse modelo permite que os clientes se concentrem nas preocupações específicas da aplicação enquanto delegam o gerenciamento da infraestrutura de cluster para a AWS. Recursos opcionais por nó podem ser incluídos em clusters por meio de complementos da AWS.

Responsabilidade da AWS

Com o Modo Automático do EKS , a AWS expande sua responsabilidade para incluir o gerenciamento de vários componentes críticos adicionais em comparação com aqueles já gerenciados em clusters do EKS que não usem o Modo Automático. Em particular, o Modo Automático do EKS assume a configuração, o gerenciamento, a segurança e o ajuste de escala das instâncias do EC2 lançadas, bem como os recursos de cluster para balanceamento de carga,

gerenciamento de endereços IP, política de rede e armazenamento em blocos. Os componentes a seguir são gerenciados pela AWS no Modo Automático do EKS:

- Instâncias EC2 lançadas no Modo Automático: a AWS gerencia o ciclo de vida completo dos nós utilizando as instâncias gerenciadas do Amazon EC2. As instâncias gerenciadas do EC2 são responsáveis pela configuração, aplicação de patches, monitoramento e manutenção da integridade do sistema operacional. Nesse modelo, tanto a instância em si quanto o sistema operacional convidado em execução nela são de responsabilidade da AWS. Os nós usam variantes das AMIs do Bottlerocket que são otimizadas para executar contêineres. As AMIs da Bottlerocket têm software bloqueado, sistemas de arquivos raiz imutáveis e acesso seguro à rede (para evitar comunicações diretas por meio de SSH ou SSM).
- Capacidades de cluster: a AWS gerencia o ajuste de escala automático da computação, redes de pod com aplicação de políticas de rede, integração com o Elastic Load Balancing e configuração de drivers de armazenamento.
- Plano de controle de cluster: a AWS continua gerenciando o servidor da API Kubernetes, os ENIs entre contas e o banco de dados etcd, como no EKS padrão.
- Serviços básicos e Infraestrutura global: a AWS mantém a responsabilidade pelos serviços subjacentes de computação, armazenamento, rede e monitoramento, bem como pela infraestrutura global de regiões, zonas locais e locais da borda.

Criação de um cluster com o modo automático do Amazon EKS

Este capítulo explica como criar um cluster do Amazon EKS com o Modo Automático habilitado usando várias ferramentas e interfaces. O Modo Automático simplifica a criação de clusters configurando e gerenciando automaticamente a infraestrutura de computação, rede e armazenamento do cluster. Você aprenderá a criar um cluster do Modo Automático usando a AWS CLI, o Console de gerenciamento da AWS ou a ferramenta de linha de comandos do eksctl.

> **Note**
>
> O Modo Automático do EKS requer a versão 1.29 ou superior do Kubernetes.

Escolha a ferramenta de preferência com base em suas necessidades: o Console de gerenciamento da AWS fornece uma interface visual ideal para aprender sobre os recursos do Modo Automático do EKS e criar clusters individuais. A AWS CLI é mais adequada para tarefas de script e automação,

especialmente ao integrar a criação de clusters em fluxos de trabalho ou pipelines de CI/CD existentes. A CLI do eksctl oferece uma experiência nativa do Kubernetes e é recomendada para usuários familiarizados com as ferramentas do Kubernetes que desejam operações de linha de comando simplificadas com padrões adequados.

Antes de começar, verifique se você tem os pré-requisitos necessários instalados e configurados, incluindo as permissões apropriadas do IAM para criar clusters do EKS. Para saber como instalar ferramentas da CLI, como `kubectl`, `aws` e `eksctl`, consulte Configuração.

Você pode usar a AWS CLI, o Console de gerenciamento da AWS ou a CLI do eksctl para criar um cluster com o Modo Automático do Amazon EKS.

Tópicos

- Criar um cluster do Modo Automático do EKS com a CLI do eksctl
- Criar um cluster do Modo Automático do EKS com a AWS CLI
- Criar um cluster do Modo Automático do EKS com o Console de gerenciamento da AWS

Criar um cluster do Modo Automático do EKS com a CLI do eksctl

Este tópico mostra como criar um cluster do Modo Automático do Amazon EKS usando a interface de linha de comandos (CLI) do eksctl. Você pode criar um cluster do Modo Automático executando um único comando da CLI ou aplicando um arquivo de configuração YAML. Ambos os métodos fornecem a mesma funcionalidade, com a abordagem de YAML oferecendo um controle mais granular das configurações do cluster.

A CLI do eksctl simplifica o processo de criação e gerenciamento de clusters do Modo Automático do EKS ao lidar com a criação e configuração de recursos subjacentes da AWS. Antes de prosseguir, certifique-se de ter as credenciais e permissões da AWS necessárias configuradas na máquina local. Este guia pressupõe que você esteja familiarizado com os conceitos básicos do Amazon EKS e já tenha instalado as ferramentas de CLI necessárias.

> **Note**
>
> Você deve instalar a versão `0.195.0` ou superior do eksctl. Para obter mais informações, consulte as versões do eksctl no GitHub.

Criar um cluster do Modo Automático do EKS com um comando da CLI

Você deve ter as ferramentas da `aws` e do `eksctl` instaladas. Você deve estar conectado à AWS CLI com permissões suficientes para gerenciar recursos da AWS, incluindo: instâncias do EC2, redes do EC2, clusters do EKS e perfis do IAM. Para ter mais informações, consulte Configuração.

Execute o comando a seguir para criar um cluster do Modo Automático do EKS com

```
eksctl create cluster --name=<cluster-name> --enable-auto-mode
```

Crie um cluster do Modo Automático do EKS com um arquivo YAML

Você deve ter as ferramentas da `aws` e do `eksctl` instaladas. Você deve estar conectado à AWS CLI com permissões suficientes para gerenciar recursos da AWS, incluindo: instâncias do EC2, redes do EC2, clusters do EKS e perfis do IAM. Para ter mais informações, consulte Configuração.

Analise as opções de configuração do Modo Automático do EKS no exemplo do recurso ClusterConfig abaixo. Para conferir a especificação completa do ClusterConfig, consulte a documentação do eksctl.

AWS sugere habilitar o Modo Automático do EKS. Se esta for a primeira vez que você cria um cluster do Modo Automático do EKS, deixe o `nodeRoleARN` não especificado criar um perfil do IAM do nó para o Modo Automático do EKS. Caso já tenha um perfil do IAM do nó na conta da AWS, a AWS sugere reutilizá-la.

AWS sugere não especificar nenhum valor para `nodePools`. O Modo Automático do EKS criará grupos de nós padrão. Você pode usar a API do Kubernetes para criar grupos de nós adicionais.

```
# cluster.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: <cluster-name>
  region: <aws-region>

iam:
  # ARN of the Cluster IAM Role
  # optional, eksctl creates a new role if not supplied
  # suggested to use one Cluster IAM Role per account
```

```
  serviceRoleARN: <arn-cluster-iam-role>

autoModeConfig:
  # defaults to false
  enabled: boolean
  # optional, defaults to [general-purpose, system].
  # suggested to leave unspecified
  # To disable creation of nodePools, set it to the empty array ([]).
  nodePools: []string
  # optional, eksctl creates a new role if this is not supplied
  # and nodePools are present.
  nodeRoleARN: string
```

Salve o arquivo `ClusterConfig` como `cluster.yaml` e use o seguinte comando para criar o cluster:

```
eksctl create cluster -f cluster.yaml
```

Criar um cluster do Modo Automático do EKS com a AWS CLI

Os clusters de modo automático do EKS automatizam as tarefas de rotina de gerenciamento de clusters para computação, armazenamento e rede. Por exemplo, os clusters do Modo Automático do EKS detectam automaticamente quando nós adicionais são necessários e provisionam novas instâncias do EC2 para atender às demandas de workloads.

Este tópico orienta você na criação de um novo cluster do Modo Automático do EKS usando a AWS CLI e, opcionalmente, na implantação de uma workload de exemplo.

Pré-requisitos

- A versão mais recente da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo. Para verificar sua versão atual, use `aws --version`. Para instalar a versão mais recente, consulte Instalar e Configuração rápida com aws configure, no Guia do usuário da AWS Command Line Interface.
  - Faça login na CLI com permissões suficientes do IAM para criar recursos da AWS, incluindo políticas do IAM, perfis do IAM e clusters do EKS.
- A ferramenta de linha de comando do kubectl instalada no dispositivo. A AWS sugere que você use a mesma versão do kubectl que a versão do Kubernetes do cluster do EKS. Para instalar ou atualizar o kubectl, consulte the section called “Configuração do kubectl e eksctl”.

Especificar sub-redes da VPC

O Modo Automático do Amazon EKS implanta nós em sub-redes da VPC. Ao criar um cluster do EKS, você deve especificar as sub-redes da VPC nas quais os nós serão implantados. Você pode usar as sub-redes da VPC padrão na conta da AWS ou criar uma VPC dedicada para workloads críticas.

- A AWS sugere a criação de uma VPC dedicada para o cluster. Saiba como the section called “Crie uma VPC”.
- O console do EKS ajuda na criação de uma nova VPC. Saiba como the section called “Console de Gerenciamento”.
- Como alternativa, você pode usar a VPC padrão da conta da AWS. Use as instruções a seguir para encontrar os IDs de sub-rede.

Para encontrar os IDs de sub-redes da VPC padrão

Usar a AWS CLI:

1. Execute o comando a seguir para listar a VPC padrão e suas sub-redes:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$(aws ec2 describe-vpcs
 --query 'Vpcs[?IsDefault==`true`].VpcId' --output text)" --query 'Subnets[*].
{ID:SubnetId,AZ:AvailabilityZone}' --output table
```

2. Salve a saída e anote os IDs das sub-redes.

   Exemplo de resultado:

```
----------------------------------------
|             DescribeSubnets          |
----------------------------------------
|   SubnetId        |   AvailabilityZone  |
|-------------------|---------------------|
|   subnet-012345678 |   us-west-2a        |
|   subnet-234567890 |   us-west-2b        |
|   subnet-345678901 |   us-west-2c        |
----------------------------------------
```

Perfis do IAM de clusters do Modo Automático do EKS

Função do IAM do cluster

O Modo Automático do EKS exige um perfil do IAM do cluster para realizar ações na conta da AWS, como provisionar novas instâncias do EC2. Você deve criar esse perfil para conceder ao EKS as permissões necessárias. A AWS recomenda anexar as seguintes políticas gerenciadas pela AWS ao perfil do IAM do cluster:

- AmazonEKSComputePolicy
- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- AmazonEKSNetworkingPolicy
- AmazonEKSClusterPolicy

Perfil do IAM do nó

Ao criar um cluster do Modo Automático do EKS, você especifica um perfil do IAM do nó. Quando o Modo Automático do EKS cria nós para processar workloads pendentes, cada novo nó de instância do EC2 recebe o perfil do IAM do nó. Esse perfil permite que o nó se comunique com o EKS, mas geralmente não é acessado pelas workloads em execução no nó.

Se você quiser conceder permissões para workloads em execução em um nó, use a Identidade de Pods do EKS. Para obter mais informações, consulte the section called “Identidade de Pods”.

Você deve criar esse perfil e anexar a seguinte política gerenciada pela AWS:

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

Função vinculada ao serviço

O Modo Automático do EKS também requer um perfil vinculado ao serviço, que é criado e configurado automaticamente pela AWS. Para obter mais informações, consulte AWSServiceRoleForAmazonEKS.

Criar um perfil do IAM do cluster do Modo Automático do EKS

Etapa 1: criar a política de confiança

Crie uma política de confiança para permitir que o serviço do Amazon EKS assuma o perfil. Salve a política como `trust-policy.json`:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Etapa 2: criar o perfil do IAM

Use a política de confiança para criar o perfil do IAM do cluster:

```
aws iam create-role \
    --role-name AmazonEKSAutoClusterRole \
    --assume-role-policy-document file://trust-policy.json
```

Etapa 3: registre o ARN do perfil

Recupere e salve o ARN do novo perfil para uso nas etapas subsequentes:

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

Etapa 4: anexar as políticas necessárias

Anexe as seguintes políticas gerenciadas pela AWS ao perfil do IAM do cluster para conceder as permissões necessárias:

AmazonEKSClusterPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
```

AmazonEKSComputePolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSComputePolicy
```

AmazonEKSBlockStoragePolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSBlockStoragePolicy
```

AmazonEKSLoadBalancingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

AmazonEKSNetworkingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSNetworkingPolicy
```

Criar um perfil do IAM do nó do Modo Automático do EKS

Etapa 1: criar a política de confiança

Crie uma política de confiança para permitir que o serviço do Amazon EKS assuma o perfil. Salve a política como `node-trust-policy.json`:

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Etapa 2: criar o perfil do IAM do nó

Use o arquivo node-trust-policy.json da etapa anterior para definir quais entidades podem assumir o perfil. Execute o seguinte comando para criar o perfil do IAM do nó:

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

Etapa 3: registre o ARN do perfil

Depois de criar o perfil, recupere e salve o ARN do perfil do IAM do nó. Você precisará desse ARN nas etapas subsequentes. Use o seguinte comando para obter o ARN:

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Etapa 4: anexar as políticas necessárias

Anexe as seguintes políticas gerenciadas pela AWS ao perfil do IAM do nó para fornecer as permissões necessárias:

AmazonEKSWorkerNodeMinimalPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

AmazonEC2ContainerRegistryPullOnly:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
```

```
    --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Criar um cluster do Modo Automático do EKS

Visão geral

Para criar um cluster de Modo Automático do EKS usando a AWS CLI, você precisará dos seguintes parâmetros:

- `cluster-name`: o nome do cluster.
- `k8s-version`: a versão do Kubernetes (por exemplo, 1.31).
- `subnet-ids`: IDs de sub-rede identificados nas etapas anteriores.
- `cluster-role-arn`: ARN do perfil do IAM do cluster.
- `node-role-arn`: ARN do perfil do IAM do nó.

Configurações padrão do cluster

Analise estes valores e recursos padrão antes de criar o cluster:

- `nodePools`: o Modo Automático do EKS inclui grupos de nós de uso geral e padrão do sistema. Saiba mais sobre Grupos de nós.

Observação: os grupos de nós no Modo Automático do EKS diferem dos grupos dos nós gerenciados pelo Amazon EKS, mas podem coexistir no mesmo cluster.

- `computeConfig.enabled`: automatiza tarefas computacionais de rotina, como criar e excluir instâncias do EC2.
- `kubernetesNetworkConfig.elasticLoadBalancing.enabled`: automatiza as tarefas de balanceamento de carga, incluindo a criação e exclusão de Elastic Load Balancers.
- `storageConfig.blockStorage.enabled`: automatiza tarefas de armazenamento, como criar e excluir volumes do Amazon EBS.
- `accessConfig.authenticationMode`: requer entradas de acesso do EKS. Saiba mais sobre os modos de autenticação do EKS.

Executar o comando

Use o seguinte comando para criar o cluster:

```
aws eks create-cluster \
  --region ${AWS_REGION} \
  --cli-input-json \
  "{
      \"name\": \"${CLUSTER_NAME}\",
      \"version\": \"${K8S_VERSION}\",
      \"roleArn\": \"${CLUSTER_ROLE_ARN}\",
      \"resourcesVpcConfig\": {
        \"subnetIds\": ${SUBNETS_JSON},
        \"endpointPublicAccess\": true,
        \"endpointPrivateAccess\": true
      },
      \"computeConfig\": {
        \"enabled\": true,
        \"nodeRoleArn\":\"${NODE_ROLE_ARN}\",
        \"nodePools\": [\"general-purpose\", \"system\"]
      },
      \"kubernetesNetworkConfig\": {
        \"elasticLoadBalancing\": {
          \"enabled\": true
        }
      },
      \"storageConfig\": {
        \"blockStorage\": {
          \"enabled\": true
        }
      },
      \"accessConfig\": {
        \"authenticationMode\": \"API\"
      }
    }
```

Verificar o status do cluster

Etapa 1: verificar a criação do cluster

Execute o comando a seguir para verificar o status do cluster. A criação do cluster normalmente leva cerca de 15 minutos.

```
aws eks describe-cluster --name "${CLUSTER_NAME}" --output json
```

Etapa 2: atualizar o kubeconfig

Quando o cluster estiver pronto, atualize o arquivo kubeconfig local para permitir que o `kubectl` se comunique com o cluster. Esta configuração usa a AWS CLI para autenticação.

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

Etapa 3: verificar grupos de Node

Listar os grupos de nós no cluster com o comando a seguir:

```
kubectl get nodepools
```

Próximas etapas

- Saiba como implantar uma workload de exemplo no novo cluster do Modo Automático do EKS.

Criar um cluster do Modo Automático do EKS com o Console de gerenciamento da AWS

Criar um cluster do Modo Automático do EKS no Console de gerenciamento da AWS requer menos configuração do que outras opções. O EKS se integra ao AWS IAM e à rede de VPC para ajudar a criar os recursos associados a um cluster do EKS.

Você tem duas opções para criar um cluster no console:

- Configuração rápida (com o Modo Automático do EKS)
- Configuração personalizada

Neste tópico, você aprenderá como criar um cluster do Modo Automático do EKS usando a opção Configuração rápida.

Criar um Modo Automático do EKS usando a opção de configuração rápida

Você deve estar conectado ao Console de gerenciamento da AWS com permissões suficientes para gerenciar recursos da AWS, incluindo: instâncias do EC2, redes do EC2, clusters do EKS e perfis do IAM.

1. Navegue até o console do EKS.
2. Clique em Criar cluster.
3. Confirme se a opção Configuração rápida está selecionada.
4. Determine os valores a seguir ou use os padrões para um cluster de teste.
   - Nome do cluster
   - Versão do Kubernetes
5. Selecione o perfil do IAM do cluster. Se esta for a primeira vez que você cria um cluster do Modo Automático do EKS, use a opção Criar perfil recomendado.
   - Opcionalmente, você pode reutilizar um único perfil do IAM do cluster na conta da AWS para todos os clusters do Modo Automático do EKS.
   - O perfil do IAM do cluster inclui as permissões necessárias para o Modo Automático do EKS gerenciar recursos, incluindo instâncias do EC2, volumes do EBS e balanceadores de carga do EC2.
   - A opção Criar perfil recomendado preenche previamente todos os campos com valores recomendados. Selecione Próximo e, em seguida, Criar. O perfil usará o nome `AmazonEKSAutoClusterRole` sugerido.
   - Se você criou recentemente um novo perfil, use o ícone de Atualizar para recarregar a lista suspensa de seleção de perfis.
6. Selecione o perfil do IAM do nó. Se esta for a primeira vez que você cria um cluster do Modo Automático do EKS, use a opção Criar perfil recomendado.
   - Opcionalmente, você pode reutilizar um único perfil do IAM do nó na conta da AWS para todos os clusters do Modo Automático do EKS.
   - O perfil do IAM do nó inclui as permissões necessárias para que os nós do Modo Automático se conectem ao cluster. O perfil do IAM do nó deve incluir permissões para recuperar imagens do ECR para os contêineres.
   - A opção Criar perfil recomendado preenche previamente todos os campos com valores recomendados. Selecione Próximo e, em seguida, Criar. O perfil usará o nome `AmazonEKSAutoNodeRole` sugerido.
   - Se você criou recentemente um novo perfil, use o ícone de Atualizar para recarregar a lista suspensa de seleção de perfis.
7. Selecione a VPC para o cluster do Modo Automático do EKS. Escolha Criar VPC para criar uma VPC para o EKS, ou escolha uma VPC que você criou anteriormente para o EKS.

- Se você usar o console da VPC para criar uma VPC, a AWS sugere que você crie pelo menos um gateway NAT por zona de disponibilidade. Do contrário, você pode usar todos os outros padrões.
- Para obter mais informações e detalhes sobre os requisitos do cluster IPv6, consulte the section called “Crie uma VPC”.

8. (opcional) O Modo Automático do EKS preenche automaticamente as sub-redes privadas da VPC selecionada. Você pode remover as sub-redes indesejadas.
   - O EKS seleciona automaticamente sub-redes privadas da VPC seguindo as práticas recomendadas. Opcionalmente, você pode selecionar sub-redes adicionais da VPC, como sub-redes públicas.
9. (opcional) Selecione Visualizar padrões de configuração rápida para analisar todos os valores de configuração do novo cluster. A tabela indica que alguns valores não são editáveis após a criação do cluster.
10. Selecione Create cluster (Criar cluster). Observe que a conclusão da criação do cluster pode levar quinze minutos.

Próximas etapas

- Saiba como implantar uma workload de exemplo no cluster do Modo Automático do EKS

Habilitar o Modo Automático do EKS em clusters existentes do EKS

Você pode habilitar o Modo Automático do EKS em clusters existentes do EKS.

A AWS oferece suporte às seguintes migrações:

- Migração do Karpenter para os nós do Modo Automático do EKS. Para obter mais informações, consulte the section called “Migrar do Karpenter”.
- Migração dos grupos de nós gerenciados pelo EKS para os nós do Modo Automático do EKS. Para obter mais informações, consulte the section called “Migrar de MNGs”.
- Migração do EKS Fargate para o Modo Automático do EKS. Para obter mais informações, consulte the section called “Migrar do Fargate”.

A AWS não oferece suporte às seguintes migrações:

- Migração de volumes do controlador EBS CSI (usando o complemento do Amazon EKS) para o controlador EBS CSI para o Modo Automático do EKS (gerenciado pelo Modo Automático do EKS). Os PVCs realizados com um não podem ser montados pelo outro, porque usam dois provisionadores de volume diferentes do Kubernetes.
  - O `eks-auto-mode-ebs-migration-tool` (projeto AWS Labs) permite a migração entre o EBS CSI StorageClass padrão (`ebs.csi.aws.com`) e o EKS Auto EBS CSI StorageClass (`ebs.csi.eks.amazonaws.com`). Observe que a migração exige a exclusão e a recriação dos recursos existentes de PersistentVolumeClaim/PersistentVolume, portanto, a validação em um ambiente que não seja de produção é essencial antes da implementação.
- Migrar balanceadores de carga do AWS Load Balancer Controller para o Modo Automático do EKS

  Você pode instalar o AWS Load Balancer Controller em um cluster do Modo Automático do Amazon EKS. Use as opções `IngressClass` ou `loadBalancerClass` para associar recursos de serviço e entrada ao Load Balancer Controller ou ao Modo Automático do EKS.
- Migração de clusters do EKS com CNIs alternativas ou outras configurações de rede não compatíveis

Referência de migração

Use a referência de migração a seguir para configurar recursos do Kubernetes para serem de propriedade de controladores autogerenciados ou do Modo Automático do EKS.

Recurso	Recurso	Campo	Autogerenciado	Modo Automático do EKS
Armazenamento em bloco	`StorageClass`	`provisioner`	`ebs.csi.aws.com`	`ebs.csi.eks.amazonaws.com`
Balanceamento de carga	`Service`	`loadBalancerClass`	`service.k8s.aws/nlb`	`eks.amazonaws.com/nlb`

Recurso	Recurso	Campo	Autogerenciado	Modo Automático do EKS
Balanceamento de carga	`IngressClass`	`controller`	`ingress.k8s.aws/alb`	`eks.amazonaws.com/alb`
Balanceamento de carga	`IngressClassParams`	`apiversion`	`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`
Balanceamento de carga	`TargetGroupBinding`	`apiversion`	`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`
Computação	`NodeClass`	`apiVersion`	`karpenter.sh/v1`	`eks.amazonaws.com/v1`

Migração de volumes do EBS

Ao migrar workloads para o Modo Automático do EKS, você precisa lidar com a migração de volumes do EBS devido aos diferentes provisionadores de drivers CSI:

- Provisionador do Modo Automático do EKS: `ebs.csi.eks.amazonaws.com`
- Provisionador do EBS CSI de código aberto: `ebs.csi.aws.com`

Siga estas etapas para migrar os volumes persistentes:

1. Modifique a política de retenção de volume: altere a versão existente da plataforma (PV) `persistentVolumeReclaimPolicy` para `Retain` para garantir que o volume subjacente do EBS não seja excluído.
2. Remova o PV do Kubernetes: exclua o recurso PV antigo enquanto mantém o volume real do EBS intacto.
3. Crie um novo PV com provisionamento estático: crie um novo PV que faça referência ao mesmo volume do EBS, mas que funcione com o driver CSI de destino.

4. Vincular a um novo PVC: crie um novo PVC que faça referência específica ao seu PV usando o campo `volumeName`.

Considerações

- Certifique-se de que suas aplicações sejam interrompidas antes de iniciar a migração.
- Faça backup dos dados antes de iniciar o processo de migração.
- Esse processo precisa ser executado para cada volume persistente.
- A workload deve ser atualizada para usar o novo PVC.

Migração de balanceadores de carga

Você não pode transferir diretamente os balanceadores de carga existentes do controlador autogerenciado de balanceadores de carga da AWS para o Modo Automático do EKS. Em vez disso, você deve implementar uma estratégia de implantação azul/verde. Isso envolve manter a configuração atual do balanceador de carga enquanto cria balanceadores de carga no controlador gerenciado.

Para minimizar a interrupção do serviço, recomendamos uma abordagem de mudança de tráfego baseada em DNS. Primeiro, crie balanceadores de carga usando o Modo Automático do EKS, mantendo a configuração operacional existente. Em seguida, use o roteamento de DNS (como o Route 53) para transferir gradualmente o tráfego dos balanceadores de carga antigos para os novos. Depois que o tráfego tiver sido migrado com êxito e você tiver verificado a nova configuração, será possível desativar os balanceadores de carga antigos e o controlador autogerenciado.

Habilitar o Modo Automático do EKS em um cluster existente

Este tópico descreve como habilitar o Modo Automático do EKS nos clusters existentes no Amazon EKS. Habilitar o Modo Automático em um cluster existente requer a atualização das permissões do IAM e a configuração das principais configurações do Modo Automático do EKS. Depois de habilitado, você pode começar a migrar as workloads computacionais existentes para aproveitar as operações simplificadas e o gerenciamento automatizado da infraestrutura do Modo Automático.

> **Important**
>
> Verifique se você tem a versão mínima necessária de determinados complementos do Amazon EKS instalada antes de habilitar o Modo Automático do EKS. Para obter mais informações, consulte the section called “Versões de complementos necessárias”.

Antes de começar, certifique-se de ter acesso de administrador ao seu cluster do Amazon EKS e permissões para modificar os perfis do IAM. As etapas neste tópico orientam você a habilitar o Modo Automático usando o Console de gerenciamento da AWS ou a AWS CLI.

Console de gerenciamento da AWS

Você deve estar conectado ao Console da AWS com permissão para gerenciar recursos do IAM, EKS e EC2.

> **Note**
>
> O perfil do IAM do cluster de um cluster do EKS não pode ser alterado após a criação do cluster. O Modo Automático do EKS requer permissões adicionais nesse perfil. Você deve anexar políticas adicionais ao perfil atual.

Atualizar perfil do IAM do cluster

1. Abra a página de visão geral do cluster no Console de gerenciamento da AWS.
2. Em ARN do perfil do IAM do cluster, selecione Visualizar no IAM.
3. Na lista suspensa Adicionar permissões, selecione Anexar políticas.
4. Use a caixa Pesquisar para encontrar e selecionar as seguintes políticas:
   - `AmazonEKSComputePolicy`
   - `AmazonEKSBlockStoragePolicy`
   - `AmazonEKSLoadBalancingPolicy`
   - `AmazonEKSNetworkingPolicy`
   - `AmazonEKSClusterPolicy`
5. Selecione Adicionar permissões.
6. Na guia Relacionamentos de confiança, selecione Editar política de confiança.

7. Insira a política de confiança a seguir do perfil do IAM do cluster e selecione Atualizar política

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Habilitar o Modo Automático do EKS

1. Abra a página de visão geral do cluster no Console de gerenciamento da AWS.
2. Em Modo Automático do EKS, selecione Gerenciar.
3. Alterne o Modo Automático do EKS para ativado.
4. No menu suspenso Grupo de nós do EKS, selecione os grupos de nós padrão que você deseja criar.
   - Saiba mais sobre grupos de nós no Modo Automático do EKS. Para obter mais informações, consulte the section called “Criar grupo de nós”.
5. Se você já criou um perfil do IAM do nó do Modo Automático do EKS nesta conta da AWS, selecione-a no menu suspenso Perfil do IAM do nó. Se você ainda não criou esse perfil, selecione Criar perfil recomendado e siga as etapas.

AWS CLI

Pré-requisitos

- O perfil do IAM do cluster do cluster existente do EKS deve incluir permissões suficientes para o Modo Automático do EKS, como as seguintes políticas:
  - `AmazonEKSComputePolicy`

- `AmazonEKSBlockStoragePolicy`
- `AmazonEKSLoadBalancingPolicy`
- `AmazonEKSNetworkingPolicy`
- `AmazonEKSClusterPolicy`

- O perfil do IAM do cluster deve ter uma política de confiança atualizada, incluindo a ação `sts:TagSession`. Para obter mais informações sobre como criar um perfil do IAM do cluster, consulte the section called “ AWS CLI”.
- `aws` CLI instalada, com login efetuado e em uma versão suficiente. Você deve ter permissão para gerenciar recursos do IAM, EKS e EC2. Para obter mais informações, consulte Configuração.

Procedimento

Use os comandos a seguir para habilitar o Modo Automático do EKS em um cluster existente.

> **Note**
>
> Os recursos de computação, armazenamento em blocos e balanceamento de carga devem estar todos habilitados ou desabilitados na mesma solicitação.

```
aws eks update-cluster-config \
 --name $CLUSTER_NAME \
 --compute-config enabled=true \
 --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": true}}' \
 --storage-config '{"blockStorage":{"enabled": true}}'
```

Versões de complementos necessárias

Se você planeja habilitar o Modo Automático do EKS em um cluster existente, talvez seja necessário atualizar alguns complementos. Observação:

- Isso se aplica somente aos clusters existentes em transição para o Modo Automático do EKS.
- Novos clusters criados com o Modo Automático do EKS habilitado não exigem essas atualizações.

Se você tiver algum dos seguintes complementos instalados, verifique se eles estão pelo menos na versão mínima especificada:

Nome do complemento	Versão mínima necessária
Plug-in CNI da Amazon VPC para Kubernetes	v1.19.0-eksbuild.1
Kube-proxy	• v1.26.15-eksbuild.19 • v1.27.16-eksbuild.14 • v1.28.15-eksbuild.4 • v1.29.10-eksbuild.3 • v1.30.6-eksbuild.3 • v1.31.2-eksbuild.3
Driver da CSI do Amazon EBS	v1.37.0-eksbuild.1
Controlador de snapshots da CSI	v8.1.0-eksbuild.2
Atendente de Identidade de Pods do EKS	v1.3.4-eksbuild.1

Para obter mais informações, consulte the section called “Atualização de um complemento”.

Próximas etapas

- Para migrar as workloads de Gerenciar grupos de nós, consulte the section called “Migrar de MNGs”.
- Para migrar do Karpenter autogerenciado, consulte the section called “Migrar do Karpenter”.

Migrar do Karpenter para o Modo Automático do EKS usando o kubectl

Este tópico orienta você pelo processo de migração de workloads do Karpenter para o Modo Automático do Amazon EKS usando o kubectl. A migração pode ser realizada gradualmente, permitindo que você mova as workloads no seu próprio ritmo, mantendo a estabilidade do cluster e a disponibilidade das aplicações durante a transição.

A abordagem passo a passo descrita abaixo permite que você execute o Karpenter e o Modo Automático do EKS lado a lado durante o período de migração. Essa estratégia de operação dupla ajuda a garantir uma transição suave, permitindo que você valide o comportamento da workload no Modo Automático do EKS antes de desativar completamente o Karpenter. Você pode migrar

aplicações individualmente ou em grupos, oferecendo flexibilidade para acomodar seus requisitos operacionais específicos e a tolerância a riscos.

Pré-requisitos

Antes de iniciar a migração, verifique se você tem:

- Karpenter v1.1 ou posterior instalado no cluster. Para obter mais informações, consulte Upgrading to 1.1.0+ na documentação do Karpenter.
- `kubectl` instalado e conectado ao cluster. Para obter mais informações, consulte Configuração.

Este tópico pressupõe que você esteja familiarizado com o Karpenter e NodePools. Para obter mais informações, consulte a documentação do Karpenter.

Etapa 1: habilitar o modo automático do EKS no cluster

Habilite o Modo Automático do EKS no cluster existente usando a CLI ou o Console de Gerenciamento da AWS. Para obter mais informações, consulte the section called “Habilitar no cluster”.

> **Note**
>
> Ao ativar o Modo Automático do EKS, não habilite o nodepool `general purpose` neste estágio durante a transição. Esse grupo de nós não é seletivo.
> Para obter mais informações, consulte the section called “Analisar grupos de nós integrados”.

Etapa 2: criar um NodePool com taint do Modo Automático do EKS

Crie um NodePool para o Modo Automático do EKS com um taint. Isso garante que os pods existentes não sejam programados automaticamente nos novos nós do Modo Automático do EKS. Esse grupo de nós usa o `NodeClass default` criado no Modo Automático do EKS. Para obter mais informações, consulte the section called “Criar classe de nó”.

Exemplo de grupo de nós com taint:

```
apiVersion: karpenter.sh/v1
kind: NodePool
```

```
metadata:
  name: eks-auto-mode
spec:
  template:
    spec:
      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      taints:
        - key: "eks-auto-mode"
          effect: "NoSchedule"
```

Atualize os requisitos do grupo de nós para que correspondam à configuração do Karpenter da qual você está migrando. Você precisa de pelo menos um requisito.

Etapa 3: atualizar workloads para migração

Identifique e atualize as workloads que você deseja migrar para o Modo Automático do EKS. Adicione tolerâncias e seletores de nós às workloads:

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
```

Essa alteração permite que a workload seja programada nos novos nós do Modo Automático do EKS.

O Modo Automático do EKS usa rótulos diferentes do Karpenter. Os rótulos relacionados às instâncias gerenciadas pelo EC2 começam com `eks.amazonaws.com`. Para obter mais informações, consulte the section called “Criar grupo de nós”.

Etapa 4: migrar gradualmente as workloads

Repita a Etapa 3 para cada workload que você deseja migrar. Isso permite que você mova workloads individualmente ou em grupos, com base nos requisitos e tolerância a riscos.

Etapa 5: remover o NodePool original do Karpenter

Depois que todas as workloads tiverem sido migradas, você poderá remover o NodePool original do Karpenter:

```
kubectl delete nodepool <original-nodepool-name>
```

Etapa 6: remover o taint do NodePool do Modo Automático do EKS (opcional)

Se você quiser que o Modo Automático do EKS se torne o padrão para novas workloads, você pode remover o taint do NodePool do Modo Automático do EKS:

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: eks-auto-mode
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      # Remove the taints section
```

Etapa 7: remover os seletores de nós das workloads (opcional)

Se você removeu o taint do NodePool do Modo Automático do EKS, você pode, opcionalmente, remover os seletores de nós das workloads, já que o Modo Automático do EKS agora é o padrão:

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
```

```
      # Remove the nodeSelector section
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
```

Etapa 8: desinstalar o Karpenter do cluster

As etapas para remover o Karpenter dependem de como você o instalou. Para obter mais informações, consulte as instruções de instalação do Karpenter.

Migrar dos grupos de nós gerenciados pelo EKS para o Modo Automático do EKS

Ao fazer a transição do cluster do Amazon EKS para usar o Modo Automático do EKS, você pode migrar facilmente as workloads existentes de grupos de nós gerenciados (MNGs) usando a ferramenta eksctl da CLI. Esse processo garante a disponibilidade contínua da aplicação, enquanto o Modo Automático do EKS otimiza os recursos computacionais. A migração pode ser realizada com o mínimo de interrupção nas aplicações em execução.

Este tópico mostra as etapas para drenar com segurança os pods dos grupos de nós gerenciados existentes e permitir que o Modo Automático do EKS os reprograme em instâncias recém-provisionadas. Seguindo esse procedimento, você pode aproveitar a consolidação inteligente de workloads do Modo Automático do EKS e, ao mesmo tempo, manter a disponibilidade da aplicação durante a migração.

Pré-requisitos

- Cluster com o Modo Automático do EKS habilitado
- `eksctl` CLI instalada e conectada ao cluster. Para obter mais informações, consulte Configuração.
- O Karpenter não está instalado no cluster.

Procedimento

Use o comando `eksctl` da CLI a seguir para iniciar a drenagem de pods das instâncias existentes do grupo de nós gerenciados. O Modo Automático do EKS criará nós para dar suporte aos pods deslocados.

```
eksctl delete nodegroup --cluster=<clusterName> --name=<nodegroupName>
```

Você precisará executar esse comando para cada grupo de nós gerenciados no cluster.

Para obter mais informações sobre esse comando, consulte Deleting and draining nodegroups na documentação do eksctl.

Migrar do EKS Fargate para o Modo Automático do EKS

Este tópico orienta você pelo processo de migração de workloads do EKS Fargate para o Modo Automático do Amazon EKS usando o `kubectl`. A migração pode ser realizada gradualmente, permitindo que você mova as workloads no seu próprio ritmo, mantendo a estabilidade do cluster e a disponibilidade das aplicações durante a transição.

A abordagem passo a passo descrita abaixo permite que você execute o EKS Fargate e o Modo Automático do EKS lado a lado durante o período de migração. Essa estratégia de operação dupla ajuda a garantir uma transição suave, permitindo que você valide o comportamento da workload no Modo Automático do EKS antes de desativar completamente o EKS Fargate. Você pode migrar aplicações individualmente ou em grupos, oferecendo flexibilidade para acomodar seus requisitos operacionais específicos e a tolerância a riscos.

Comparação do Modo Automático do Amazon EKS com o EKS com o AWS Fargate

O Amazon EKS com o AWS Fargate continua sendo uma opção para os clientes que desejam executar o EKS, mas o Modo Automático do Amazon EKS é a abordagem recomendada no futuro. O Modo Automático do EKS é totalmente compatível com o Kubernetes, oferecendo suporte a todas as ferramentas de plataforma e primitivas upstream do Kubernetes, como o Istio, com o qual o Fargate não é compatível. O Modo Automático do EKS também tem compatibilidade total com todas as opções de compra de runtime do EC2, incluindo instâncias de GPU e spot, permitindo que os clientes aproveitem descontos negociados do EC2 e outros mecanismos de economia. Esses recursos não estão disponíveis ao usar o EKS com o Fargate.

Além disso, o Modo Automático do EKS permite que os clientes obtenham o mesmo modelo de isolamento do Fargate, usando recursos de programação padrão do Kubernetes para garantir que cada instância do EC2 execute um único contêiner de aplicação. Ao adotar o Modo Automático do EKS, os clientes podem aproveitar todos os benefícios de executar o Kubernetes na AWS: uma plataforma totalmente compatível com o Kubernetes que oferece a flexibilidade de aproveitar toda a gama de opções de compras e do EC2, preservando a facilidade de uso e a abstração do gerenciamento de infraestrutura que o Fargate oferece.

Pré-requisitos

Antes de iniciar a migração, verifique se você tem:

- Configure um cluster com o Fargate. Para obter mais informações, consulte the section called "Conceitos básicos".
- O `kubectl` instalado e conectado ao cluster. Para obter mais informações, consulte Configuração.

Etapa 1: verificar o cluster do Fargate

1. Verifique se o cluster do EKS com o Fargate está em execução:

```
kubectl get node
```

```
NAME STATUS ROLES AGE VERSION
fargate-ip-192-168-92-52.ec2.internal Ready <none> 25m v1.30.8-eks-2d5f260
fargate-ip-192-168-98-196.ec2.internal Ready <none> 24m v1.30.8-eks-2d5f260
```

2. Verifique os pods em execução:

```
kubectl get pod -A
```

```
NAMESPACE NAME READY STATUS RESTARTS AGE
kube-system coredns-6659cb98f6-gxpjz 1/1 Running 0 26m
kube-system coredns-6659cb98f6-gzzsx 1/1 Running 0 26m
```

3. Crie uma implantação em um arquivo denominado `deployment_fargate.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
```

```
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: fargate
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

4. Aplique a implantação:

```
kubectl apply -f deployment_fargate.yaml
```

```
deployment.apps/nginx-deployment created
```

5. Confira os pods e as implantações:

```
kubectl get pod,deploy
```

```
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-5c7479459b-6trtm   1/1     Running   0          61s
pod/nginx-deployment-5c7479459b-g8ssb   1/1     Running   0          61s
pod/nginx-deployment-5c7479459b-mq4mf   1/1     Running   0          61s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3            3           61s
```

6. Verifique o nó:

```
kubectl get node -owide
```

```
NAME                                    STATUS  ROLES  AGE VERSION
   INTERNAL-IP     EXTERNAL-IP OS-IMAGE       KERNEL-VERSION
 CONTAINER-RUNTIME
fargate-ip-192-168-111-43.ec2.internal  Ready   <none> 31s v1.30.8-eks-2d5f260
 192.168.111.43  <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
```

```
fargate-ip-192-168-117-130.ec2.internal Ready   <none> 36s v1.30.8-eks-2d5f260
 192.168.117.130 <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
fargate-ip-192-168-74-140.ec2.internal  Ready   <none> 36s v1.30.8-eks-2d5f260
 192.168.74.140  <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
```

Etapa 2: habilitar o Modo Automático do EKS no cluster

1. Habilite o Modo Automático do EKS no cluster existente usando a CLI ou o Console de Gerenciamento da AWS. Para obter mais informações, consulte the section called “Habilitar no cluster”.
2. Confira o nodepool:

```
kubectl get nodepool
```

```
NAME              NODECLASS   NODES   READY   AGE
general-purpose   default     1       True    6m58s
system            default     0       True    3d14h
```

Etapa 3: atualizar workloads para migração

Identifique e atualize as workloads que você deseja migrar para o Modo Automático do EKS.

Para migrar uma workload do Fargate para o Modo Automático do EKS, aplique a anotação `eks.amazonaws.com/compute-type: ec2`. Isso garante que a workload não seja programada pelo Fargate, apesar do perfil do Fargate, e que seja detectada pelo NodePool do Modo Automático do EKS. Para obter mais informações, consulte the section called “Criar grupo de nós”.

1. Modifique as implantações (por exemplo, o arquivo `deployment_fargate.yaml`) para alterar o tipo de computação para `ec2`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
```

```
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

2. Aplique a implantação. Essa alteração permite que a workload seja programada nos novos nós do Modo Automático do EKS:

```
kubectl apply -f deployment_fargate.yaml
```

3. Verifique se a implantação está em execução no cluster do Modo Automático do EKS:

```
kubectl get pod -o wide
```

```
NAME                               READY   STATUS    RESTARTS   AGE     IP
    NODE                   NOMINATED NODE   READINESS GATES
nginx-deployment-97967b68d-ffxxh   1/1     Running   0          3m31s
 192.168.43.240   i-0845aafcb51630ffb   <none>           <none>
nginx-deployment-97967b68d-mbcgj   1/1     Running   0          2m37s
 192.168.43.241   i-0845aafcb51630ffb   <none>           <none>
nginx-deployment-97967b68d-qpd8x   1/1     Running   0          2m35s
 192.168.43.242   i-0845aafcb51630ffb   <none>           <none>
```

4. Verifique se não há nenhum nó do Fargate nem nenhuma implantação em execução nos nós gerenciados pelo Modo Automático do EKS:

```
kubectl get node -owide
```

```
NAME                  STATUS ROLES  AGE   VERSION              INTERNAL-IP     EXTERNAL-
IP OS-IMAGE                                       KERNEL-VERSION CONTAINER-RUNTIME
i-0845aafcb51630ffb Ready  <none> 3m30s v1.30.8-eks-3c20087 192.168.41.125
 3.81.118.95 Bottlerocket (EKS Auto) 2025.3.14 (aws-k8s-1.30) 6.1.129
 containerd://1.7.25+bottlerocket
```

Etapa 4: migrar gradualmente as workloads

Repita a Etapa 3 para cada workload que você deseja migrar. Isso permite que você mova workloads individualmente ou em grupos, com base nos requisitos e tolerância a riscos.

Etapa 5: remover o perfil original do Fargate

Depois que todas as workloads tiverem sido migradas, você poderá remover o perfil original do `fargate`. Substitua *`<fargate profile name>`* pelo nome do seu perfil do Fargate:

```
aws eks delete-fargate-profile --cluster-name eks-fargate-demo-cluster --fargate-
profile-name <fargate profile name>
```

Etapa 6: reduzir a escala verticalmente do CoreDNS

Como o Modo Automático do EKS processa o CoreDNS, você reduz a escala verticalmente da implantação do `coredns` para 0:

```
kubectl scale deployment coredns -n kube-system —-replicas=0
```

Execução de workloads de amostra em clusters do modo automático do EKS

Este capítulo fornece exemplos de como implantar diferentes tipos de workloads em clusters do Amazon EKS executados no Modo Automático. Os exemplos demonstram os principais padrões de workloads, incluindo aplicações de exemplo, aplicações Web com balanceamento de carga, workloads com estado usando armazenamento persistente e workloads com requisitos específicos de posicionamento de nós. Cada exemplo inclui manifestos completos e instruções de implantação passo a passo que você pode usar como modelos para suas próprias aplicações.

Antes de prosseguir com os exemplos, certifique-se de que você tenha um cluster do EKS em execução no Modo Automático e que tenha instalado a AWS CLI e o kubectl. Para obter mais informações, consulte Configuração. Os exemplos pressupõem familiaridade básica com os conceitos do Kubernetes e comandos do kubectl.

Você pode usar essas amostras baseadas em casos de uso para executar workloads em clusters do Modo Automático do EKS.

the section called “Implantar workload de ampliação”

Mostra como implantar uma workload de exemplo em um cluster do Modo Automático do EKS usando comandos do `kubectl`.

the section called “Implantar balanceador de carga”

Mostra como implantar uma versão em contêiner do jogo 2048 no Amazon EKS.

the section called “Implantar workload com estado”

Mostra como implantar uma aplicação com estado de exemplo em um cluster do Modo Automático do EKS.

the section called “Implementar workload acelerada”

Mostra como implantar workloads com aceleração de hardware em nós gerenciados pelo modo automático do EKS.

the section called “Controlar a implantação”

Mostra como usar uma anotação para controlar se uma workload estiver implantada em nós gerenciados pelo Modo Automático do EKS.

Implantar uma workload de ampliação de exemplo em um cluster do Modo Automático do Amazon EKS

Neste tutorial, você aprenderá como implantar uma workload de exemplo em um cluster do Modo Automático do EKS e observará como ela provisiona automaticamente os recursos computacionais necessários. Você usará os comandos do `kubectl` para observar o comportamento do cluster e acompanhar de perto como o Modo Automático simplifica as operações do Kubernetes na AWS. Ao final deste tutorial, você saberá como o Modo Automático do EKS responde às implantações de workloads gerenciando automaticamente os recursos computacionais subjacentes, sem exigir a configuração manual do grupo de nós.

Pré-requisitos

- Um cluster do Modo Automático do Amazon EKS. Registre o nome e a região da AWS do cluster.
- Uma entidade principal do IAM, como um usuário ou perfil, com permissões suficientes para gerenciar recursos de rede, de computação e do EKS.
  - Para obter mais informações, consulte a Criar perfis e anexar políticas no Guia do usuário do IAM.
- `aws` CLI instalada e configurada com uma identidade do IAM.
- `kubectl` CLI instalada e conectada ao cluster.
  - Para obter mais informações, consulte Configuração.

Etapa 1: analisar os recursos computacionais existentes (opcional)

Primeiro, use `kubectl` para listar os grupos de nós no cluster.

```
kubectl get nodepools
```

Exemplo de saída:

```
general-purpose
```

Neste tutorial, implantaremos uma workload configurada para usar o grupo de nós `general-purpose`. Esse grupo de nós é incorporado ao Modo Automático do EKS e inclui padrões razoáveis para workloads gerais, como microsserviços e aplicações Web. Você pode criar seu próprio grupo de nós. Para obter mais informações, consulte the section called “Criar grupo de nós”.

Depois, use `kubectl` para listar os nós conectados ao cluster.

```
kubectl get nodes
```

Caso tenha acabado de criar um cluster do Modo Automático do EKS, você não terá nós.

Neste tutorial, você implantará uma workload de exemplo. Se você não tiver nós ou se a workload não se adequar aos nós existentes, o Modo Automático do EKS provisionará um novo nó.

Etapa 2: implantar uma aplicação de exemplo no cluster.

Analise a implantação do Kubernetes a seguir e salve-a como `inflate.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate
  template:
    metadata:
      labels:
        app: inflate
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
      securityContext:
        runAsUser: 1000
        runAsGroup: 3000
        fsGroup: 2000
      containers:
        - name: inflate
          image: public.ecr.aws/eks-distro/kubernetes/pause:3.7
          resources:
            requests:
              cpu: 1
          securityContext:
            allowPrivilegeEscalation: false
```

Observe que o seletor `eks.amazonaws.com/compute-type: auto` requer que a workload seja implantada em um nó do Modo Automático do Amazon EKS.

Aplique a implantação ao cluster.

```
kubectl apply -f inflate.yaml
```

Etapa 3: observar os eventos do Kubernetes

Use o comando a seguir para observar os eventos do Kubernetes, incluindo a criação de um nó. Use `ctrl+c` para parar de observar os eventos.

```
kubectl get events -w --sort-by '.lastTimestamp'
```

Use `kubectl` para listar novamente os nós conectados ao cluster. Registre o nó recém-criado.

```
kubectl get nodes
```

Etapa 4: visualizar nós e instâncias no console da AWS

Você pode visualizar os nós do Modo Automático do EKS no console do EKS e as instâncias associadas do EC2 no console do EC2.

As instâncias do EC2 implantadas pelo Modo Automático do EKS são restritas. Você não pode executar comandos arbitrários nos nós do Modo Automático do EKS.

Etapa 5: excluir a implantação

Use `kubectl` para excluir uma implantação de exemplo

```
kubectl delete -f inflate.yaml
```

Se você não tiver outras workloads implantadas no cluster, o nó criado pelo Modo Automático do EKS ficará vazio.

Na configuração padrão, o Modo Automático do EKS detecta nós que ficaram vazios por trinta segundos e os encerra.

Use `kubectl` ou o console do EC2 para confirmar que a instância associada foi excluída.

Implantação de uma workload de amostra do balanceador de carga no modo automático do EKS

Este guia orienta você na implantação de uma versão em contêiner do jogo 2048 no Amazon EKS, completa com balanceamento de carga e acessibilidade à internet.

Pré-requisitos

- Um cluster do Modo Automático do EKS
- `kubectl` configurado para interagir com o cluster
- Permissões do IAM apropriadas para criar recursos do ALB

Etapa 1: criar o namespace

Primeiro, crie um namespace dedicado para a aplicação do jogo 2048.

Crie um arquivo chamado `01-namespace.yaml`:

```
apiVersion: v1
kind: Namespace
metadata:
  name: game-2048
```

Aplique a configuração do namespace:

```
kubectl apply -f 01-namespace.yaml
```

Etapa 2: implantar uma aplicação

A aplicação executa várias réplicas do contêiner do jogo 2048.

Crie um arquivo chamado `02-deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 5
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: app-2048
          ports:
            - containerPort: 80
          resources:
```

```
            requests:
              cpu: "0.5"
```

> **Note**
>
> Se você receber um erro ao carregar a imagem `public.ecr.aws/l6m2t8p7/docker-2048:latest`, verifique se o perfil do IAM do nó tem permissões suficientes para extrair imagens do ECR. Para obter mais informações, consulte the section called “Perfil do IAM do nó”. A imagem de `docker-2048` no exemplo também é uma imagem de `x86_64` e não será executada em outras arquiteturas.

Principais componentes:

- Implanta cinco réplicas da aplicação
- Usa uma imagem pública do ECR
- Solicita 0,5 núcleo de CPU por pod
- Expõe a porta 80 para o tráfego HTTP

Aplique a implantação:

```
kubectl apply -f 02-deployment.yaml
```

Etapa 3: criar o serviço

O serviço expõe a implantação à rede do cluster.

Crie um arquivo chamado `03-service.yaml`:

```
apiVersion: v1
kind: Service
metadata:
  namespace: game-2048
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
```

```
  selector:
    app.kubernetes.io/name: app-2048
```

Principais componentes:

- Cria um serviço do NodePort
- Mapeia a porta 80 para a porta 80 do contêiner
- Usa o seletor de rótulos para encontrar pods

Aplique o serviço:

```
kubectl apply -f 03-service.yaml
```

Etapa 4: configurar o balanceamento de carga

Você vai configurar uma entrada para expor a aplicação à internet.

Primeiro, crie a `IngressClass`. Crie um arquivo chamado `04-ingressclass.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  labels:
    app.kubernetes.io/name: LoadBalancerController
  name: alb
spec:
  controller: eks.amazonaws.com/alb
```

> **Note**
>
> O Modo Automático do EKS precisa de tags de sub-rede para identificar sub-redes públicas e privadas.
> Se você criou o cluster com o `eksctl`, você já tem essas tags.
> Aprenda como the section called “Marcar sub-redes”.

Em seguida, crie o recurso Ingress. Crie um arquivo chamado `05-ingress.yaml`:

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
metadata:
  namespace: game-2048
  name: ingress-2048
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

Principais componentes:

- Cria um ALB voltado para a internet
- Usa o tipo de destino IP para roteamento direto de pods
- Roteia todo o tráfego (/) para o serviço do jogo

Aplique as configurações de entrada:

```
kubectl apply -f 04-ingressclass.yaml
kubectl apply -f 05-ingress.yaml
```

Etapa 5: verificar a implantação

1. Verifique se todos os pods estão em execução:

   ```
   kubectl get pods -n game-2048
   ```

2. Verifique se o serviço foi criado:

   ```
   kubectl get svc -n game-2048
   ```

3. Obtenha o endpoint do ALB:

```
kubectl get ingress -n game-2048
```

O campo ADDRESS na saída da entrada mostrará o endpoint do ALB. Aguarde de dois a três minutos para que o ALB provisione e registre todos os destinos.

Etapa 6: acessar o jogo

Abra seu navegador da Web e navegue até o URL do endpoint do ALB da etapa anterior. Você deve ver a interface do jogo 2048.

Etapa 7: limpeza

Para remover todos os recursos criados neste tutorial:

```
kubectl delete namespace game-2048
```

Isso excluirá todos os recursos no namespace, incluindo os recursos de implantação, serviço e entrada.

O que está acontece por detrás

1. A implantação cria cinco pods que executam o jogo 2048
2. O serviço fornece acesso de rede estável a esses pods
3. Modo Automático do EKS:
   - Cria um Application Load Balancer na AWS
   - Configura grupos de destino para os pods
   - Configura as regras de roteamento para direcionar o tráfego ao serviço

Solução de problemas

Se o jogo não carregar:

- Certifique-se de que todos os pods estejam em execução: `kubectl get pods -n game-2048`
- Verifique o status de entrada: `kubectl describe ingress -n game-2048`

- Confira as verificações de integridade do ALB: consulte a integridade do grupo de destino no Console da AWS

Implantar uma workload com estado de exemplo no Modo Automático do EKS

Este tutorial o orientará na implantação de uma aplicação com estado de exemplo no cluster do Modo Automático do EKS. A aplicação grava carimbos de data e hora em um volume persistente, demonstrando os recursos automáticos de provisionamento e persistência de volume do EBS do Modo Automático do EKS.

Pré-requisitos

- Um cluster do Modo Automático do EKS
- A AWS CLI configurada com as permissões apropriadas
- `kubectl` instalado e configurado
  - Para ter mais informações, consulte Configuração.

Etapa 1: configurar o ambiente

1. Defina as variáveis de ambiente:

```
export CLUSTER_NAME=my-auto-cluster
export AWS_REGION="us-west-2"
```

2. Atualize o kubeconfig:

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

Etapa 2: criar a classe de armazenamento

A `StorageClass` define como o Modo Automático do EKS provisionará volumes do EBS.

O Modo Automático do EKS não cria uma `StorageClass` para você. Você deve criar uma `StorageClass` referenciando `ebs.csi.eks.amazonaws.com` para usar o recurso de armazenamento do Modo Automático do EKS.

1. Crie um arquivo chamado `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. Aplique a `StorageClass`:

```
kubectl apply -f storage-class.yaml
```

Principais componentes:

- `provisioner: ebs.csi.eks.amazonaws.com`: usa o Modo Automático do EKS
- `volumeBindingMode: WaitForFirstConsumer`: atrasa a criação do volume até que um pod precise dele
- `type: gp3`: especifica o tipo de volume do EBS
- `encrypted: "true"`: o EBS usará a chave `aws/ebs` padrão para criptografar volumes criados com essa classe. Isso é opcional, mas recomendado.
- `storageclass.kubernetes.io/is-default-class: "true"`: o Kubernetes usará essa classe de armazenamento por padrão, a menos que você especifique uma classe de volume diferente em uma reivindicação de volume persistente. Tenha cuidado ao definir esse valor se estiver migrando de outro controlador de armazenamento. (opcional)

Etapa 3: criar a declaração de volume persistente

A PVC solicita armazenamento da `StorageClass`.

1. Crie um arquivo chamado `pvc.yaml`:

```
apiVersion: v1
```

```
kind: PersistentVolumeClaim
metadata:
  name: auto-ebs-claim
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: auto-ebs-sc
  resources:
    requests:
      storage: 8Gi
```

2. Aplique a PVC:

```
kubectl apply -f pvc.yaml
```

Principais componentes:

- `accessModes: ReadWriteOnce`: o volume pode ser montado por um nó por vez
- `storage: 8Gi`: solicita um volume de 8 GiB
- `storageClassName: auto-ebs-sc`: faz referência à `StorageClass` que criamos

Etapa 4: implantar uma aplicação

A implantação executa um contêiner que grava carimbos de data e hora no volume persistente.

1. Crie um arquivo chamado `deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate-stateful
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate-stateful
  template:
    metadata:
      labels:
        app: inflate-stateful
```

```
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
      containers:
        - name: bash
          image: public.ecr.aws/docker/library/bash:4.4
          command: ["/usr/local/bin/bash"]
          args: ["-c", "while true; do echo $(date -u) >> /data/out.txt; sleep 60;
done"]
          resources:
            requests:
              cpu: "1"
          volumeMounts:
            - name: persistent-storage
              mountPath: /data
      volumes:
        - name: persistent-storage
          persistentVolumeClaim:
            claimName: auto-ebs-claim
```

2. Aplique a implantação:

```
kubectl apply -f deployment.yaml
```

Principais componentes:

- Contêiner bash simples que grava carimbos de data e hora em um arquivo
- Monta o PVC em `/data`
- Solicita um núcleo de CPU
- Usa o seletor de nós para nós gerenciados pelo EKS

Etapa 5: verificar a configuração

1. Verifique se o pod está em execução:

```
kubectl get pods -l app=inflate-stateful
```

2. Verifique se a PVC está vinculada:

```
kubectl get pvc auto-ebs-claim
```

3. Verifique o volume do EBS:

```
# Get the PV name
PV_NAME=$(kubectl get pvc auto-ebs-claim -o jsonpath='{.spec.volumeName}')
# Describe the EBS volume
aws ec2 describe-volumes \
  --filters Name=tag:CSIVolumeName,Values=${PV_NAME}
```

4. Verifique se os dados estão sendo gravados:

```
kubectl exec "$(kubectl get pods -l app=inflate-stateful \
  -o=jsonpath='{.items[0].metadata.name}')" -- \
  cat /data/out.txt
```

Etapa 6: limpeza

Execute o comando a seguir para remover todos os recursos criados neste tutorial:

```
# Delete all resources in one command
kubectl delete deployment/inflate-stateful pvc/auto-ebs-claim storageclass/auto-ebs-sc
```

O que está acontece por detrás

1. A PVC solicita armazenamento da `StorageClass`
2. Quando o pod está programado:
   a. O Modo Automático do EKS provisiona um volume do EBS
   b. Cria um PersistentVolume
   c. Anexa o volume ao nó
3. O pod monta o volume e começa a gravar carimbos de data e hora

Controlador de snapshots

O Modo Automático do EKS é compatível com o Kubernetes CSI Snapshotter, também conhecido como controlador de snapshots. No entanto, o Modo Automático do EKS não inclui o controlador de

snapshots. Você é responsável por instalar e configurar o controlador de snapshots. Para ter mais informações, consulte the section called “Controlador de snapshots da CSI”.

Analise a `VolumeSnapshotClass` a seguir que faz referência à capacidade de armazenamento do Modo Automático do EKS.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: auto-ebs-vsclass
driver: ebs.csi.eks.amazonaws.com
deletionPolicy: Delete
```

Saiba mais sobre o Kubernetes CSI Snapshotter.

Implementar uma workload acelerada

Este tutorial demonstra como o modo automático do Amazon EKS simplifica a inicialização de workloads com a aceleração de hardware. O Modo Automático do Amazon EKS simplifica as operações além do próprio cluster, automatizando os principais componentes da infraestrutura e fornecendo recursos de computação, rede, balanceamento de carga, armazenamento e do Identity and Access Management prontos para uso.

O Modo Automático do Amazon EKS inclui os drivers e plug-ins de dispositivos necessários para determinados tipos de instância, como drivers NVIDIA e AWS Neuron. Você não precisa instalar ou atualizar esses componentes.

O Modo Automático do EKS gerencia automaticamente os drivers destes aceleradores:

- AWS Trainium
- AWS Inferentia
- GPUs NVIDIA em instâncias aceleradas do Amazon EC2

> **Note**
>
> O Modo Automático do EKS inclui o plug-in de dispositivo NVIDIA para o Kubernetes. Esse plug-in é executado automaticamente e não fica visível como um conjunto de daemons no cluster.

Suporte adicional de rede:

- Elastic Fabric Adapter (EFA)

O Modo Automático do Amazon EKS elimina o trabalho de gerenciamento de drivers de aceleradores e plug-ins de dispositivos.

Você também pode aproveitar uma redução de custos escalando o cluster para zero. Você pode configurar o Modo Automático do EKS para encerrar instâncias quando nenhuma workload estiver em execução. Essa configuração é útil para workloads de inferência baseadas em lotes.

Veja a seguir um exemplo de como executar workloads aceleradas com o Modo Automático do Amazon EKS.

Pré-requisitos

- Um cluster do Kubernetes com o Modo Automático do Amazon EKS configurado.
- Uma classe de nó `default` do EKS conforme criada quando os grupos de nós gerenciados `general-purpose` ou `system` estão habilitados.

Etapa 1: implantar uma workload de GPU

Neste exemplo, você criará um NodePool para workloads baseadas em NVIDIA que requer 45 GB de memória de GPU. Com o Modo Automático do EKS, você usa as restrições de agendamento do Kubernetes para definir os requisitos de sua instância.

Para implantar o `NodePool` do Modo Automático do Amazon EKS e o exemplo de `workload`, consulte a seguinte definição de NodePool e pod e salve como `nodepool-gpu.yaml` e `pod.yaml`:

nodepool-gpu.yaml

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    budgets:
    - nodes: 10%
    consolidateAfter: 1h
```

```
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
          - g6e
          - g6
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      terminationGracePeriod: 24h0m0s
```

pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  nodeSelector:
    eks.amazonaws.com/compute-type: auto
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
    args:
    - "nvidia-smi"
    resources:
      requests:
        memory: "30Gi"
```

```
        cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        memory: "30Gi"
        nvidia.com/gpu: 1
  tolerations:
  - key: nvidia.com/gpu
    effect: NoSchedule
    operator: Exists
```

Observe que o seletor `eks.amazonaws.com/compute-type: auto` requer que a workload seja implantada em um nó do Modo Automático do Amazon EKS. O NodePool também define uma taint que só permite que pods com tolerâncias para GPUs Nvidia sejam programados.

Aplique o NodePool e a workload ao cluster.

```
kubectl apply -f nodepool-gpu.yaml
kubectl apply -f pod.yaml
```

A seguinte saída deverá ser mostrada:

```
nodepool.karpenter.sh/gpu configured created
pod/nvidia-smi created
```

Aguarde alguns segundos e verifique os nós no cluster. Agora você deve ver um novo nó provisionado no cluster do Modo Automático do Amazon EKS:

```
> kubectl get nodes

NAME        TYPE           CAPACITY    ZONE         NODE                   READY   AGE
gpu-dnknr   g6e.2xlarge    on-demand   us-west-2b   i-02315c7d7643cdee6   True    76s
```

Etapa 2: validar

Você pode ver que o Modo Automático do Amazon EKS executou `g6e.2xlarge` em vez de `g6.2xlarge`, pois a workload exigia uma instância com GPU l40s, de acordo com as seguintes restrições de programação do Kubernetes:

```
...
  nodeSelector:
```

```
    eks.amazonaws.com/instance-gpu-name: l40s
...
    requests:
        memory: "30Gi"
        cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        memory: "30Gi"
        nvidia.com/gpu: 1
```

Agora, veja os logs dos contêineres executando o seguinte comando:

```
kubectl logs nvidia-smi
```

Exemplo de resultado:

```
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI 535.230.02             Driver Version: 535.230.02   CUDA Version: 12.2
 |
|-----------------------------------------+----------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
 |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M.
 |
|                                         |                      |               MIG M.
 |
|=========================================+======================
+======================|
|   0  NVIDIA L40S                    On  | 00000000:30:00.0 Off |                    0
 |
| N/A   27C    P8              23W / 350W |      0MiB / 46068MiB |      0%      Default
 |
|                                         |                      |                  N/A
 |
+-----------------------------------------+----------------------
+----------------------+

+-----------------------------------------------------------------------------------------
+
| Processes:
 |
```

```
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory
 |
|        ID   ID                                                             Usage
 |
|
=============================================================================================|
|  No running processes found
 |
+---------------------------------------------------------------------------------------
+
```

Você pode ver que o contêiner detectou que está sendo executado em uma instância com uma GPU `NVIDIA` e que você não precisou instalar nenhum driver de dispositivo, pois é gerenciado pelo Modo Automático do Amazon EKS.

Etapa 3: limpar

Para remover todos os objetos criados, use o `kubectl` para excluir a implantação da amostra e o NodePool para que o nó seja encerrado:

```
kubectl delete -f nodepool-gpu.yaml
kubectl delete -f pod.yaml
```

Exemplo de referência de NodePools

Criar um NodePool NVIDIA

O seguinte NodePool define:

- Executa somente instâncias da família `g6e` e `g6`
- Consolida os nós quando estiverem vazios por uma hora
  - O valor de 1 hora para `consolodateAfter` é compatível com workloads elevadas e reduz a rotatividade de nós. Você pode ajustar `consolidateAfter` com base em seus requisitos de workload.

Exemplo de NodePool com consolidação e família de instâncias de GPU

```
apiVersion: karpenter.sh/v1
kind: NodePool
```

```
metadata:
  name: gpu
spec:
  disruption:
    budgets:
    - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
          - g6e
          - g6
      terminationGracePeriod: 24h0m0s
```

Em vez de definir `eks.amazonaws.com/instance-gpu-name`, você pode usar `eks.amazonaws.com/instance-family` para especificar a família de instâncias. Para outros rótulos conhecidos que influenciam a revisão da programação, consulte the section called “Rótulos compatíveis com o Modo Automático do EKS”.

Se você tiver requisitos específicos de armazenamento, poderá ajustar `iops`, `size` e `throughput` do armazenamento temporário de nós criando o próprio NodeClass para referência no NodePool. Saiba mais sobre as opções configuráveis do NodeClass.

Exemplo de configuração de armazenamento para NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
```

```
  name: gpu
spec:
  ephemeralStorage:
    iops: 3000
    size: 80Gi
    throughput: 125
```

Definir um AWS Trainium e AWS Inferentia NodePool

O NodePool abaixo tem uma configuração de `eks.amazonaws.com/instance-category` que instrui a executar apenas instâncias da família Inferentia e Trainium:

```
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values:
            - inf
            - trn
```

Configurar as configurações do modo automático do EKS

Este capítulo descreve como configurar aspectos específicos dos clusters do Modo automático do Amazon Elastic Kubernetes Service (EKS). Embora o Modo automático do EKS gerencie a maioria dos componentes da infraestrutura automaticamente, você pode personalizar certos recursos para atender aos requisitos de workloads.

Usando as opções de configuração descritas neste tópico, você pode modificar as configurações de rede, os recursos computacionais e os comportamentos de balanceamento de carga, mantendo os benefícios do gerenciamento automatizado da infraestrutura. Antes de fazer qualquer alteração na configuração, analise as opções disponíveis nas seções a seguir para determinar qual abordagem melhor atende às suas necessidades.

Quais recursos você deseja configurar?	Opção de configuração
Rede e armazenamento de nós • Configurar o posicionamento de nós em sub-redes públicas e privadas • Definir grupos de segurança personalizados para controle de acesso aos nós	the section called “Criar classe de nó”

Quais recursos você deseja configurar?	Opção de configuração
• Personalizar políticas de conversão de endereços de rede (SNAT) • Habilitar políticas de rede do Kubernetes, registro em log e monitoramento detalhados de políticas de rede • Definir parâmetros de armazenamento efêmeros (tamanho, IOPS, throughput) • Configurar o armazenamento efêmero criptografado com chaves do KMS personali zadas • Isole o tráfego de pods em sub-redes separadas dos nós	
Recursos de computação de nós • Selecionar tipos e famílias de instâncias do EC2 específicos • Definir arquiteturas de CPU (x86_64, ARM64) • Configurar tipos de capacidade (sob demanda, spot) • Especificar zonas de disponibilidade • Configurar taints e rótulos de nós • Definir limites de recursos para uso de CPU e memória	the section called “Criar grupo de nós”

Quais recursos você deseja configurar?	Opção de configuração
Grupos de nós de capacidade estática • Manter um número fixo de nós, independentemente da demanda de pods • Configurar capacidade previsível para instâncias reservadas • Definir contagens de réplicas específicas para uma área de ocupação de infraestrutura consistente • Escalar grupos de nós estáticos manualmente • Monitorar o status do grupo de nós de capacidade estática	the section called “Grupos de nós de capacidade estática”
Configurações do Application Load Balancer • Implementar balanceadores de carga internos ou voltados para a internet • Configurar o balanceamento de carga entre zonas • Definir períodos de tempo limite de ociosidade • Habilitar a compatibilidade com HTTP/2 e WebSocket • Configurar parâmetros de verificação de integridade • Especificar configurações de certificado TLS • Definir atributos do grupo de destino • Definir o tipo de endereço IP (IPv4, pilha dupla)	the section called “Criar classe de entrada”

Quais recursos você deseja configurar?	Opção de configuração
Configurações do Network Load Balancer • Configurar o roteamento de IP direto do pod • Habilitar o balanceamento de carga entre zonas • Definir tempo limite de ociosidade da conexão • Configurar parâmetros de verificação de integridade • Especificar o posicionamento da sub-rede • Definir o tipo de endereço IP (IPv4, pilha dupla) • Configurar a preservação do IP de origem do cliente • Definir atributos do grupo de destino	the section called “Create service (Criar serviço)”
Configurações da classe de armazenamento • Definir tipos de volume do EBS (gp3, io1, io2 etc.) • Configurar a criptografia de volume e o uso da chave do KMS • Definir parâmetros de IOPS e throughput • Definir como classe de armazenamento padrão • Definir tags personalizadas para volumes provisionados	the section called “Criar StorageClass”

Quais recursos você deseja configurar?	Opção de configuração
Controlar o uso de ODCRs • Configure a implantação de workloads nas reservas de capacidade sob demanda do EC2 • Selecione explicitamente ODCRs específicas por ID para o uso direcionado da capacidade • Use a seleção baseada em tags para direcionar grupos de ODCRs • Filtre as ODCRs por uma conta de proprieda de da AWS para cenários entre contas • Controle se as workloads usam automatic amente ODCRs abertas	the section called “Controle de reservas de capacidade”

Criar uma classe de nó para o Amazon EKS

As classes de nós do Amazon EKS são modelos que fornecem controle granular sobre a configuração dos nós gerenciados pelo Modo automático do EKS. Uma classe de nó define as configurações no nível de infraestrutura que se aplicam a grupos de nós no cluster do EKS, incluindo configuração de rede, configurações de armazenamento e marcação de recursos. Este tópico explica como criar e configurar uma classe de nó para atender aos requisitos operacionais específicos.

Quando você precisa personalizar como o Modo automático do EKS provisiona e configura instâncias do EC2 além das configurações padrão, a criação de uma classe de nó oferece controle preciso sobre os parâmetros críticos da infraestrutura. Por exemplo, você pode especificar o posicionamento em sub-rede privada para aumentar a segurança, configurar o armazenamento efêmero da instância para workloads sensíveis à performance ou aplicar marcações personalizadas para alocação de custos.

Criar uma classe de nó

Para criar um `NodeClass`, siga estas etapas:

1. Criar um arquivo YAML (por exemplo, `nodeclass.yaml`) com sua configuração de classe de nó
2. Aplicar a configuração ao cluster usando o `kubectl`

3. Faça referência à classe de nó na configuração do grupo de nós. Para obter mais informações, consulte the section called “Criar grupo de nós”.

Você precisa ter o `kubectl` instalado e configurado. Para obter mais informações, consulte Configuração.

Exemplo básico de classe de nó

Veja a seguir um exemplo de classe de nó:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: private-compute
spec:
  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
  ephemeralStorage:
    size: "160Gi"
```

Essa NodeClass aumenta a quantidade de armazenamento efêmero no nó.

Aplique essa configuração usando:

```
kubectl apply -f nodeclass.yaml
```

Em seguida, faça referência à classe de nó na configuração do grupo de nós. Para obter mais informações, consulte the section called “Criar grupo de nós”.

Criar entrada de acesso de classe de nós

Se você criar uma classe de nós personalizada, precisará criar uma entrada de acesso do EKS para permitir que os nós se juntem ao cluster. O EKS cria automaticamente entradas de acesso quando você usa a classe de nós e os pools de nós integrados.

Para obter informações sobre como as entradas de acesso funcionam, consulte the section called "Entradas de acesso".

Ao criar entradas de acesso para as classes de nós do Modo automático do EKS, você precisa usar o tipo de entrada de acesso do `EC2`.

Criar entrada de acesso com a CLI

Para criar uma entrada de acesso para os nós do EC2 e associar a política de nós automáticos do EKS:

Atualize os comandos da CLI a seguir com o nome do cluster e o ARN do perfil do nó. O ARN do perfil de nó é especificado no YAML da classe de nós.

```
# Create the access entry for EC2 nodes
aws eks create-access-entry \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --type EC2

# Associate the auto node policy
aws eks associate-access-policy \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy \
  --access-scope type=cluster
```

Criar entrada de acesso com o CloudFormation

Para criar uma entrada de acesso para os nós do EC2 e associar a política de nós automáticos do EKS:

Atualize o CloudFormation a seguir com o nome do cluster e o ARN do perfil do nó. O ARN do perfil de nó é especificado no YAML da classe de nós.

```
EKSAutoNodeRoleAccessEntry:
  Type: AWS::EKS::AccessEntry
  Properties:
    ClusterName: <cluster-name>
    PrincipalArn: <node-role-arn>
    Type: "EC2"
    AccessPolicies:
      - AccessScope:
```

```
          Type: cluster
        PolicyArn: arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy
  DependsOn: [ <cluster-name> ] # previously defined in CloudFormation
```

Para obter informações sobre a implantação de pilhas do CloudFormation, consulte Conceitos básicos do CloudFormation

Especificação das classes de nós

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Required fields

  # role and instanceProfile are mutually exclusive fields.
  role: MyNodeRole  # IAM role for EC2 instances
  # NOTE: instance profile names must start with the 'eks' prefix. this is a
  # temporary limitation to be lifted soon.
  # instanceProfile: eks-MyNodeInstanceProfile  # IAM instance-profile for EC2
 instances

  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
    # Alternative using direct subnet ID
    # - id: "subnet-0123456789abcdef0"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
    # Alternative approaches:
    # - id: "sg-0123456789abcdef0"
    # - name: "eks-cluster-security-group"

  # Optional: Pod subnet selector for advanced networking
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"
    # Alternative using direct subnet ID
```

```
    # - id: "subnet-0987654321fedcba0"
# must include Pod security group selector also
 podSecurityGroupSelectorTerms:
  - tags:
      Name: "eks-pod-sg"
    # Alternative using direct security group ID
    # - id: "sg-0123456789abcdef0"

  # Optional: Selects on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        Name: "targeted-odcr"
      # Optional owning account ID filter
      owner: "012345678901"

  # Optional fields
  snatPolicy: Random  # or Disabled

  networkPolicy: DefaultAllow  # or DefaultDeny
  networkPolicyEventLogs: Disabled  # or Enabled

  ephemeralStorage:
    size: "80Gi"    # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000      # Range: 3000-16000
    throughput: 125 # Range: 125-1000
    # Optional KMS key for encryption
    kmsKeyID: "arn:aws:kms:region:account:key/key-id"
    # Accepted formats:
    # KMS Key ID
    # KMS Key ARN
    # Key Alias Name
    # Key Alias ARN

  advancedNetworking:
    # Optional: Controls whether public IP addresses are assigned to instances that are
 launched with the nodeclass.
    # If not set, defaults to the MapPublicIpOnLaunch setting on the subnet.
    associatePublicIPAddress: false

    # Optional: Forward proxy, commonly requires certificateBundles as well
```

```
    # for EC2, see https://repost.aws/knowledge-center/eks-http-proxy-containerd-
automation
    httpsProxy: http://192.0.2.4:3128 #commonly port 3128 (Squid) or 8080 (NGINX) #Max
 255 characters
    #httpsProxy: http://[2001:db8::4]:3128 # IPv6 address with port, use []
    noProxy: #Max 50 entries
        - localhost #Max 255 characters each
        - 127.0.0.1
        #- ::1 # IPv6 localhost
        #- 0:0:0:0:0:0:0:1 # IPv6 localhost
        - 169.254.169.254 # EC2 Instance Metadata Service
        #- [fd00:ec2::254] # IPv6 EC2 Instance Metadata Service
        # Domains to exclude, put all VPC endpoints here
        - .internal
        - .eks.amazonaws.com

  advancedSecurity:
    # Optional, US regions only: Specifying `fips: true` will cause nodes in the
 nodeclass to run FIPS compatible AMIs.
    fips: false

  # Optional: Custom certificate bundles.
  certificateBundles:
    - name: "custom-cert"
      data: "base64-encoded-cert-data"

  # Optional: Additional EC2 tags (with restrictions)
  tags:
    Environment: "production"
    Team: "platform"
    # Note: Cannot use restricted tags like:
    # - kubernetes.io/cluster/*
    # - karpenter.sh/provisioner-name
    # - karpenter.sh/nodepool
    # - karpenter.sh/nodeclaim
    # - karpenter.sh/managed-by
    # - eks.amazonaws.com/nodeclass
```

Considerações

- Se você quiser verificar a quantidade de armazenamento local de uma instância, é possível descrever o nó para ver o recurso de armazenamento efêmero.

- Criptografia de volume: o EKS usa a chave do KMS personalizada configurada para criptografar o volume raiz somente leitura da instância e o volume de dados de leitura/gravação.
- Substituir o perfil do IAM do nó: caso altere o perfil do IAM do nó associado a um `NodeClass`, você precisará criar uma nova entrada de acesso. O EKS cria automaticamente uma entrada de acesso para o perfil do IAM do nó durante a criação do cluster. O perfil do IAM do nó requer a política de acesso `AmazonEKSAutoNodePolicy` do EKS. Para obter mais informações, consulte the section called “Entradas de acesso”.
- Densidade máxima dos pods: o EKS limita o número máximo de pods em um nó a 110. Esse limite é aplicado após o cálculo do máximo de pods existentes. Para obter mais informações, consulte the section called “Tipos de instância do Amazon EC2”.
- Tags: caso queira propagar tags do Kubernetes para o EC2, você precisará configurar permissões adicionais do IAM. Para obter mais informações, consulte the section called “Identidade e acesso”.
- Classe de nó padrão: não nomeie sua classe de nó personalizada `default`. Isso ocorre porque o Modo automático do EKS inclui um `NodeClass` denominado `default` que é provisionado automaticamente quando você habilita pelo menos um `NodePool` integrado. Para obter informações sobre como habilitar `NodePools` integrados, consulte the section called “Analisar grupos de nós integrados”.
- Comportamento dos **`subnetSelectorTerms`** com várias sub-redes: se houver várias sub-redes que correspondam às condições dos `subnetSelectorTerms` ou que você forneça por ID, o Modo automático do EKS criará nós distribuídos pelas sub-redes.
  - Se as sub-redes estiverem em zonas de disponibilidade (AZ) diferentes, você poderá usar recursos do Kubernetes, como Restrições de distribuição da topologia de pods e Roteamento com reconhecimento de topologia, para distribuir os pods e o tráfego pelas zonas, respectivamente.
  - Se houver várias sub-redes na mesma AZ que correspondam aos `subnetSelectorTerms`, o Modo automático do EKS criará pods em cada nó distribuídos pelas sub-redes nessa AZ. O Modo automático do EKS cria interfaces de rede secundárias em cada nó nas outras sub-redes na mesma AZ. Ele escolhe com base no número de endereços IP disponíveis em cada sub-rede para usar as sub-redes com mais eficiência. No entanto, você não pode especificar qual sub-rede o Modo automático do EKS usará para cada pod; se você precisar que os pods sejam executados em sub-redes específicas, então use o the section called “Seleção de sub-rede para pods”.

Seleção de sub-rede para pods

Os campos `podSubnetSelectorTerms` e `podSecurityGroupSelectorTerms` permitem configurações avançadas de rede, possibilitando que os pods sejam executados em sub-redes diferentes de seus nós. Essa separação fornece controle aprimorado do roteamento do tráfego de rede e das políticas de segurança. Observe que os `podSecurityGroupSelectorTerms` são obrigatórios com os `podSubnetSelectorTerms`.

Casos de uso

Use os `podSubnetSelectorTerms` quando precisar:

- Separar o tráfego de infraestrutura (comunicação de nó a nó) do tráfego de aplicações (comunicação de pod a pod)
- Aplicar configurações de rede diferentes às sub-redes de nós e às sub-redes de pods.
- Implementar diferentes políticas de segurança ou regras de roteamento para nós e pods.
- Configurar proxies reversos ou filtragem de rede especificamente para o tráfego de nós sem afetar o tráfego de pods. Use `advancedNetworking` e `certificateBundles` para definir seu proxy reverso e quaisquer certificados autoassinados ou privados para o proxy.

Exemplo de configuração

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  # Subnets for EC2 instances (nodes)
  subnetSelectorTerms:
    - tags:
        Name: "node-subnet"
        kubernetes.io/role/internal-elb: "1"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"

  # Separate subnets for Pods
```

```
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"

  podSecurityGroupSelectorTerms:
  - tags:
      Name: "eks-pod-sg"
```

Considerações sobre seletores de sub-rede para pods

- Densidade reduzida de pods: menos pods podem ser executados em cada nó ao usar os `podSubnetSelectorTerms`, porque a interface de rede primária do nó está na sub-rede de nós e não pode ser usada para pods na sub-rede de pods.
- Limitações do seletor de sub-rede: as configurações padrão de `subnetSelectorTerms` e `securityGroupSelectorTerms` não se aplicam à seleção de sub-rede de pods.
- Planejamento de rede: garanta um espaço de endereços IP adequado nas sub-redes de nós e de pods para ser compatível com os seus requisitos de workload.
- Configuração de roteamento: verifique se a tabela de rotas e a lista de controle de acesso (ACL) de rede das sub-redes de pods estão configuradas corretamente para comunicação entre as sub-redes de nós e de pods.
- Zonas de disponibilidade: verifique se você criou sub-redes de pod em várias AZs. Se você estiver usando uma sub-rede específica de pod, ela deve estar na mesma AZ que a sub-rede do nó.

Criar um grupo de nós para o Modo Automático do EKS

Os grupos de nós do Amazon EKS oferecem uma forma flexível de gerenciar recursos computacionais no cluster do Kubernetes. Este tópico demonstra como criar e configurar grupos de nós usando o Karpenter, uma ferramenta de provisionamento de nós que ajuda a otimizar a escalabilidade de clusters e a utilização de recursos. Com o recurso NodePool do Karpenter, você pode definir requisitos específicos para os recursos computacionais, incluindo tipos de instância, zonas de disponibilidade, arquiteturas e tipos de capacidade.

Você não pode modificar os grupos de nós integrados `system` e `general-purpose`. Você não pode habilitá-los nem desabilitá-los. Para obter mais informações, consulte the section called “Analisar grupos de nós integrados”.

A especificação NodePool permite um controle refinado sobre os recursos computacionais do cluster do EKS por meio de vários rótulos e requisitos compatíveis. Isso inclui opções para especificar categorias de instâncias do EC2, configurações de CPU, zonas de disponibilidade, arquiteturas (ARM64/AMD64) e tipos de capacidade (spot ou sob demanda). Você também pode definir limites de recursos para o uso de CPU e memória, garantindo que o cluster permaneça dentro dos limites operacionais desejados.

O Modo Automático do EKS utiliza rótulos conhecidos do Kubernetes para fornecer maneiras consistentes e padronizadas de identificar as características dos nós. Esses rótulos, como `topology.kubernetes.io/zone` para zonas de disponibilidade e `kubernetes.io/arch` para arquitetura de CPU, seguem as convenções estabelecidas do Kubernetes. Além disso, rótulos específicos do EKS (prefixados com `eks.amazonaws.com/`) ampliam essa funcionalidade com atributos específicos da AWS, como tipos de instância, fabricantes de CPU, recursos de GPU e especificações de rede. Esse sistema de rotulagem padronizado permite uma integração perfeita com as ferramentas existentes do Kubernetes, ao mesmo tempo em que fornece uma integração profunda da infraestrutura da AWS.

Criar um NodePool

Siga estas etapas para criar um NodePool para o cluster do Amazon EKS:

1. Crie um arquivo YAML denominado `nodepool.yaml` com a configuração necessária do NodePool. Você pode usar o exemplo de configuração abaixo.
2. Aplique o NodePool ao cluster:

```
kubectl apply -f nodepool.yaml
```

3. Verifique se o NodePool foi criado com êxito:

```
kubectl get nodepools
```

4. (Opcional) Monitore o status do NodePool:

```
kubectl describe nodepool default
```

Certifique-se de que o NodePool faça referência a um NodeClass válido que exista no cluster. O NodeClass define as configurações específicas da AWS para os recursos computacionais. Para obter mais informações, consulte the section called “Criar classe de nó”.

Exemplo de NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: my-node-pool
spec:
  template:
    metadata:
      labels:
        billing-team: my-team
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
          values: ["4", "8", "16", "32"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["arm64", "amd64"]

  limits:
    cpu: "1000"
    memory: 1000Gi
```

Rótulos compatíveis com o Modo Automático do EKS

O Modo Automático do EKS é compatível com os rótulos conhecidos a seguir.

> **Note**
>
> O Modo Automático do EKS usa rótulos diferentes do Karpenter. Os rótulos relacionados às instâncias gerenciadas pelo EC2 começam com `eks.amazonaws.com`.

Rótulo	Exemplo	Descrição
topology.kubernetes.io/zone	us-east-2a	Região AWS
node.kubernetes.io/instance-type	g4dn.8xlarge	Tipo de instância AWS
kubernetes.io/arch	amd64	As arquiteturas são definidas pelos valores GOARCH na instância
karpenter.sh/capacity-type	spot	Os tipos de capacidade incluem `spot`, `on-demand`
eks.amazonaws.com/instance-hypervisor	nitro	Tipos de instância que usam um hipervisor específico
eks.amazonaws.com/compute-type	auto	Identifica os nós gerenciados pelo Modo Automático do EKS
eks.amazonaws.com/instance-encryption-in-transit-supported	true	Tipos de instância que são compatíveis (ou não) com a criptografia em trânsito
eks.amazonaws.com/instance-category	g	Tipos de instância da mesma categoria , geralmente a string antes do número de geração
eks.amazonaws.com/instance-generation	4	Número de geração do tipo de instância em uma categoria de instância
eks.amazonaws.com/instance-family	g4dn	Tipos de instância de propriedades semelhantes, mas com quantidades de recursos diferentes

Rótulo	Exemplo	Descrição
eks.amazonaws.com/instance-size	8xlarge	Tipos de instância com quantidades de recursos semelhantes, mas com propriedades diferentes
eks.amazonaws.com/instance-cpu	32	Número de CPUs na instância
eks.amazonaws.com/instance-cpu-manufacturer	`aws`	Nome do fabricante da CPU
eks.amazonaws.com/instance-memory	131072	Número de mebibytes de memória na instância
eks.amazonaws.com/instance-ebs-bandwidth	9500	Número de megabits máximos do EBS disponíveis na instância
eks.amazonaws.com/instance-network-bandwidth	131072	Número de megabits de linha de base disponíveis na instância
eks.amazonaws.com/instance-gpu-name	t4	Nome da GPU na instância, se disponível
eks.amazonaws.com/instance-gpu-manufacturer	nvidia	Nome do fabricante da GPU
eks.amazonaws.com/instance-gpu-count	1	Número de GPUs na instância
eks.amazonaws.com/instance-gpu-memory	16384	Número de mebibytes de memória na GPU
eks.amazonaws.com/instance-local-nvme	900	Número de gibibytes de armazenamento NVMe local na instância

> **Note**
>
> O Modo Automático do EKS é compatível apenas com determinadas instâncias e tem requisitos mínimos de tamanho. Para obter mais informações, consulte the section called "Referência de instâncias compatíveis com o Modo automático do EKS".

Rótulos não compatíveis com o Modo Automático do EKS

O Modo Automático do EKS não é compatível com os rótulos a seguir.

- O Modo Automático do EKS é compatível apenas com o Linux
  - `node.kubernetes.io/windows-build`
  - `kubernetes.io/os`

Desabilitar pools de nós integrados

Se você criar pools de nós personalizados, poderá desabilitar os pools de nós integrados. Para obter mais informações, consulte the section called "Analisar grupos de nós integrados".

Cluster sem pools de nós integrados

Você pode criar um cluster sem os pools de nós integrados. Isso é útil quando sua organização criou pools de nós personalizados.

> **Note**
>
> Quando você cria um cluster sem grupos de nós integrados, o NodeClass `default` não é provisionado automaticamente. Você precisará criar um NodeClass personalizado. Para obter mais informações, consulte the section called "Criar classe de nó".

Visão geral:

1. Crie um cluster do EKS com os valores de `nodePools` e `nodeRoleArn` vazios.
   - Exemplo de `autoModeConfig` eksctl:

```
autoModeConfig:
```

```
  enabled: true
  nodePools: []
  # Do not set a nodeRoleARN
```

Para obter mais informações, consulte the section called “CLI do eksctl”.

2. Crie uma classe de nós personalizada com um ARN de perfil de nó
   - Para obter mais informações, consulte the section called “Criar classe de nó”.
3. Crie uma entrada de acesso para a classe de nós personalizada
   - Para obter mais informações, consulte the section called “Criar entrada de acesso de classe de nós”.
4. Crie um pool de nós personalizado, conforme descrito acima.

Interrupção

Você pode configurar o Modo Automático do EKS para interromper os nós por meio do seu NodePool de várias maneiras. Você pode usar `spec.disruption.consolidationPolicy`, `spec.disruption.consolidateAfter` ou `spec.template.spec.expireAfter`. Você também pode limitar a taxa de interrupção do Modo Automático do EKS por meio de `spec.disruption.budgets` do NodePool. Você também pode controlar as janelas de tempo e o número de nós interrompidos simultaneamente. Para obter instruções sobre como configurar esse comportamento, consulte Interrupção na documentação do Karpenter.

Você pode configurar a interrupção de pools de nós para:

- Identifique quando as instâncias estão subutilizadas e consolide as workloads.
- Crie um orçamento de interrupção de pools de nós para limitar a taxa de encerramentos de nós devido à oscilação, a um vazio e à consolidação.

Por padrão, o Modo Automático do EKS:

- Consolida instâncias subutilizadas.
- Encerra as instâncias após 336 horas.
- Define um único orçamento de interrupção de 10% dos nós.
- Permite que os nós sejam substituídos devido ao desvio quando uma nova AMI do Modo Automático é lançada, o que ocorre aproximadamente uma vez por semana.

Término do período de carência

Quando o `terminationGracePeriod` não está explicitamente definido em um NodePool automático do EKS, o sistema aplica automaticamente um período de carência padrão com término em 24 horas ao NodeClaim associado. Embora os clientes do Modo Automático do EKS não vejam um padrão de `terminationGracePeriod` em suas configurações personalizadas do NodePool, eles constatarão esse valor padrão no NodeClaim. A funcionalidade permanece consistente, independentemente de o período de carência ser definido explicitamente no NodePool ou ser padronizado no NodeClaim, garantindo um comportamento previsível de encerramento do nó em todo o cluster.

Grupos de nós de capacidade estática no modo automático do EKS

O modo automático do Amazon EKS oferece suporte a grupos de nós de capacidade estática que mantêm um número fixo de nós, independentemente da demanda de pods. Os grupos de nós com capacidade estática são úteis para workloads que demandam previsibilidade de capacidade, instâncias reservadas ou requisitos de conformidade específicos nos quais é necessário manter uma área de ocupação de infraestrutura consistente.

Ao contrário dos grupos de nós dinâmicos que escalam com base nas demandas de agendamento de pods, os grupos de nós de capacidade estática mantêm o número de nós que você configurou.

Exemplo básico

A seguir, apresentamos um exemplo de um grupo de nós simples com capacidade estática que mantém cinco nós:

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: my-static-nodepool
spec:
  replicas: 5  # Maintain exactly 5 nodes

  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
```

```
    requirements:
      - key: "eks.amazonaws.com/instance-category"
        operator: In
        values: ["m", "c"]
      - key: "topology.kubernetes.io/zone"
        operator: In
        values: ["us-west-2a", "us-west-2b"]

limits:
  nodes: 8
```

Configuração de um grupo de nós de capacidade estática

Para criar um grupo de nós de capacidade estática, defina o campo `replicas` em sua especificação de NodePool. O campo `replicas` define o número exato de nós que o grupo de nós manterá.

Restrições de grupos de nós de capacidade estática

Os grupos de nós de capacidade estática apresentam vários comportamentos e restrições importantes:

Restrições de configuração:

- Não é possível alternar modos: depois de definir o campo `replicas` em um grupo de nós, você não pode removê-lo. O grupo de nós não pode alternar entre os modos estático e dinâmico.
- Limites de recursos limitados: há suporte apenas para o campo `limits.nodes` na seção de limites. Os limites de CPU e de memória não são aplicáveis.
- Ausência do campo destinado ao peso: não é possível configurar o campo `weight` em grupos de nós com capacidade estática, já que a seleção de nós não ocorre por prioridade.

Comportamento operacional:

- Sem consolidação: os nós presentes em grupos de nós de capacidade estática não são considerados para consolidação com base na utilização.
- Operações de escalabilidade: as operações de escalabilidade ignoram os orçamentos de interrupção de nós, mas continuam respeitando os PodDisruptionBudgets.
- Substituição de nós: os nós ainda são substituídos por desvio (como atualizações de AMI) e expiração com base na sua configuração.

Escalabilidade de um grupo de nós de capacidade estática

É possível alterar o número de réplicas em um grupo de nós de capacidade estática usando o comando `kubectl scale`:

```
# Scale down to 5 nodes
kubectl scale nodepool static-nodepool --replicas=5
```

Ao reduzir a escala verticalmente, o modo automático do EKS encerra os nós de maneira controlada, respeitando os PodDisruptionBudgets e possibilitando o reagendamento dos pods em execução nos nós remanescentes.

Monitoramento dos grupos de nós de capacidade estática

Use os seguintes comandos para monitorar os grupos de nós de capacidade estática:

```
# View node pool status
kubectl get nodepool static-nodepool

# Get detailed information including current node count
kubectl describe nodepool static-nodepool

# Check the current number of nodes
kubectl get nodepool static-nodepool -o jsonpath='{.status.nodes}'
```

O campo `status.nodes` mostra o número atual de nós gerenciados pelo grupo de nós, que deve corresponder à sua contagem de `replicas` desejada em condições normais.

Exemplos de configuração

Grupo de nós de capacidade estática básico

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: basic-static
spec:
  replicas: 5

  template:
    spec:
```

```
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["m"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a"]

  limits:
    nodes: 8  # Allow scaling up to 8 during operations
```

Capacidade estática com tipos de instância específicos

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: reserved-instances
spec:
  replicas: 20

  template:
    metadata:
      labels:
        instance-type: reserved
        cost-center: production
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "node.kubernetes.io/instance-type"
          operator: In
          values: ["m5.2xlarge"]  # Specific instance type
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
```

```
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b", "us-west-2c"]

  limits:
    nodes: 25

  disruption:
    # Conservative disruption for production workloads
    budgets:
      - nodes: 10%
```

Grupo de nós de capacidade estática em diversas zonas

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: multi-zone-static
spec:
  replicas: 12  # Will be distributed across specified zones

  template:
    metadata:
      labels:
        availability: high
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
          values: ["8", "16"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b", "us-west-2c"]
        - key: "karpenter.sh/capacity-type"
          operator: In
```

```
          values: ["on-demand"]

  limits:
    nodes: 15

  disruption:
    budgets:
      - nodes: 25%
```

Práticas recomendadas

Planejamento de capacidade:

- Defina `limits.nodes` como um valor superior a `replicas` para permitir a escalabilidade temporária durante as operações de substituição de nós.
- Considere a capacidade máxima necessária durante o desvio de nós ou as atualizações de AMI ao definir os limites.

Seleção de instâncias:

- Use tipos de instância específicos quando você tiver instâncias reservadas ou requisitos de hardware específicos.
- Evite requisitos excessivamente restritivos que possam limitar a disponibilidade de instâncias durante a escalabilidade.

Gerenciamento de interrupções:

- Configure orçamentos de interrupção apropriados para equilibrar a disponibilidade com as operações de manutenção.
- Considere a tolerância da aplicação para a substituição de nós ao definir as porcentagens do orçamento.

Como monitorar o:

- Monitore regularmente o campo `status.nodes` para garantir que sua capacidade desejada seja mantida.
- Configure alertas para quando a contagem real de nós divergir das réplicas desejadas.

Distribuição de zonas:

- Para obter alta disponibilidade, distribua a capacidade estática por várias zonas de disponibilidade.
- Quando você cria um grupo de nós de capacidade estática que abrange várias zonas de disponibilidade, o modo automático do EKS distribui os nós entre as zonas especificadas, mas não há garantia de que a distribuição será uniforme.
- Para obter uma distribuição previsível e uniforme entre as zonas de disponibilidade, crie grupos de nós de capacidade estática separados, cada um fixado a uma zona de disponibilidade específica usando o requisito `topology.kubernetes.io/zone`.
- Se você precisar de 12 nós distribuídos uniformemente em três zonas, crie três grupos de nós com quatro réplicas cada, em vez de um único grupo de nós com 12 réplicas abrangendo as três zonas.

Solução de problemas

Nós não atingem o número de réplicas desejado:

- Verifique se o valor de `limits.nodes` é suficiente
- Verifique se seus requisitos não estão restringindo excessivamente a seleção de instâncias
- Revise as cotas de serviço da AWS para os tipos de instância e regiões que você está usando

Substituição de nós demorando muito:

- Ajuste os orçamentos de interrupção para permitir mais substituições simultâneas
- Verifique se os PodDisruptionBudgets estão impedindo o encerramento dos nós

Encerramento inesperado de nós:

- Analise as configurações `expireAfter` e `terminationGracePeriod`
- Verifique se houve encerramentos manuais de nós ou eventos de manutenção da AWS

Criar uma IngressClass para configurar um Application Load Balancer

O Modo automático do EKS automatiza tarefas de rotina de balanceamento de carga, incluindo a exposição de aplicações de cluster à internet.

A AWS sugere o uso de Application Load Balancers (ALB) para fornecer tráfego HTTP e HTTPS. Os Application Load Balancers podem rotear solicitações com base no conteúdo da solicitação. Para obter mais informações sobre Application Load Balancers, consulte What is Elastic Load Balancing?.

O Modo automático do EKS cria e configura Application Load Balancers (ALBs). Por exemplo, o Modo automático do EKS cria um balanceador de carga quando você cria um objeto `Ingress` do Kubernetes e o configura para rotear o tráfego para a workload do cluster.

Visão geral

1. Crie uma workload que você deseja expor à internet.
2. Crie um recurso `IngressClassParams`, especificando valores de configuração da AWS específicos, como o certificado a ser usado para sub-redes SSL/TLS e VPC.
3. Crie um recurso `IngressClass`, especificando que o Modo automático do EKS será o controlador do recurso.
4. Crie um recurso `Ingress` que associe um caminho HTTP e uma porta a uma workload do cluster.

O Modo automático do EKS criará um Application Load Balancer que aponte para a workload específica no recurso `Ingress`, usando a configuração do balanceador de carga específica no recurso `IngressClassParams`.

Pré-requisitos

- Modo automático do EKS habilitado em um cluster do Amazon EKS
- Kubectl configurado para se conectar ao cluster
  - Você pode usar `kubectl apply -f <filename>` para aplicar os arquivos YAML de configuração de exemplo abaixo ao cluster.

> **Note**
>
> O Modo automático do EKS precisa de tags de sub-rede para identificar sub-redes públicas e privadas.
> Se você criou o cluster com o `eksctl`, você já tem essas tags.
> Saiba como the section called “Marcar sub-redes”.

Etapa 1: criar uma workload

Para começar, crie uma workload que você deseja expor à internet. Pode ser qualquer recurso do Kubernetes que envia tráfego HTTP, como um Deployment e um Service.

Este exemplo usa um serviço HTTP simples, chamado `service-2048`, que atua como receptor na porta `80`. Crie este serviço e a sua implantação aplicando o seguinte manifesto, `2048-deployment-service.yaml`:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 2
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: app-2048
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: NodePort
  selector:
    app.kubernetes.io/name: app-2048
```

Aplicar a configuração ao cluster:

```
kubectl apply -f 2048-deployment-service.yaml
```

Os recursos listados acima serão criados no namespace padrão. Você pode verificar isso executando o seguinte comando:

```
kubectl get all -n default
```

Etapa 2: criar IngressClassParams

Crie um objeto `IngressClassParams` para especificar as opções de configuração da AWS específicas do Application Load Balancer. Neste exemplo, criamos um recurso `IngressClassParams` denominado `alb` (que você usará na próxima etapa) que especifica o esquema do balanceador de carga como `internet-facing` em um arquivo nomeado como `alb-ingressclassparams.yaml`.

```
apiVersion: eks.amazonaws.com/v1
kind: IngressClassParams
metadata:
  name: alb
spec:
  scheme: internet-facing
```

Aplicar a configuração ao cluster:

```
kubectl apply -f alb-ingressclassparams.yaml
```

Etapa 3: criar IngressClass

Crie um `IngressClass` que faça referência aos valores de configuração específicos da AWS definidos no recurso `IngressClassParams` em um arquivo nomeado como `alb-ingressclass.yaml`. Registre o nome da `IngressClass`. Neste exemplo, tanto `IngressClass` como `IngressClassParams` são denominados `alb`.

Use a anotação `is-default-class` para controlar se os recursos `Ingress` devem usar essa classe por padrão.

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
```

```
metadata:
  name: alb
  annotations:
    # Use this annotation to set an IngressClass as Default
    # If an Ingress doesn't specify a class, it will use the Default
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  # Configures the IngressClass to use EKS Auto Mode
  controller: eks.amazonaws.com/alb
  parameters:
    apiGroup: eks.amazonaws.com
    kind: IngressClassParams
    # Use the name of the IngressClassParams set in the previous step
    name: alb
```

Para obter mais informações sobre as opções de configuração, consulte the section called “Referência IngressClassParams”.

Aplicar a configuração ao cluster:

```
kubectl apply -f alb-ingressclass.yaml
```

Etapa 4: criar Ingress

Crie um recurso `Ingress` em um arquivo nomeado como `alb-ingress.yaml`. O objetivo desse recurso é associar caminhos e portas no Application Load Balancer às workloads no cluster. Neste exemplo, criamos um recurso `Ingress` denominado `2048-ingress` que roteia o tráfego para um serviço denominado `service-2048` na porta 80.

Para obter mais informações sobre como configurar esse recurso, consulte Ingress na documentação do Kubernetes.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: 2048-ingress
spec:
  # this matches the name of IngressClass.
  # this can be omitted if you have a default ingressClass in cluster: the one with
 ingressclass.kubernetes.io/is-default-class: "true"  annotation
  ingressClassName: alb
  rules:
```

```
    - http:
        paths:
          - path: /*
            pathType: ImplementationSpecific
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

Aplicar a configuração ao cluster:

```
kubectl apply -f alb-ingress.yaml
```

Etapa 5: verificar o status

Use `kubectl` para encontrar o status de `Ingress`. Pode levar alguns minutos para que o balanceador de carga esteja disponível.

Use o nome do recurso `Ingress` que você definiu na etapa anterior. Por exemplo:

```
kubectl get ingress 2048-ingress
```

Quando o recurso estiver pronto, recupere o nome de domínio do balanceador de carga.

```
kubectl get ingress 2048-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}'
```

Para visualizar o serviço em um navegador da Web, analise a porta e o caminho especificados no recurso `Ingress`.

Etapa 6: limpeza

Para limpar o balanceador de carga, use o seguinte comando:

```
kubectl delete ingress 2048-ingress
kubectl delete ingressclass alb
kubectl delete ingressclassparams alb
```

O Modo automático do EKS excluirá automaticamente o balanceador de carga associado na conta da AWS.

Referência IngressClassParams

A tabela abaixo é uma referência rápida das opções de configuração mais usadas.

Campo	Descrição	Valor de exemplo
`scheme`	Define se o ALB é interno ou voltado para a internet	`internet-facing`
`namespaceSelector`	Restringe quais namespaces podem usar esta IngressClass	`environment: prod`
`group.name`	Agrupa várias entradas para compartilhar um único ALB	`retail-apps`
`ipAddressType`	Define o tipo de endereço IP para o ALB	`dualstack`
`subnets.ids`	Lista de IDs de sub-rede para implantação do ALB	`subnet-xxxx, subnet-yyyy`
`subnets.tags`	Filtros de tag para selecionar sub-redes do ALB	`Environment: prod`
`certificateARNs`	ARNs de certificados SSL para usar	`arn:aws:acm:region:account:certificate/id`
`tags`	Tags personalizadas de recursos da AWS	`Environment: prod, Team: platform`
`loadBalancerAttributes`	Atributos específicos do balanceador de carga	`idle_timeout.timeout_seconds: 60`

Considerações

- Você não pode usar Anotações em uma IngressClass para configurar balanceadores de carga com o Modo automático do EKS.
- Você não pode definir ListenerAttribute com o modo automático do EKS.

- Você deve atualizar o perfil do IAM do cluster para permitir a propagação de tags do Kubernetes para os recursos do AWS Load Balancer. Para obter mais informações, consulte the section called “Tags da AWS personalizadas para recursos do Modo Automático do EKS.”.
- Para obter informações sobre como associar recursos ao Modo automático do EKS ou ao AWS Load Balancer Controller autogerenciado, consulte the section called “Referência de migração”.
- Para obter informações sobre como corrigir problemas com balanceadores de carga, consulte the section called “Solução de problemas”.
- Para obter mais considerações sobre o uso do recurso de balanceamento de carga do Modo automático do EKS, consulte the section called “Balanceamento de carga”.

As tabelas a seguir fornecem uma comparação detalhada das alterações nas configurações de IngressClassParams, anotações de Ingress e TargetGroupBinding do Modo automático do EKS. Essas tabelas destacam as principais diferenças entre a capacidade de balanceamento de carga do Modo automático do EKS e o controlador de balanceador de carga de código aberto, incluindo alterações na versão da API, recursos obsoletos e nomes de parâmetros atualizados.

IngressClassParams

Anterior	Novo	Descrição
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Alteração de versão da API
`spec.certificateArn`	`spec.certificateARNs`	Compatibilidade com vários ARNs de certificados
`spec.subnets.tags`	`spec.subnets.matchTags`	Esquema de correspondência de sub-rede alterado
`spec.listeners.listenerAttributes`	Não compatível	Ainda não há suporte para o modo automático do EKS

Anotações de Ingress

Anterior	Novo	Descrição
`kubernetes.io/ingress.class`	Não compatível	Use `spec.ingressClassName` em objetos Ingress
`alb.ingress.kubernetes.io/group.name`	Não compatível	Especifique grupos somente em IngressClass
`alb.ingress.kubernetes.io/waf-acl-id`	Não compatível	Use WAF v2 em vez disso
`alb.ingress.kubernetes.io/web-acl-id`	Não compatível	Use WAF v2 em vez disso
`alb.ingress.kubernetes.io/shield-advanced-protection`	Não compatível	Integração com Shield desabilitada
`alb.ingress.kubernetes.io/auth-type: oidc`	Não compatível	No momento, o tipo OIDC Auth não é compatível.

TargetGroupBinding

Anterior	Novo	Descrição
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Alteração de versão da API
`spec.targetType` opcional	`spec.targetType` obrigatório	Especificação explícita do tipo de destino
`spec.networking.ingress.from`	Não compatível	Não oferece mais compatibilidade com NLB sem grupos de segurança

Para usar o recurso personalizado TargetGroupBinding, é necessário marcar o grupo de destino com a etiqueta “eks:eks-cluster-name” contendo o nome do cluster para conceder ao controlador as permissões do IAM necessárias. Esteja ciente de que o controlador excluirá o grupo de destino quando o recurso TargetGroupBinding ou o cluster for excluído.

Usar anotações de serviço para configurar Network Load Balancers

Saiba como configurar Network Load Balancers (NLB) no Amazon EKS usando anotações de serviço do Kubernetes. Este tópico explica as anotações compatíveis com o Modo Automático do EKS para personalizar o comportamento do NLB, incluindo acessibilidade à internet, verificações de integridade, terminação SSL e TLS e modos de direcionamento IP.

Quando você cria um serviço do Kubernetes do tipo `LoadBalancer` no Modo Automático do EKS, o EKS provisiona e configura automaticamente um AWS Network Load Balancer com base nas anotações que você especifica. Essa abordagem declarativa permite que você gerencie as configurações do balanceador de carga diretamente por meio dos manifestos do Kubernetes, mantendo práticas de infraestrutura como código.

O Modo Automático do EKS gerencia o provisionamento do Network Load Balancer por padrão para todos os serviços do tipo LoadBalancer. Nenhuma instalação ou configuração adicional de controlador é necessária. A especificação `loadBalancerClass: eks.amazonaws.com/nlb` é automaticamente definida como padrão do cluster, simplificando o processo de implantação e mantendo a compatibilidade com as workloads existentes do Kubernetes.

> **Note**
>
> O Modo Automático do EKS precisa de tags de sub-rede para identificar sub-redes públicas e privadas.
> Se você criou o cluster com o `eksctl`, você já tem essas tags.
> Saiba como the section called “Marcar sub-redes”.

Serviço de exemplo

Para obter mais informações sobre o recurso `Service` do Kubernetes, consulte a documentação do Kubernetes.

Analise o recurso `Service` de exemplo abaixo:

```
apiVersion: v1
```

```
kind: Service
metadata:
  name: echoserver
  annotations:
    # Specify the load balancer scheme as internet-facing to create a public-facing
 Network Load Balancer (NLB)
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  selector:
    app: echoserver
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: LoadBalancer
  # Specify the new load balancer class for NLB as part of EKS Auto Mode feature
  # For clusters with Auto Mode enabled, this field can be omitted as it's the default
  loadBalancerClass: eks.amazonaws.com/nlb
```

Anotações comumente usadas

A tabela a seguir lista as anotações comumente usadas compatíveis com o Modo Automático do EKS. Observe que o Modo Automático do EKS pode não ser compatível com todas as anotações.

> **Tip**
>
> Todas as anotações a seguir precisam ser prefixadas com `service.beta.kubernetes.io/`

Campo	Descrição	Exemplo
`aws-load-balancer-type`	Especifica o tipo de balanceador de carga. Use `external` para novas implantações.	`external`
`aws-load-balancer-nlb-target-type`	Especifica se o tráfego deve ser roteado para instâncias de nós ou diretamente para IPs de pod. Use `instance`	`instance`

Campo	Descrição	Exemplo
	para implantações padrão ou `ip` para roteamento direto de pods.	
`aws-load-balancer-scheme`	Controla se o balanceador de carga é interno ou voltado para a internet.	`internet-facing`
`aws-load-balancer-healthcheck-protocol`	Protocolo de verificação de integridade do grupo de destino. As opções comuns são `TCP` (padrão) ou `HTTP`.	`HTTP`
`aws-load-balancer-healthcheck-path`	O caminho HTTP para verificações de integridade ao usar o protocolo HTTP e HTTPS.	`/healthz`
`aws-load-balancer-healthcheck-port`	Porta usada para verificações de integridade. Pode ser um número de porta específico ou `traffic-port` .	`traffic-port`
`aws-load-balancer-subnets`	Especifica em quais sub-redes criar o balanceador de carga. Pode usar IDs ou nomes de sub-rede.	`subnet-xxxx, subnet-yyyy`
`aws-load-balancer-ssl-cert`	ARN do certificado SSL do AWS Certificate Manager para HTTPS e TLS.	`arn:aws:acm:region:account:certificate/cert-id`
`aws-load-balancer-ssl-ports`	Especifica quais portas devem usar SSL e TLS.	`443, 8443`

Campo	Descrição	Exemplo
`load-balancer-source-ranges`	Intervalos CIDR permitidos para acessar o balanceador de carga.	`10.0.0.0/24, 192.168.1.0/24`
`aws-load-balancer-additional-resource-tags`	Tags adicionais da AWS a serem aplicadas ao balanceador de carga e aos recursos relacionados.	`Environment=prod,Team=platform`
`aws-load-balancer-ip-address-type`	Especifica se o balanceador de carga usa IPv4 ou pilha dupla (IPv4 + IPv6).	`ipv4` ou `dualstack`

Considerações

- Você deve atualizar o perfil do IAM do cluster para permitir a propagação de tags do Kubernetes para os recursos do AWS Load Balancer. Para obter mais informações, consulte the section called “Tags da AWS personalizadas para recursos do Modo Automático do EKS.”.
- Para obter informações sobre como associar recursos ao Modo automático do EKS ou ao AWS Load Balancer Controller autogerenciado, consulte the section called “Referência de migração”.
- Para obter informações sobre como corrigir problemas com balanceadores de carga, consulte the section called “Solução de problemas”.
- Para obter mais considerações sobre o uso do recurso de balanceamento de carga do Modo Automático do EKS, consulte the section called “Balanceamento de carga”.

Ao migrar para o Modo Automático do EKS de balanceamento de carga, várias alterações nas anotações de serviço e nas configurações de recursos são necessárias. As tabelas a seguir descrevem as principais diferenças entre as implementações anteriores e as novas, incluindo opções não compatíveis e alternativas recomendadas.

Anotações do serviço

Anterior	Novo	Descrição
`service.beta.kubernetes.io/load-balancer-source-ranges`	Não compatível	Use `spec.loadBalancerSourceRanges` no serviço
`service.beta.kubernetes.io/aws-load-balancer-type`	Não compatível	Use `spec.loadBalancerClass` no serviço
`service.beta.kubernetes.io/aws-load-balancer-internal`	Não compatível	Use `service.beta.kubernetes.io/aws-load-balancer-scheme`
`service.beta.kubernetes.io/aws-load-balancer-proxy-protocol`	Não compatível	Use `service.beta.kubernetes.io/aws-load-balancer-target-group-attributes` em vez disso
Vários atributos do balanceador de carga	Não compatível	Use `service.beta.kubernetes.io/aws-load-balancer-attributes`
`service.beta.kubernetes.io/aws-load-balancer-access-log-enabled`	Não compatível	Use `service.beta.kubernetes.io/aws-load-balancer-attributes` em vez disso
`service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-name`	Não compatível	Use `service.beta.kubernetes.io/aws-load-balancer-attributes` em vez disso
`service.beta.kubernetes.io/aws-load-`	Não compatível	Use `service.beta.kubernetes.io/aws-load-`

Anterior	Novo	Descrição
`balancer-access-log-s3-bucket-prefix`		`balancer-attributes` em vez disso
`service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled`	Não compatível	Use `service.beta.kubernetes.io/aws-load-balancer-attributes` em vez disso

Para migrar das anotações de atributos obsoletas do balanceador de carga, consolide essas configurações na anotação `service.beta.kubernetes.io/aws-load-balancer-attributes`. Essa anotação aceita uma lista de pares chave/valor separados por vírgula de vários atributos do balanceador de carga. Por exemplo, para especificar o registro de acesso em log e o balanceamento de carga entre zonas, use o seguinte formato:

```
service.beta.kubernetes.io/aws-load-balancer-attributes:
 access_logs.s3.enabled=true,access_logs.s3.bucket=my-bucket,access_logs.s3.prefix=my-prefix,load_balancing.cross_zone.enabled=true
```

Esse formato consolidado fornece uma maneira mais consistente e flexível de configurar os atributos do balanceador de carga e, ao mesmo tempo, reduz o número de anotações individuais necessárias. Analise suas configurações de serviço existentes e atualize-as para usar esse formato consolidado.

TargetGroupBinding

Anterior	Novo	Descrição
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Alteração de versão da API
`spec.targetType` opcional	`spec.targetType` obrigatório	Especificação explícita do tipo de destino
`spec.networking.ingress.from`	Não compatível	Não oferece mais compatibilidade com NLB sem grupos de segurança

Observação: para usar o recurso personalizado TargetGroupBinding, é necessário marcar o grupo de destino com a etiqueta `eks:eks-cluster-name` contendo o nome do cluster para conceder ao controlador as permissões do IAM necessárias. Esteja ciente de que o controlador excluirá o grupo de destino quando o recurso TargetGroupBinding ou o cluster for excluído.

Criar uma classe de armazenamento

Uma `StorageClass` no Modo automático do Amazon EKS define como os volumes do Amazon EBS são provisionados automaticamente quando as aplicações solicitam armazenamento persistente. Esta página explica como criar e configurar uma `StorageClass` que funcione com o Modo automático do Amazon EKS para provisionar volumes do EBS.

Ao configurar uma `StorageClass`, você pode especificar as configurações padrão para os volumes do EBS, incluindo tipo de volume, criptografia, IOPS e outros parâmetros de armazenamento. Você também pode configurar a `StorageClass` para usar chaves do AWS KMS para gerenciamento de criptografia.

O Modo automático do EKS não cria uma `StorageClass` para você. Você deve criar uma `StorageClass` referenciando `ebs.csi.eks.amazonaws.com` para usar o recurso de armazenamento do Modo automático do EKS.

Primeiro, crie um arquivo denominado `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
allowedTopologies:
- matchLabelExpressions:
  - key: eks.amazonaws.com/compute-type
    values:
    - auto
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

Segundo, aplique a classe de armazenamento ao cluster.

```
kubectl apply -f storage-class.yaml
```

Principais componentes:

- `provisioner: ebs.csi.eks.amazonaws.com`: usa o modo automático do EKS
- `allowedTopologies`: especificar `matchLabelExpressions` para combinar com `eks.amazonaws.com/compute-type:auto` garantirá que, se seus pods precisarem de um volume provisionado automaticamente usando o Modo automático, os pods não serão agendados em nós não automáticos.
- `volumeBindingMode: WaitForFirstConsumer`: atrasa a criação do volume até que um pod precise dele
- `type: gp3`: especifica o tipo de volume do EBS
- `encrypted: "true"`: o EBS criptografará todos os volumes criados usando a `StorageClass`. O EBS usará o alias de chave `aws/ebs` padrão. Para obter mais informações, consulte How Amazon EBS encryption works no Guia do usuário do Amazon EBS. Este valor é opcional, mas sugerido.
- `storageclass.kubernetes.io/is-default-class: "true"`: o Kubernetes usará essa classe de armazenamento por padrão, a menos que você especifique uma classe de volume diferente em uma reivindicação de volume persistente. Este valor é opcional. Tenha cuidado ao definir esse valor se estiver migrando de um controlador de armazenamento diferente.

Usar uma chave autogerenciada do KMS para criptografar volumes do EBS

Para usar uma chave autogerenciada do KMS para criptografar volumes do EBS automatizados pelo Modo automático do EKS, você precisa:

1. Criar uma chave autogerenciada do KMS.
   - Para obter mais informações, consulte Create a symmetric encryption KMS key ou How Amazon Elastic Block Store (Amazon EBS) uses KMS no Guia do usuário do KMS.
2. Crie uma nova política que permita o acesso à chave do KMS.
   - Use o exemplo de política do IAM abaixo para criar a política. Insira o ARN da nova chave autogerenciada do KMS. Para obter mais informações, consulte a seção Criar perfis e anexar políticas (console) do Guia do usuário do AWS IAM.
3. Anexe a política ao perfil do cluster do EKS.

- Use o console da AWS para encontrar o ARN do perfil do cluster do EKS. As informações do perfil estão visíveis na seção Visão geral. Para obter mais informações, consulte the section called “Função do IAM do cluster”.

4. Atualize a `StorageClass` para fazer referência ao ID da chave do KMS no campo `parameters.kmsKeyId`.

Exemplo de política do IAM autogerenciada do KMS

Atualize os seguintes valores na política abaixo:

- `<account-id>`: o ID da conta da AWS, como `111122223333`
- `<aws-region>`: a região da AWS do cluster, como `us-west-2`

```
{
  "Version":"2012-10-17",
  "Id": "key-auto-policy-3",
  "Statement": [
      {
          "Sid": "Enable IAM User Permissions",
          "Effect": "Allow",
          "Principal": {
              "AWS": "arn:aws:iam::123456789012:root"
          },
          "Action": "kms:*",
          "Resource": "*"
      },
      {
        "Sid": "Allow access through EBS for all principals in the account that are
 authorized to use EBS",
        "Effect": "Allow",
        "Principal": {
            "AWS": "*"
        },
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:CreateGrant",
            "kms:DescribeKey"
```

```
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:CallerAccount": "123456789012",
                "kms:ViaService": "ec2.us-east-1.amazonaws.com"
            }
        }
    }
  ]
}
```

Exemplo da **`StorageClass`** autogerenciada do KMS

```
parameters:
  type: gp3
  encrypted: "true"
  kmsKeyId: <custom-key-arn>
```

Referência a parâmetros da **`StorageClass`**

Para obter informações gerais sobre os recursos da `StorageClass` do Kubernetes, consulte Storage Classes na documentação do Kubernetes.

A seção `parameters` do recurso `StorageClass` é específica para a AWS. Use a tabela a seguir para analisar as opções disponíveis.

Parâmetros	Valores	Padrão	Descrição
"csi.storage.k8s.io/fstype"	xfs, ext2, ext3, ext4	ext4	Tipo de sistema de arquivos que será formatado durante a criação do volume. Esse parâmetro diferencia maiúsculas de minúsculas.
"type"	io1, io2, gp2, gp3, sc1, st1, standard, sbp1, sbg1	gp3	Tipo de volume do EBS.

Parâmetros	Valores	Padrão	Descrição
"iopsPerGB"			Operações de E/S por segundo por GiB. Pode ser especificado para volumes IO1, IO2 e GP3.
"allowAutoIOPSPerGBIncrease"	true, false	false	Quando `"true"`, o driver CSI aumenta o IOPS para um volume quando `iopsPerGB * <volume size>` está muito baixo para se enquadrar na faixa de IOPS compatível com a AWS. Isso permite que o provisionamento dinâmico sempre tenha êxito, mesmo quando o usuário especificar uma capacidade de PVC ou valor de `iopsPerGB` muito pequeno. Por outro lado, isso pode introduzir custos adicionais, pois esses volumes têm IOPS mais alto do que o solicitado em `iopsPerGB` .

Parâmetros	Valores	Padrão	Descrição
"iops"			Operações de E/S por segundo. Pode ser especificado para volumes IO1, IO2 e GP3.
"throughput"		125	Throughput em MiB/s. Eficaz somente quando o tipo de volume gp3 é especificado.
"encrypted"	true, false	false	Especifica se o volume deve ser criptografado ou não. Os valores válidos são "true" ou "false".
"blockExpress"	true, false	false	Permite a criação de volumes io2 Block Express.
"kmsKeyId"			O ARN completo da chave a ser usada ao criptografar o volume. Se não for especific ado, a AWS usará a chave padrão do KMS para a região em que o volume está. Será uma chave gerada automaticamente, denominada `/aws/ebs` caso não seja alterada.

Parâmetros	Valores	Padrão	Descrição
"blockSize"			O tamanho do bloco a ser usado ao formatar o sistema de arquivos subjacente. Compatível apenas com nós Linux e com o fstype `ext2`, `ext3`, `ext4` ou `xfs`.
"inodeSize"			O tamanho do inode a ser usado ao formatar o sistema de arquivos subjacente. Compatível apenas com nós Linux e com o fstype `ext2`, `ext3`, `ext4` ou `xfs`.
"bytesPerInode"			O `bytes-per-inode` a ser usado ao formatar o sistema de arquivos subjacente. Compatível apenas com nós Linux e com o fstype `ext2`, `ext3`, `ext4`.
"numberOfInodes"			O `number-of-inodes` a ser usado ao formatar o sistema de arquivos subjacent e. Compatível apenas com nós Linux e com o fstype `ext2`, `ext3`, `ext4`.

Parâmetros	Valores	Padrão	Descrição
"ext4BigAlloc"	true, false	false	Altera o sistema de arquivos `ext4` para usar a alocação de blocos em cluster habilitando a opção de formatação `bigalloc`. Aviso: `bigalloc` pode não ser totalmente compatível com o Kernel Linux do nó.
"ext4ClusterSize"			O tamanho do cluster a ser usado ao formatar um sistema de arquivos `ext4` quando o recurso `bigalloc` está habilitado. Observação: o parâmetro `ext4BigAlloc` deve ser definido como true.

Para obter mais informações, consulte AWS EBS CSI Driver no GitHub.

Considerações

> **Note**
>
> Você só pode implantar workloads que dependam das StorageClasses no Modo automático do EKS em nós do Modo automático do EKS. Se você tiver um cluster com tipos mistos de nós, precisará configurar suas workloads para serem executadas somente em nós do Modo

automático do EKS. Para obter mais informações, consulte the section called “Controlar a implantação”.

A capacidade de armazenamento em blocos do Modo automático do EKS é diferente do driver CSI do EBS.

- Provisionamento estático
  - Caso queira usar volumes do EBS criados externamente com o Modo automático do EKS, você precisará adicionar manualmente uma tag da AWS com a chave `eks:eks-cluster-name` e o valor do nome do cluster.
- Taint de inicialização de nós
  - Você não pode usar o recurso de taint de inicialização de nós para impedir a programação de pods antes que a capacidade de armazenamento esteja pronta.
- Tags personalizadas em volumes provisionados dinamicamente
  - Você não pode usar o sinalizador extra-tag da CLI para configurar tags personalizadas em volumes do EBS provisionados dinamicamente
  - Você pode usar a marcação de `StorageClass` para adicionar tags personalizadas. O Modo automático do EKS adicionará tags aos recursos associados da AWS. Você precisará atualizar o perfil do IAM do cluster para tags personalizadas. Para obter mais informações, consulte the section called “Tags da AWS personalizadas para recursos do Modo Automático do EKS.”.
- Métricas de performance detalhadas do EBS
  - Você não pode acessar as métricas do Prometheus da performance detalhada do EBS

Instalar o complemento do Controlador de Snapshots da CSI

O Modo automático do EKS é compatível com o complemento do Controlador de Snapshots da CSI do Amazon EKS.

A AWS sugere que você configure esse complemento para ser executado no grupo de nós `system` integrado.

Para obter mais informações, consulte:

- the section called “Executar complementos críticos”
- the section called “Analisar grupos de nós integrados”

- the section called “Controlador de snapshots da CSI”

Para instalar o controlador de snapshots no grupo de nós do sistema

1. Abra o cluster do EKS no console da AWS
2. Na guia Complementos, selecione Obter mais complementos
3. Selecione Controlador de Snapshots da CSI e depois Próximo
4. Na página Configurar definições de complementos selecionados, selecione Configurações opcionais para visualizar o Esquema de configuração do complemento
   a. Insira o YAML a seguir para associar o controlador de snapshots ao grupo de nós `system`. O controlador de snapshots inclui uma tolerância ao taint `CriticalAddonsOnly`.

```
{
        "nodeSelector": {
            "karpenter.sh/nodepool": "system"
        }
}
```

   b. Selecione Próximo.
5. Revise a configuração do complemento e selecione Criar

Desabilitar o Modo Automático do EKS

Você pode desabilitar o Modo Automático do EKS em um cluster existente do EKS. É uma operação destrutiva.

- O EKS encerrará todas as instâncias do EC2 operadas pelo Modo Automático do EKS.
- O EKS excluirá todos os balanceadores de carga operados pelo Modo Automático do EKS.
- O EKS não excluirá volumes do EBS provisionados pelo Modo Automático do EKS.

O modo automático do EKS foi projetado para gerenciar completamente os recursos que cria. As intervenções manuais podem fazer com que o modo automático do EKS não consiga limpar completamente esses recursos quando for desabilitado. Por exemplo, se você referenciar um grupo de segurança gerenciado a partir de regras externas do grupo de segurança e esquecer de remover essa referência antes de desabilitar o modo automático do EKS para um cluster, o grupo de

segurança gerenciado “vazará” (não será excluído). As etapas abaixo descrevem como remover um grupo de segurança vazado, caso isso aconteça.

Desabilitar o Modo Automático do EKS (console da AWS)

1. Abra a página de visão geral do cluster no Console de gerenciamento da AWS.
2. Em Modo Automático do EKS, selecione Gerenciar.
3. Alterne o Modo Automático do EKS para `off`.

Se algum grupo de segurança gerenciado não for excluído ao final deste processo, é possível excluí-lo manualmente usando as descrições da página Exclua um grupo de segurança.

Desabilitar o Modo Automático do EKS (AWS CLI)

Use o comando a seguir para desabilitar o Modo Automático do EKS em um cluster existente.

Você precisa ter a `aws` CLI instalada e estar logado com permissões suficientes para gerenciar clusters do EKS. Para obter mais informações, consulte Configuração.

> **Note**
>
> Os recursos de computação, armazenamento em blocos e balanceamento de carga devem estar todos habilitados ou desabilitados na mesma solicitação.

```
aws eks update-cluster-config \
 --name $CLUSTER_NAME \
 --compute-config enabled=false \
 --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": false}}' \
 --storage-config '{"blockStorage":{"enabled": false}}'
```

É possível verificar se um grupo de segurança vazado do modo automático do EKS não foi excluído após desabilitar o modo automático do EKS da seguinte maneira:

```
aws ec2 describe-security-groups \
    --filters Name=tag:eks:eks-cluster-name,Values=<cluster-Name> Name=tag-
key,Values=ingress.eks.amazonaws.com/resource,service.eks.amazonaws.com/resource --
query "SecurityGroups[*].[GroupName]"
```

Para, em seguida, excluir o grupo de segurança:

```
aws ec2 delete-security-group --group-name=<sg-name>
```

Atualizar a versão do Kubernetes de um cluster do Modo Automático do EKS.

Este tópico explica como atualizar a versão do Kubernetes do cluster no Modo Automático. O Modo Automático simplifica o processo de atualização de versões gerenciando a coordenação das atualizações do ambiente de gerenciamento com as substituições de nós, mantendo a disponibilidade de workloads por meio de orçamentos de interrupção de pods.

Ao atualizar um cluster do Modo Automático, muitos componentes que tradicionalmente exigiam atualizações manuais agora são gerenciados como parte do serviço. Compreender os aspectos automatizados do processo de atualização e suas responsabilidades ajuda a garantir uma transição suave de versão do cluster.

Saber mais sobre as atualizações com o Modo Automático do EKS

Depois de iniciar uma atualização do ambiente de gerenciamento, o Modo Automático do EKS atualizará os nós no cluster. À medida que os nós expiram, o Modo Automático do EKS os substitui por novos nós. Os novos nós têm a nova versão correspondente do Kubernetes. O Modo Automático do EKS observa os orçamentos de interrupção de pods ao atualizar os nós.

Além disso, não é mais necessário atualizar componentes como:

- CNI da Amazon VPC
- AWS Load Balancer Controller
- CoreDNS
- `kube-proxy`
- Karpenter
- Driver CSI do AWS EBS

O Modo Automático do EKS substitui esses componentes pela funcionalidade de serviço.

Você ainda é responsável por atualizar:

- Aplicações e workloads implantadas no cluster
- Complementos e controladores autogerenciados
- Complementos do Amazon EKS
  - Saiba como the section called “Atualização de um complemento”

Conheça as práticas recomendadas para atualizações de clusters

Iniciar atualização do cluster

Para iniciar uma atualização de cluster, consulte the section called “Atualizar a versão do Kubernetes”.

Habilitar ou desabilitar NodePools integrados

O Modo Automático do EKS tem dois NodePools integrados. Você pode habilitar ou desabilitar esses NodePools usando o console da AWS, a CLI ou a API.

Referência de NodePool integrado

- `system`
  - Esse NodePool tem um taint `CriticalAddonsOnly`. Muitos complementos do EKS, como o CoreDNS, toleram esse taint. Use esse grupo de nós do sistema para separar aplicações críticas de clusters.
  - É compatível com as arquiteturas `amd64` e `arm64`.
- `general-purpose`
  - Esse NodePool oferece suporte para iniciar nós de workloads de uso geral no cluster.
  - Usa somente arquitetura `amd64`.

Os dois NodePools integrados:

- Usam o NodeClass padrão do EKS
- Usam somente a capacidade sob demanda do EC2
- Usam as famílias de instâncias C, M e R do EC2
- Requerem instâncias da 5.ª geração ou mais recentes do EC2

> **Note**
>
> É necessário habilitar pelo menos um NodePool integrado para que o EKS provisione o NodeClass “default”. Caso desabilite todos os NodePools integrados, você precisará criar um NodeClass personalizado e configurar um NodePool para usá-lo. Para obter mais informações NodeClasses, consulte the section called “Criar classe de nó”.

Procedimento

Pré-requisitos

- A versão mais recente da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo. Para verificar sua versão atual, use `aws --version`. Para instalar a versão mais recente, consulte Instalar e Configuração rápida com aws configure, no Guia do usuário da AWS Command Line Interface.
  - Faça login na CLI com permissões suficientes do IAM para criar recursos da AWS, incluindo políticas do IAM, perfis do IAM e clusters do EKS.

Habilitar com a AWS CLI

Use o seguinte comando para habilitar os dois NodePools integrados:

```
aws eks update-cluster-config \
  --name <cluster-name> \
  --compute-config '{
    "nodeRoleArn": "<node-role-arn>",
    "nodePools": ["general-purpose", "system"],
    "enabled": true
  }' \
  --kubernetes-network-config '{
  "elasticLoadBalancing":{"enabled": true}
  }' \
  --storage-config '{
  "blockStorage":{"enabled": true}
  }'
```

Você pode modificar o comando para habilitar seletivamente os NodePools.

Desabilitar com a AWS CLI

Use o seguinte comando para desabilitar os dois NodePools integrados:

```
aws eks update-cluster-config \
  --name <cluster-name> \
  --compute-config '{
  "enabled": true,
  "nodePools": []
  }' \
  --kubernetes-network-config '{
  "elasticLoadBalancing":{"enabled": true}}' \
  --storage-config '{
  "blockStorage":{"enabled": true}
  }'
```

Controlar se uma workload é implantada nos nós do Modo Automático do EKS

Ao executar workloads em um cluster do EKS com o Modo Automático do EKS, talvez você precise controlar se workloads específicas são executadas nos nós do Modo Automático do EKS ou em outros tipos de computação. Este tópico descreve como usar seletores de nós e regras de afinidade para garantir que as workloads sejam agendadas na infraestrutura de computação pretendida.

Os exemplos neste tópico demonstram como usar o rótulo `eks.amazonaws.com/compute-type` para exigir ou impedir a implantação de uma workload nos nós do Modo Automático do EKS. Isso é particularmente útil em clusters de modo misto em que você está executando o Modo Automático do EKS e outros tipos de computação, como provisionadores do Karpenter autogerenciados ou grupos de nós gerenciados pelo EKS.

Os nós do Modo Automático do EKS definiram o valor do rótulo `eks.amazonaws.com/compute-type` como `auto`. Você pode usar esse rótulo para controlar se uma workload é implantada em nós gerenciados pelo Modo Automático do EKS.

Exigir que uma workload seja implantada nos nós do Modo Automático do EKS

> **Note**
>
> Esse valor de `nodeSelector` não é necessário para o Modo Automático do EKS. Esse valor de `nodeSelector` só será relevante se você estiver executando um cluster em um

modo misto, com tipos de nós não gerenciados pelo Modo Automático do EKS. Por exemplo, você pode ter capacidade computacional estática implantada no cluster com grupos de nós gerenciados pelo EKS e capacidade computacional dinâmica gerenciada pelo Modo Automático do EKS.

Você pode adicionar esse `nodeSelector` às implantações ou a outras workloads para exigir que o Kubernetes as programe nos nós do Modo Automático do EKS.

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    nodeSelector:
      eks.amazonaws.com/compute-type: auto
```

Exigir que uma workload não seja implantada nos nós do Modo Automático do EKS

Você pode adicionar esse `nodeAffinity` às implantações ou a outras workloads para exigir que o Kubernetes não as programe nos nós do Modo Automático do EKS.

```
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - auto
```

Executar complementos críticos em instâncias dedicadas

Neste tópico, você aprenderá como implantar uma workload com uma tolerância `CriticalAddonsOnly` para que o Modo Automático do EKS a programe no grupo de nós `system`.

O grupo de nós `system` integrado do Modo Automático do EKS foi projetado para executar complementos críticos em instâncias dedicadas. Essa segmentação garante que os componentes essenciais tenham recursos dedicados e fiquem isolados das workloads gerais, aprimorando a estabilidade e a performance gerais do cluster.

Este guia demonstra como implantar complementos no grupo de nós `system` usando a tolerância `CriticalAddonsOnly` e os seletores de nós apropriados. Seguindo estas etapas, você pode garantir que as aplicações críticas sejam programadas nos nós `system` dedicados, aproveitando os benefícios de isolamento e alocação de recursos fornecidos pela estrutura especializada de grupo de nós do Modo Automático do EKS.

O Modo Automático do EKS tem dois grupos de nós integrados: `general-purpose` e `system`. Para ter mais informações, consulte the section called “Analisar grupos de nós integrados”.

O objetivo do grupo de nós `system` é segregar complementos críticos em nós diferentes. Os nós provisionados pelo grupo de nós `system` têm um taint `CriticalAddonsOnly` do Kubernetes. O Kubernetes apenas programará pods nesses nós se eles tiverem uma tolerância correspondente. Para obter mais informações, consulte Taints and Tolerations na documentação do Kubernetes.

Pré-requisitos

- Cluster do Modo Automático do EKS com o grupo de nós `system` integrado habilitado. Para ter mais informações, consulte the section called “Analisar grupos de nós integrados”.
- `kubectl` instalado e configurado. Para ter mais informações, consulte Configuração.

Procedimento

Analise o exemplo de YAML abaixo. Observe as seguintes configurações:

- `nodeSelector`: associa a workload ao grupo de nós `system` integrado. Esse grupo de nós deve ser habilitado com a API da AWS. Para ter mais informações, consulte the section called “Analisar grupos de nós integrados”.
- `tolerations`: esta tolerância supera o taint `CriticalAddonsOnly` nos nós do grupos de nós `system`.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
```

```
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      nodeSelector:
        karpenter.sh/nodepool: system
      tolerations:
      - key: "CriticalAddonsOnly"
        operator: "Exists"
      containers:
      - name: app
        image: nginx:latest
        resources:
          requests:
            cpu: "500m"
            memory: "512Mi"
```

Para atualizar uma workload para executar no grupo de nós `system`, você precisa:

1. Atualizar a workload existente para adicionar as seguintes configurações descritas acima:
   - `nodeSelector`
   - `tolerations`
2. Implantar a workload atualizada no cluster com `kubectl apply`

Depois de atualizar a workload, ela será executada em nós dedicados.

Usar políticas de rede com o Modo Automático do EKS

Visão geral

À medida que os clientes escalam os ambientes de aplicação usando o EKS, o isolamento do tráfego de rede torna-se cada vez mais fundamental para evitar o acesso não autorizado a recursos internos e externos ao cluster. Isso é especialmente importante em um ambiente de multilocação, no qual diversas workloads distintas são executadas simultaneamente no mesmo cluster. As políticas de rede do Kubernetes permitem reforçar a postura de segurança de rede das workloads do Kubernetes, incluindo integrações com endpoints externos ao cluster. O modo automático do EKS oferece suporte a diversos tipos de políticas de rede.

Isolamento nas camadas 3 e 4

As políticas de rede padrão do Kubernetes operam nas camadas 3 e 4 do modelo de rede OSI e permitem que você controle o fluxo de tráfego no nível de endereço IP ou de porta dentro do cluster do Amazon EKS.

Casos de uso

- Realize a segmentação do tráfego de rede entre as workloads para assegurar que a comunicação ocorra apenas entre aplicações relacionadas.
- Realize o isolamento dos locatários por namespace, usando políticas para garantir a separação de rede.

Aplicação baseada em DNS

É comum que os clientes implementem workloads no EKS integradas a ambientes distribuídos maiores, sendo que algumas delas devem estabelecer comunicação com sistemas e serviços externos ao cluster (tráfego em direção ao norte). Esses sistemas e serviços podem estar localizados na Nuvem AWS ou completamente fora do ambiente da AWS. As políticas baseadas em Sistema de Nomes de Domínio (DNS) permitem que você fortaleça a postura de segurança ao adotar uma abordagem mais estável e previsível a fim de evitar o acesso não autorizado de pods a recursos ou a endpoints externos ao cluster. Este mecanismo elimina a necessidade de rastrear e de permitir manualmente endereços IP específicos nas listas de permissões. Ao proteger os recursos com uma abordagem baseada em DNS, você também tem mais flexibilidade para atualizar a infraestrutura externa sem precisar flexibilizar a postura de segurança ou modificar as políticas de rede devido a mudanças nas versões originais de servidores e hosts. É possível filtrar o tráfego de saída para endpoints externos utilizando um Nome de Domínio Totalmente Qualificado (FQDN, na sigla em inglês) ou um padrão de correspondência de nome de domínio do DNS. Isso proporciona ainda mais flexibilidade ao ampliar o acesso para diversos subdomínios vinculados a um endpoint específico externo ao cluster.

Casos de uso

- Adote o padrão de abordagem baseada em DNS para filtrar o acesso de um ambiente do Kubernetes a endpoints externos ao cluster.
- Garanta a segurança do acesso aos serviços da AWS em ambientes com múltiplos locatários.
- Gerencie o acesso da rede de pods a workloads on-premises em seus ambientes de nuvem híbrida.

Regras administrativas (ou com abrangência de cluster)

Em alguns casos, como em cenários multilocatários, os clientes podem ter o requisito de aplicar um padrão de segurança de rede que seja válido em todo o cluster. Em vez de criar e gerenciar uma política diferente para cada namespace de forma repetitiva, é possível usar uma política única para o controle centralizado do acesso à rede de diversas workloads no cluster, independentemente do namespace em que estejam. Esses tipos de políticas permitem que você amplie o escopo de aplicação para as regras de filtragem de rede aplicadas nas camadas 3 e 4 ou por meio de regras de DNS.

Casos de uso

- Gerencie de forma centralizada o acesso à rede de todas as workloads (ou de um grupo específico delas) no seu cluster do EKS.
- Defina uma postura de segurança de rede padrão em todo o cluster.
- Amplie os padrões de segurança organizacionais ao escopo do cluster com maior eficiência operacional.

Introdução

Pré-requisitos

- Um cluster do Amazon EKS com o Modo Automático do EKS habilitado
- kubectl configurado para se conectar ao cluster

Etapa 1: habilitar o Network Policy Controller

Para usar políticas de rede com o Modo Automático do EKS, primeiro você precisa habilitar o Network Policy Controller aplicando um ConfigMap ao cluster.

1. Crie um arquivo chamado `enable-network-policy.yaml` com o seguinte conteúdo:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
```

```
  enable-network-policy-controller: "true"
```

2. Aplicar o ConfigMap ao cluster:

```
kubectl apply -f enable-network-policy.yaml
```

Etapa 2: habilitar políticas de rede na classe de nó

Antes de usar políticas de rede, você precisa garantir que sua classe de nó esteja configurada para que seja compatível com elas. Siga estas etapas:

1. Crie ou edite um arquivo YAML da classe de nó (por exemplo, `nodeclass-network-policy.yaml`) com o seguinte conteúdo:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: network-policy-enabled
spec:
  # Enables network policy support
  networkPolicy: DefaultAllow
  # Optional: Enables logging for network policy events
  networkPolicyEventLogs: Enabled
  # Include other Node Class configurations as needed
```

2. Aplique a configuração da classe de nó ao cluster:

```
kubectl apply -f nodeclass-network-policy.yaml
```

3. Verifique se a classe de nó foi criada:

```
kubectl get nodeclass network-policy-enabled
```

4. Atualize o grupo de nós para usar essa classe de nó. Para obter mais informações, consulte the section called “Criar grupo de nós”.

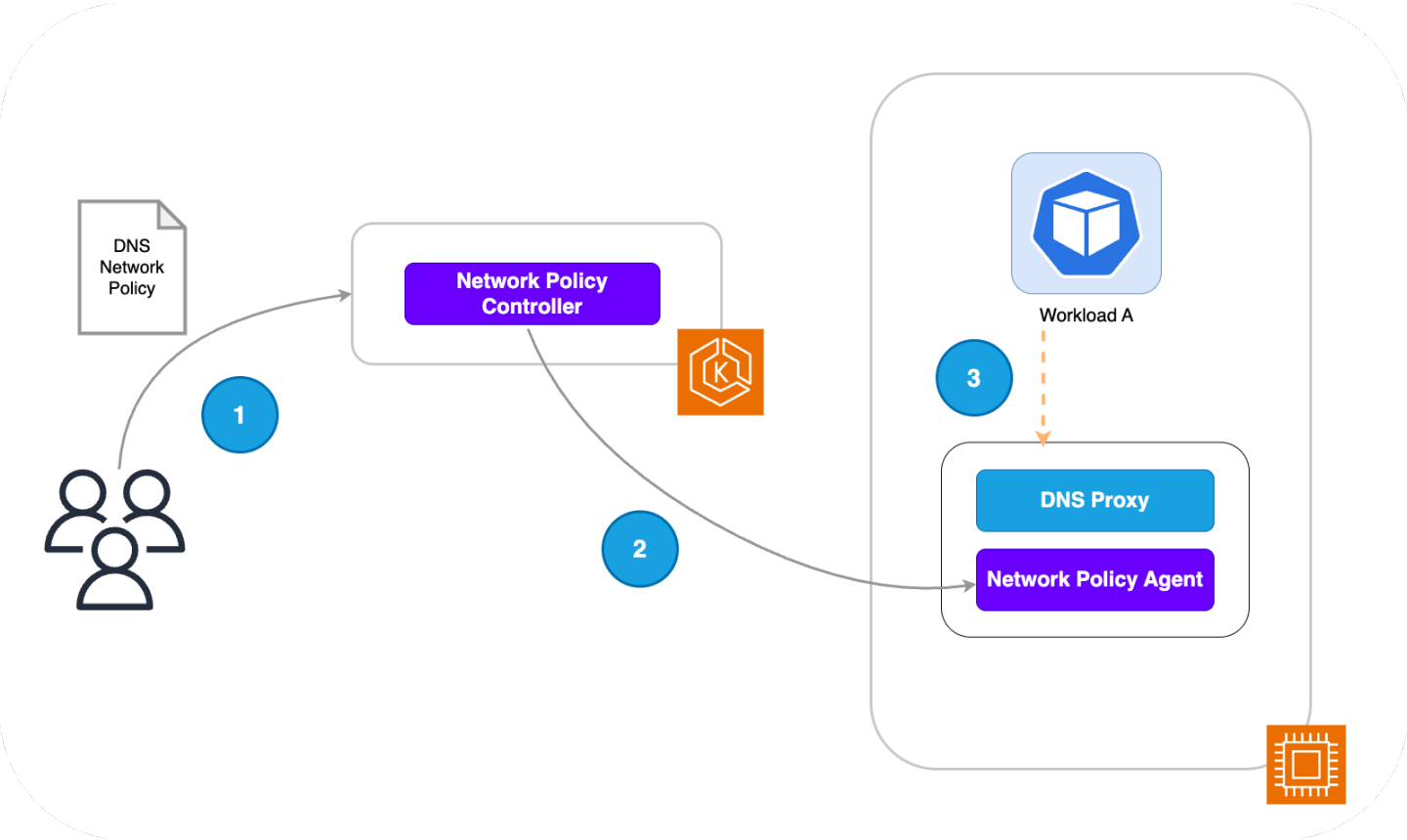
Depois que os nós estiverem usando essa classe de nó, eles poderão aplicar políticas de rede. Agora você pode prosseguir para criar e aplicar políticas de rede para controlar o tráfego no cluster. Para todas as opções de configuração da classe de nó, consulte the section called “Criar classe de nó”.

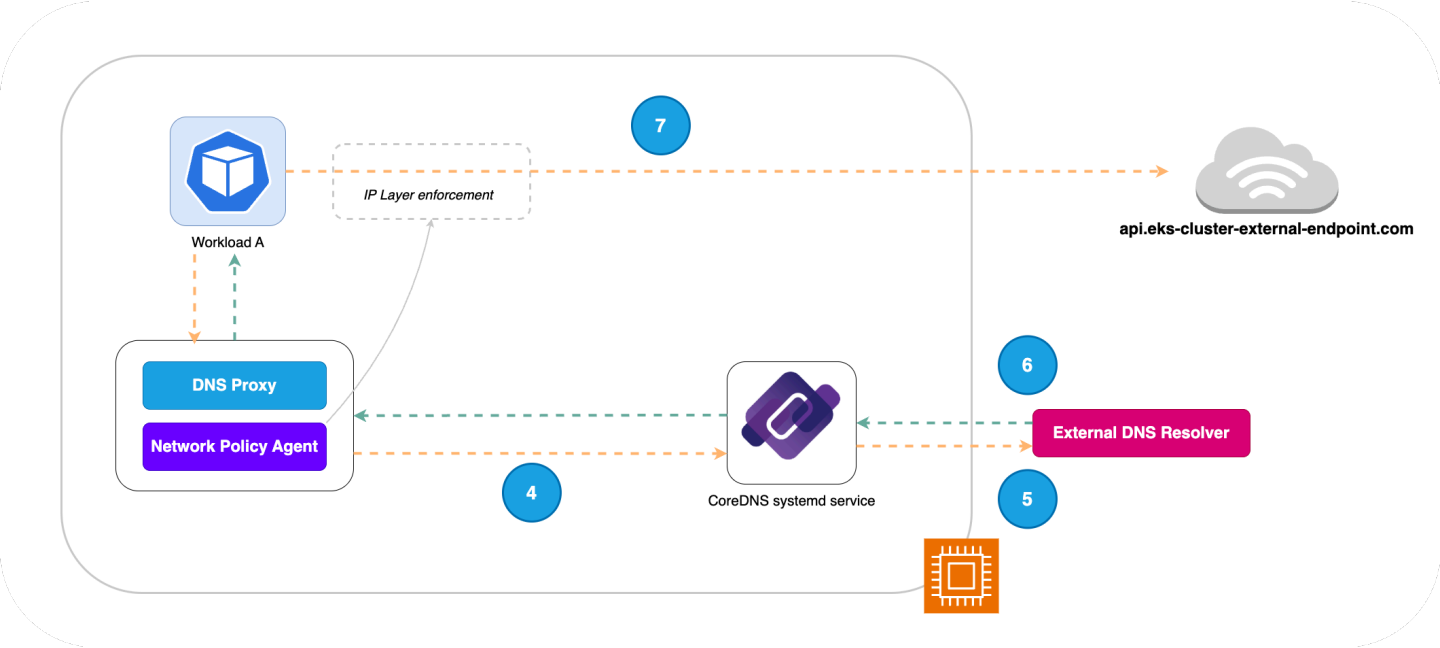
Etapa 3: criar e testar políticas de rede

O cluster do Modo Automático do EKS agora está configurado para ser compatível com as políticas de rede do Kubernetes. Você pode testar isso com the section called “Demonstração de política Stars”.

Como funciona?

Política de rede baseada em DNS





1. A equipe responsável pela plataforma aplica uma política baseada em DNS ao cluster do EKS.
2. O Network Policy Controller é responsável por monitorar a criação de políticas no cluster e, em seguida, reconciliar os endpoints das políticas. Neste caso de uso, o Network Policy Controller orienta o agente do nó a realizar a filtragem das solicitações de DNS, seguindo a lista de domínios autorizados na política correspondente. Os nomes de domínio são incluídos na lista de permissões usando o FQDN ou um nome de domínio que corresponda a um padrão definido na configuração de recursos do Kubernetes.
3. A Workload A faz uma tentativa de resolução de IP para um endpoint externo ao cluster. A solicitação de DNS é enviada primeiro a um proxy, que realiza a filtragem das solicitações com base na lista de permissões aplicada por meio da política de rede.
4. Uma vez que a solicitação de DNS é aprovada pela lista de permissões do filtro, ela é encaminhada ao CoreDNS.
5. O CoreDNS, por sua vez, envia a solicitação ao resolvedor de DNS externo (Amazon Route 53 Resolver) para obter a lista de endereços IP associados ao nome de domínio.
6. Os endereços IP resolvidos, juntamente com o TTL, são enviados de volta como resposta à solicitação de DNS. Em seguida, esses IPs são gravados em um mapa eBPF, que é utilizado na etapa seguinte para a aplicação da política na camada de IP.
7. As sondas eBPF anexadas à interface Veth do pod filtrarão o tráfego de saída da Workload A para o endpoint externo ao cluster, com base nas regras em vigor. Isso garante que os pods possam enviar tráfego externo ao cluster apenas para os IPs dos domínios incluídos na lista

de permissões. O tempo de validade desses endereços IP baseia-se no TTL retornado pelo resolvedor de DNS externo (Amazon Route 53 Resolver).

Uso da Application Network Policy

A `ApplicationNetworkPolicy` combina as funcionalidades das políticas de rede padrão do Kubernetes com a filtragem baseada em DNS no nível do namespace, utilizando uma única definição de recurso personalizado (CRD, na sigla em inglês). Portanto, a `ApplicationNetworkPolicy` pode ser usada para:

1. Definir restrições nas camadas 3 e 4 da pilha de rede usando blocos de IP e números de porta.
2. Definir regras que operam na camada 7 da pilha de rede e permitem filtrar o tráfego com base em FQDNs.

Observação importante: as regras baseadas em DNS, definidas por meio da `ApplicationNetworkPolicy`, são aplicáveis somente a workloads executadas em instâncias do EC2 iniciadas pelo modo automático do EKS.

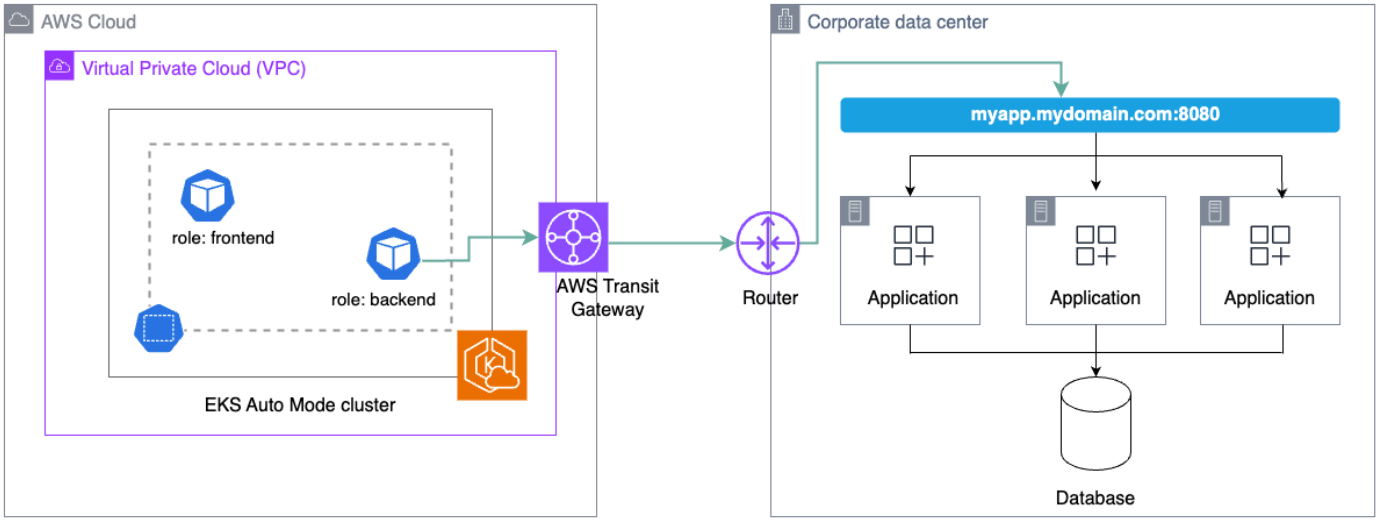
Exemplo

Você tem uma workload em seu cluster no modo automático do EKS que precisa se comunicar com uma aplicação on-premise, a qual usa um balanceador de carga com um nome de DNS. É possível implementar essa configuração usando a seguinte política de rede:

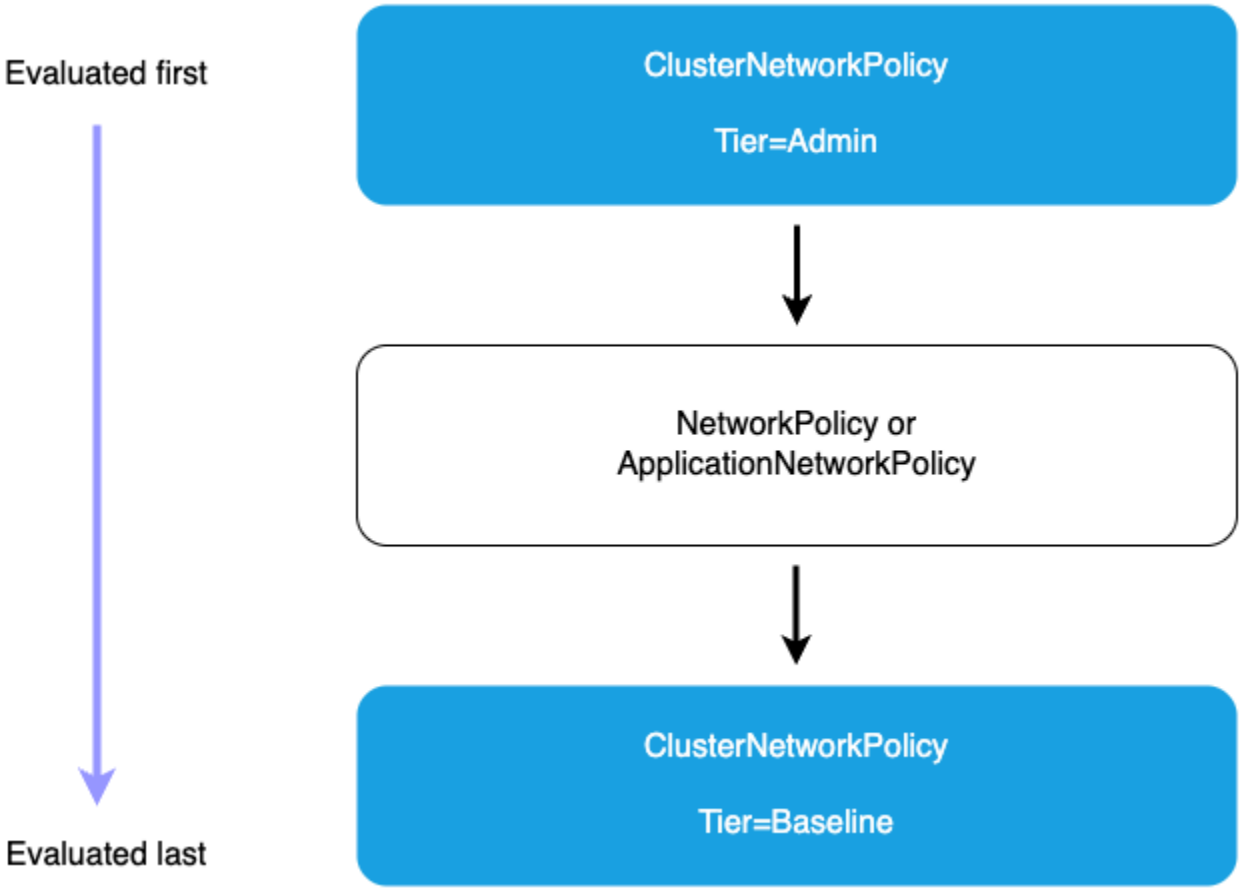
```
apiVersion: networking.k8s.aws/v1alpha1
kind: ApplicationNetworkPolicy
metadata:
  name: my-onprem-app-egress
  namespace: galaxy
spec:
  podSelector:
    matchLabels:
      role: backend
  policyTypes:
  - Egress
  egress:
  - to:
    - domainNames:
      - "myapp.mydomain.com"
    ports:
```

```
    - protocol: TCP
      port: 8080
```

No nível de rede do Kubernetes, isso permitiria a saída de quaisquer pods no namespace “galaxy” rotulados com `role: backend` para se conectarem ao nome de domínio myapp.mydomain.com na porta TCP 8080. Além disso, seria necessário estabelecer a conectividade de rede para que o tráfego seja enviado da VPC em direção ao data center corporativo.



Política de rede do administrador (ou do cluster)



Uso da Cluster Network Policy

Quando se usa uma `ClusterNetworkPolicy`, as políticas da camada Administrador têm prioridade na avaliação e não podem ser sobrescritas. Após a avaliação das políticas do nível de Administrador, as políticas padrão com escopo de namespace são utilizadas para executar as regras de segmentação de rede aplicadas. Você pode fazer isso usando a `ApplicationNetworkPolicy` ou `NetworkPolicy`. Por fim, as regras do nível Linha de Base, que definem as restrições de rede padrão para as workloads do cluster, serão aplicadas. Essas regras do nível de Linha de Base podem ser substituídas pelas políticas com escopo de namespace, se necessário.

Exemplo

Imagine que você tem uma aplicação no cluster e deseja isolá-la das workloads pertencentes a outros locatários. É possível bloquear explicitamente o tráfego do cluster proveniente de outros namespaces para impedir o acesso de rede ao namespace da workload sensível.

```
apiVersion: networking.k8s.aws/v1alpha1
kind: ClusterNetworkPolicy
metadata:
  name: protect-sensitive-workload
spec:
  tier: Admin
  priority: 10
  subject:
    namespaces:
      matchLabels:
        kubernetes.io/metadata.name: earth
  ingress:
    - action: Deny
      from:
      - namespaces:
          matchLabels: {} # Match all namespaces.
      name: select-all-deny-all
```

Considerações

Compreensão da ordem de avaliação das políticas

As funcionalidades da política de rede disponíveis no EKS seguem uma ordem de avaliação específica, garantindo que o gerenciamento do tráfego seja sempre seguro e previsível. Dessa forma, compreender o fluxo de avaliação é essencial para projetar uma estratégia de segurança de rede eficiente em seu ambiente.

1. Políticas do nível de Administrador (avaliadas primeiro): todas as ClusterNetworkPolicies do nível de Administrador são avaliadas antes de quaisquer outras políticas. No nível de Administrador, o processamento das políticas segue a ordem de prioridade (o menor número de prioridade tem precedência). O tipo de ação determina o que acontece em seguida no processamento.
   - Ação de negação (maior precedência): quando uma política de Administrador com uma ação de negação corresponde ao tráfego, esse tráfego é bloqueado imediatamente, independentemente de quaisquer outras políticas. O processamento de quaisquer outras regras de ClusterNetworkPolicy ou NetworkPolicy é interrompido. Isso garante que os controles de segurança de nível organizacional não possam ser substituídos por políticas de nível de namespace.
   - Ação de permissão: uma vez avaliadas as regras de negação, as políticas de Administrador com a ação de permissão entram em processamento seguindo a ordem de prioridade (o menor número de prioridade tem precedência). Se o tráfego coincidir com uma regra de permissão, ele

é liberado imediatamente e o processo de avaliação de políticas é encerrado. Essas políticas podem conceder acesso entre diversos namespaces com base em seletores de rótulos, oferecendo controle centralizado sobre quais workloads podem acessar recursos específicos.

- Ação de aprovação: as ações de aprovação nas políticas do nível de Administrador delegam a tomada de decisão para os níveis inferiores. Quando o tráfego corresponde a uma regra de aprovação, a avaliação ignora todas as regras restantes do nível de Administrador para esse tráfego e prossegue diretamente para o nível de NetworkPolicy. Dessa forma, os administradores conseguem transferir formalmente o controle de padrões específicos de tráfego para as equipes responsáveis pelas aplicações. Por exemplo, é possível usar regras de aprovação para delegar o gerenciamento do tráfego interno do namespace aos administradores do namespace, enquanto mantém controles rígidos sobre o acesso externo.

2. Nível de política de rede: se nenhuma política do nível de Administrador corresponder às ações de negação ou de aprovação, ou se uma ação de aprovação for acionada, os recursos de ApplicationNetworkPolicy com escopo de namespace e as NetworkPolicy tradicionais são avaliados em seguida. Essas políticas oferecem controle granular nos namespaces individuais e são gerenciadas pelas equipes responsáveis pela aplicação. As políticas com escopo de namespace conseguem apenas aumentar a restrição imposta pelas políticas de Administrador. Eles não podem substituir a decisão de negação de uma política de Administrador, mas podem restringir ainda mais o tráfego que foi permitido ou aprovado pelas políticas de Administrador.
3. Políticas de Administrador do nível de Linha de Base: se nenhuma política do nível de Administrador ou de escopo de namespace corresponder ao tráfego, as ClusterNetworkPolicies do nível de Linha de Base são avaliadas. Estas políticas fornecem posturas de segurança padrão que podem ser substituídas por políticas com escopo de namespace, permitindo que os administradores definam padrões para toda a organização, enquanto oferecem flexibilidade às equipes para realizar personalizações conforme necessário. A avaliação das políticas de Linha de Base respeita a ordem de prioridade, na qual o menor número de prioridade tem precedência.
4. Negação padrão (se nenhuma política corresponder): este comportamento de negação por padrão garante que apenas conexões explicitamente permitidas sejam autorizadas, mantendo uma postura de segurança robusta.

Aplicação do princípio de privilégio mínimo

- Comece com políticas restritivas e adicione permissões gradualmente conforme necessário: inicie implementando políticas de negação por padrão no nível do cluster e, em seguida, adicione regras de permissão incrementalmente à medida que validar os requisitos legítimos de conectividade.

Essa abordagem obriga as equipes a justificarem explicitamente cada conexão externa, criando um ambiente mais seguro e auditável.

- Audite regularmente e remova regras de política não utilizadas: as políticas de rede podem se acumular com o tempo à medida que as aplicações evoluem, deixando para trás regras obsoletas que expandem desnecessariamente a superfície de ataque. Implemente um processo de análise regular para identificar e remover regras de política que não são mais necessárias, garantindo que sua postura de segurança permaneça rígida e sustentável.
- Use nomes de domínio específicos em vez de padrões abrangentes sempre que possível: embora padrões com caracteres curinga, como `*.amazonaws.com`, ofereçam conveniência, eles também concedem acesso a uma ampla variedade de serviços. Sempre que viável, especifique nomes de domínio exatos, como `s3.us-west-2.amazonaws.com`, para limitar o acesso apenas aos serviços específicos que suas aplicações exigem, reduzindo o risco de movimentação lateral caso uma workload seja comprometida.

Uso de políticas baseadas em DNS no EKS

- As regras baseadas em DNS, definidas por meio da `ApplicationNetworkPolicy`, são aplicáveis somente a workloads executadas em instâncias do EC2 iniciadas pelo modo automático do EKS. Se você estiver operando um cluster em modo misto (composto por nós de processamento do modo automático do EKS e sem ser nós do modo automático do EKS), as regras baseadas em DNS serão eficazes apenas nos nós de processamento do modo do EKS (instâncias gerenciadas do EC2).

Validação das políticas de DNS

- Use clusters de preparação que espelhem a topologia de rede da produção para testes: o ambiente de preparação deve replicar a arquitetura de rede, as dependências externas e os padrões de conectividade da produção para garantir testes de política precisos. Isso inclui configurações de VPC compatíveis, comportamento de resolução de DNS e acesso aos mesmos serviços externos que as workloads de produção exigem.
- Implemente testes automatizados para caminhos de rede críticos: desenvolva testes automatizados que validem a conectividade com serviços externos essenciais como parte do pipeline de CI/CD. Esses testes devem verificar se os fluxos de tráfego legítimos são permitidos enquanto as conexões não autorizadas são bloqueadas, fornecendo validação contínua de que as políticas de rede mantêm a postura de segurança correta à medida que sua infraestrutura evolui.

- Monitore o comportamento da aplicação após alterações de política: após implantar políticas de rede novas ou modificadas em ambientes de produção, monitore minuciosamente os logs da aplicação, as taxas de erro e as métricas de performance para identificar rapidamente quaisquer problemas de conectividade. Estabeleça procedimentos claros de reversão para que você possa reverter rapidamente as alterações de política caso elas causem comportamento inesperado da aplicação ou interrupções de serviço.

Interação com o firewall de DNS do Amazon Route 53

A avaliação das políticas de Administrador e de rede do EKS ocorre primeiro no nível do pod, assim que o tráfego é gerado. Se uma política de rede do EKS permitir a saída para um domínio específico, o pod realiza então uma consulta ao DNS que é encaminhada ao Route 53 Resolver. Nesse ponto, as regras do firewall de DNS do Route 53 são avaliadas. Se o firewall de DNS bloquear a consulta do domínio, a resolução de DNS falha e a conexão não pode ser estabelecida, mesmo que a política de rede do EKS a tenha permitido. Isso cria camadas de segurança complementares. As políticas de rede baseadas em DNS do EKS fornecem controle de saída em nível de pod para requisitos de acesso específicos da aplicação e limites de segurança multilocatários, enquanto o firewall de DNS oferece proteção em toda a VPC contra domínios mal-intencionados conhecidos e aplica listas de bloqueio em toda a organização.

Marcar sub-redes para o Modo Automático do EKS

Se você usa o recurso de balanceamento de carga do Modo Automático do EKS, é necessário adicionar tags da AWS às sub-redes da VPC.

Contexto

Essas tags identificam as sub-redes como associadas ao cluster e, o mais importante, se a sub-rede é pública ou privada.

Sub-redes públicas têm acesso direto à internet por meio de um gateway da internet. Elas são usadas para recursos que precisam estar acessíveis ao público, como balanceadores de carga.

As sub-redes privadas não têm acesso direto à internet e usam gateways NAT para tráfego de saída. Elas são usadas para recursos internos, como nós do EKS que não precisam de IPs públicos.

Para saber mais sobre gateways NAT e gateways da internet, consulte Conectar sua VPC a outras redes no Guia do usuário da Amazon Virtual Private Cloud (VPC).

Requisito

No momento, as sub-redes usadas para balanceamento de carga pelo Modo Automático do EKS precisam ter uma das tags a seguir.

Sub-redes públicas

As sub-redes públicas são usadas para balanceadores de carga voltados para a internet. Essas sub-redes devem ter as tags a seguir:

Chave	Valor
`kubernetes.io/role/elb`	1 ou ``

Sub-redes privadas

As sub-redes privadas são usadas para balanceadores de carga internos. Essas sub-redes devem ter as tags a seguir:

Chave	Valor
`kubernetes.io/role/internal-elb`	1 ou ``

Procedimento

Antes de começar, identifique quais sub-redes são públicas (com acesso ao gateway da internet) e quais são privadas (usando o gateway NAT). Você precisará de permissões para modificar os recursos da VPC.

Console de gerenciamento da AWS

1. Abra o console da Amazon VPC e navegue até Sub-redes.
2. Selecione a sub-rede a ser marcada.
3. Escolha a guia Tags e selecione Adicionar tag.
4. Adicione a tag apropriada:
   - Para sub-redes públicas: Key=`kubernetes.io/role/elb`

- Para sub-redes privadas: Key=`kubernetes.io/role/internal-elb`

5. Defina Value como 1 ou deixe em branco.
6. Salve e repita para as sub-redes restantes.

AWS CLI

Para sub-redes públicas:

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/elb,Value=1
```

Para sub-redes privadas:

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/internal-elb,Value=1
```

Substitua `subnet-ID` pelo ID da sub-rede atual.

Gerar relatórios de conformidade do CIS de nós do Kubernetes usando o comando kubectl debug

Este tópico descreve como gerar relatórios de conformidade do CIS (Center for Internet Security) para os nós do Amazon EKS usando o comando `kubectl debug`. O comando permite criar temporariamente um contêiner de depuração em um nó do Kubernetes e executar verificações de conformidade do CIS usando a ferramenta `apiclient`. A ferramenta `apiclient` faz parte do sistema operacional do Bottlerocket, o sistema operacional usado pelos nós do Modo Automático do EKS.

Pré-requisitos

Antes de começar, verifique se você tem:

- Acesso a um cluster do Amazon EKS com o `kubectl` configurado (a versão deve ser pelo menos v1.32.0; digite `kubectl version` para verificar).
- As permissões apropriadas do IAM para depurar nós.
- Um perfil válido que permita operações de depuração (por exemplo, `sysadmin`).

Para obter mais informações sobre como usar perfis de depuração com o `kubectl`, consulte Depuração de um Pod ou Nó aplicando um perfil na documentação do Kubernetes.

Procedimento

1. Determine o ID da instância da AWS do nó em que você deseja executar o relatório. Use o comando a seguir para listar todos os nós no cluster. O ID da instância pode ser encontrado na coluna de nome e começa com `i-`:

```
kubectl get nodes
```

```
NAME                  STATUS   ROLES    AGE   VERSION
i-0ea0ba0f8ef9ad609   Ready    <none>   62s   v1.30.10-eks-1a9dacd
```

2. Execute o seguinte comando, substituindo `<instance-id>` pelo ID da instância do nó que você deseja consultar:

```
kubectl debug node/<instance-id> -it --profile=sysadmin --image=public.ecr.aws/
amazonlinux/amazonlinux:2023 -- bash -c "yum install -q -y util-linux-core; nsenter -
t 1 -m apiclient report cis --level 1 --format text"
```

   Os componentes desse comando incluem:

   - `kubectl debug node/<instance-id>`: cria uma sessão de depuração no ID da instância especificada do EC2.
   - `-it`: aloca um TTY (shell de linha de comando) e mantém o stdin aberto para uso interativo.
   - `--profile=sysadmin`: usa o perfil especificado do `kubectl` com as permissões apropriadas.
   - `--image=public.ecr.aws/amazonlinux/amazonlinux:2023`: usa `amazonlinux:2023` como a imagem do contêiner para depuração.
   - `bash -c "…"` : executa os seguintes comandos em um shell bash:
     - `yum install -q -y util-linux-core`: instala silenciosamente o pacote de utilitários necessário.
     - `nsenter -t 1 -m`: executa `nsenter` para inserir o namespace do processo de host (PID 1).
     - `apiclient report cis --level 1 --format text`: executa o relatório de conformidade do CIS no nível 1 com saída de texto.

3. Analise a saída de texto do relatório.

Interpretar a saída

O comando gera um relatório baseado em texto mostrando o status de conformidade de vários controles do CIS. A saída inclui:

- IDs individuais de controles do CIS
- Descrição de cada controle
- Status de aprovado, reprovado ou ignorado para cada verificação
- Detalhes que explicam quaisquer problemas de conformidade

Veja abaixo um exemplo da saída do relatório executado em uma instância do Bottlerocket:

```
Benchmark name:  CIS Bottlerocket Benchmark
Version:         v1.0.0
Reference:       https://www.cisecurity.org/benchmark/bottlerocket
Benchmark level: 1
Start time:      2025-04-11T01:40:39.055623436Z

[SKIP] 1.2.1     Ensure software update repositories are configured (Manual)
[PASS] 1.3.1     Ensure dm-verity is configured (Automatic)[PASS] 1.4.1     Ensure
 setuid programs do not create core dumps (Automatic)
[PASS] 1.4.2     Ensure address space layout randomization (ASLR) is enabled
 (Automatic)
[PASS] 1.4.3     Ensure unprivileged eBPF is disabled (Automatic)
[PASS] 1.5.1     Ensure SELinux is configured (Automatic)
[SKIP] 1.6       Ensure updates, patches, and additional security software are
 installed (Manual)
[PASS] 2.1.1.1   Ensure chrony is configured (Automatic)
[PASS] 3.2.5     Ensure broadcast ICMP requests are ignored (Automatic)
[PASS] 3.2.6     Ensure bogus ICMP responses are ignored (Automatic)
[PASS] 3.2.7     Ensure TCP SYN Cookies is enabled (Automatic)
[SKIP] 3.4.1.3   Ensure IPv4 outbound and established connections are configured
 (Manual)
[SKIP] 3.4.2.3   Ensure IPv6 outbound and established connections are configured
 (Manual)
[PASS] 4.1.1.1   Ensure journald is configured to write logs to persistent disk
 (Automatic)
[PASS] 4.1.2     Ensure permissions on journal files are configured (Automatic)

Passed:          11
Failed:          0
```

```
Skipped:         4
Total checks:    15
```

Para obter informações sobre a avaliação comparativa, consulte Kubernetes Benchmark do Center for Internet Security (CIS).

Recursos relacionados

- Bottlerocket CIS Benchmark na documentação do sistema operacional do Bottlerocket.
- Depuração de Pods em Execução na documentação do Kubernetes.
- Kubernetes Benchmark do Center for Internet Security (CIS)

Habilitar a criptografia de volume do EBS com chaves do KMS gerenciadas pelo cliente para o modo automático do EKS

Você pode criptografar o volume raiz efêmero das instâncias do modo automático do EKS com uma chave do KMS gerenciada pelo cliente.

O modo automático do Amazon EKS usa perfis vinculados a serviços para delegar permissões a outros serviços da AWS ao gerenciar volumes criptografados do EBS para seus clusters do Kubernetes. Este tópico descreve como configurar a política de chaves necessária para especificar uma chave gerenciada pelo cliente para criptografia do Amazon EBS com o modo automático do EKS.

Considerações:

- O modo automático do EKS não precisa de autorização adicional para usar a chave gerenciada pela AWS padrão para proteger os volumes criptografados em sua conta.
- Este tópico aborda a criptografia de volumes efêmeros, os volumes raiz para instâncias do EC2. Para obter mais informações sobre a criptografia de volumes de dados usados em workloads, consulte the section called “Criar StorageClass”.

Visão geral

As seguintes chaves do AWS KMS podem ser usadas para criptografia do volume raiz do Amazon EBS quando o modo automático do EKS executa instâncias:

- Chave gerenciada pela AWS: uma chave de criptografia em sua conta que é criada por, pertencente a e é gerenciada pelo Amazon EBS. Essa é a chave de criptografia padrão para uma nova conta.
- Chave gerenciada pelo cliente: uma chave de criptografia personalizada que você cria, possui e gerencia.

> **Note**
>
> A chave deve ser simétrica. O Amazon EBS não oferece suporte a chaves gerenciadas pelo cliente assimétricas.

Etapa 1: configurar a política de chave

Suas chaves do KMS devem ter uma política de chaves que permita que o modo automático do EKS execute instâncias com volumes do Amazon EBS criptografados com uma chave gerenciada pelo cliente.

Configure sua política de chaves com a seguinte estrutura:

> **Note**
>
> Esta política inclui apenas permissões para o modo automático do EKS. A política de chaves poderá precisar de permissões adicionais se outras identidades precisarem usar a chave ou gerenciar concessões.

```
{
    "Version":"2012-10-17",
    "Id": "MyKeyPolicy",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:role/ClusterServiceRole"
                ]
```

```
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:role/ClusterServiceRole"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

Certifique-se de substituir <account-id> pelo ID real da sua conta da AWS.

Ao configurar a política de chaves:

- O `ClusterServiceRole` deve ter as permissões do IAM necessárias para usar a chave do KMS para operações de criptografia
- A condição `kms:GrantIsForAWSResource` garante que concessões só possam ser criadas para serviços da AWS

Etapa 2: configurar o NodeClass com sua chave gerenciada pelo cliente

Depois de configurar a política de chaves, faça referência à chave do KMS na configuração NodeClass do modo automático do EKS:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Insert existing configuration

  ephemeralStorage:
    size: "80Gi"  # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000    # Range: 3000-16000
    throughput: 125  # Range: 125-1000

    # KMS key for encryption
    kmsKeyID: "arn:aws:kms:<region>:<account-id>:key/<key-id>"
```

Substitua os valores dos espaços reservados por seus valores reais:

- `<region>` com sua região da AWS
- `<account-id>` com o ID da sua conta da AWS
- `<key-id>` com o ID da sua chave do KMS

Você pode especificar a chave do KMS usando qualquer um dos seguintes itens:

- ID da chave do KMS: `1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- ARN da chave KMS: `arn:aws:kms:us-west-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- Nome do alias da chave: `alias/eks-auto-mode-key`
- ARN do alias da chave: `arn:aws:kms:us-west-2:111122223333:alias/eks-auto-mode-key`

Aplique a configuração de NodeClass usando kubectl:

```
kubectl apply -f nodeclass.yaml
```

Recursos relacionados

- Criar uma classe de nó para o Amazon EKS
- Veja mais informações no Guia do desenvolvedor do AWS Key Management Service.
  - Permissões para serviços da AWS em políticas de chaves
  - Alterar uma política de chave
  - Concessões no AWS KMS

Atualizar os controles da organização para o Modo Automático do EKS

Alguns controles organizacionais podem impedir que o Modo Automático do EKS funcione corretamente. Nesse caso, você deve atualizar esses controles para permitir que o Modo Automático do EKS tenha as permissões necessárias para gerenciar instâncias do EC2 em seu nome.

O Modo Automático do EKS usa um perfil de serviço para inicializar as instâncias do EC2 que dão suporte aos nós do Modo Automático do EKS. Um perfil de serviço é um perfil do IAM criado em sua conta que um serviço assume para realizar ações em seu nome na sua conta. As políticas de controle de serviço (SCPs) sempre se aplicam às ações realizadas com perfis de serviço. Isso permite que uma SCP iniba as operações do Modo Automático. A ocorrência mais comum é quando uma SCP é usada para restringir as imagens de máquina da Amazon (AMIs) que podem ser executadas. Para permitir que o Modo Automático do EKS funcione, modifique a SCP para permitir a execução de AMIs de contas do Modo Automático do EKS.

Você também pode usar o recurso de AMIs permitidas pelo EC2 para limitar a visibilidade das AMIs em outras contas. Se usar esse recurso, você deverá expandir os critérios de imagens para incluir também as contas das AMIs do Modo Automático do EKS nas regiões de interesse.

Exemplo de SCP para bloquear todas as AMIs, exceto as AMIs do Modo Automático do EKS

A SCP abaixo impede chamadas a `ec2:RunInstances`, a menos que a AMI pertença à conta da AMI do Modo Automático do EKS para us-west-2 ou us-east-1.

> **Note**
>
> É importante não usar a chave de contexto `ec2:Owner`. A Amazon possui as contas das AMIs do Modo Automático do EKS, e o valor dessa chave sempre será `amazon`. A criação

de uma SCP que permita a execução de AMIs, se o `ec2:Owner` for `amazon`, permitirá a execução de qualquer AMI de propriedade da Amazon, não apenas as do Modo Automático do EKS.*

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAMI",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:*:ec2:*::image/ami-*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "767397842682",
            "992382739861"
          ]
        }
      }
    }
  ]
}
```

Contas de AMIs do Modo Automático do EKS

Contas da AWS que variam de acordo com a região hospedam as AMIs públicas do Modo Automático do EKS.

Região da AWS	Conta
af-south-1	471112993317
ap-east-1	590183728416
ap-northeast-1	851725346105
ap-northeast-2	992382805010
ap-northeast-3	891377407544

ap-south-1	975049899075
ap-south-2	590183737426
ap-southeast-1	339712723301
ap-southeast-2	58264376476
ap-southeast-3	471112941769
ap-southeast-4	590183863144
ap-southeast-5	654654202513
ap-southeast-7	533267217478
ca-central-1	992382439851
ca-west-1	767397959864
eu-central-1	891376953411
eu-central-2	381492036002
eu-north-1	339712696471
eu-south-1	975049955519
eu-south-2	471112620929
eu-west-1	381492008532
eu-west-2	590184142468
eu-west-3	891376969258
il-central-1	590183797093
me-central-1	637423494195
me-south-1	905418070398

mx-central-1	211125506622
sa-east-1	339712709251
us-east-1	992382739861
us-east-2	975050179949
us-west-1	975050035094
us-west-2	767397842682

Associação de um endereço IP público

Quando `ec2:RunInstances` é chamado, o campo `AssociatePublicIpAddress` para o lançamento de uma instância é determinado automaticamente pelo tipo de sub-rede na qual a instância está sendo lançada. Um SCP pode ser usado para garantir que esse valor seja explicitamente definido como falso, independentemente do tipo de sub-rede na qual está sendo iniciada. Nesse caso, o campo `spec.advancedNetworking.associatePublicIPAddress` do NodeClass também pode ser definido como falso para satisfazer os requisitos do SCP.

```
{
      "Sid": "DenyPublicEC2IPAddesses",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
          "BoolIfExists": {
              "ec2:AssociatePublicIpAddress": "true"
          }
      }
  }
```

Controle da implantação de workloads em reservas de capacidade com o modo automático do EKS

É possível controlar a implantação de workloads em reservas de capacidade. O modo automático do EKS oferece suporte a reservas de capacidade sob demanda (ODCRs, na sigla em inglês) do EC2 e a blocos de capacidade do EC2 para ML.

> **Tip**
>
> Por padrão, o modo automático do EKS é iniciado automaticamente em ODCRs abertos e blocos de capacidade para ML. Ao usar `capacityReservationSelectorTerms` na definição de NodeClass, o modo automático do EKS deixará de usar automaticamente quaisquer reservas de capacidade em aberto.

Reservas de capacidade sob demanda (ODCRs) do EC2

As reservas de capacidade sob demanda (ODCRs) do EC2 permitem que você reserve capacidade computacional para suas instâncias do Amazon EC2 em uma determinada zona de disponibilidade por qualquer duração. Ao usar o Modo Automático do EKS, talvez você queira controlar se suas workloads do Kubernetes serão implantadas nessas instâncias reservadas para maximizar a utilização da capacidade pré-adquirida ou para garantir que workloads críticas tenham acesso aos recursos garantidos.

Por padrão, o Modo Automático do EKS é inicializado automaticamente em ODCRs abertas. No entanto, ao configurar `capacityReservationSelectorTerms` em um NodeClass, você pode controlar explicitamente quais ODCRs suas workloads usam. Os nós provisionados usando ODCRs configuradas terão `karpenter.sh/capacity-type: reserved` e serão priorizados em relação aos sob demanda e spot. Depois que esse recurso for habilitado, o Modo Automático do EKS não usará mais automaticamente ODCRs abertas. Elas deverão ser selecionadas explicitamente por um NodeClass, oferecendo controle preciso sobre o uso da reserva de capacidade em todo o cluster.

> **Warning**
>
> Se você configurar `capacityReservationSelectorTerms` em um NodeClass em um cluster, o Modo Automático do EKS não usará mais automaticamente ODCRs abertas para nenhum NodeClass no cluster.

NodeClass de exemplo

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
spec:
  # Optional: Selects upon on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
```

```
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        app: "my-app"
      # Optional owning account ID filter
      owner: "012345678901"
```

Este exemplo de NodeClass demonstra duas abordagens para selecionar ODCRs. O primeiro método faz referência direta a uma ODCR específica pelo seu ID (`cr-56fac701cc1951b03`). O segundo método usa a seleção baseada em tags, visando ODCRs com a tag `Name: "targeted-odcr"`. Você também pode, opcionalmente, filtrar pela conta da AWS proprietária da reserva, o que é particularmente útil em cenários entre contas ou ao trabalhar com reservas de capacidade compartilhada.

Blocos de capacidade do EC2 para ML

Os blocos de capacidade para ML reservam instâncias de computação acelerada baseadas em GPU em uma data futura para fornecer suporte às workloads de machine learning (ML) de curta duração. As instâncias executadas em um bloco de capacidade são automaticamente colocadas próximas umas das outras nos Amazon EC2 UltraClusters para redes sem bloqueio de baixa latência com escala de petabits.

Para obter mais informações sobre as plataformas e os tipos de instância compatíveis, consulte Blocos de capacidade para ML no Guia do usuário do EC2.

Você pode criar uma NodeClass para o modo automático do EKS que usa um bloco de capacidade para ML, semelhante a um ODCR (descrito anteriormente).

As seguintes definições de amostra criam três recursos:

1. Uma NodeClass que faz referência à sua reserva de bloco de capacidade
2. Um NodePool que usa a NodeClass e aplica um taint
3. Uma especificação de pod com tolerância a taint e que solicita recursos de GPU

NodeClass de exemplo

Esta NodeClass faz referência a um bloco de capacidade específico para ML por meio do ID de reserva. Você pode obter esse ID no console do EC2.

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu
spec:
  # Specify your Capacity Block reservation ID
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
```

Para obter mais informações, consulte the section called “Criar classe de nó”.

Exemplo de NodePool

Este NodePool se refere a NodeClass da gpu e especifica configurações importantes:

- Utiliza apenas a capacidade reservada, definindo `karpenter.sh/capacity-type: reserved`
- Solicita famílias de instâncias de GPU específicas, apropriadas para workloads de ML
- Aplica um taint `nvidia.com/gpu` para garantir que apenas workloads de GPU sejam agendadas nesses nós

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: gpu
      requirements:
        - key: eks.amazonaws.com/instance-family
          operator: In
          values:
            - g6
            - p4d
            - p4de
            - p5
            - p5e
            - p5en
```

```
            - p6
            - p6-b200
        - key: karpenter.sh/capacity-type
          operator: In
          values:
            - reserved
            # Enable other capacity types
            # - on-demand
            # - spot
      taints:
        - effect: NoSchedule
          key: nvidia.com/gpu
```

Para obter mais informações, consulte the section called “Criar grupo de nós”.

Exemplo de pod

Este exemplo de pod demonstra como configurar uma workload para ser executada nos nós do bloco de capacidade:

- Usa um nodeSelector para direcionar tipos específicos de GPU (neste caso, GPUs H200)
- Inclui uma tolerância para o taint `nvidia.com/gpu` aplicado pelo NodePool
- Solicita explicitamente recursos de GPU usando o tipo de recurso `nvidia.com/gpu`

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  nodeSelector:
    # Select specific GPU type - uncomment as needed
    # eks.amazonaws.com/instance-gpu-name: l4
    # eks.amazonaws.com/instance-gpu-name: a100
    eks.amazonaws.com/instance-gpu-name: h200
    # eks.amazonaws.com/instance-gpu-name: b200
    eks.amazonaws.com/compute-type: auto
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
    args:
```

```
    - "nvidia-smi"
    resources:
      requests:
        # Uncomment if needed
        # memory: "30Gi"
        # cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        # Uncomment if needed
        # memory: "30Gi"
        nvidia.com/gpu: 1
  tolerations:
  - key: nvidia.com/gpu
    effect: NoSchedule
    operator: Exists
```

Para obter mais informações, consulte Pods na documentação do Kubernetes.

Recursos relacionados

- Blocos de capacidade para ML no Guia do usuário do Amazon EC2
- Encontrar e comprar blocos de capacidade no Guia do usuário do Amazon EC2
- Gerencie recursos computacionais para workloads de IA/ML no Amazon EKS
- Otimização de recursos de GPU e gerenciamento de custos no Guia de práticas recomendadas do EKS

Implantação de nós do modo automático do EKS em zonas locais

O modo automático do EKS fornece gerenciamento simplificado de clusters com provisionamento automático de nós. As zonas locais da AWS ampliam a infraestrutura da AWS para locais geográficos mais próximos dos seus usuários finais, reduzindo a latência para aplicações sensíveis a esse atraso. Este guia orienta você no processo de implantação de nós do modo automático do EKS em zonas locais da AWS, permitindo a execução de aplicações conteinerizadas com menor latência para usuários em áreas geográficas específicas.

Este guia também demonstra como usar taints e tolerâncias do Kubernetes para garantir que apenas workloads específicas sejam executadas em nós das zonas locais, ajudando você a controlar custos e otimizar o uso de recursos.

Pré-requisitos

Antes de começar a implantar nós do modo automático do EKS em zonas locais, certifique-se de que os seguintes pré-requisitos estejam configurados:

- Um cluster existente no modo automático do EKS
- Zonas locais ativadas em sua conta da AWS

Etapa 1: criação de uma sub-rede na zona local

A primeira etapa para a implantação de nós do modo automático do EKS em uma zona local é criar uma sub-rede nessa zona local. Essa sub-rede fornece a infraestrutura de rede aos seus nós e permite que eles se comuniquem com o restante da VPC. Siga as instruções presentes em Criação de uma sub-rede em uma zona local (no Guia do usuário das zonas locais da AWS) para criar uma sub-rede na zona local escolhida.

> **Tip**
>
> Anote o nome da sua sub-rede da zona local.

Etapa 2: criação de uma NodeClass para a sub-rede na zona local

Após criar a sub-rede da zona local, é necessário definir uma NodeClass que faça referência a essa sub-rede. A NodeClass é um recurso personalizado do Kubernetes que especifica os atributos de infraestrutura para os seus nós, incluindo quais sub-redes, grupos de segurança e configurações de armazenamento devem ser usados. No exemplo abaixo, criamos uma NodeClass chamada “local-zone” que direciona para uma sub-rede da zona local com base em seu nome. Você também pode usar o ID da sub-rede. Será necessário adaptar essa configuração para direcionar para a sua sub-rede da zona local.

Para obter mais informações, consulte the section called “Criar classe de nó”.

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: local-zone
spec:
```

```
  subnetSelectorTerms:
    - id: <local-subnet-id>
```

Etapa 3: criação do NodePool com NodeClass e Taint

Com sua NodeClass configurada, agora você precisa criar um NodePool que utilize essa NodeClass. Um NodePool define as características de computação dos seus nós, incluindo os tipos de instância. O NodePool emprega a NodeClass como referência para determinar em qual local as instâncias devem ser iniciadas.

No exemplo abaixo, criamos um NodePool que faz referência à NodeClass “local-zone”. Também adicionamos um taint aos nós para garantir que apenas pods com uma tolerância correspondente possam ser agendados nesses nós das zonas locais. Isso é particularmente importante para nós das zonas locais, que normalmente têm custos mais elevados e devem ser usados apenas por workloads que se beneficiam especificamente da latência reduzida.

Para obter mais informações, consulte the section called “Criar grupo de nós”.

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: my-node-pool
spec:
  template:
    metadata:
      labels:
        node-type: local-zone
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: local-zone
      taints:
        - key: "aws.amazon.com/local-zone"
          value: "true"
          effect: NoSchedule

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
        - key: "eks.amazonaws.com/instance-cpu"
```

```
          operator: In
          values: ["4", "8", "16", "32"]
```

O taint com a chave `aws.amazon.com/local-zone` e o efeito `NoSchedule` garante que pods sem uma tolerância correspondente não sejam agendados nesses nós. Isso evita que workloads comuns sejam executadas acidentalmente na zona local, o que poderia gerar custos inesperados.

Etapa 4: implantação de workloads com tolerância e afinidade de nó

Para um controle ideal sobre a alocação de workloads em nós das zonas locais, use taints e tolerâncias e a afinidade de nó simultaneamente. Essa abordagem combinada oferece os seguintes benefícios:

1. Controle de custos: o taint garante que apenas pods com tolerâncias explícitas possam usar recursos das zonas locais, que podem ser mais caros.
2. Posicionamento garantido: a afinidade de nó garante que suas aplicações sensíveis à latência sejam executadas exclusivamente na zona local, e não em nós comuns do cluster.

A seguir, apresentamos um exemplo de uma Deployment configurada para ser executada especificamente em nós das zonas locais:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: low-latency-app
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: low-latency-app
  template:
    metadata:
      labels:
        app: low-latency-app
    spec:
      tolerations:
      - key: "aws.amazon.com/local-zone"
        operator: "Equal"
        value: "true"
        effect: "NoSchedule"
```

```
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: "node-type"
                operator: "In"
                values: ["local-zone"]
      containers:
      - name: application
        image: my-low-latency-app:latest
        resources:
          limits:
            cpu: "1"
            memory: "1Gi"
          requests:
            cpu: "500m"
            memory: "512Mi"
```

Esta Deployment conta com duas configurações de agendamento fundamentais:

1. A tolerância permite que os pods sejam agendados em nós que contam com o taint `aws.amazon.com/local-zone`.
2. O requisito de afinidade de nó garante que esses pods sejam executados apenas em nós com o rótulo `node-type: local-zone`.

Juntas, essas configurações garantem que sua aplicação sensível à latência seja executada apenas em nós das zonas locais, e que as aplicações comuns não consumam os recursos dessas zonas locais, a menos que sejam explicitamente configuradas para isso.

Etapa 5: verificação com o Console da AWS

Após configurar a NodeClass, o NodePool e as Deployments, você deve verificar se os nós estão sendo provisionados na zona local conforme o esperado e se as workloads estão sendo executadas neles. É possível usar o Console de Gerenciamento da AWS para verificar se as instâncias do EC2 estão sendo iniciadas na sub-rede mais adequada da zona local.

Além disso, você pode verificar a lista de nós do Kubernetes usando o comando `kubectl get nodes -o wide` para confirmar se os nós estão sendo integrados ao cluster com os rótulos e os taints configurados corretamente:

```
kubectl get nodes -o wide
kubectl describe node <node-name> | grep -A 5 Taints
```

Você também pode verificar se os pods das workloads estão agendados nos nós das zonas locais:

```
kubectl get pods -o wide
```

Essa abordagem garante que apenas workloads com tolerância específica para o taint da zona local sejam agendadas nesses nós, ajudando você a controlar custos e a fazer o uso mais eficiente dos seus recursos de zonas locais.

Saiba como o Modo Automático do EKS funciona

Use este capítulo para saber mais sobre como os componentes dos clusters do Modo Automático do Amazon EKS funcionam.

Tópicos

- Saiba mais sobre as instâncias gerenciadas do modo automático do Amazon EKS
- Saber mais sobre identidade e acesso no Modo Automático do EKS
- Saiba mais sobre as redes de VPC e sobre o balanceamento de carga no modo automático do EKS

Saiba mais sobre as instâncias gerenciadas do modo automático do Amazon EKS

Este tópico explica como o Modo automático do Amazon EKS gerencia as instâncias do Amazon EC2 no cluster do EKS. Quando você habilita o Modo automático do EKS, os recursos computacionais do cluster são automaticamente provisionados e gerenciados pelo EKS, mudando a forma como você interage com as instâncias do EC2 que servem como nós no cluster.

Entender como o Modo automático do Amazon EKS gerencia as instâncias é essencial para planejar a estratégia de implantação de workloads e procedimentos operacionais. Diferentemente das instâncias do EC2 tradicionais ou dos grupos de nós gerenciados, essas instâncias seguem um modelo de ciclo de vida diferente, em que o EKS assume a responsabilidade por muitos aspectos operacionais, ao mesmo tempo em que restringe certos tipos de acesso e personalização.

O Modo automático do Amazon EKS automatiza tarefas de rotina para criar novas instâncias do EC2 e as anexa como nós ao cluster do EKS. O Modo automático do EKS detecta quando uma workload não cabe nos nós existentes e cria uma instância do EC2.

O Modo automático do Amazon EKS é responsável por criar, excluir e aplicar patches em instâncias do EC2. Você é responsável pelos contêineres e pods implantados na instância.

As instâncias do EC2 criadas pelo Modo automático do EKS são diferentes das outras instâncias do EC2. Elas são instâncias gerenciadas. Essas instâncias gerenciadas são de propriedade do EKS e são mais restritas. Você não pode acessar diretamente ou instalar software em instâncias gerenciadas pelo Modo automático do EKS.

A AWS sugere executar o Modo automático do EKS ou o Karpenter autogerenciado. Você pode instalar ambos durante uma migração ou em uma configuração avançada. Se você tiver ambos instalados, configure os grupos de nós para que as workloads sejam associadas ao Karpenter ou ao Modo automático do EKS.

Para obter mais informações, consulte Amazon EC2 managed instance no Guia do usuário do Amazon EC2.

Tabela de comparação

Instância padrão do EC2	Instância gerenciada do Modo automático do EKS
Você é responsável por aplicar patches e atualizar a instância.	A AWS aplica patches e atualiza automatic amente a instância.
O EKS não é responsável pelo software na instância.	O EKS é responsável por determinados softwares na instância, como `kubelet`, o runtime do contêiner e o sistema operacional.
Você pode excluir a instância do EC2 usando a API do EC2.	O EKS determina o número de instância s implantadas na conta. Se você excluir uma workload, o EKS reduzirá o número de instâncias na conta.
Você pode usar o SSH para acessar a instância do EC2.	Você pode implantar pods e contêineres na instância gerenciada.

Instância padrão do EC2	Instância gerenciada do Modo automático do EKS
Você determina o sistema operacional e a imagem (AMI).	A AWS determina o sistema operacional e a imagem.
Você pode implantar workloads que dependam da funcionalidade do Windows ou do Ubuntu.	Você pode implantar contêineres baseados em Linux, mas sem dependências específicas do sistema operacional.
Você determina qual família e tipo de instância executar.	A AWS determina qual família e tipo de instância executar. Você pode usar um grupo de nós para limitar os tipos de instância selecionados pelo Modo automático do EKS.

A seguinte funcionalidade funciona tanto para instâncias gerenciadas quanto para instâncias padrão do EC2:

- Você pode visualizar a instância no console da AWS.
- Você pode usar o armazenamento de instâncias como armazenamento efêmero para workloads.

Compatibilidade com a AMI

Com o Modo automático do EKS, a AWS determina a imagem (AMI) usada para seus nós de computação. A AWS monitora o lançamento das novas versões de AMIs do Modo automático do EKS. Se você tiver problemas com as workloads relacionados a uma versão de AMI, crie um caso de suporte. Para obter mais informações, consulte Creating support cases and case management no Guia do usuário do AWS Support.

Geralmente, o EKS lança uma nova AMI toda semana contendo um CVE e correções de segurança.

Referência de instâncias compatíveis com o Modo automático do EKS

O Modo automático do EKS cria apenas instâncias de tipos compatíveis e que atendem a um requisito de tamanho mínimo.

O modo automático do EKS é compatível com os seguintes tipos de instância:

Família	Tipos de instância
Otimizada para computação (C)	c8g, c7a, c7g, c7gn, c7gd, c7i, c7i-flex, c6a, c6g, c6i, c6gn, c6id, c6in, c6gd, c5, c5a, c5d, c5ad, c5n, c4
Uso geral (M)	m8g, m7i, m7a, m7g, m7gd, m7i-flex, m6a, m6i, m6in, m6g, m6idn, m6id, m6gd, m5, m5a, m5ad, m5n, m5dn, m5d, m5zn, m4
Otimizada para memória (R)	r8g, r7a, r7iz, r7gd, r7i, r7g, r6a, r6i, r6id, r6in, r6idn, r6g, r6gd, r5, r5n, r5a, r5dn, r5b, r5ad, r5d, r4
Com capacidade de intermitência (T)	t4g, t3, t3a, t2
Alta memória (Z/X)	z1d, x8g, x2gd
Otimizada para armazenamento (I/D)	i8g, i7ie, i4g, i4i, i3, i3en, is4gen, d3, d3en, im4gn
Computação acelerada (P/G/Inf/Trn)	p5, p4d, p4de, p3, p3dn, gr6, g6, g6e, g5g, g5, g4dn, inf2, inf1, trn1, trn1n
Computação de alta performance (X2)	x2iezn, x2iedn, x2idn

Além disso, o Modo automático do EKS só criará instâncias do EC2 que atendam aos seguintes requisitos:

- Mais de uma CPU
- O tamanho da instância não é nano, micro ou pequeno

Para obter mais informações, consulte Convenções de nomenclatura de tipos de instância do Amazon EC2.

Serviço de metadados da instância

- O Modo automático do EKS impõe o IMDSv2 com um limite de salto de um por padrão, aderindo às práticas recomendadas de segurança da AWS.
- Essa configuração padrão não pode ser modificada no Modo automático.
- Para complementos que normalmente exigem acesso ao IMDS, forneça parâmetros (como região da AWS) durante a instalação para evitar consultas ao IMDS. Para obter mais informações, consulte the section called “Campos que você pode personalizar”.
- Se for absolutamente necessário que um pod tenha acesso ao IMDS ao ser executado no Modo automático, ele deverá ser configurado para ser executado com `hostNetwork: true`. Isso permite que o pod acesse diretamente o serviço de metadados da instância.
- Considere as implicações de segurança ao conceder aos pods acesso aos metadados da instância.

Para obter mais informações sobre o serviço de metadados de instância (IMDS) do Amazon EC2, consulte Configurar as opções de serviço de metadados de instância no Guia do usuário do Amazon EC2.

Considerações

- Se o armazenamento efêmero configurado no NodeClass for menor do que o armazenamento local NVMe da instância, o Modo automático do EKS eliminará a necessidade de configuração manual executando automaticamente as seguintes ações:
  - Usa um volume de dados menor (20 GiB) do Amazon EBS para reduzir custos.
  - Formata e configura o armazenamento local NVMe para uso efêmero de dados. Isso inclui a configuração de uma matriz RAID 0 se houver várias unidades NVMe.
- Quando `ephemeralStorage.size` for igual ou exceder a capacidade local do NVMe, as seguintes ações ocorrerão:
  - O modo automático ignora o pequeno volume de EBS.
  - Drives NVMe são diretamente expostos à sua workload.
- O modo automático do Amazon EKS não é compatível com as seguintes ações do Serviço de Injeção de Falhas da AWS:

- `ec2:RebootInstances`
- `ec2:SendSpotInstanceInterruptions`
- `ec2:StartInstances`
- `ec2:StopInstances`
- `ec2:TerminateInstances`
- `ec2:PauseVolumeIO`

- O modo automático do Amazon EKS é compatível com as ações de pod do EKS do Serviço de Injeção de Falhas da AWS. Para obter mais informações, consulte Gerenciamento de experimentos do Serviço de Injeção de Falhas e Uso das ações aws:eks:pod do AWS FIS no Guia do usuário do Hub de Resiliência da AWS.
- Você não precisa instalar o `Neuron Device Plugin` nos nós do Modo automático do EKS.

  Caso tenha outros tipos de nós no cluster, você precisará configurar o plug-in Neuron Device para não ser executado em nós do Modo automático. Para obter mais informações, consulte the section called “Controlar a implantação”.

Saber mais sobre identidade e acesso no Modo Automático do EKS

Este tópico descreve os perfis e as permissões do Identity and Access Management (IAM) necessários para usar o Modo Automático do EKS. O Modo Automático do EKS usa dois perfis principais do IAM: um perfil do IAM do cluster e um perfil do IAM do nó. Esses perfis trabalham em conjunto com a Identidade de Pods do EKS e as entradas de acesso do EKS para fornecer gerenciamento de acesso abrangente para os clusters do EKS.

Ao configurar o Modo Automático do EKS, você precisará configurar esses perfis do IAM com permissões específicas que permitam que os serviços da AWS interajam com os recursos do cluster. Isso inclui permissões para gerenciar recursos computacionais, volumes de armazenamento, balanceadores de carga e componentes de rede. Compreender essas configurações de perfil é essencial para a operação e a segurança adequadas do cluster.

No Modo Automático do EKS, os perfis do AWS IAM são mapeados automaticamente para as permissões do Kubernetes por meio de entradas de acesso do EKS, eliminando a necessidade de configuração manual de ConfigMaps `aws-auth` ou vinculações personalizadas. Quando você cria um novo cluster no modo automático, o EKS cria automaticamente as permissões correspondentes do Kubernetes usando entradas de acesso, garantindo que os componentes do cluster e serviços da AWS tenham os níveis de acesso apropriados nos sistemas de autorização da AWS e do

Kubernetes. Essa integração automatizada reduz a complexidade da configuração e ajuda a evitar problemas relacionados a permissões que geralmente ocorrem ao gerenciar clusters do EKS.

Função do IAM do cluster

O perfil do IAM do cluster é um perfil do AWS Identity and Access Management (IAM) usado pelo Amazon EKS para gerenciar permissões para clusters do Kubernetes. Esse perfil concede ao Amazon EKS as permissões necessárias para interagir com outros serviços da AWS em nome do cluster, e é configurado automaticamente com as permissões do Kubernetes usando entradas de acesso do EKS.

- Você também deve anexar políticas do AWS IAM ao perfil.
- O Modo Automático do EKS anexa permissões do Kubernetes a esse perfil automaticamente usando entradas de acesso do EKS.
- Com o Modo Automático do EKS, a AWS sugere a criação de um único perfil do IAM do cluster por conta da AWS.
- A AWS sugere nomear esse perfil `AmazonEKSAutoClusterRole`.
- Esse perfil exige permissões para vários serviços da AWS gerenciarem recursos, incluindo volumes do EBS, Elastic Load Balancers e instâncias do EC2.
- A configuração sugerida para esse perfil inclui várias políticas do IAM gerenciadas pela AWS, relacionadas aos diferentes recursos do Modo Automático do EKS.
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicy`
  - `AmazonEKSLoadBalancingPolicy`
  - `AmazonEKSNetworkingPolicy`
  - `AmazonEKSClusterPolicy`

Para obter mais informações sobre o perfil do IAM do cluster e políticas do IAM gerenciadas pela AWS, consulte:

- the section called “ Políticas gerenciadas pela AWS”
- the section called “Função do IAM do cluster”

Para obter mais informações sobre o acesso do Kubernetes, consulte:

- the section called “Analisar as políticas de acesso”

Perfil do IAM do nó

O perfil do IAM do nó é um perfil do AWS Identity and Access Management (IAM) usado pelo Amazon EKS para gerenciar permissões para nós de processamento em clusters do Kubernetes. Esse perfil concede às instâncias do EC2 executadas como nós do Kubernetes as permissões necessárias para interagir com os serviços e recursos da AWS, e é configurado automaticamente com permissões de RBAC do Kubernetes usando entradas de acesso do EKS.

- Você também deve anexar políticas do AWS IAM ao perfil.
- O Modo Automático do EKS anexa permissões de RBAC do Kubernetes a esse perfil automaticamente usando entradas de acesso do EKS.
- A AWS sugere nomear esse perfil `AmazonEKSAutoNodeRole`.
- Com o Modo Automático do EKS, a AWS sugere a criação de um único perfil do IAM do nó por conta da AWS.
- Esse perfil tem permissões limitadas. As principais permissões incluem assumir um perfil de Identidade de Pods e extrair imagens do ECR.
- A AWS sugere as seguintes políticas do IAM gerenciadas pela AWS:
  - `AmazonEKSWorkerNodeMinimalPolicy`
  - `AmazonEC2ContainerRegistryPullOnly`

Para obter mais informações sobre o perfil do IAM do cluster e políticas do IAM gerenciadas pela AWS, consulte:

- the section called “ Políticas gerenciadas pela AWS”
- the section called “Perfil do IAM do nó”

Para obter mais informações sobre o acesso do Kubernetes, consulte:

- the section called “Analisar as políticas de acesso”

Perfil vinculado a serviço

O Amazon EKS usa um perfil vinculado ao serviço (SLR) para determinadas operações. A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

A AWS cria e configura automaticamente o SLR. Você poderá excluir um SLR somente depois de primeiro excluir seus recursos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

A política de SLR concede ao Amazon EKS permissões para observar e excluir os principais componentes da infraestrutura: recursos do EC2 (instâncias, interfaces de rede, grupos de segurança), recursos do ELB (balanceadores de carga, grupos de destino), recursos do CloudWatch (registro em log e métricas) e perfis do IAM com o prefixo "eks". Também habilita a rede de endpoints privados por meio da associação da VPC e zona hospedada e inclui permissões para o monitoramento do EventBridge e limpeza dos recursos marcados com o EKS.

Para obter mais informações, consulte:

- the section called “Política gerenciada pela AWS: AmazonEKSServiceRolePolicy”
- the section called “Permissões de função vinculada ao serviço para o Amazon EKS”

Tags da AWS personalizadas para recursos do Modo Automático do EKS.

Por padrão, as políticas gerenciadas relacionadas ao Modo Automático do EKS não permitem a aplicação de tags definidas pelo usuário aos recursos da AWS provisionados pelo Modo Automático. Caso queira aplicar tags definidas pelo usuário aos recursos da AWS, você deverá anexar permissões adicionais ao perfil do IAM do cluster com permissões suficientes para criar e modificar tags nos recursos da AWS. Segue abaixo um exemplo de uma política que permitirá acesso irrestrito à marcação:

Visualizar exemplo de política de tags personalizadas

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Compute",
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:CreateFleet",
                "ec2:RunInstances",
                "ec2:CreateLaunchTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                },
                "StringLike": {
                    "aws:RequestTag/eks:kubernetes-node-class-name": "*",
                    "aws:RequestTag/eks:kubernetes-node-pool-name": "*"
                }
            }
        },
        {
            "Sid": "Storage",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVolume",
                "ec2:CreateSnapshot"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:snapshot/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "Networking",
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterface",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
```

```
                },
                "StringLike": {
                    "aws:RequestTag/eks:kubernetes-cni-node-name": "*"
                }
            }
        },
        {
            "Sid": "LoadBalancer",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateRule",
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "ShieldProtection",
            "Effect": "Allow",
            "Action": [
                "shield:CreateProtection"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "ShieldTagResource",
            "Effect": "Allow",
            "Action": [
                "shield:TagResource"
            ],
```

```
            "Resource": "arn:aws:shield::*:protection/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        }
    ]
}
```

Referência de política de acesso

Para obter mais informações sobre as permissões do Kubernetes usadas pelo Modo Automático do EKS, consulte the section called “Analisar as políticas de acesso”.

Saiba mais sobre as redes de VPC e sobre o balanceamento de carga no modo automático do EKS

Este tópico explica como configurar os recursos de balanceamento de carga e rede da Virtual Private Cloud (VPC) no Modo Automático do EKS. Embora o Modo Automático do EKS gerencie a maioria dos componentes de rede automaticamente, você ainda pode personalizar certos aspectos da configuração de rede do cluster por meio dos recursos do `NodeClass` e das anotações do balanceador de carga.

Quando você usa o Modo Automático do EKS, a AWS gerencia a configuração da interface de rede de contêineres (CNI) da VPC e o provisionamento de balanceador de carga para o cluster. Você pode influenciar os comportamentos de rede definindo objetos `NodeClass` e aplicando anotações específicas aos seus recursos de Service e Ingress, mantendo o modelo operacional automatizado fornecido pelo Modo Automático do EKS.

Funcionalidade de redes

O modo automático do EKS oferece uma nova funcionalidade de redes para gerenciar a rede de nós e de pods. É possível configurá-la criando um objeto `NodeClass` no Kubernetes.

As opções de configuração para o plug-in CNI da AWS VPC anterior não se aplicarão ao modo automático do EKS.

Configurar a rede com um `NodeClass`

O recurso `NodeClass` no Modo Automático do EKS possibilita a personalização de determinados aspectos dos recursos de redes. Por meio do `NodeClass`, você pode especificar seleções de grupos de segurança, controlar o posicionamento dos nós em sub-redes da VPC, definir políticas de SNAT, configurar políticas de rede e habilitar o registro em log de eventos de rede. Essa abordagem mantém o modelo operacional automatizado do Modo Automático do EKS e, ao mesmo tempo, fornece flexibilidade para a personalização da rede.

Você pode usar um `NodeClass` para:

- Selecionar um grupo de segurança para nós
- Controlar como os nós são posicionados nas sub-redes da VPC
- Definir a política de SNAT do nó como `random` ou `disabled`
- Habilitar as políticas de rede do Kubernetes, incluindo:
  - Definir a política de rede como Negar por padrão ou Permitir por padrão
  - Habilite o registro em log de eventos de rede em um arquivo.
- Isole o tráfego de pods do tráfego de nós anexando os pods a sub-redes diferentes.

Saiba como Criar um NodeClass do Amazon EKS.

Considerações

Modo Automático do EKS é compatível com:

- Políticas de rede do EKS.
- As opções `HostPort` e `HostNetwork` para pods do Kubernetes.
- Pods e nós em sub-redes públicas ou privadas.
- Armazenamento em cache de consultas ao DNS no nó.

O Modo Automático do EKS não é compatível com:

- Grupos de segurança por pod (SGPP).
- Rede personalizada no `ENIConfig`. Você pode colocar pods em várias sub-redes ou isolá-los exclusivamente do tráfego de nós com the section called “Seleção de sub-rede para pods”.

- Configurações de IP warm, prefixo warm e ENI warm.
- Configuração de destinos de IP mínimos.
- Outras configurações compatíveis com a CNI de código aberto da AWS VPC.
- Configurações de política de rede, como personalização do temporizador conntrack (o padrão é 300 s).
- Exportação de logs de eventos de rede para o CloudWatch.

Gerenciamento de recursos de rede

O Modo Automático do EKS lida com o gerenciamento de prefixo, endereçamento IP e interface de rede monitorando os recursos do NodeClass para as configurações de rede. O serviço executa várias operações importantes automaticamente:

Delegação de prefixo

O modo automático do EKS utiliza por padrão a delegação de prefixos (prefixos /28) para a rede dos pods e mantém um grupo de recursos IP definido previamente que é escalado com base no número de pods agendados. Quando é detectada fragmentação da sub-rede do pod, o modo automático provisiona endereços IP secundários (prefixos /32). Devido a esse algoritmo de rede de pods padrão, o modo automático calcula o número máximo de pods por nó com base na quantidade de ENIs e de IPs com suporte por tipo de instância (considerando o pior cenário de fragmentação). Para obter mais informações sobre o número máximo de ENIs e de IPs com suporte por tipo de instância, consulte Máximo de endereços IP por interface de rede no Guia do usuário do EC2. As famílias de instâncias da geração mais recente (Nitro v6 e versões posteriores) geralmente têm um número maior de ENIs e de IPs com suporte por tipo de instância, e o modo automático ajusta o cálculo do número máximo de pods de acordo.

Para clusters IPv6, apenas a delegação de prefixo é utilizada, e o modo automático sempre usa um limite máximo de 110 pods por nó.

Gerenciamento do período de espera

O serviço implementa um grupo de espera para prefixos ou endereços IPv4 secundários que não estão mais em uso. Depois que o período de espera expirar, esses recursos serão liberados de volta para a VPC. No entanto, se os pods reutilizarem esses recursos durante o período de espera, eles serão restaurados do grupo de espera.

Compatibilidade com IPv6

Para clusters IPv6, o Modo Automático do EKS provisiona um prefixo IPv6 `/80` por nó na interface de rede primária.

O serviço também garante o gerenciamento adequado e a coleta de resíduos de todas as interfaces de rede.

Balanceamento de carga

Você configura os AWS Elastic Load Balancers provisionados pelo Modo Automático do EKS usando anotações nos recursos de Service e Ingress.

Para ter mais informações, consulte the section called “Criar classe de entrada” ou the section called “Create service (Criar serviço)”.

Considerações sobre o balanceamento de carga com o Modo Automático do EKS

- O modo de direcionamento padrão é o modo de IP, não o modo de instância.
- O Modo Automático do EKS é compatível apenas com o modo de grupo de segurança para Network Load Balancers.
- A AWS não é compatível com a migração de balanceadores de carga do controlador de balanceador de carga autogerenciado da AWS para o gerenciamento pelo Modo Automático do EKS.
- O campo `networking.ingress.ipBlock` na especificação `TargetGroupBinding` não é compatível.
- Se os nós de processamento usarem grupos de segurança personalizados (não o padrão de nomenclatura `eks-cluster-sg-`), o perfil do cluster precisará de permissões adicionais do IAM. A política padrão gerenciada pelo EKS só permite que o EKS modifique grupos de segurança denominados `eks-cluster-sg-`. Sem permissão para modificar os grupos de segurança personalizados, o EKS não pode adicionar as regras de entrada necessárias que permitem que o tráfego de ALB e NLB chegue aos pods.

Considerações sobre o CoreDNS

O modo automático do EKS não usa a implantação tradicional do CoreDNS para fornecer resolução de DNS dentro do cluster. Em vez disso, os nós do modo automático empregam o CoreDNS, que é executado como um serviço do sistema diretamente em cada nó. Ao migrar um cluster tradicional para o modo automático, você pode remover a implantação do CoreDNS do cluster assim que as workloads forem movidas para os nós do modo automático.

> **Important**
>
> Se você planeja manter um cluster com nós no modo automático e nós que não estejam no modo automático, deve manter a implantação do CoreDNS. Os nós que não operam no modo automático dependem dos pods do CoreDNS tradicionais para a resolução de DNS, pois não têm acesso ao serviço de DNS em nível de nó fornecido pelo modo automático.

Solucionar problemas do Modo Automático do EKS

Com o Modo Automático do EKS, a AWS assume mais responsabilidade pelas instâncias do EC2 na conta da AWS. O EKS assume a responsabilidade pelo runtime do contêiner nos nós, pelo sistema operacional nos nós e por determinados controladores. Isso inclui um controlador de armazenamento em blocos, um controlador de balanceamento de carga e um controlador de computação.

Você deve usar as APIs da AWS e do Kubernetes para solucionar problemas de nós. É possível:

- Use um recurso `NodeDiagnostic` do Kubernetes para recuperar os logs dos nós usando o the section called “Agente de monitoramento de nós”. Para conferir as etapas, consulte the section called “Obter logs de nós”.
- Use o comando `get-console-output` da CLI do AWS EC2 para recuperar a saída do console dos nós. Para conferir as etapas, consulte the section called “Obter a saída do console de uma instância gerenciada do EC2 usando a CLI do AWS EC2”.
- Use os contêineres de depuração do Kubernetes para recuperar os logs dos nós. Para conferir as etapas, consulte the section called “Obter logs de nós usando contêineres de depuração e a CLI do `kubectl`”.

> **Note**
>
> O Modo Automático do EKS usa instâncias gerenciadas pelo EC2. Você não pode acessar diretamente as instâncias gerenciadas pelo EC2, inclusive por SSH.

Você pode ter os seguintes problemas que têm soluções específicas para os componentes do Modo Automático do EKS:

- Pods presos no estado `Pending`, que não estão sendo agendados nos nós do Modo Automático. Para conferir as soluções, consulte the section called “Solucionar problemas de pods com falhas ao agendar o nó do Modo Automático”.
- Instâncias gerenciadas do EC2 que não se juntam ao cluster como nós do Kubernetes. Para conferir as soluções, consulte the section called “Solucionar problemas de nós que não estão se associando ao cluster”.
- Erros e problemas com `NodePools`, `PersistentVolumes` e `Services` que usam os controladores incluídos no Modo Automático do EKS. Para conferir as soluções, consulte the section called “Solucionar problemas de controladores incluídos no Modo Automático”.
- A segurança aprimorada de pods impede o compartilhamento de volumes entre pods. Para conferir as soluções, consulte the section called “Compartilhar volumes entre pods”.

Você pode usar os seguintes métodos para solucionar problemas de componentes do Modo Automático do EKS:

- the section called “Obter a saída do console de uma instância gerenciada do EC2 usando a CLI do AWS EC2”
- the section called “Obter logs de nós usando contêineres de depuração e a CLI do `kubectl`”
- the section called “Visualizar recursos associados ao Modo Automático do EKS no Console da AWS”
- the section called “Visualizar erros do IAM na conta da AWS”
- the section called “Detectar problemas de conectividade de nós com o `VPC Reachability Analyzer`”

Agente de monitoramento de nós

O Modo Automático do EKS inclui o agente de monitoramento de nós do Amazon EKS. Você pode usar esse agente para visualizar informações de solução de problemas e depuração sobre os nós. O agente de monitoramento de nós publica os `events` do Kubernetes e as `conditions` dos nós. Para obter mais informações, consulte the section called “Integridade de nós”.

Obter a saída do console de uma instância gerenciada do EC2 usando a CLI do AWS EC2

Este procedimento ajuda na solução de problemas de tempo de inicialização ou no nível do kernel.

Primeiro, você precisa determinar o ID da instância do EC2 associada à workload. Depois, use a AWS CLI para recuperar a saída do console.

1. Confirme se você tem o `kubectl` instalado e se está conectado ao cluster.
2. (Opcional) Use o nome de uma implantação do Kubernetes para listar os pods associados.

```
kubectl get pods -l app=<deployment-name>
```

3. Use o nome do pod do Kubernetes para determinar o ID da instância do EC2 do nó associado.

```
kubectl get pod <pod-name> -o wide
```

4. Use o ID da instância do EC2 para recuperar a saída do console.

```
aws ec2 get-console-output --instance-id <instance id> --latest --output text
```

Obter logs de nós usando contêineres de depuração e a CLI do **`kubectl`**

A forma recomendada de recuperar logs de um nó do Modo Automático do EKS é usar o recurso `NodeDiagnostic`. Para conferir essas etapas, consulte the section called “Obter logs de nós”.

No entanto, você pode transmitir logs ao vivo de uma instância usando o comando `kubectl debug node`. Esse comando inicia um novo pod no nó que você deseja depurar e que pode ser usado interativamente.

1. Inicie um contêiner de depuração. O comando a seguir usa `i-0123456789012345`6 para o ID da instância do nó, `-it` aloca um `tty` e anexa `stdin` para uso interativo e usa o perfil `sysadmin` do arquivo kubeconfig.

```
kubectl debug node/i-0123456789012345 -it --profile=sysadmin --image=public.ecr.aws/
amazonlinux/amazonlinux:2023
```

   Veja abaixo um exemplo de saída.

```
Creating debugging pod node-debugger-i-0123456789012345-nxb9c with container
 debugger on node i-0123456789012345.
If you don't see a command prompt, try pressing enter.
bash-5.2#
```

2. No shell, agora você pode instalar o `util-linux-core`, que fornece o comando `nsenter`. Use `nsenter` para inserir o namespace de montagem do PID 1 (`init`) no host e execute o comando `journalctl` para transmitir logs do `kubelet`:

```
yum install -y util-linux-core
nsenter -t 1 -m journalctl -f -u kubelet
```

Por segurança, a imagem do contêiner do Amazon Linux não instala muitos binários por padrão. Você pode usar o comando `yum whatprovides` para identificar o pacote que deve ser instalado para fornecer um determinado binário.

```
yum whatprovides ps
```

```
Last metadata expiration check: 0:03:36 ago on Thu Jan 16 14:49:17 2025.
procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo        : @System
Matched from:
Filename    : /usr/bin/ps
Provide    : /bin/ps

procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo        : amazonlinux
Matched from:
Filename    : /usr/bin/ps
Provide    : /bin/ps
```

Visualizar recursos associados ao Modo Automático do EKS no Console da AWS

Você pode usar o Console da AWS para visualizar o status dos recursos associados ao cluster do Modo Automático do EKS.

- Volumes do EBS
  - Visualize os volumes do Modo Automático do EKS pesquisando pela chave de tag `eks:eks-cluster-name`
- Load balancers

  - Visualize os balanceadores de carga do Modo Automático do EKS pesquisando pela chave de tag `eks:eks-cluster-name`
- Instâncias do EC2
  - Visualize as instâncias do Modo Automático do EKS pesquisando pela chave de tag `eks:eks-cluster-name`

Visualizar erros do IAM na conta da AWS

1. Navegue até o console do CloudTrail
2. Selecione "Histórico de eventos" no painel de navegação à esquerda
3. Aplique filtros de código de erro:
   - AccessDenied
   - UnauthorizedOperation
   - InvalidClientTokenId

Procure erros relacionados ao cluster do EKS. Use as mensagens de erro para atualizar as entradas de acesso ao EKS, o perfil do IAM do cluster ou o perfil do IAM do nó. Pode ser necessário anexar uma nova política a esses perfis com permissões para o Modo Automático do EKS.

Solucionar problemas de pods com falhas ao agendar o nó do Modo Automático

Caso os pods estejam ficando no estado `Pending` e não estejam sendo agendados no nó do modo automático, verifique se o seu manifesto de pod ou de implantação contém um `nodeSelector`. Se um `nodeSelector` estiver presente, certifique-se de que ele esteja usando `eks.amazonaws.com/compute-type: auto` para ser agendado em nós criados pelo Modo Automático do EKS. Para obter mais informações sobre os rótulos dos nós usados pelo Modo Automático do EKS, consulte the section called “Controlar a implantação”.

Solucionar problemas de nós que não estão se associando ao cluster

O Modo Automático do EKS configura automaticamente novas instâncias do EC2 com as informações corretas para se associar ao cluster, incluindo o endpoint do cluster e a autoridade de certificação (CA) do cluster. No entanto, essas instâncias ainda podem falhar ao se associarem ao

cluster do EKS como um nó. Execute os seguintes comandos para identificar as instâncias que não se associaram ao cluster:

1. Execute `kubectl get nodeclaim` para verificar os `NodeClaims` que são `Ready = False`.

```
kubectl get nodeclaim
```

2. Execute `kubectl describe nodeclaim <node_claim>` e verifique a seção Status para identificar possíveis problemas que impedem o nó de se associar ao cluster.

```
kubectl describe nodeclaim <node_claim>
```

Mensagens de erro comuns:

`Error getting launch template configs`

Você pode receber esse erro caso esteja configurando tags personalizadas no `NodeClass` com as permissões padrão do perfil do IAM do cluster. Consulte the section called “Identidade e acesso”.

`Error creating fleet`

Pode ocorrer um problema de autorização ao fazer a chamada `RunInstances` da API do EC2. Verifique o AWS CloudTrail em busca de erros e consulte the section called “Perfil do IAM para o cluster do modo automático” para obter as permissões necessárias do IAM.

Detectar problemas de conectividade de nós com o **`VPC Reachability Analyzer`**

> **Note**
>
> Há uma cobrança por cada análise executada no VPC Reachability Analyzer. Para obter detalhes dos preços, consulte Preços da Amazon VPC.

Um dos motivos pelos quais uma instância não se associou ao cluster é um problema de conectividade de rede que a impede de acessar o servidor da API. Para diagnosticar esse problema, você pode usar o VPC Reachability Analyzer para realizar uma análise da conectividade entre um nó que não está conseguindo se associar ao cluster e o servidor de API. Você precisará de duas informações:

- ID da instância de um nó que não consegue se associar ao cluster
- Endereço IP do endpoint do servidor de API do Kubernetes

Para obter o ID da instância, você precisará criar uma workload no cluster para fazer com que o Modo Automático do EKS inicie uma instância do EC2. Isso também cria um objeto `NodeClaim` no seu cluster que terá o ID da instância. Execute `kubectl get nodeclaim -o yaml` para imprimir todos os `NodeClaims` no cluster. Cada `NodeClaim` contém o ID da instância como um campo, e novamente no providerID:

```
kubectl get nodeclaim -o yaml
```

Veja abaixo um exemplo de saída.

```
    nodeName: i-01234567890123456
    providerID: aws:///us-west-2a/i-01234567890123456
```

Você pode determinar o endpoint do servidor de API do Kubernetes executando `kubectl get endpoint kubernetes -o yaml`. Os endereços estão no campo de endereços:

```
kubectl get endpoints kubernetes -o yaml
```

Veja abaixo um exemplo de saída.

```
apiVersion: v1
kind: Endpoints
metadata:
  name: kubernetes
  namespace: default
subsets:
- addresses:
  - ip: 10.0.143.233
  - ip: 10.0.152.17
  ports:
  - name: https
    port: 443
    protocol: TCP
```

Com essas duas informações, você pode realizar a análise. Primeiro, navegue até o VPC Reachability Analyzer no Console de gerenciamento da AWS.

1. Clique em "Criar e analisar caminho"
2. Forneça um nome para a análise (por exemplo, "Falha na associação do nó")
3. Para o "Tipo de origem", selecione "Instâncias"
4. Insira o ID da instância do nó com falha como a "Origem"
5. Para o "Destino do caminho", selecione "Endereço IP"
6. Insira um dos endereços IP do servidor da API como "Endereço de destino"
7. Expanda a "Seção de configuração adicional do cabeçalho de pacote"
8. Insira uma "Porta de destino" de 443
9. Selecione "Protocolo" como TCP caso ainda não tenha selecionado
10. Clique em “Criar e analisar caminho”.
11. A análise pode levar alguns minutos para ser concluída. Se os resultados da análise indicarem falha na acessibilidade, eles indicarão onde a falha ocorreu no caminho da rede para que você possa resolver o problema.

Compartilhar volumes entre pods

Os nós do Modo Automático do EKS são configurados com o SELinux no modo de imposição, o que fornece mais isolamento entre os pods que estão sendo executados no mesmo nó. Quando o SELinux está habilitado, a maioria dos pods sem privilégios terá automaticamente seu próprio rótulo de segurança de várias categorias (MCS) aplicado a eles. Esse rótulo MCS é exclusivo por pod e foi projetado para garantir que um processo em um pod não possa manipular um processo em nenhum outro pod ou no host. Mesmo que um pod rotulado seja executado como raiz e tenha acesso ao sistema de arquivos do host, ele não conseguirá manipular arquivos, fazer chamadas de sistema sensíveis no host, acessar o runtime do contêiner ou obter o material da chave secreta do kubelet.

Devido a essa questão, você pode encontrar problemas ao tentar compartilhar dados entre pods. Por exemplo, um `PersistentVolumeClaim` com um modo de acesso de `ReadWriteOnce` não permitirá que vários pods acessem o volume simultaneamente.

Para habilitar esse compartilhamento entre pods, você pode usar as `seLinuxOptions` do pod para configurar o mesmo rótulo MCS nesses pods. Neste exemplo, atribuímos as três categorias `c123,c456,c789` ao pod. Isso não entrará em conflito com nenhuma categoria atribuída automaticamente aos pods no nó, pois eles só receberão duas categorias.

```
securityContext:
  seLinuxOptions:
```

```
    level: "s0:c123,c456,c789"
```

Exibir eventos do Karpenter nos logs do ambiente de gerenciamento

Para clusters do EKS com logs do ambiente de gerenciamento habilitados, é possível obter informações sobre as ações e o processo de tomada de decisão do Karpenter consultando os logs. Isso pode ser particularmente útil para solucionar problemas do Modo Automático do EKS relacionados ao provisionamento, ajuste de escala e encerramento de nós. Para visualizar eventos relacionados ao Karpenter, use a seguinte consulta do CloudWatch Logs Insights:

```
fields @timestamp, @message
| filter @logStream like /kube-apiserver-audit/
| filter @message like 'DisruptionBlocked'
or @message like 'DisruptionLaunching'
or @message like 'DisruptionTerminating'
or @message like 'DisruptionWaitingReadiness'
or @message like 'Unconsolidatable'
or @message like 'FailedScheduling'
or @message like 'NoCompatibleInstanceTypes'
or @message like 'NodeRepairBlocked'
or @message like 'Disrupted'
or @message like 'Evicted'
or @message like 'FailedDraining'
or @message like 'TerminationGracePeriodExpiring'
or @message like 'TerminationFailed'
or @message like 'FailedConsistencyCheck'
or @message like 'InsufficientCapacityError'
or @message like 'UnregisteredTaintMissing'
or @message like 'NodeClassNotReady'
| sort @timestamp desc
```

Essa consulta filtra eventos relacionados ao Karpenter específicos nos logs de auditoria do kube-apiserver. Os eventos incluem vários estados de interrupção, falhas de agendamento, problemas de capacidade e problemas relacionados ao nó. Ao analisar esses logs, é possível compreender melhor:

- Por que Karpenter está tomando certas ações.
- Qualquer problema que impeça o provisionamento, o ajuste de escala ou o encerramento adequados dos nós.
- Possíveis problemas de capacidade ou compatibilidade com tipos de instância.

- Eventos do ciclo de vida do nó, como interrupções, remoções ou encerramentos.

Para usar essa consulta:

1. Navegue até o console do CloudWatch
2. No painel de navegação à esquerda, selecione "Logs Insights"
3. Escolha o grupo de logs para os logs do ambiente de gerenciamento do seu cluster do EKS
4. Cole a consulta no editor de consultas
5. Ajuste o intervalo de tempo conforme necessário
6. Execute a consulta

Os resultados mostrarão um cronograma de eventos relacionados ao Karpenter, ajudando você a solucionar problemas e a entender o comportamento do Modo Automático do EKS em seu cluster. Para revisar as ações do Karpenter em um nó específico, é possível adicionar o filtro de linha abaixo especificando o ID da instância à consulta mencionada acima:

```
|filter @message like /[.replaceable]`i-12345678910123456`/
```

> **Note**
>
> Para usar essa consulta, os logs do ambiente de gerenciamento devem estar habilitados em seu cluster do EKS. Se você ainda não fez isso, consulte the section called “Logs do ambiente de gerenciamento”.

Solucionar problemas de controladores incluídos no Modo Automático

Caso tenha um problema com um controlador, você deve pesquisar:

- Se os recursos associados a esse controlador estão devidamente formatados e válidos.
- Se os recursos de RBAC do AWS IAM e do Kubernetes estão configurados corretamente para o cluster. Para obter mais informações, consulte the section called “Identidade e acesso”.

Recursos relacionados

Use estes artigos do AWS re:Post para obter etapas avançadas de solução de problemas:

- How to troubleshoot common scaling issues in EKS Auto-Mode?
- How do I troubleshoot custom nodepool and nodeclass provisioning issues in Amazon EKS Auto Mode?
- How do I troubleshoot EKS Auto Mode built-in node pools with Unknown Status?

Analisar as notas de release do Modo automático do EKS

Esta página documenta as atualizações do Modo automático do Amazon EKS. Você pode verificar periodicamente esta página para ver anúncios sobre recursos, correções de erros, problemas conhecidos e funcionalidades obsoletas.

Para receber notificações de todas as alterações do arquivo de origem nesta página específica da documentação, você pode se inscrever no seguinte URL com um leitor de RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/automode/auto-change.adoc.atom
```

19 de novembro de 2025

Recurso: habilitamos a disponibilização e a descompactação em paralelo de Seekable OCI (SOCI) para as famílias de instâncias G, P e Trn que contam com o armazenamento NVMe local. A disponibilização e o desempacotamento paralelos de SOCI são sempre usados para essas famílias de instâncias com o modo automático do EKS, e não são necessárias alterações de configuração para ativá-los. Para obter mais informações sobre o SOCI, consulte o blog de lançamento.

19 de novembro de 2025

Recurso: adicionamos suporte para grupos de nós de capacidade estática que mantêm um número fixo de nós. Para obter mais informações, consulte the section called “Grupos de nós de capacidade estática”.

23 de outubro de 2025

Recurso: os usuários com clusters em regiões dos EUA agora podem solicitar o uso de AMIs compatíveis com FIPS especificando `spec.advancedSecurity.fips` na definição de NodeClass.

1.º de outubro de 2025

Recurso: o modo automático do EKS agora fornece suporte para a implantação de nós em zonas locais da AWS. Para obter mais informações, consulte the section called “Uso das zonas locais”.

30 de setembro de 2025

Recurso: adicionamos suporte para instanceProfile ao `spec.instanceProfile` da NodeClass, que é mutuamente exclusivo em relação ao campo `spec.role`.

29 de setembro de 2025

No momento, o DRA não é compatível com o modo automático do EKS.

10 de setembro de 2025

Obrigação: os eventos disparados pelo controlador de computação do modo automático do EKS agora usarão o nome `eks-auto-mode/compute` em vez de `karpenter`.

24 de agosto de 2025

Correção de erro: as VPCs que usavam um conjunto de opções DHCP com um nome de domínio personalizado contendo letras maiúsculas faziam com que os nós não conseguissem se juntar ao cluster devido à geração de um nome de host inválido. Isso foi resolvido e nomes de domínio com letras maiúsculas agora funcionam corretamente.

15 de agosto de 2025

Correção de erro: o Atendente de Identidade de Pods agora só escuta no endereço local do link IPv4 em um cluster EKS IPv4 para evitar problemas em que o pod não consiga acessar o endereço IPv6.

6 de agosto de 2025

Recurso: adicionada nova configuração em `spec.advancedNetworking.associatePublicIPAddress` no NodeClass que pode ser usada para evitar que endereços IP públicos sejam atribuídos aos nós do modo automático do EKS

30 de junho de 2025

Atributo: o NodeClass do modo automático agora usa a chave do KMS personalizada configurada para criptografar o volume raiz somente para leitura da instância, além do volume de dados de

leitura/gravação. Anteriormente, a chave do KMS personalizada era usada somente para criptografar o volume de dados.

20 de junho de 2025

Recurso: suporte para controlar a implantação de workloads em reservas de capacidade sob demanda (ODCRs) do EC2. Isso adiciona a chave opcional `capacityReservationSelectorTerms` ao NodeClass, permitindo que você controle explicitamente quais ODCRs suas workloads usam. Para obter mais informações, consulte the section called “Controle de reservas de capacidade”.

13 de junho de 2025

Recurso: suporte para sub-redes de pod separadas no `NodeClass`. Isso adiciona as chaves opcionais `` `podSubnetSelectorTerms `` e `podSecurityGroupSelectorTerms` para definir as sub-redes e os grupos de segurança para os pods. Para obter mais informações, consulte the section called “Seleção de sub-rede para pods”.

30 de abril de 2025

Recurso: suporte para proxies de rede de encaminhamento no `NodeClass`. Isso adiciona a chave opcional `advancedNetworking` para configurar seu proxy HTTPS. Para obter mais informações, consulte the section called “Especificação das classes de nós”.

18 de abril de 2025

Funcionalidade: suporte para resolver domínios locais (normalmente reservados para DNS multicast) via DNS unicast.

11 de abril de 2025

Funcionalidade: adicionados `certificateBundles` e `ephemeralStorage.kmsKeyID` ao `NodeClass`. Para obter mais informações, consulte the section called “Especificação das classes de nós”.

Funcionalidade: velocidade de extração de imagens aprimorada, especialmente para tipos de instância com armazenamento de instância local que podem aproveitar a descompactação mais rápida da imagem.

Correção de erro: resolvida uma condição de corrida que causava FailedCreatePodSandBox , Error while dialing: dial tcp 127.0.0.1:50051: connect: a conexão, às vezes, não ocorria para a programação de pods para um nó imediatamente na inicialização.

04 de abril de 2025

Funcionalidade: aumento de `registryPullQPS` de 5 para 25 e de `registryBurst` de 10 para 50 para reduzir a controle de utilização de extração de imagens imposta pelo cliente (`Failed to pull image xyz: pull QPS exceeded`)

31 de março de 2025

Correção de erro: corrigiu um problema em que, se um pod central do DNS estivesse sendo executado em um nó do Modo automático, as consultas ao DNS dos pods no nó atingiriam esse pod central do DNS em vez do servidor DNS local do nó. As consultas ao DNS de pods em um nó do Modo automático sempre irão para o DNS local do nó.

21 de março de 2025

Correção de erro: os nós do Modo automático agora resolvem `kube-dns.kube-system.svc.cluster.local` corretamente quando não há um serviço de `kube-dns` instalado no cluster. Resolve o problema #2546 do GitHub.

14 de março de 2025

Funcionalidade: saída `IPv4` habilitada em clusters `IPv6`. O tráfego `IPv4` que sai dos clusters `IPv6` do Modo automático agora será convertido automaticamente para o endereço `v4` da ENI primária do nó.

Ciclo de vida e configuração do cluster do Amazon EKS

Esta seção fornece orientações detalhadas sobre o gerenciamento completo do ciclo de vida dos clusters Kubernetes e as diferentes maneiras de configurá-los. Ele abrange a criação de clusters, o monitoramento da integridade do cluster, a atualização de versões do Kubernetes e a exclusão de clusters. Além disso, inclui tópicos avançados sobre como configurar o acesso ao endpoint, habilitar ou desabilitar o suporte ao Windows, configurar clusters privados, implementar o ajuste de escala automático e como aumentar a resiliência com a mudança de zona para o redirecionamento de tráfego em configurações multi-AZ.

Tópicos

- Criar um cluster do Modo Automático do Amazon EKS
- Criar um cluster do Amazon EKS
- Ambiente de gerenciamento provisionado do Amazon EKS
- Prepare-se para as atualizações da versão e a solução de problemas de configurações incorretas do Kubernetes com insights do cluster
- Atualizar um cluster existente para a nova versão do Kubernetes
- Excluir um cluster
- Endpoint do servidor de API do cluster
- Implantar nós Windows em clusters EKS
- Desativar o suporte ao Windows
- Implementar clusters privados com acesso limitado à internet
- Escale a computação em cluster com o Karpenter e o Cluster Autoscaler
- Saiba mais sobre a mudança de zona do Controlador de Recuperação de Aplicações da Amazon (ARC, na sigla em inglês) no Amazon EKS
- Habilitação da mudança de zona do EKS para evitar zonas de disponibilidade comprometidas

Criar um cluster do Modo Automático do Amazon EKS

Este tópico fornece instruções detalhadas para criar um cluster do Modo Automático do Amazon EKS usando opções avançadas de configuração. Ele aborda pré-requisitos, opções de rede e configurações de complementos. O processo inclui configurar perfis do IAM, definir configurações

de cluster, especificar parâmetros de rede e selecionar complementos. Os usuários podem criar clusters usando o Console de gerenciamento da AWS ou a AWS CLI, com orientação passo a passo fornecida para ambos os métodos.

Para usuários que buscam um processo de configuração menos complexo, consulte o seguinte para ver etapas simplificadas da criação de um cluster:

- the section called “CLI do eksctl”
- the section called “ AWS CLI”
- the section called “Console de Gerenciamento”

Este guia de configuração avançada é destinado a usuários que precisam de um controle mais granular sobre a configuração do cluster do Modo Automático do EKS e estão familiarizados com os conceitos e requisitos do Amazon EKS. Antes de prosseguir com a configuração avançada, certifique-se de ter atendido a todos os pré-requisitos e ter uma compreensão completa dos requisitos de rede e do IAM para clusters do Modo Automático do EKS.

O Modo Automático do EKS requer permissões adicionais do IAM. Para obter mais informações, consulte:

- the section called “Perfis do IAM de clusters do Modo Automático do EKS”
- the section called “Identidade e acesso”

> **Note**
>
> Se você quiser criar um cluster sem o Modo Automático do EKS, consulte the section called “Criar um cluster”.
> Este tópico aborda a configuração avançada. Se você deseja começar a usar o Modo Automático do EKS, consulte the section called “Criar cluster”.

Pré-requisitos

- Uma VPC e sub-redes existentes que atendem aos requisitos do Amazon EKS. Antes de implantar um cluster para uso em ambientes de produção, convém ter uma compreensão integral dos requisitos da VPC e da sub-rede. Se você não tiver um VPC e sub-redes, poderá criá-los usando um modelo do Amazon EKS fornecido pelo AWS CloudFormation.

- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version`. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface.
- Uma entidade principal do IAM com permissões para criar e modificar recursos do EKS e do IAM.

Criar cluster - AWS console

1. Abra o console do Amazon EKS.
2. Escolha Add cluster (Adicionar cluster) e, em seguida, Create (Criar).
3. Em Opções de configuração, selecione Configuração personalizada.
   - Este tópico aborda a configuração personalizada. Para obter informações sobre a configuração rápida, consulte the section called “Console de Gerenciamento”.
4. Confirme se Usar Modo Automático do EKS está habilitado.
   - Este tópico aborda a criação de clusters com o Modo Automático do EKS. Para obter mais informações sobre como criar clusters sem o Modo Automático do EKS, consulte the section called “Criar um cluster”.
5. Na página Configure cluster (Configurar cluster), preencha os seguintes campos:
   - Name (Nome): um nome exclusivo para o cluster. O nome pode conter apenas caracteres alfanuméricos (diferencia maiúsculas de minúsculas), hifens e sublinhados. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
   - Perfil do IAM do cluster: escolha o perfil do IAM do cluster do Amazon EKS que você criou para permitir que o ambiente de gerenciamento do Kubernetes gerencie os recursos da AWS em seu nome. Se você ainda não criou um perfil do IAM do cluster para o Modo Automático do EKS, selecione o botão Criar perfil recomendado para criar o perfil com as permissões necessárias no console do IAM.

- Kubernetes version (Versão do Kubernetes) – a versão do Kubernetes a ser usada para o cluster. Recomendamos que você selecione a versão mais recente, a menos que precise de uma versão anterior.
- Política de atualização: a política de versão do Kubernetes que você deseja definir para o cluster. Se quiser que o cluster seja executado somente em uma versão com suporte padrão, você pode escolher Padrão. Se quiser que o cluster entre no suporte estendido ao final do suporte padrão para uma versão, você pode escolher Estendido. Caso selecione uma versão do Kubernetes que esteja atualmente com suporte estendido, você não poderá selecionar o suporte padrão como opção.

6. Na seção Computação do Modo Automático da página de configuração do cluster, preencha os seguintes campos:
   - Grupos de nós: determine se você deseja usar o build nos grupos de nós. Para obter mais informações, consulte the section called “Analisar grupos de nós integrados”.
   - Perfil do IAM do nó: se habilitar qualquer um dos grupos de nós integrados, você precisará selecionar um perfil do IAM do nó. O Modo Automático do EKS atribuirá esse perfil a novos nós. Você não poderá alterar o valor depois que o cluster for criado. Se você ainda não criou um perfil do IAM do nó para o Modo Automático do EKS, selecione o botão Criar perfil recomendado para criar o perfil com as permissões necessárias. Para obter mais informações sobre essa função, consulte the section called “Identidade e acesso”.
7. Na seção Acesso ao cluster da página de configuração do cluster, preencha os seguintes campos:
   - Inicialização do acesso de administrador do cluster: o criador do cluster é automaticamente um administrador do Kubernetes. Caso deseje desabilitar essa opção, selecione Desabilitar acesso de administrador ao cluster.
   - Modo de autenticação do cluster: o Modo Automático do EKS requer entradas de acesso ao EKS, o modo de autenticação da API do EKS. Opcionalmente, você pode habilitar o modo de autenticação `ConfigMap` selecionando ConfigMap e API do EKS.
8. Na página de configuração do cluster, preencha os seguintes campos:
   - Secrets encryption (Criptografia de segredos): (Opcional) Escolha habilitar a criptografia de segredos do Kubernetes usando uma chave do KMS. Também é possível isso depois de criar o cluster. Antes de habilitar esse recurso, familiarize-se com as informações em the section called “Habilitar a criptografia de segredos”.
   - Mudança de zona do ARC: o Modo Automático do EKS não é compatível com a mudança de zona do ARC.

- Tags (Etiquetas) – (opcional) adicione etiquetas ao cluster. Para obter mais informações, consulte the section called “Marcando seus Recursos”.

  Após terminar com essa página, escolha Próximo.

9. Na página Specify networking (Especificar redes), selecione valores para os seguintes campos:

   - VPC: escolha uma VPC existente que atenda aos Requisitos de VPC do Amazon EKS na qual criar o cluster. Antes de escolher uma VPC, recomendamos familiarizar-se com todos os requisitos e considerações em the section called “Requisitos para sub-rede e VPC”. Não será possível alterar quais VPCs você deseja utilizar depois de criar o cluster. Se nenhuma VPC estiver listada, você precisará criar uma primeiro. Para obter mais informações, consulte the section called “Crie uma VPC”.
   - Subnets (Sub-redes): por padrão, as sub-redes disponíveis na VPC especificada no campo anterior são pré-selecionadas. É necessário selecionar pelo menos duas.

     As sub-redes escolhidas devem atender aos Requisitos para sub-redes do Amazon EKS. Antes de selecionar sub-redes, convém estar familiarizado com todos os Requisitos e considerações sobre VPCs e sub-redes do Amazon EKS.

     Security groups (Grupos de segurança) (Opcional): especifique um ou mais grupos de segurança que você deseja que o Amazon EKS associe às interfaces de rede que ele cria.

     Independentemente de você escolher algum grupo de segurança, o Amazon EKS cria um grupo de segurança que permite comunicação entre seu cluster e sua VPC. O Amazon EKS associa esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte the section called “Requisitos para grupos de segurança”. É possível modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS.
   - Escolher família de endereços IP do cluster: é possível escolher IPv4 e IPv6.

     Por padrão, o Kubernetes atribui endereços `IPv4` aos pods e serviços. Antes de decidir usar a família `IPv6`, verifique se você está familiarizado com todas as considerações e requisitos dos tópicos Requisitos e considerações da VPC, the section called “Requisitos e considerações para sub-redes”, the section called “Requisitos para grupos de segurança” e the section called “IPv6”. Caso escolha a família `IPv6`, você não poderá especificar um intervalo de endereços para o Kubernetes atribuir endereços de serviço `IPv6` da mesma forma que faz para a família `IPv4`. O Kubernetes atribui endereços de serviço do intervalo exclusivo de endereços locais (`fc00::/7`).

- (Opcional) Escolha Configurar intervalo de endereços IP do Kubernetes e especifique um Intervalo **IPv4** de serviços.

  Especificar seu próprio intervalo pode ajudar a evitar conflitos entre serviços do Kubernetes e outras redes com emparelhamento ou conectadas à VPC. Insira um intervalo em notação CIDR. Por exemplo: `10.2.0.0/16`.

  O bloco CIDR deve atender aos seguintes requisitos:

  - Estar dentro de um dos seguintes intervalos: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.
  - Ter um tamanho mínimo de `/24` e máximo de `/12`.
  - Não se sobrepor ao intervalo da VPC para seus recursos do Amazon EKS.

  Apenas é possível especificar essa opção ao utilizar a família de endereços `IPv4` e somente ao criar o cluster. Se isso não for especificado, o Kubernetes atribuirá endereços IP de serviço de qualquer um dos blocos CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Para Cluster endpoint access (Acesso ao endpoint do cluster), selecione uma opção. Depois de criar o cluster, você poderá alterar essa opção. Antes de selecionar uma opção não padrão, familiarize-se com as opções e suas implicações. Para obter mais informações, consulte the section called “Acesso ao endpoint do cluster”.

  Após terminar com essa página, escolha Próximo.

10. (Opcional) Na página Configurar observabilidade, escolha quais opções de Métricas e Log do ambiente de gerenciamento deseja ativar. Por padrão, o cada tipo de log está desativado.

    - Para obter mais informações sobre as opções de métricas do Prometheus, consulte the section called “Etapa 1: ativar as métricas do Prometheus”.
    - Para obter mais informações sobre as opções do Log do ambiente de gerenciamento, consulte the section called “Logs do ambiente de gerenciamento”.
    - Após terminar com essa página, escolha Próximo.

11. Na página Selecionar complementos, escolha os complementos que você deseja adicionar ao cluster. Você pode escolher quantos complementos do Amazon EKS e do AWS Marketplace desejar. Se os complementos do AWS Marketplace que você deseja instalar não estiverem listados, será possível clicar na numeração da página para visualizar resultados de páginas adicionais ou pesquisar os complementos disponíveis no AWS Marketplace inserindo algum texto na caixa de pesquisa. Você também poderá pesquisar por categoria, fornecedor ou modelo de preços e depois escolher os complementos nos resultados da pesquisa. Ao criar um cluster, é

possível visualizar, selecionar e instalar qualquer complemento que ofereça suporte às Identidade de Pods do EKS, conforme detalhado em the section called “Identidade de Pods”.

- O Modo Automático do EKS automatiza a funcionalidade de determinados complementos. Se você planeja implantar grupos de nós gerenciados pelo EKS no cluster do Modo Automático do EKS, selecione Complementos adicionais do Amazon EKS e analise as opções. Talvez seja necessário instalar complementos como o CoreDNS e kube-proxy. O EKS só vai instalar os complementos desta seção em nós e grupos de nós autogerenciados.
- Após terminar com essa página, escolha Próximo.

12 Na página Configurar opções de complementos selecionados, selecione a versão que deseja instalar. Você sempre pode atualizar para uma versão posterior após a criação do cluster.

Para complementos com suporte para Identidade de Pods do EKS, é possível usar o console para gerar automaticamente o perfil com o nome, a política gerenciada pela AWS e a política de confiança pré-preenchidos especificamente para o complemento. É possível reutilizar perfis existentes ou criar novos perfis para complementos com suporte. Para obter as etapas de uso do console para criar perfis para complementos que ofereçam suporte às Identidades de Pods do EKS, consulte the section called “Criar complemento (console do AWS)”. Se um complemento não oferecer suporte à Identidade de Pods do EKS, uma mensagem será exibida com instruções para usar o assistente para criar os perfis do IAM para contas de serviço (IRSA) após a criação do cluster.

Você pode atualizar a configuração de cada complemento após a criação do cluster. Para obter mais informações sobre a configuração de complementos, consulte the section called “Atualização de um complemento”. Após terminar com essa página, escolha Próximo.

13 Na página Review and create (Revisar e criar), revise as informações que você inseriu ou selecionou nas páginas anteriores. Se precisar fazer alterações, escolha Edit (Editar). Quando estiver satisfeito, escolha Criar. O campo Status mostra o campo CREATING (Criando) enquanto o cluster é provisionado.

> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas

zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte the section called “Capacidade insuficiente”.

O provisionamento de cluster leva alguns minutos.

Crar cluster: AWS CLI

As instruções da CLI a seguir abrangem a criação de recursos do IAM e a criação do cluster.

Criar um perfil do IAM do cluster do Modo Automático do EKS

Etapa 1: criar a política de confiança

Crie uma política de confiança para permitir que o serviço do Amazon EKS assuma o perfil. Salve a política como `trust-policy.json`:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Etapa 2: criar o perfil do IAM

Use a política de confiança para criar o perfil do IAM do cluster:

```
aws iam create-role \
    --role-name AmazonEKSAutoClusterRole \
    --assume-role-policy-document file://trust-policy.json
```

Etapa 3: registre o ARN do perfil

Recupere e salve o ARN do novo perfil para uso nas etapas subsequentes:

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

Etapa 4: anexar as políticas necessárias

Anexe as seguintes políticas gerenciadas pela AWS ao perfil do IAM do cluster para conceder as permissões necessárias:

AmazonEKSClusterPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
```

AmazonEKSComputePolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSComputePolicy
```

AmazonEKSBlockStoragePolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSBlockStoragePolicy
```

AmazonEKSLoadBalancingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

AmazonEKSNetworkingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSNetworkingPolicy
```

Criar um perfil do IAM do nó do Modo Automático do EKS

Etapa 1: criar a política de confiança

Crie uma política de confiança para permitir que o serviço do Amazon EKS assuma o perfil. Salve a política como `node-trust-policy.json`:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Etapa 2: criar o perfil do IAM do nó

Use o arquivo node-trust-policy.json da etapa anterior para definir quais entidades podem assumir o perfil. Execute o seguinte comando para criar o perfil do IAM do nó:

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

Etapa 3: registre o ARN do perfil

Depois de criar o perfil, recupere e salve o ARN do perfil do IAM do nó. Você precisará desse ARN nas etapas subsequentes. Use o seguinte comando para obter o ARN:

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Etapa 4: anexar as políticas necessárias

Anexe as seguintes políticas gerenciadas pela AWS ao perfil do IAM do nó para fornecer as permissões necessárias:

AmazonEKSWorkerNodeMinimalPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

AmazonEC2ContainerRegistryPullOnly:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Criação de cluster

1. Crie o cluster usando o comando a seguir. Antes da execução do comando, realize as seguintes substituições:
   - Substitua *region-code* pela região AWS na qual você deseja criar o cluster.
   - Substitua *my-cluster* por um nome para seu cluster. O nome pode conter apenas caracteres alfanuméricos (diferencia maiúsculas de minúsculas), hifens e sublinhados. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
   - Substitua *1.30* por qualquer versão compatível do Amazon EKS.
   - Substitua *111122223333* pelo ID da sua conta.
   - Se você criou perfis do IAM com nomes diferentes para os perfis de cluster e de nó, substitua os ARNs.
   - Substitua os valores de `subnetIds` pelos seus próprios valores. Também é possível adicionar outras IDs. Você deve especificar pelo menos dois IDs de sub-rede.

     As sub-redes escolhidas devem atender aos Requisitos para sub-redes do Amazon EKS. Antes de selecionar sub-redes, convém estar familiarizado com todos os Requisitos e considerações sobre VPCs e sub-redes do Amazon EKS.
   - Se não quiser especificar um ID de grupo de segurança, remova `,securityGroupIds=sg-<ExampleID1>` do comando. Se quiser especificar um ou mais IDs de grupo de segurança, substitua os valores de `securityGroupIds` pelos seus próprios. Também é possível adicionar outras IDs.

     Independentemente de você escolher algum grupo de segurança, o Amazon EKS cria um grupo de segurança que permite comunicação entre seu cluster e sua VPC. O Amazon EKS associa

esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte the section called “Requisitos para grupos de segurança”. É possível modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS.

```
aws eks create-cluster \
  --region region-code \
  --name my-cluster \
  --kubernetes-version 1.30 \
  --role-arn arn:aws:iam::111122223333:role/AmazonEKSAutoClusterRole \
  --resources-vpc-config '{"subnetIds": ["subnet-ExampleID1","subnet-ExampleID2"],
 "securityGroupIds": ["sg-ExampleID1"], "endpointPublicAccess": true,
 "endpointPrivateAccess": true}' \
  --compute-config '{"enabled": true, "nodeRoleArn":
 "arn:aws:iam::111122223333:role/AmazonEKSAutoNodeRole", "nodePools": ["general-
purpose", "system"]}' \
  --kubernetes-network-config '{"elasticLoadBalancing": {"enabled": true}}' \
  --storage-config '{"blockStorage": {"enabled": true}}' \
  --access-config '{"authenticationMode": "API"}'
```

> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte the section called “Capacidade insuficiente”.

Veja a seguir as configurações opcionais que, se necessário, devem ser adicionadas ao comando anterior. Apenas é possível habilitar essas opções ao criar o cluster, e não depois.

- Se quiser especificar de qual bloco de Encaminhamento Entre Domínios Sem Classificação (CIDR) `IPv4` o Kubernetes atribui endereços IP de serviço, será necessário especificá-lo adicionando `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>` ao comando a seguir.

Especificar seu próprio intervalo pode ajudar a evitar conflitos entre serviços do Kubernetes e outras redes com emparelhamento ou conectadas à VPC. Insira um intervalo em notação CIDR. Por exemplo: `10.2.0.0/16`.

O bloco CIDR deve atender aos seguintes requisitos:

- Estar dentro de um dos seguintes intervalos: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.
- Ter um tamanho mínimo de `/24` e máximo de `/12`.
- Não se sobrepor ao intervalo da VPC para seus recursos do Amazon EKS.

Apenas é possível especificar essa opção ao utilizar a família de endereços `IPv4` e somente ao criar o cluster. Se isso não for especificado, o Kubernetes atribuirá endereços IP de serviço de qualquer um dos blocos CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Se estiver criando um cluster e quiser que o cluster atribua endereços `IPv6` a pods e serviços em vez de endereços `IPv4`, adicione `--kubernetes-network-config ipFamily=ipv6` ao comando a seguir.

Por padrão, o Kubernetes atribui endereços `IPv4` aos pods e serviços. Antes de decidir usar a família `IPv6`, verifique se você está familiarizado com todas as considerações e requisitos dos tópicos Requisitos e considerações da VPC, the section called “Requisitos e considerações para sub-redes”, the section called “Requisitos para grupos de segurança” e the section called “IPv6”. Caso escolha a família `IPv6`, você não poderá especificar um intervalo de endereços para o Kubernetes atribuir endereços de serviço `IPv6` da mesma forma que faz para a família `IPv4`. O Kubernetes atribui endereços de serviço do intervalo exclusivo de endereços locais (`fc00::/7`).

2. Leva alguns minutos para provisionar o cluster. Você pode consultar o status do cluster com o comando a seguir.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

Próximas etapas

- the section called “Acessar meu cluster com o kubectl”
- the section called “Entradas de acesso”

- Habilitar a criptografia de segredos para o seu cluster.
- Configurar o registro em log para o seu cluster.
- Adicionar nós ao seu cluster.

Criar um cluster do Amazon EKS

> **Note**
>
> Este tópico aborda a criação de clusters do Amazon EKS sem o modo automático do EKS. Para obter instruções detalhadas sobre como criar um cluster do Modo Automático do EKS, consulte the section called “Criar um cluster automático”.
> Para começar a usar o Modo Automático do EKS, consulte the section called “Criar cluster (Modo Automático do EKS)”.

Este tópico dá uma visão geral das opções disponíveis e descreve o que deve ser considerado ao criar um cluster do Amazon EKS. Se você precisar criar um cluster com a infraestrutura on-premises como a computação para nós, consulte the section called “Criação de cluster”. Se esta for a primeira vez que você cria um cluster do Amazon EKS, recomendamos que siga um de nossos guias em Conceitos básicos. Esses guias ajudam a criar um cluster simples e padrão sem expandir para todas as opções que estão disponíveis.

Pré-requisitos

- Uma VPC e sub-redes existentes que atendem aos requisitos do Amazon EKS. Antes de implantar um cluster para uso em ambientes de produção, convém ter uma compreensão integral dos requisitos da VPC e da sub-rede. Se você não tiver um VPC e sub-redes, poderá criá-los usando um modelo do Amazon EKS fornecido pelo AWS CloudFormation.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões

atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.

- Uma entidade principal do IAM com permissões para `create` e `describe` um cluster do Amazon EKS. Para obter mais informações, consulte the section called “Criar um cluster local do Kubernetes em um Outpost” e the section called “Listar ou descrever todos os clusters”.

Etapa 1: Criar perfil IAM de cluster

1. Se já tiver um perfil do IAM de cluster ou se for criar seu cluster com `eksctl`, você poderá ignorar essa etapa. Por padrão, `eksctl` cria um perfil para você.
2. Execute o comando a seguir para criar um arquivo JSON de política de confiança do IAM:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. Crie o perfil do IAM do cluster do Amazon EKS. Se necessário, prefixe *`eks-cluster-role-trust-policy.json`* com o caminho no computador no qual você gravou o arquivo na etapa anterior. O comando associa a política de confiança criada na etapa anterior à função. Para criar um perfil do IAM, a entidade principal do IAM que estiver criando o perfil deverá ser atribuída à seguinte ação `iam:CreateRole` (permissão):

```
aws iam create-role --role-name myAmazonEKSClusterRole --assume-role-policy-document
 file://"eks-cluster-role-trust-policy.json"
```

4. Você pode atribuir a política gerenciada do Amazon EKS ou criar sua própria política personalizada. Para obter as permissões mínimas que você deve usar em sua política personalizada, consulte the section called “Função do IAM do cluster”.

   Anexe a política gerenciada do Amazon EKS chamada AmazonEKSClusterPolicy à perfil. Para anexar uma política do IAM a uma entidade principal do IAM a entidade principal do IAM que está anexando a política deve receber uma das seguintes ações do IAM (permissões): `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

   ```
   aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
   AmazonEKSClusterPolicy --role-name myAmazonEKSClusterRole
   ```

Perfil vinculado ao serviço

O Amazon EKS cria automaticamente um perfil vinculado ao serviço chamado `AWSServiceRoleForAmazonEKS`.

Isso é adicional ao perfil do IAM do cluster. A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. O perfil permite que o Amazon EKS gerencie clusters em sua conta. Para obter mais informações, consulte the section called “Função do do cluster”.

A identidade do IAM que você usa para criar o cluster do EKS deve ter permissão para criar o perfil vinculado ao serviço. Isso inclui a permissão `iam:CreateServiceLinkedRole`.

Caso o perfil vinculado ao serviço não exista e seu perfil do IAM atual não tenha permissões suficientes para criá-lo, a operação de criação do cluster apresentará falhas.

Etapa 2: Criar cluster

Você pode criar um cluster usando:

- `eksctl`
- o Console de gerenciamento da AWS
- a AWS CLI

Criar cluster - eksctl

1. Você precisa da versão `0.215.0` ou posterior da ferramenta de linha de comando da `eksctl` instalada no seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
2. Crie um cluster `IPv4` do Amazon EKS com a versão padrão do Kubernetes para Amazon EKS em sua região padrão da AWS. Antes da execução do comando, realize as seguintes substituições:
3. Substitua *`region-code`* pela região AWS na qual você deseja criar o cluster.
4. Substitua *`my-cluster`* por um nome para seu cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
5. Substitua *`1.33`* por qualquer versão compatível do Amazon EKS.
6. Altere os valores para `vpc-private-subnets` para atender às suas necessidades. Também é possível adicionar outras IDs. Você deve especificar pelo menos dois IDs de sub-rede. Se preferir especificar sub-redes públicas, poderá alterar `--vpc-private-subnets` para `--vpc-public-subnets`. Sub-redes públicas têm uma tabela de rotas associada a uma rota para um gateway da Internet, mas sub-redes privadas não têm tabelas de rotas associadas. Convém utilizar sub-redes privadas sempre que possível.

   As sub-redes escolhidas devem atender aos Requisitos para sub-redes do Amazon EKS. Antes de selecionar sub-redes, convém estar familiarizado com todos os Requisitos e considerações sobre VPCs e sub-redes do Amazon EKS.
7. Execute o seguinte comando:

   ```
   eksctl create cluster --name my-cluster --region region-code --version 1.33 --vpc-private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
   ```

   O provisionamento de cluster leva alguns minutos. Durante a criação do cluster, várias linhas de saída são exibidas. A última linha do resultado é semelhante ao exemplo de linha a seguir.

   ```
   [#]  EKS cluster "my-cluster" in "region-code" region is ready
   ```
8. Continue com the section called “Etapa 3: atualizar o kubeconfig”

Configurações opcionais

Para ver a maioria das opções que podem ser especificadas ao criar um cluster com `eksctl`, use o comando `eksctl create cluster --help`. Para ver todas as opções disponíveis, é possível utilizar um arquivo `config`. Para obter mais informações, consulte Uso dos arquivos de configuração e o esquema de arquivo de configuração na documentação do `eksctl`. Você pode encontrar exemplos de arquivos de configuração no GitHub.

Veja a seguir as configurações opcionais que, se necessário, devem ser adicionadas ao comando anterior. Apenas é possível habilitar essas opções ao criar o cluster, e não depois. Se precisar especificar essas opções, você deverá criar o cluster com um arquivo de configuração eksctl e especificar as configurações, em vez de usar o comando anterior.

- Se você quiser especificar um ou mais grupos de segurança que o Amazon EKS atribui às interfaces de rede que ele cria, especifique a opção securityGroup.

  Independentemente de você escolher algum grupo de segurança, o Amazon EKS cria um grupo de segurança que permite comunicação entre seu cluster e sua VPC. O Amazon EKS associa esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte the section called “Requisitos para grupos de segurança”. É possível modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS.
- Se quiser especificar de qual bloco `IPv4` de Encaminhamento Entre Domínios Sem Classificação (CIDR) o Kubernetes atribui endereços IP de serviço, especifique a opção serviceIPv4CIDR.

  Especificar seu próprio intervalo pode ajudar a evitar conflitos entre serviços do Kubernetes e outras redes com emparelhamento ou conectadas à VPC. Insira um intervalo em notação CIDR. Por exemplo: `10.2.0.0/16`.

  O bloco CIDR deve atender aos seguintes requisitos:

  - Estar dentro de um dos seguintes intervalos: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.
  - Ter um tamanho mínimo de `/24` e máximo de `/12`.
  - Não se sobrepor ao intervalo da VPC para seus recursos do Amazon EKS.

  Apenas é possível especificar essa opção ao utilizar a família de endereços `IPv4` e somente ao criar o cluster. Se isso não for especificado, o Kubernetes atribuirá endereços IP de serviço de qualquer um dos blocos CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Se estiver criando um cluster e quiser que ele atribua endereços `IPv6` a pods e serviços em vez de endereços `IPv4`, especifique a opção ipFamily.

  Por padrão, o Kubernetes atribui endereços `IPv4` aos pods e serviços. Antes de optar por utilizar a família `IPv6`, familiarize-se com todas as considerações e requisitos nos tópicos the section called "Requisitos e considerações para VPCs", the section called "Requisitos e considerações para sub-redes", the section called "Requisitos para grupos de segurança" e the section called "IPv6". Caso escolha a família `IPv6`, você não poderá especificar um intervalo de endereços para o Kubernetes atribuir endereços de serviço `IPv6` da mesma forma que faz para a família `IPv4`. O Kubernetes atribui endereços de serviço do intervalo exclusivo de endereços locais (`fc00::/7`).

Criar cluster - AWS console

1. Abra o console do Amazon EKS.
2. Escolha Add cluster (Adicionar cluster) e, em seguida, Create (Criar).
3. Em Opções de configuração, selecione Configuração personalizada.
   - Para obter informações sobre como criar rapidamente um cluster com o Modo Automático do EKS, consulte the section called "Console de Gerenciamento".
4. Em Modo Automático do EKS, desative a opção Usar Modo Automático do EKS.
   - Para obter informações sobre como criar um cluster do Modo Automático do EKS com configuração personalizada, consulte the section called "Criar um cluster automático".
5. Na página Configure cluster (Configurar cluster), preencha os seguintes campos:
   - Name (Nome): um nome exclusivo para o cluster. O nome pode conter apenas caracteres alfanuméricos (diferencia maiúsculas de minúsculas), hifens e sublinhados. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
   - Perfil do IAM do cluster: escolha o perfil do IAM do cluster do Amazon EKS que você criou para permitir que o ambiente de gerenciamento do Kubernetes gerencie os recursos da AWS em seu nome.
   - Kubernetes version (Versão do Kubernetes) – a versão do Kubernetes a ser usada para o cluster. Recomendamos que você selecione a versão mais recente, a menos que precise de uma versão anterior.
   - Tipo de suporte: a política de versão do Kubernetes que você deseja definir para o cluster. Se desejar que seu cluster seja executado somente em uma versão com suporte padrão, escolher Suporte padrão é uma opção possível. Se desejar que seu cluster entre no suporte estendido

ao final do suporte padrão para uma versão, escolha Suporte estendido. Caso selecione uma versão do Kubernetes que esteja atualmente com suporte estendido, você não poderá selecionar o suporte padrão como opção.

- Secrets encryption (Criptografia de segredos): (Opcional) Escolha habilitar a criptografia de segredos do Kubernetes usando uma chave do KMS. Também é possível isso depois de criar o cluster. Antes de habilitar esse recurso, familiarize-se com as informações em the section called “Habilitar a criptografia de segredos”.
- Tags (Etiquetas) – (opcional) adicione etiquetas ao cluster. Para obter mais informações, consulte the section called “Marcando seus Recursos”.
- Mudança de zona do ARC: (opcional) você pode usar o Controlador de Recuperação de Aplicações do Route 53 para mitigar zonas de disponibilidade prejudicadas. Para obter mais informações, consulte the section called “Saiba mais sobre a mudança de zona”.

6. Na seção Acesso ao cluster da página de configuração do cluster, preencha os seguintes campos:
   - Inicialização do acesso de administrador do cluster: o criador do cluster é automaticamente um administrador do Kubernetes. Caso deseje desabilitar essa opção, selecione Desabilitar acesso de administrador ao cluster.
   - Modo de autenticação do cluster: determine como você deseja conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes. Para obter mais informações, consulte the section called “Definir modo de autenticação do cluster”.

     Após terminar com essa página, escolha Próximo.
7. Na página Specify networking (Especificar redes), selecione valores para os seguintes campos:
   - VPC: escolha uma VPC existente que atenda aos Requisitos de VPC do Amazon EKS na qual criar o cluster. Antes de escolher uma VPC, recomendamos familiarizar-se com todos os requisitos e considerações em the section called “Requisitos para sub-rede e VPC”. Não será possível alterar quais VPCs você deseja utilizar depois de criar o cluster. Se nenhuma VPC estiver listada, você precisará criar uma primeiro. Para obter mais informações, consulte the section called “Crie uma VPC”.
   - Subnets (Sub-redes): por padrão, as sub-redes disponíveis na VPC especificada no campo anterior são pré-selecionadas. É necessário selecionar pelo menos duas.

     As sub-redes escolhidas devem atender aos Requisitos para sub-redes do Amazon EKS. Antes de selecionar sub-redes, convém estar familiarizado com todos os Requisitos e considerações sobre VPCs e sub-redes do Amazon EKS.

Security groups (Grupos de segurança) (Opcional): especifique um ou mais grupos de segurança que você deseja que o Amazon EKS associe às interfaces de rede que ele cria.

Independentemente de você escolher algum grupo de segurança, o Amazon EKS cria um grupo de segurança que permite comunicação entre seu cluster e sua VPC. O Amazon EKS associa esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte the section called “Requisitos para grupos de segurança”. É possível modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS.

- Escolher família de endereços IP do cluster: é possível escolher IPv4 e IPv6.

  Por padrão, o Kubernetes atribui endereços `IPv4` aos pods e serviços. Antes de decidir usar a família `IPv6`, verifique se você está familiarizado com todas as considerações e requisitos dos tópicos Requisitos e considerações da VPC, the section called “Requisitos e considerações para sub-redes”, the section called “Requisitos para grupos de segurança” e the section called “IPv6”. Caso escolha a família `IPv6`, você não poderá especificar um intervalo de endereços para o Kubernetes atribuir endereços de serviço `IPv6` da mesma forma que faz para a família `IPv4`. O Kubernetes atribui endereços de serviço do intervalo exclusivo de endereços locais (`fc00::/7`).

- (Opcional) Escolha Configurar intervalo de endereços IP do Kubernetes e especifique um Intervalo **`IPv4`** de serviços.

  Especificar seu próprio intervalo pode ajudar a evitar conflitos entre serviços do Kubernetes e outras redes com emparelhamento ou conectadas à VPC. Insira um intervalo em notação CIDR. Por exemplo: `10.2.0.0/16`.

  O bloco CIDR deve atender aos seguintes requisitos:

  - Estar dentro de um dos seguintes intervalos: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.
  - Ter um tamanho mínimo de `/24` e máximo de `/12`.
  - Não se sobrepor ao intervalo da VPC para seus recursos do Amazon EKS.

Apenas é possível especificar essa opção ao utilizar a família de endereços `IPv4` e somente ao criar o cluster. Se isso não for especificado, o Kubernetes atribuirá endereços IP de serviço de qualquer um dos blocos CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Para Cluster endpoint access (Acesso ao endpoint do cluster), selecione uma opção. Depois de criar o cluster, você poderá alterar essa opção. Antes de selecionar uma opção não padrão, familiarize-se com as opções e suas implicações. Para obter mais informações, consulte the section called “Acesso ao endpoint do cluster”.

  Após terminar com essa página, escolha Próximo.

8. (Opcional) Na página Configurar observabilidade, escolha quais opções de Métricas e Log do ambiente de gerenciamento deseja ativar. Por padrão, o cada tipo de log está desativado.
   - Para obter mais informações sobre as opções de métricas do Prometheus, consulte the section called “Etapa 1: ativar as métricas do Prometheus”.
   - Para obter mais informações sobre as opções do Log do ambiente de gerenciamento, consulte the section called “Logs do ambiente de gerenciamento”.

   Após terminar com essa página, escolha Próximo.
9. Na página Selecionar complementos, escolha os complementos que você deseja adicionar ao cluster. Alguns complementos são pré-selecionados. Você pode escolher quantos complementos do Amazon EKS e do AWS Marketplace desejar. Se os complementos do AWS Marketplace que você deseja instalar não estiverem listados, será possível clicar na numeração da página para visualizar resultados de páginas adicionais ou pesquisar os complementos disponíveis no AWS Marketplace inserindo algum texto na caixa de pesquisa. Você também poderá pesquisar por categoria, fornecedor ou modelo de preços e depois escolher os complementos nos resultados da pesquisa. Ao criar um cluster, é possível visualizar, selecionar e instalar qualquer complemento que ofereça suporte às Identidade de Pods do EKS, conforme detalhado em the section called “Identidade de Pods”.

   Após terminar com essa página, escolha Próximo.

   Alguns complementos, como Amazon VPC CNI, CoreDNS e kube-proxy, são instalados por padrão. Se você desabilitar qualquer um dos complementos padrão, isso poderá afetar sua capacidade de executar aplicações do Kubernetes.
10. Na página Configurar opções de complementos selecionados, selecione a versão que deseja instalar. Você sempre pode atualizar para uma versão posterior após a criação do cluster.

    Para complementos com suporte para Identidade de Pods do EKS, é possível usar o console para gerar automaticamente o perfil com o nome, a política gerenciada pela AWS e a política de confiança pré-preenchidos especificamente para o complemento. É possível reutilizar perfis existentes ou criar novos perfis para complementos com suporte. Para obter as etapas de uso

do console para criar perfis para complementos que ofereçam suporte às Identidades de Pods do EKS, consulte the section called “Criar complemento (console do AWS)”. Se um complemento não oferecer suporte à Identidade de Pods do EKS, uma mensagem será exibida com instruções para usar o assistente para criar os perfis do IAM para contas de serviço (IRSA) após a criação do cluster.

Você pode atualizar a configuração de cada complemento após a criação do cluster. Para obter mais informações sobre a configuração de complementos, consulte the section called “Atualização de um complemento”. Após terminar com essa página, escolha Próximo.

11. Na página Review and create (Revisar e criar), revise as informações que você inseriu ou selecionou nas páginas anteriores. Se precisar fazer alterações, escolha Edit (Editar). Quando estiver satisfeito, escolha Criar. O campo Status mostra o campo CREATING (Criando) enquanto o cluster é provisionado.

> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte the section called “Capacidade insuficiente”.

O provisionamento de cluster leva alguns minutos.

12. Continue com the section called “Etapa 3: atualizar o kubeconfig”

Crar cluster: AWS CLI

1. Crie o cluster usando o comando a seguir. Antes da execução do comando, realize as seguintes substituições:
   - Substitua *region-code* pela região AWS na qual você deseja criar o cluster.
   - Substitua *my-cluster* por um nome para seu cluster. O nome pode conter apenas caracteres alfanuméricos (diferencia maiúsculas de minúsculas), hifens e sublinhados. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.

- Substitua *1.33* por qualquer versão compatível do Amazon EKS.
- Substitua *111122223333* pelo ID da sua conta e *myAmazonEKSClusterRole* pelo nome do perfil IAM do cluster.
- Substitua os valores de `subnetIds` pelos seus próprios valores. Também é possível adicionar outras IDs. Você deve especificar pelo menos dois IDs de sub-rede.

  As sub-redes escolhidas devem atender aos Requisitos para sub-redes do Amazon EKS. Antes de selecionar sub-redes, convém estar familiarizado com todos os Requisitos e considerações sobre VPCs e sub-redes do Amazon EKS.
- Se não quiser especificar um ID de grupo de segurança, remova `,securityGroupIds=sg-<ExampleID1>` do comando. Se quiser especificar um ou mais IDs de grupo de segurança, substitua os valores de `securityGroupIds` pelos seus próprios. Também é possível adicionar outras IDs.

  Independentemente de você escolher algum grupo de segurança, o Amazon EKS cria um grupo de segurança que permite comunicação entre seu cluster e sua VPC. O Amazon EKS associa esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte the section called “Requisitos para grupos de segurança”. É possível modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS.

```
aws eks create-cluster --region region-code --name my-cluster --kubernetes-version
 1.33 \
   --role-arn arn:aws:iam::111122223333:role/myAmazonEKSClusterRole \
   --resources-vpc-config subnetIds=subnet-ExampleID1,subnet-
ExampleID2,securityGroupIds=sg-ExampleID1
```

> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte the section called “Capacidade insuficiente”.

Veja a seguir as configurações opcionais que, se necessário, devem ser adicionadas ao comando anterior. Apenas é possível habilitar essas opções ao criar o cluster, e não depois.

- Por padrão, o EKS instala vários complementos de rede durante a criação do cluster. Isso inclui o Amazon VPC CNI, o CoreDNS e o kube-proxy.

  Se você quiser desabilitar a instalação desses complementos de rede padrão, use o parâmetro abaixo. Essa opção poderá ser usada para CNIs alternativos, como Cilium. Para obter mais informações, confira a API Reference do EKS.

  ```
  aws eks create-cluster --bootstrapSelfManagedAddons false
  ```

- Se quiser especificar de qual bloco de Encaminhamento Entre Domínios Sem Classificação (CIDR) `IPv4` o Kubernetes atribui endereços IP de serviço, será necessário especificá-lo adicionando `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>` ao comando a seguir.

  Especificar seu próprio intervalo pode ajudar a evitar conflitos entre serviços do Kubernetes e outras redes com emparelhamento ou conectadas à VPC. Insira um intervalo em notação CIDR. Por exemplo: `10.2.0.0/16`.

  O bloco CIDR deve atender aos seguintes requisitos:

  - Estar dentro de um dos seguintes intervalos: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.
  - Ter um tamanho mínimo de `/24` e máximo de `/12`.
  - Não se sobrepor ao intervalo da VPC para seus recursos do Amazon EKS.

Apenas é possível especificar essa opção ao utilizar a família de endereços `IPv4` e somente ao criar o cluster. Se isso não for especificado, o Kubernetes atribuirá endereços IP de serviço de qualquer um dos blocos CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Se estiver criando um cluster e quiser que o cluster atribua endereços `IPv6` a pods e serviços em vez de endereços `IPv4`, adicione `--kubernetes-network-config ipFamily=ipv6` ao comando a seguir.

  Por padrão, o Kubernetes atribui endereços `IPv4` aos pods e serviços. Antes de decidir usar a família `IPv6`, verifique se você está familiarizado com todas as considerações e requisitos dos tópicos Requisitos e considerações da VPC, the section called “Requisitos e considerações para sub-redes”, the section called “Requisitos para grupos de segurança” e the section called

"IPv6". Caso escolha a família `IPv6`, você não poderá especificar um intervalo de endereços para o Kubernetes atribuir endereços de serviço `IPv6` da mesma forma que faz para a família `IPv4`. O Kubernetes atribui endereços de serviço do intervalo exclusivo de endereços locais (`fc00::/7`).

2. Leva alguns minutos para provisionar o cluster. Você pode consultar o status do cluster com o comando a seguir.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

Não prossiga para a próxima etapa até que a saída recebida esteja `ACTIVE`.

3. Continue com the section called "Etapa 3: atualizar o kubeconfig"

Etapa 3: atualizar o kubeconfig

1. Se você criou seu cluster usando `eksctl`, pode ignorar esta etapa. Isso ocorre porque `eksctl` já concluiu essa etapa para você. Habilite o `kubectl` para se comunicar com o cluster adicionando um novo contexto ao arquivo `kubectl config`. Para obter mais informações sobre como criar e atualizar o arquivo, consulte the section called "Acessar meu cluster com o kubectl".

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Veja um exemplo de saída abaixo.

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

2. Confirme a comunicação com o cluster, executando o seguinte comando:

```
kubectl get svc
```

Veja abaixo um exemplo de saída.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

Etapa 4: configuração do cluster

1. (Recomendado) Para usar alguns complementos do Amazon EKS ou para permitir que workloads individuais do Kubernetes tenham permissões específicas do AWS Identity and Access Management (IAM), crie um provedor OpenID Connect (OIDC) do IAM para o cluster. Você só precisa criar um provedor OIDC do IAM para o cluster uma vez. Para saber mais sobre os complementos do Amazon EKS, consulte the section called “Complementos do Amazon EKS”. Para saber mais sobre a atribuição de permissões específicas do IAM às workloads, consulte the section called “Credenciais com IRSA”.
2. (Recomendado) Configure o cluster para o plug-in CNI da Amazon VPC para Kubernetes antes de implantar nós do Amazon EC2 no cluster. Por padrão, o plugin foi instalado com o seu cluster. Quando nós do Amazon EC2 são adicionados ao cluster, o plugin é implantado automaticamente em cada nó do Amazon EC2 que você adicionar. O plugin requer a anexação de uma das seguintes políticas do IAM a um perfil do IAM. Se o seu cluster usar a família `IPv4`, use a política do IAM gerenciada AmazonEKS_CNI_Policy. Se o seu cluster usar a família `IPv6`, use uma política do IAM criada por você.

   O perfil do IAM ao qual a política é anexada pode ser o perfil do IAM do nó ou um perfil dedicado utilizado apenas para o plugin. Convém anexar a política a esse perfil. Para obter mais informações sobre como criar o perfil, consulte the section called “Configuração para IRSA” ou perfil do IAM de nó do Amazon EKS.
3. Se você implantou seu cluster usando o Console de gerenciamento da AWS, pode ignorar esta etapa. O Console de gerenciamento da AWS implanta os complementos CoreDNS, plug-in CNI da Amazon VPC para Kubernetes e `kube-proxy` do Amazon EKS por padrão.

   Se você implementar o cluster usando o `eksctl` ou a AWS CLI, os complementos autogerenciados `kube-proxy`, CoreDNS e plug-in CNI da Amazon VPC para Kubernetes e serão implantados. É possível migrar os complementos autogerenciados `kube-proxy`, CoreDNS e plug-in CNI da Amazon VPC para Kubernetes que são implantados com o cluster, para complementos do Amazon EKS. Para obter mais informações, consulte the section called “Complementos do Amazon EKS”.
4. (Opcional) Se ainda não fez isso, você poderá habilitar as métricas do Prometheus para o cluster. Para obter mais informações, consulte Criar um extrator no Guia do usuário do Amazon Managed Service for Prometheus.
5. Caso planeje implementar workloads no cluster que usem volumes do Amazon EBS, você deverá instalar a CSI do Amazon EBS no cluster antes de implementar as workloads.

Próximas etapas

- A entidade principal do IAM que criou o cluster é a única que tem acesso ao cluster. Conceda permissões a outras entidades principais do IAM para que elas possam acessar o cluster.
- Se a entidade principal do IAM que criou o cluster tiver apenas as permissões mínimas do IAM referenciadas nos pré-requisitos, então talvez seja interessante adicionar mais permissões do Amazon EKS a essa entidade principal. Para obter mais informações sobre como conceder permissões do Amazon EKS a entidades principais do IAM, consulte the section called “Referência do IAM”.
- Se quiser que a entidade principal do IAM que criou o cluster, ou qualquer outra entidade principal, visualize os recursos do Kubernetes no console do Amazon EKS, conceda as permissões necessárias às entidades.
- Se quiser que os nós e as entidades principais do IAM acessem o cluster do dentro da VPC, habilite o endpoint privado do seu cluster. Por padrão, o endpoint público está habilitado. É possível desativar o endpoint público depois de habilitar o endpoint privado, se desejado. Para obter mais informações, consulte the section called “Acesso ao endpoint do cluster”.
- Habilitar a criptografia de segredos para o seu cluster.
- Configurar o registro em log para o seu cluster.
- Adicionar nós ao seu cluster.

Ambiente de gerenciamento provisionado do Amazon EKS

Visão geral

O ambiente de gerenciamento provisionado do Amazon EKS é um recurso que permite aos administradores de cluster selecionar entre diferentes níveis de escalabilidade, definindo o nível ideal para garantir que o ambiente de gerenciamento do cluster tenha alta performance e previsibilidade. Com isso, administradores de cluster podem assegurar que o ambiente de gerenciamento esteja sempre provisionado conforme a capacidade definida.

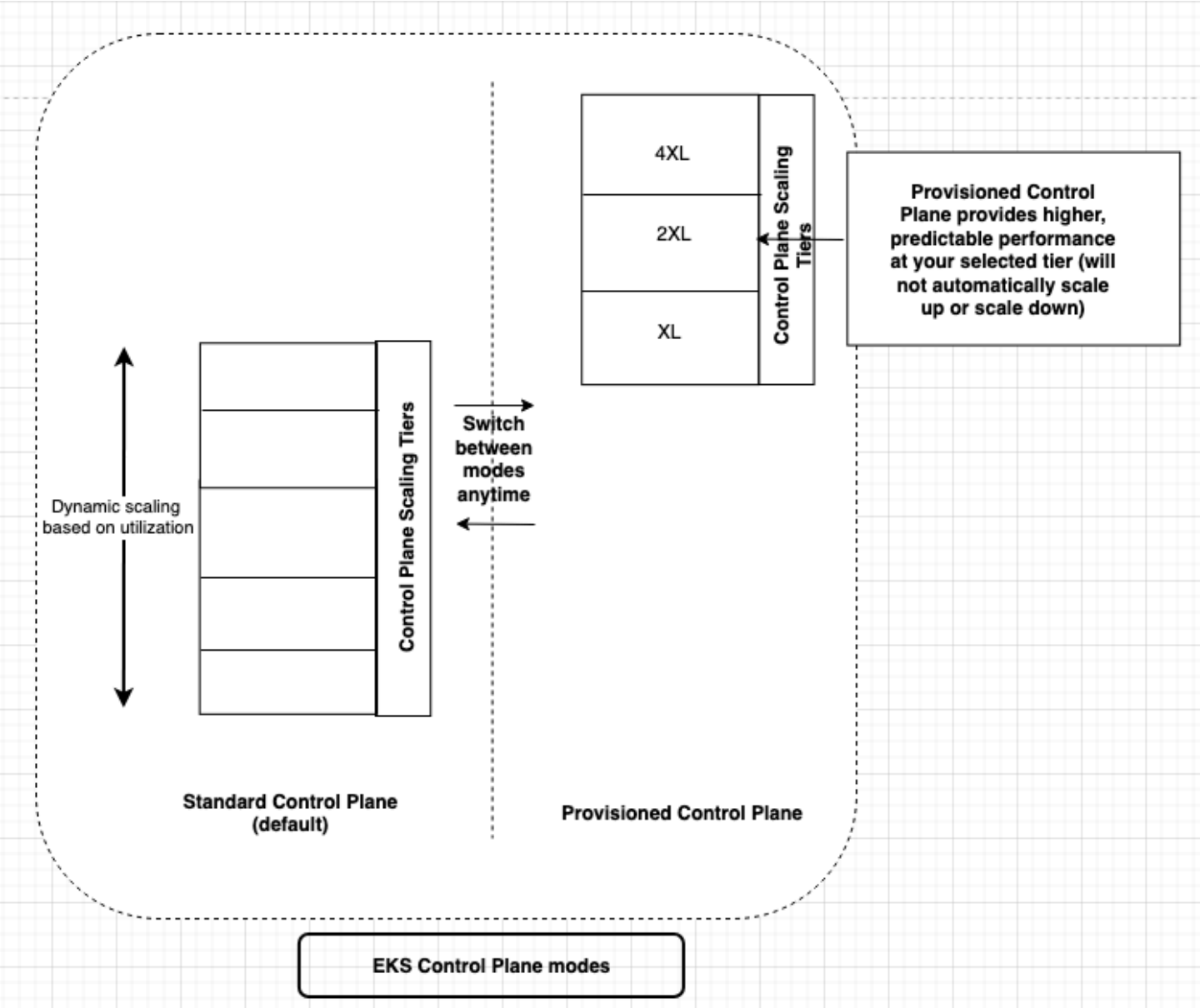
O Amazon EKS oferece dois modos de operação para o ambiente de gerenciamento do cluster. Por padrão, os clusters do Amazon EKS usam o modo padrão, no qual o ambiente de gerenciamento aumenta ou reduz a escala verticalmente, de forma automática, com base nas demandas da workload. O modo padrão aloca dinamicamente capacidade suficiente do ambiente de gerenciamento para atender às necessidades da workload e é a solução recomendada para a

maioria dos casos de uso. No entanto, para workloads especializadas que não admitem qualquer variabilidade da performance devido à escalabilidade do ambiente de gerenciamento, ou para aquelas que exigem grandes quantidades de capacidade, é possível optar pelo modo provisionado. Com o modo provisionado, você pode realizar a alocação prévia da capacidade do ambiente de gerenciamento, garantindo prontidão para atender a demandas rigorosas da workload.

> **Note**
>
> O modo provisionado é um modo de operação adicional do ambiente de gerenciamento, disponível em conjunto com o modo padrão original. A introdução do modo provisionado não altera o comportamento do modo padrão.

Com o ambiente de gerenciamento provisionado do EKS, os administradores de cluster podem realizar o provisionamento prévio da capacidade desejada de forma antecipada, garantindo alta performance e previsibilidade constante ao ambiente de gerenciamento do cluster. O ambiente de gerenciamento provisionado do EKS também permite que os administradores de cluster provisionem a mesma capacidade do ambiente de gerenciamento em diversos ambientes, desde o ambiente de preparação até a produção e locais de recuperação de desastres. Isso é fundamental para assegurar a consistência e a previsibilidade da performance do ambiente de gerenciamento em todos os ambientes. Por fim, o ambiente de gerenciamento provisionado do EKS oferece acesso a níveis altíssimos de performance no ambiente de gerenciamento, permitindo a execução de workloads de IA com escalabilidade massiva, computação de alta performance e processamento de dados em grande escala no Kubernetes.

Por padrão, todos os clusters do Amazon EKS, tanto os novos quanto os já existentes, operam no modo padrão. Para clusters que requerem uma performance alta e previsível do ambiente de gerenciamento, você pode optar pelo recurso de ambiente de gerenciamento provisionado do EKS. A cobrança será feita com base na tarifa horária do nível de escalabilidade escolhido para o ambiente de gerenciamento, somada às taxas horárias do EKS para suporte padrão ou estendido. Para obter mais informações sobre preços, consulte Preços do Amazon EKS.



Casos de uso

O ambiente de gerenciamento provisionado do EKS foi projetado para atender a cenários específicos em que a performance alta e previsível do ambiente de gerenciamento é fundamental para as operações. Compreender esses casos de uso pode ajudar você a determinar se o ambiente de gerenciamento provisionado do EKS é a solução ideal para as workloads.

Workloads sensíveis à performance: para as workloads que demandam baixa latência e máxima performance do ambiente de gerenciamento do Kubernetes, o ambiente de gerenciamento provisionado do EKS oferece uma capacidade que remove instabilidades causadas pela escalabilidade do ambiente de gerenciamento.

Workloads com escalabilidade massiva: se você executa workloads altamente escaláveis, como treinamento e inferência de IA, computação de alta performance ou processamento de dados em grande escala, que exigem um grande número de nós no cluster, o ambiente de gerenciamento provisionado fornece a capacidade necessária para suportar essas workloads exigentes.

Eventos de alta demanda antecipados: quando você espera um aumento súbito nas solicitações ao ambiente de gerenciamento devido a um evento próximo, como vendas ou promoções de comércio eletrônico, lançamentos de produtos, temporadas de compras ou grandes eventos esportivos e de entretenimento, o ambiente de gerenciamento provisionado permite ajustar a escalabilidade da capacidade antecipadamente. Essa abordagem proativa garante que o ambiente de gerenciamento esteja pronto para lidar com o aumento de carga, sem a necessidade de esperar que a escalabilidade automática responda à demanda.

Consistência de ambientes: o ambiente de gerenciamento provisionado permite replicar a capacidade e a performance do ambiente de gerenciamento entre os ambientes de preparação e de produção, ajudando você a identificar problemas potenciais antecipadamente, antes da implantação em produção. Ao manter o mesmo nível do ambiente de gerenciamento em diferentes ambientes, você garante que os resultados dos testes reflitam com precisão o comportamento em ambientes de produção, evitando imprevistos de performance no momento da implantação.

Recuperação de desastres e continuidade de negócios: para cenários de recuperação de desastres, o ambiente de gerenciamento provisionado permite que você provisione ambientes de failover com o mesmo nível de capacidade do ambiente primário. Isso garante o mínimo de interrupção e uma recuperação rápida durante eventos de failover, pois o cluster de recuperação de desastres terá características de performance do ambiente de gerenciamento idênticas às do cluster de produção desde o instante de sua ativação.

Níveis de escalabilidade do ambiente de gerenciamento

O ambiente de gerenciamento provisionado do EKS disponibiliza níveis de escalabilidade que usam tamanhos de referência (XL, 2XL e 4XL). Cada nível define a capacidade por meio de três atributos principais do Kubernetes que determinam as características de performance do ambiente de gerenciamento do cluster. Compreender esses atributos ajuda você a selecionar o nível adequado para os requisitos da workload.

A simultaneidade de solicitações de API mede o número de solicitações que o servidor de API do ambiente de gerenciamento do Kubernetes pode processar simultaneamente, o que é fundamental para workloads com alto throughput.

A taxa de agendamento de pods indica a rapidez com que o agendador padrão do Kubernetes consegue agendar pods em nós, medida em pods por segundo.

O tamanho do banco de dados do cluster indica o espaço de armazenamento alocado para o etcd, o banco de dados que contém o estado e os metadados do cluster.

Ao provisionar o ambiente de gerenciamento do cluster em um determinado nível de escalabilidade usando o ambiente de gerenciamento provisionado, o EKS garante que o ambiente de gerenciamento mantenha os limites correspondentes a esse nível. Os limites dos níveis de escalabilidade do ambiente de gerenciamento variam conforme a versão do Kubernetes, como mostrado nas tabelas a seguir.

EKS v1.28 e v1.29

Nível de escalabilidade do ambiente de gerenciamento provisionado	Simultaneidade de solicitações de API (estações)	Taxa de agendamento de pods (pods por segundo)	Tamanho do banco de dados do cluster (GB)
XL	1.700	100	16
2XL	3.400	100	16
4XL	6.800	100	16

EKS v1.30 e versões posteriores

Nível de escalabilidade do ambiente de gerenciamento provisionado	Simultaneidade de solicitações de API (estações)	Taxa de agendamento de pods (pods por segundo)	Tamanho do banco de dados do cluster (GB)
XL	1.700	167	16
2XL	3.400	283	16
4XL	6.800	400	16

Monitoramento da utilização do nível de escalabilidade do ambiente de gerenciamento

O Amazon EKS disponibiliza várias métricas que auxiliam no monitoramento do uso do nível de escalabilidade do ambiente de gerenciamento. Essas métricas são publicadas como métricas do Amazon CloudWatch e estão acessíveis por meio dos consoles do CloudWatch e do EKS. Além disso, essas métricas podem ser coletadas via scraping do endpoint de Prometheus do cluster do EKS (acesse aqui).

	Métricas do Prometheus	Métrica do CloudWatch
Concorrência de solicitações de API	apiserver_flowcontrol_current_executing_seats	apiserver_flowcontrol_current_executing_seats
Taxa de agendamento de pods	scheduler_schedule_attempts_total	scheduler_schedule_attempts_total, scheduler_schedule_attempts_SCHEDULED, scheduler_schedule_attempts_UNSCHEDULABLE
Tamanho do banco de dados do cluster	apiserver_storage_size_bytes	apiserver_storage_size_bytes

Noções básicas sobre a diferença entre a capacidade do nível e a performance real

Ao selecionar um nível de escalabilidade do ambiente de gerenciamento provisionado, os atributos desse nível representam as configurações subjacentes que o Amazon EKS aplica ao ambiente de gerenciamento. Contudo, a performance real alcançada varia de acordo com os padrões da workload, as configurações e a aderência às práticas recomendadas do Kubernetes. Por exemplo, enquanto o nível 4XL define o recurso de prioridade e de equidade da API (APF, na sigla em inglês) com 6.800 estações de solicitações simultâneas, o throughput real de solicitações que você obtém do ambiente de gerenciamento depende dos tipos de operações realizadas. Por exemplo, o Kubernetes penaliza solicitações do tipo “list” mais do que as do tipo “get” e, por isso, o número efetivo de solicitações “list” processadas simultaneamente pelo ambiente de gerenciamento é menor do que o de solicitações “get” (confira aqui). De forma semelhante, ainda que o QPS do agendador padrão seja 400 no nível 4XL, a taxa real de agendamento de pods depende de fatores como o estado de prontidão e a integridade dos nós para o agendamento. Para alcançar a performance

ideal, certifique-se de que as aplicações sigam as práticas recomendadas do Kubernetes (confira aqui) e estejam configuradas adequadamente para as características da workload.

Considerações

- Capacidade do ambiente de gerenciamento padrão: o modo padrão de ambiente de gerenciamento do EKS oferece a melhor relação entre custo e benefício e é a opção recomendada para a maioria dos casos de uso. No entanto, para workloads especializadas que não admitem qualquer variabilidade da performance devido à escalabilidade do ambiente de gerenciamento, ou para aquelas que exigem grandes quantidades de capacidade, considere usar o modo provisionado.
- Adesão necessária: os clusters atuais não realizam o aumento da escala verticalmente de forma automática do ambiente de gerenciamento padrão para um nível com preços superiores do ambiente de gerenciamento provisionado do EKS. É necessário aderir explicitamente a um dos novos níveis de escalabilidade do ambiente de gerenciamento provisionado do EKS.
- Restrição de saída: o modo padrão do ambiente de gerenciamento fornece suporte a um tamanho de banco de dados do cluster (etcd) de até 8 GB. Se o tamanho do banco de dados do cluster exceder 8 GB durante o uso do modo provisionado, você não poderá retornar ao modo padrão até reduzir o tamanho do banco de dados para menos de 8 GB. Por exemplo, se você estiver usando 14 GB de armazenamento de banco de dados no modo provisionado, deverá primeiro reduzir a utilização do banco de dados para menos de 8 GB antes de retornar ao modo padrão.
- Não há escalabilidade automática de nível: o ambiente de gerenciamento provisionado do EKS não realiza a transição automática entre os níveis. Após escolher um nível de escalabilidade, o ambiente de gerenciamento do cluster fica vinculado a esse nível, o que assegura uma performance constante e previsibilidade. No entanto, você tem a flexibilidade de implementar sua própria solução de ajuste de escala automático ao monitorar as métricas de utilização do nível e usar as APIs do ambiente de gerenciamento provisionado do EKS para reduzir ou aumentar a escala verticalmente quando as métricas ultrapassarem os limites definidos por você, garantindo total controle sobre a estratégia de escalabilidade e otimização de custos.
- Consulta do nível atual: é possível usar o console do Amazon EKS, a Amazon Web Services CLI ou a API para verificar em qual nível de escalabilidade o ambiente de gerenciamento se encontra. Na CLI, você pode executar o comando `describe-cluster`: `aws eks describe-cluster --name cluster-name`
- Tempo de transição de nível: você pode usar o console do Amazon EKS, as APIs do Amazon EKS ou a CLI para sair de um nível ou alternar entre os níveis de escalabilidade. O Amazon EKS introduziu um novo tipo de atualização de cluster chamado `ScalingTierConfigUpdate`, que

você pode inspecionar para monitorar o progresso da transição. Após executar um comando de alteração de nível, é possível listar as atualizações no cluster para obter uma nova atualização do tipo `ScalingTierConfigUpdate` com o status `Updating`. Esse status passará para `Successful` quando a alteração for finalizada ou para `Failed` em caso de falha. O campo de erro na atualização indica o motivo da falha Não há restrições quanto à frequência com que você pode alternar entre os níveis. A alteração do nível do ambiente de gerenciamento leva alguns minutos para ser concluída.

- Seleção do nível ideal: para determinar o nível de escalabilidade do ambiente de gerenciamento provisionado ideal para o cluster, você pode realizar testes de carga provisionando o cluster no nível máximo (4XL). Depois, execute um teste de carga para simular o cenário de maior demanda no ambiente de gerenciamento do cluster. Monitore as métricas de utilização do nível do ambiente de gerenciamento no momento de carga máxima e empregue esses dados para escolher o nível de escalabilidade adequado no modo provisionado.
- Preços do ambiente de gerenciamento provisionado: a cobrança será feita com base na taxa horária correspondente ao nível de escalabilidade do ambiente de gerenciamento provisionado em que o cluster estiver operando. Este valor é adicional às taxas horárias do suporte padrão ou do suporte estendido. Consulte a página de preços do Amazon EKS para obter mais detalhes.
- Nível de escalabilidade superior: se você pretende executar o cluster em um nível de escalabilidade superior ao 4XL, entre em contato com a equipe da conta da Amazon Web Services para obter informações adicionais sobre preços.
- Suporte a versões e regiões do Kubernetes: o ambiente de gerenciamento provisionado do EKS é compatível com todas as regiões comerciais da Amazon Web Services, da GovCloud e da China. O ambiente de gerenciamento provisionado é compatível com o EKS na versão 1.28 ou em versões posteriores.

Conceitos básicos do ambiente de gerenciamento provisionado do Amazon EKS

Este guia apresenta o passo a passo para configurar e usar o ambiente de gerenciamento provisionado do EKS por meio da AWS CLI e do Console de gerenciamento da AWS.

Pré-requisitos

Antes de começar, verifique se você tem:

- AWS CLI - Uma ferramenta de linha de comando para trabalhar com os serviços AWS, incluindo o Amazon EKS. Para obter mais informações, consulte Instalação no Guia do usuário da AWS Command Line Interface. Depois de instalar a AWS CLI, recomendamos que você também a configure. Para obter mais informações, consulte Configuração rápida com o comando "aws configure" no Guia do usuário da AWS Command Line Interface. Observe que o AWS CLI v2 é necessário para usar a opção update-kubeconfig mostrada nesta página.
- Permissões do IAM necessárias: a entidade principal de segurança do IAM que você está usando deve ter permissões para trabalhar com perfis do IAM do Amazon EKS, perfis vinculadas a serviços, AWS CloudFormation, uma VPC e recursos relacionados. Para obter mais informações, consulte Ações e Uso de perfis vinculados ao serviço no Guia do usuário do IAM. Você deve concluir todas as etapas deste manual como o mesmo usuário. Execute o seguinte comando para verificar o usuário atual:

```
aws sts get-caller-identity
```

> **Note**
>
> Convém concluir as etapas neste tópico em um shell Bash. Se não estiver utilizando um shell Bash, alguns comandos de script, como caracteres de continuação de linha e a forma como as variáveis são definidas e utilizadas, exigirão o ajuste do seu shell. Além disso, as regras de citação e de escape do seu shell podem ser diferentes. Para obter mais informações, consulte Usar de aspas com strings na AWS CLI no Guia do usuário da AWS Command Line Interface.

Ambiente de gerenciamento provisionado do EKS: AWS CLI

Crie um cluster com o nível de escalabilidade do ambiente de gerenciamento provisionado do EKS

```
aws eks create-cluster --name prod-cluster \
--role-arn arn:aws:iam::012345678910:role/eks-service-role-AWSServiceRoleForAmazonEKS-
J7ONKE3BQ4PI \
--resources-vpc-config subnetIds=subnet-6782e71e,subnet-
e7e761ac,securityGroupIds=sg-6979fe18 \
--control-plane-scaling-config tier=tier-xl
```

Resposta:

```
{
    "cluster": {
        "name": "my-eks-cluster",
        "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
        "createdAt": "2024-03-14T11:31:44.348000-04:00",
        "version": "1.26",
        "endpoint": "https://JSA79429HJDASKJDJ8223829MNDNASW.yl4.us-
east-2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0fb75d2d8401716e7",
                "subnet-02184492f67a3d0f9",
                "subnet-04098063527aab776",
                "subnet-0e2907431c9988b72",
                "subnet-04ad87f71c6e5ab4d",
                "subnet-09d912bb63ef21b9a"
            ],
            "securityGroupIds": [
                "sg-0c1327f6270afbb36"
            ],
            "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
            "vpcId": "vpc-0012b8e1cc0abb17d",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": true,
            "publicAccessCidrs": [
                "22.19.18.2/32"
            ]
        },
        "controlPlaneScalingConfig": {
            "tier": "tier-xl"
        },
        "kubernetesNetworkConfig": {
            "serviceIpv4Cidr": "10.100.0.0/16",
            "ipFamily": "ipv4"
        },
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
```

```
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
JSA79429HJDASKJDJ8223829MNDNASW"
            }
        },
        "status": "CREATING",
        "certificateAuthority": {
            "data": "CA_DATA_STRING..."
        },
        "platformVersion": "eks.14",
        "tags": {
            "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
            "alpha.eksctl.io/cluster-name": "my-eks-cluster",
            "karpenter.sh/discovery": "my-eks-cluster",
            "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
            "auto-delete": "no",
            "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
            "EKS-Cluster-Name": "my-eks-cluster",
            "alpha.eksctl.io/cluster-oidc-enabled": "true",
            "aws:cloudformation:logical-id": "ControlPlane",
            "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
            "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
        },
        "health": {
            "issues": []
        },
        "accessConfig": {
            "authenticationMode": "API_AND_CONFIG_MAP"
        }
    }
}
```

Visualize o nível de escalabilidade do ambiente de gerenciamento provisionado do cluster

```
aws eks describe-cluster --name prod-cluster
```

Resposta:

```
{
    "cluster": {
        "name": "my-eks-cluster",
        "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
        "createdAt": "2024-03-14T11:31:44.348000-04:00",
        "version": "1.26",
        "endpoint": "https://JSA79429HJDASKJDJ8223829MNDNASW.yl4.us-
east-2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0fb75d2d8401716e7",
                "subnet-02184492f67a3d0f9",
                "subnet-04098063527aab776",
                "subnet-0e2907431c9988b72",
                "subnet-04ad87f71c6e5ab4d",
                "subnet-09d912bb63ef21b9a"
            ],
            "securityGroupIds": [
                "sg-0c1327f6270afbb36"
            ],
            "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
            "vpcId": "vpc-0012b8e1cc0abb17d",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": true,
            "publicAccessCidrs": [
                "22.19.18.2/32"
            ]
        },
        "controlPlaneScalingConfig": {
            "tier": "tier-xl"
        },
        "kubernetesNetworkConfig": {
            "serviceIpv4Cidr": "10.100.0.0/16",
            "ipFamily": "ipv4"
        },
```

```
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
JSA79429HJDASKJDJ8223829MNDNASW"
            }
        },
        "status": "ACTIVE",
        "certificateAuthority": {
            "data": "CA_DATA_STRING..."
        },
        "platformVersion": "eks.14",
        "tags": {
            "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
            "alpha.eksctl.io/cluster-name": "my-eks-cluster",
            "karpenter.sh/discovery": "my-eks-cluster",
            "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
            "auto-delete": "no",
            "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
            "EKS-Cluster-Name": "my-eks-cluster",
            "alpha.eksctl.io/cluster-oidc-enabled": "true",
            "aws:cloudformation:logical-id": "ControlPlane",
            "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
            "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
        },
        "health": {
            "issues": []
        },
```

```
        "accessConfig": {
            "authenticationMode": "API_AND_CONFIG_MAP"
        }
    }
}
```

Atualize o cluster para usar o ambiente de gerenciamento provisionado do EKS

```
aws eks update-cluster-config --name prod-cluster \
--control-plane-scaling-config tier=tier-2xl
```

Resposta:

```
{
    "update": {
        "id": "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd",
        "status": "InProgress",
        "type": "ScalingTierConfigUpdate",
        "params": [
            {
                "type": "UpdatedTier",
                "value": "tier-2xl"
            },
            {
                "type": "PreviousTier",
                "value": "tier-xl"
            }
        ],
        "createdAt": 1565807210.37,
        "errors": []
    }
}
```

Visualize a atualização de escalabilidade do ambiente de gerenciamento provisionado

```
aws eks list-updates --name example
```

Resposta:

```
{
    "updateIds": [
```

```
        "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd1"
    ]
}
```

Migre do ambiente de gerenciamento provisionado para o ambiente de gerenciamento padrão

```
aws eks update-cluster-config --name prod-cluster \
--control-plane-scaling-config tier=standard
```

Resposta:

```
{
    "update": {
        "id": "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd",
        "status": "InProgress",
        "type": "ScalingTierConfigUpdate",
        "params": [
            {
                "type": "UpdatedTier",
                "value": "standard"
            },
            {
                "type": "PreviousTier",
                "value": "tier-2xl"
            }
        ],
        "createdAt": 1565807210.37,
        "errors": []
    }
}
```

Ambiente de gerenciamento provisionado do EKS: Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Selecione Criar cluster.
3. Em Opções de configuração, selecione Configuração personalizada.
4. Role a tela para baixo até o nível de escalabilidade do ambiente de gerenciamento. Selecione Usar um nível de escalabilidade para habilitar o ambiente de gerenciamento provisionado.
5. Selecione o nível de escalabilidade do ambiente de gerenciamento que você gostaria de provisionar para o cluster entre as várias opções, como XL, 2XL e 4XL.

6. Escolha outras opções para a configuração do cluster conforme a necessidade. Na etapa final, selecione Criar cluster. Lembre-se de que a criação do cluster pode levar vários minutos para ser concluída.

Prepare-se para as atualizações da versão e a solução de problemas de configurações incorretas do Kubernetes com insights do cluster

Os insights do cluster do Amazon EKS fornecem detecção de problemas e recomendações para resolvê-los e ajudar você a gerenciar o cluster. Cada cluster do Amazon EKS passa por verificações automáticas e recorrentes em relação a uma lista de insights com curadoria do Amazon EKS. Essas verificações de insights são totalmente gerenciadas pelo Amazon EKS e oferecem recomendações sobre como lidar com quaisquer descobertas.

Tipos de insights de cluster

- Insights de configurações: identificam configurações incorretas na configuração do EKS Hybrid Nodes que podem prejudicar a funcionalidade do cluster ou das workloads.
- Insights de atualizações: identificam problemas que podem afetar a capacidade de atualizar para novas versões do Kubernetes.

Considerações

- Frequência: o Amazon EKS atualiza os insights do cluster a cada 24 horas, mas você também pode atualizá-los manualmente para ver o status mais recente. Por exemplo, você pode atualizar manualmente os insights do cluster depois de resolver um problema para ver se o problema foi resolvido.
- Permissões: o Amazon EKS cria automaticamente uma entrada de acesso ao cluster para insights de cluster em cada cluster do EKS. Essa entrada dá permissão ao EKS para visualizar as informações sobre o cluster. O Amazon EKS usa essas informações para gerar os insights. Para obter mais informações, consulte the section called “AmazonEKSClusterInsightsPolicy”.

Casos de uso

Os insights de cluster no Amazon EKS fornecem verificações automatizadas para ajudar a manter a integridade, a confiabilidade e a configuração ideal de seus clusters Kubernetes. Abaixo estão os principais casos de uso para insights de cluster, incluindo prontidão para upgrade e solução de problemas de configuração.

Insights de atualização

Os Insights de atualizações são um tipo específico de verificações de insights nos insights do cluster. Essas verificações retornam insights relacionados à prontidão para a atualização da versão do Kubernetes. O Amazon EKS executa verificações de insights de atualização em cada cluster do EKS.

> **Important**
>
> O Amazon EKS reverteu temporariamente um recurso que exigia que você usasse um sinalizador `--force` para atualizar o cluster quando havia certos problemas de insights do cluster. Para obter mais informações, consulte Temporary rollback of enforcing upgrade insights on update cluster version no GitHub.
> Para obter mais informações sobre como atualizar o cluster, consulte the section called "Etapa 3: atualizar o ambiente de gerenciamento do cluster".

Antes de atualizar a versão do cluster do Kubernetes, é possível usar a guia Insights sobre upgrade do painel de observabilidade no console do Amazon EKS. Se seu cluster identificou problemas, revise-os e faça as correções apropriadas. Os problemas incluem links para o Amazon EKS e a documentação do Kubernetes. Depois de corrigir o problema, atualize os insights do cluster sob demanda para obter os insights mais recentes. Se todos os problemas tiverem sido resolvidos, atualize seu cluster.

O Amazon EKS retorna insights relacionados à prontidão para a atualização da versão do Kubernetes. Os insights de atualização identificam possíveis problemas que podem afetar as atualizações do cluster do Kubernetes. Isso minimiza o esforço que os administradores despendem na preparação das atualizações e aumenta a confiabilidade das aplicações nas versões mais recentes do Kubernetes. Os clusters são verificados automaticamente pelo Amazon EKS em relação a uma lista de possíveis problemas que afetam a atualização da versão do Kubernetes. O

Amazon EKS atualiza frequentemente a lista de verificações de insights com base nas análises das alterações feitas em cada versão do Kubernetes.

Os insights de atualização do Amazon EKS aceleram o processo de teste e verificação de novas versões. Eles também permitem que administradores de clusters e desenvolvedores de aplicações aproveitem os recursos mais recentes do Kubernetes, destacando os problemas e oferecendo orientações sobre remediação.

Insights de configuração

Os insights do cluster do EKS verificam automaticamente os clusters do Amazon EKS com nós híbridos para identificar problemas de configuração que prejudicam a comunicação entre webhook e o ambiente de gerenciamento do Kubernetes, os comandos do kubectl como exec e logs e muito mais. Os insights de configurações revelam problemas e fornecem recomendações de correção, acelerando o tempo necessário para uma configuração de nós híbridos totalmente funcionais.

Conceitos básicos

Para ver a lista de verificações de insights realizadas e quaisquer problemas relevantes identificados pelo Amazon EKS, você usar o Console de gerenciamento da AWS, a AWS CLI, os SDKs da AWS e a operação da API `ListInsights` do Amazon EKS. Para começar, consulte o the section called “Visualizar insights de clusters”.

Visualizar insights de clusters

O Amazon EKS fornece dois tipos de insights: Insights de configurações e Insights de atualizações. Os Insights de configurações identificam configurações incorretas na configuração do EKS Hybrid Nodes que podem prejudicar a funcionalidade do cluster ou das workloads. Os Insights de atualizações identificam problemas que podem afetar a capacidade de atualizar para novas versões do Kubernetes.

Para ver a lista de verificações de insights realizadas e quaisquer problemas relevantes identificados pelo Amazon EKS, você pode chamar a aparência no Console de gerenciamento da AWS, a AWS CLI, os SDKs da AWS e a operação da API `ListInsights` do Amazon EKS.

Visualizar insights de configurações (console)

1. Abra o console do Amazon EKS.
2. Na lista Clusters, selecione o nome do cluster de Amazon EK cujos insights deseja visualizar.

3. Escolha Monitorar cluster.
4. Escolha a guia Integridade do cluster.
5. Na tabela Insights de configuração, você verá as colunas a seguir:
   - Nome: a verificação que foi realizada pelo Amazon EKS em relação ao cluster.
   - Status do insight: um insight com status de `Error` significa que há uma configuração incorreta que provavelmente está afetando a funcionalidade do cluster. Um insight com status de `Warning` significa que a configuração não corresponde à abordagem documentada, mas que a funcionalidade do cluster poderá funcionar se você a tiver configurado intencionalmente. Um insight com status de `Passing` significa que o Amazon EKS não encontrou nenhum problema associado a essa verificação de insights no cluster.
   - Versão: a versão aplicável.
   - Horário da última atualização: a hora em que o status do insight foi atualizado pela última vez para o cluster.
   - Descrição: informações da verificação de insights, que incluem o alerta e as ações recomendadas para remediação.

Visualizar insights de atualizações (console)

1. Abra o console do Amazon EKS.
2. Na lista Clusters, selecione o nome do cluster de Amazon EK cujos insights deseja visualizar.
3. Escolha Monitorar cluster.
4. Escolha a guia Insights de atualização.
5. Para visualizar os dados mais recentes, escolha o botão Atualizar insights e aguarde a conclusão da atualização.
6. Na tabela Insights de atualização, você verá as colunas a seguir:
   - Nome: a verificação que foi realizada pelo Amazon EKS em relação ao cluster.
   - Status do insight: um insight com status de "Erro" normalmente significa que a versão do Kubernetes afetada é N+1 da versão atual do cluster, enquanto um status de "Aviso" significa que o insight se aplica a uma versão do Kubernetes futura N+2 ou mais. Um insight com status de "Aprovado" significa que o Amazon EKS não encontrou nenhum problema associado a essa verificação de insights em seu cluster. Um status de insight de "Desconhecido" significa que o Amazon EKS não consegue determinar se seu cluster é afetado por essa verificação de insights.

- Versão: a versão do Kubernetes que o insight verificou em busca de possíveis problemas.
- Horário da última atualização: a hora em que o status do insight foi atualizado pela última vez para o cluster.
- Horário da última transição: a hora em que o status do insight foi alterado pela última vez.
- Descrição: informações da verificação de insights, que incluem o alerta e as ações recomendadas para remediação.

Exibir insights do cluster (AWS CLI)

1. Para visualizar os dados mais recentes, atualize os insights de um cluster especificado. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.
   - Substitua *region-code* pelo código da sua região da AWS.
   - Substitua *my-cluster* pelo nome do cluster.

   ```
   aws eks start-insight-refresh --region region-code --cluster-name my-cluster
   ```

2. Para acompanhar o status de uma atualização de insights, execute o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

   ```
   aws eks describe-insights-refresh --cluster-name my-cluster
   ```

   Veja abaixo um exemplo de saída.

   ```
   {
       "message": "Insights refresh is in progress",
       "status": "IN_PROGRESS",
       "startedAt": "2025-07-30T13:36:09-07:00"
   }
   ```

3. Liste todos os insights de um cluster especificado. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.
   - Substitua *region-code* pelo código da sua região da AWS.
   - Substitua *my-cluster* pelo nome do cluster.

   ```
   aws eks list-insights --region region-code --cluster-name my-cluster
   ```

Veja abaixo um exemplo de saída.

```
{
"insights":
    [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
 for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
 updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
 days.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "name": "Kubelet version skew",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557309.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for kubelet versions of worker nodes in the
 cluster to see if upgrade would cause non compliance with supported Kubernetes
 kubelet version skew policy.",
            "insightStatus":
            {
                "status": "UNKNOWN",
                "reason": "Unable to determine status of node kubelet versions.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
```

```
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
days.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "name": "Cluster health issues",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for any cluster health issues that prevent
successful upgrade to the next Kubernetes version on EKS.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No cluster health issues detected.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "name": "EKS add-on version compatibility",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of installed EKS add-ons to ensure they
are compatible with the next version of Kubernetes. ",
            "insightStatus": { "status": "PASSING", "reason": "All installed EKS
add-on versions are compatible with next Kubernetes version."},
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
            "name": "kube-proxy version skew",
```

```
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of kube-proxy in cluster to see if
 upgrade would cause non compliance with supported Kubernetes kube-proxy version
 skew policy.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "kube-proxy versions match the cluster control plane
 version.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEddddd",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
 for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
 updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
 days.",
            },
        },
    ],
"nextToken": null,
}
```

4. Para obter informações descritivas sobre um insight, execute o comando a seguir. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.
   - Substitua *region-code* pelo código da sua região da AWS.
   - Substitua *a1b2c3d4-5678-90ab-cdef-EXAMPLE22222* por um ID de insight recuperado da listagem dos insights do cluster.
   - Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-insight --region region-code --id a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222 --cluster-name my-cluster
```

Veja abaixo um exemplo de saída.

```
{
  "insight":
    {
      "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "name": "Kubelet version skew",
      "category": "UPGRADE_READINESS",
      "kubernetesVersion": "1.27",
      "lastRefreshTime": 1734557309.000,
      "lastTransitionTime": 1734557309.000,
      "description": "Checks for kubelet versions of worker nodes in the cluster
 to see if upgrade would cause non compliance with supported Kubernetes kubelet
 version skew policy.",
      "insightStatus":
        {
          "status": "UNKNOWN",
          "reason": "Unable to determine status of node kubelet versions.",
        },
      "recommendation": "Upgrade your worker nodes to match the Kubernetes version
 of your cluster control plane.",
      "additionalInfo":
        {
          "Kubelet version skew policy": "https://kubernetes.io/releases/version-
skew-policy/#kubelet",
          "Updating a managed node group": "https://docs.aws.amazon.com/eks/latest/
userguide/update-managed-node-group.html",
        },
      "resources": [],
      "categorySpecificSummary":
        { "deprecationDetails": [], "addonCompatibilityDetails": [] },
    },
}
```

Atualizar um cluster existente para a nova versão do Kubernetes

Quando uma nova versão do Kubernetes está disponível no Amazon EKS, você pode atualizar o cluster do Amazon EKS para a versão mais recente.

> **Important**
>
> Depois de atualizar um cluster, não é possível revertê-lo uma versão anterior. Antes de atualizar para uma nova versão do Kubernetes, recomendamos que você analise as informações indicadas em Entender o ciclo de vida da versão do Kubernetes no EKS e também as etapas de atualização neste tópico.

Novas versões do Kubernetes às vezes introduzem alterações significativas. Assim, recomendamos que você teste o comportamento das aplicações com relação a uma nova versão do Kubernetes antes de atualizar os clusters de produção. Você pode fazer isso, criando um fluxo de trabalho de integração contínua para testar o comportamento da aplicação antes de passar para uma nova versão do Kubernetes.

O processo de atualização consiste na execução de novos nós do servidor da API pelo Amazon EKS com a versão do Kubernetes atualizada para substituir as existentes. O Amazon EKS executa verificações padrão de integridade e prontidão da infraestrutura para o tráfego de rede nesses novos nós para confirmar que eles estão funcionando conforme o esperado. Porém, depois que você iniciou a atualização do cluster, não pode pausá-la nem interrompê-la. Se qualquer uma dessas verificações falhar, o Amazon EKS reverterá a implantação de infraestrutura e seu cluster permanecerá na versão do Kubernetes anterior. A execução de aplicações não será afetada e o cluster nunca será deixado em um estado irrecuperável ou não determinista. O Amazon EKS faz backup regularmente de todos os clusters gerenciados, além de ter mecanismos para recuperar clusters, se necessário. Estamos sempre avaliando e melhorando nossos processos de gerenciamento de infraestrutura do Kubernetes.

Para atualizar o cluster, o Amazon EKS requer até cinco endereços IP disponíveis das sub-redes que você especificou ao criar o cluster. O Amazon EKS cria novas interfaces de rede elásticas (interfaces de rede) de cluster em qualquer uma das sub-redes especificadas. Como as interfaces de rede podem ser criadas em sub-redes diferentes das interfaces de rede existentes, certifique-se de que as regras do grupo de segurança permitam a comunicação de cluster necessária para qualquer uma das sub-redes que você especificou ao criar o cluster. Se alguma das sub-redes que você especificou ao criar o cluster não existir, não tiver endereços IP disponíveis suficientes ou não tiver

regras de grupo de segurança que permitam a comunicação necessária do cluster, a atualização poderá falhar.

Para garantir que o endpoint do servidor de API do cluster sempre esteja acessível, o Amazon EKS fornece um ambiente de gerenciamento do Kubernetes com alta disponibilidade e executa atualizações contínuas das instâncias do servidor de API durante as operações de atualização. Para contabilizar a alteração de endereços IP das instâncias do servidor de API compatíveis com o endpoint do servidor de API do Kubernetes, você deve garantir que seus clientes do servidor de API gerenciem as reconexões de modo eficaz. Versões recentes do `kubectl` e das bibliotecas de cliente do Kubernetes oficialmente compatíveis executam esse processo de reconexão de modo transparente.

> **Note**
>
> Para saber mais sobre o que acontece em uma atualização de cluster, consulte Best Practices for Cluster Upgrades no Guia de práticas recomendadas do EKS. Esse recurso ajuda você a planejar uma atualização e a entender a estratégia de atualização de um cluster.

Considerações sobre o Modo automático do Amazon EKS

- A capacidade computacional do Modo automático do Amazon EKS controla a versão do Kubernetes dos nós. Depois de atualizar o ambiente de gerenciamento, o Modo automático do EKS começará a atualizar incrementalmente os nós gerenciados. O Modo automático do Amazon EKS respeita os orçamentos de interrupção de pods.
- Você não precisa atualizar manualmente as capacidades do Modo automático do Amazon EKS, incluindo as capacidades de ajuste de escala automático de computação, armazenamento em blocos e balanceamento de carga.

Resumo

Veja abaixo o resumo de alto nível do processo de atualização do cluster do Amazon EKS:

1. Certifique-se de que seu cluster esteja em um estado que seja compatível com uma atualização. Isso inclui verificar as APIs do Kubernetes usadas pelos recursos implantados no cluster,

garantindo que o cluster não tenha qualquer problema de integridade. Você deve usar os insights de atualização do Amazon EKS ao avaliar a prontidão de atualização do cluster.

2. Atualize o ambiente de gerenciamento para a próxima versão secundária (por exemplo, de 1.32 para 1.33).
3. Atualize os nós no plano de dados para que correspondam aos do ambiente de gerenciamento.
4. Atualize todas as aplicações adicionais executadas no cluster (por exemplo, `cluster-autoscaler`).
5. Atualize os complementos fornecidos pelo Amazon EKS, como os incluídos por padrão:
   - Versão recomendada da CNI da Amazon VPC
   - Versão recomendada do CoreDNS
   - `kube-proxy` Versão recomendada de
6. Atualize todos os clientes que se comunicam com o cluster (por exemplo, `kubectl`).

Etapa 1: preparar-se para o upgrade

Compare a versão do Kubernetes do ambiente de gerenciamento de cluster com a versão do Kubernetes dos seus nós.

- Obtenha a versão do Kubernetes do ambiente de gerenciamento do cluster.

```
kubectl version
```

- Obtenha a versão do Kubernetes dos nós. Esse comando retorna todos os nós autogerenciados e gerenciados pelo Amazon EC2, do Fargate e híbridos. Cada pod do Fargate é listado como seu próprio nó.

```
kubectl get nodes
```

Antes de atualizar seu ambiente de gerenciamento para uma nova versão do Kubernetes, certifique-se de que a versão secundária do Kubernetes dos nós gerenciados e dos nós Fargate no seu cluster sejam iguais à versão do seu ambiente de gerenciamento. Por exemplo, se o ambiente de gerenciamento estiver executando a versão `1.29` e um dos nós estiver executando a versão `1.28`, será necessário atualizar os nós para a versão `1.29` antes de atualizar o ambiente de gerenciamento para 1.30. Também recomendamos que você atualize os nós autogerenciados e híbridos para a mesma versão do ambiente de gerenciamento antes de atualizá-lo. Para obter mais

informações, consulte the section called “Atualizar”, the section called “Métodos de atualização” e the section called “Atualizar os nós híbridos”. Se você tiver nós do Fargate com uma versão secundária inferior à versão do ambiente de gerenciamento, primeiro exclua o pod que é representado pelo nó. Em seguida, atualize seu ambiente de gerenciamento. Todos os pods restantes serão atualizados para a nova versão depois de reimplantados.

Etapa 2: analisar as considerações sobre o upgrade

Os insights do cluster do Amazon EKS verificam automaticamente os clusters em uma lista de possíveis problemas que afetam a atualização da versão do Kubernetes, como uso de APIs obsoletas do Kubernetes. O Amazon EKS atualiza periodicamente a lista de verificações de insights a serem realizadas com base nas avaliações das alterações no projeto do Kubernetes. O Amazon EKS também atualiza a lista de verificações de insights à medida que ocorrem alterações no serviço do Amazon EKS junto com as novas versões. Para obter mais informações, consulte the section called “Insights de cluster”.

Analise o Deprecated API Migration Guide na documentação do Kubernetes.

Revisão dos insights de atualização

Use os insights de atualização do Amazon EKS para identificar problemas. Para obter mais informações, consulte the section called “Visualizar insights de atualizações (console)”.

Considerações detalhadas

- Como o Amazon EKS executa um ambiente de gerenciamento de alta disponibilidade, você pode atualizar apenas uma versão secundária por vez. Para saber mais sobre esse requisito, consulte Política de suporte da distorção da versão e da versão do Kubernetes. Suponha que a versão atual do cluster seja a `1.28` e você queira atualizar para a versão `1.30`. Você deve primeiro atualizar o cluster versão `1.28` para versão `1.29` e depois atualizar o cluster versão `1.29` para a versão `1.30`.
- Analise a distorção da versão entre o `kube-apiserver` do Kubernetes e o `kubelet` nos nós.
  - Com a versão `1.28` do Kubernetes e mais recentes, o `kubelet` pode ter até três versões secundárias anteriores ao do `kube-apiserver`. Consulte a política de distorção de versão upstream do Kubernetes.
  - Se o `kubelet` em seus nós gerenciados e do Fargate estiver na versão `1.25` ou mais recente do Kubernetes, você poderá atualizar o cluster para até três versões à frente sem atualizar a versão do `kubelet`. Por exemplo, se a versão `1.25` do `kubelet` estiver ativa, você poderá

atualizar a versão do cluster do Amazon EKS de `1.25` para `1.26` e de `1.27` para `1.28`, enquanto o `kubelet` estiver na versão `1.25`.

- Como prática recomendada, antes de iniciar uma atualização, verifique se o `kubelet` nos nós está na mesma versão do Kubernetes que a do seu ambiente de gerenciamento.
- Se o cluster estiver configurado com uma versão do plug-in CNI da Amazon VPC para Kubernetes anterior à versão `1.8.0`, recomendamos atualizar o plug-in para a versão mais recente recomendada antes de atualizar o cluster. Para atualizar o plug-in, consulte the section called “CNI da Amazon VPC”.
- Você pode fazer um backup do seu cluster do Amazon EKS para permitir a restauração do estado do cluster e do armazenamento persistente em caso de falhas durante o processo de atualização. Consulte the section called “ AWS Backup”

Etapa 3: atualizar o ambiente de gerenciamento do cluster

> **Important**
>
> O Amazon EKS reverteu temporariamente um recurso que exigia que você usasse um sinalizador `--force` para atualizar o cluster quando havia certos problemas de insights do cluster. Para obter mais informações, consulte Temporary rollback of enforcing upgrade insights on update cluster version no GitHub.
> O Amazon EKS atualiza um insight de cluster 24 horas após o "horário da última atualização". Você pode comparar a hora em que resolveu um problema com a "hora da última atualização" do insight do cluster.
> Além disso, pode levar até 30 dias para que o status do insight seja atualizado após resolver a questão do uso da API obsoleta. O insights de atualização sempre busca o uso de APIs obsoletas em um período contínuo de 30 dias.

Você pode enviar a solicitação de atualização da versão do ambiente de gerenciamento do EKS usando:

- eksctl
- o console da AWS
- a AWS CLI

Atualizar o cluster - eksctl

Este procedimento exige a versão `eksctl 0.215.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar e atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Atualize a versão do Kubernetes do ambiente de gerenciamento do Amazon EKS. Substitua `<cluster-name>` pelo nome do cluster. Substitua `<version-number>` pelo número da versão compatível com o Amazon EKS para a qual você deseja atualizar o cluster. Para obter a lista de números das versões compatíveis, consulte as versões compatíveis com o Amazon EKS.

```
eksctl upgrade cluster --name <cluster-name> --version <version-number> --approve
```

O processo pode demorar alguns minutos para ser concluído.

Avance para the section called “Etapa 4: atualizar componentes do cluster”.

Atualizar o cluster: console da AWS

1. Abra o console do Amazon EKS.
2. Escolha Atualizar agora para um cluster que você deseja atualizar.
3. Selecione a versão para a qual o cluster será atualizado e escolha Atualizar.
4. O processo pode demorar alguns minutos para ser concluído. Avance para the section called “Etapa 4: atualizar componentes do cluster”.

Atualizar o cluster: AWS CLI

1. Verifique se a AWS CLI está instalada e se você se conectou. Para obter mais informações, consulte Instalar ou atualizar a versão mais recente da AWS CLI.
2. Atualize seu cluster do Amazon EKS com o seguinte comando da AWS CLI. Substitua `<cluster-name>` e `<region-code>` do cluster que você deseja atualizar. Substitua `<version-number>` pelo número da versão compatível com o Amazon EKS para a qual você deseja atualizar o cluster. Para obter a lista de números das versões compatíveis, consulte as versões compatíveis com o Amazon EKS.

```
aws eks update-cluster-version --name <cluster-name> \
  --kubernetes-version <verion-number> --region <region-code>
```

Veja abaixo um exemplo de saída.

```
{
    "update": {
        "id": "<update-id>",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "<version-number>"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
```

3. O processo pode demorar alguns minutos para ser concluído. Monitore o status do cluster com o comando a seguir. Além de usar o mesmo `<cluster-name>` e `<region-code>`, use o `<update-id>` que o comando anterior retornou.

```
aws eks describe-update --name <cluster-name> \
   --region <region-code> --update-id <update-id>
```

Quando um status de `Successful` for exibido, significa que a atualização está concluída.

4. Avance para the section called “Etapa 4: atualizar componentes do cluster”.

Etapa 4: atualizar componentes do cluster

1. Depois que a atualização do cluster for concluída, atualize os nós para a mesma versão secundária do Kubernetes do cluster atualizado. Para obter mais informações, consulte the section called “Métodos de atualização”, the section called “Atualizar” e the section called

"Atualizar os nós híbridos". Quaisquer novos pods iniciados no Fargate têm uma versão do `kubelet` que corresponde à versão do cluster. Os pods existentes do Fargate não são alterados.

2. (Opcional) Se você implantou o Kubernetes Cluster Autoscaler no cluster antes de atualizá-lo, atualize o Cluster Autoscaler para a versão mais recente que corresponde à versão principal e secundária do Kubernetes para a qual você atualizou.

   a. Abra a página de versões do Cluster Autoscaler em um navegador da web e encontre a versão mais recente do Cluster Autoscaler que corresponda às versões principal e secundária do Kubernetes do seu cluster. Por exemplo, se a versão do Kubernetes do cluster for `1.30`, localize a versão mais recente do Cluster Autoscaler que começa com `1.30`. Registre o número de versão semântico (`1.30.n`, por exemplo) dessa versão para usar na próxima etapa.

   b. Defina a tag de imagem do Cluster Autoscaler como a versão que você registrou na etapa anterior com o comando a seguir. Se necessário, substitua `X.XX.X` pelo seu próprio valor.

```
kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-
autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:vX.XX.X
```

3. (Somente clusters com nós de GPU) Se o cluster tiver grupos de nós compatíveis com GPU (por exemplo, `p3.2xlarge`), você deverá atualizar o plug-in de dispositivo NVIDIA para KubernetesDaemonSet no seu cluster. Substitua `<vX.X.X>` pela versão desejada de NVIDIA/k8s-device-plugin antes de executar o comando a seguir.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/<vX.X.X>/
deployments/static/nvidia-device-plugin.yml
```

4. Atualize o plug-in CNI da Amazon VPC para Kubernetes, o CoreDNS e os complementos `kube-proxy`. Recomendamos atualizar os complementos para as versões mínimas listadas nos tokens da conta de serviço.

   - Se estiver usando complementos do Amazon EKS, selecione Clusters no console do Amazon EKS e, em seguida, o nome do cluster que você atualizou no painel de navegação esquerdo. As notificações serão exibidas no console. Elas informam que uma nova versão está disponível para cada complemento que tenha uma atualização disponível. Para atualizar um complemento, selecione a guia Complementos. Em uma das caixas de um complemento que tenha uma atualização disponível, selecione Atualizar agora, depois, uma versão disponível e, em seguida, selecione Atualizar.
   - Como alternativa, você pode usar a AWS CLI ou o `eksctl` para atualizar os complementos. Para obter mais informações, consulte the section called "Atualização de um complemento".

5. Se necessário, atualize a versão do `kubectl`. Você deve usar uma versão do `kubectl` que esteja em uma versão secundária de diferença do ambiente de gerenciamento do cluster do Amazon EKS.

Fazer o downgrade da versão do Kubernetes de um cluster do Amazon EKS

Não é possível fazer o downgrade do Kubernetes de um cluster do Amazon EKS. Em vez disso, crie um novo cluster em uma versão anterior do Amazon EKS e migre as workloads.

Excluir um cluster

Quando terminar de usar um cluster do Amazon EKS, você deverá excluir os recursos associados a ele para não causar custos desnecessários.

Você pode excluir um cluster com `eksctl`, Console de gerenciamento da AWS ou AWS CLI.

Considerações

- Se receber um erro porque o criador do cluster foi removido, consulte este artigo para resolvê-lo.
- Os recursos do Amazon Managed Service for Prometheus estão fora do ciclo de vida do cluster e precisam ser mantidos separados do cluster. Ao excluir seu cluster, certifique-se de excluir também todos os extratores relevantes para interromper os custos aplicáveis. Para obter mais informações, consulte Encontrar e excluir extratores no Guia do usuário do Amazon Managed Service for Prometheus.
- Para remover um cluster conectado, consulte the section called “Cancelar o registro de um cluster”

Considerações para o Modo Automático do EKS

- Todos os nós do Modo Automático do EKS serão excluídos, incluindo as instâncias gerenciadas pelo EC2
- Todos os balanceadores de carga serão excluídos

Para obter mais informações, consulte the section called “Desabilitar o Modo Automático do EKS”.

Excluir cluster (eksctl)

Este procedimento exige a versão `eksctl 0.215.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

1. Liste todos os serviços em execução no cluster.

   ```
   kubectl get svc --all-namespaces
   ```

   a. Exclua todos os serviços que têm um valor `EXTERNAL-IP` associado. Esses serviços são liderados por um balanceador de carga do Elastic Load Balancing, e você deverá excluí-los no Kubernetes para permitir que o balanceador de carga e os recursos associados sejam liberados corretamente. Substitua *`service-name`* pelo nome de cada serviço listado, conforme descrito.

      ```
      kubectl delete svc service-name
      ```

2. Exclua o cluster e seus nós associados com o seguinte comando, substituindo *`prod`* pelo nome do cluster.

   ```
   eksctl delete cluster --name prod
   ```

   Resultado:

   ```
   [#]  using region region-code
   [#]  deleting EKS cluster "prod"
   [#]  will delete stack "eksctl-prod-nodegroup-standard-nodes"
   [#]  waiting for stack "eksctl-prod-nodegroup-standard-nodes" to get deleted
   [#]  will delete stack "eksctl-prod-cluster"
   [#]  the following EKS cluster resource(s) for "prod" will be deleted: cluster. If in
    doubt, check CloudFormation console
   ```

Excluir cluster (console do AWS)

1. Liste todos os serviços em execução no cluster.

```
kubectl get svc --all-namespaces
```

2. Exclua todos os serviços que têm um valor `EXTERNAL-IP` associado. Esses serviços são liderados por um balanceador de carga do Elastic Load Balancing, e você deverá excluí-los no Kubernetes para permitir que o balanceador de carga e os recursos associados sejam liberados corretamente. Substitua *`service-name`* pelo nome de cada serviço listado, conforme descrito.

```
kubectl delete svc service-name
```

3. Exclua todos os grupos de nós e perfis do Fargate.
   a. Abra o console do Amazon EKS.
   b. No painel de navegação à esquerda, escolha Clusters do Amazon EKS e, em seguida, na lista de clusters com guias, escolha o nome do cluster que você deseja excluir.
   c. Escolha a guia Compute (Computação) e escolha um grupo de nós para exclusão. Escolha Delete (Excluir), insira o nome do grupo de nós e depois escolha Delete (Excluir). Exclua todos os grupos de nós no cluster.

> **Note**
>
> Os grupos de nós listados são somente grupos de nós gerenciados .

   d. Escolha um perfil do Fargate para excluir, selecione Delete (Excluir), insira o nome do perfil e selecione Delete (Excluir). Exclua todos os perfis do Fargate no cluster.
4. Exclua todas as pilhas do nó autogerenciado do AWS CloudFormation.
   a. Abra o console do AWS CloudFormation.
   b. Escolha a pilha de nós a ser excluída e, então, escolha Delete (Excluir).
   c. Na caixa de diálogo de confirmação Delete stack (Excluir pilha), escolha Delete stack (Excluir pilha). Exclua todas as pilhas de nós autogerenciados do cluster.
5. Excluir o cluster.
   a. Abra o console do Amazon EKS.
   b. escolha o cluster a ser excluído e escolha Delete (Excluir).
   c. Na tela de confirmação de excluir cluster, selecione Delete (Excluir).

6. (Opcional) Exclua a pilha do VPC AWS CloudFormation.
   a. Abra o console do AWS CloudFormation.
   b. Selecione a pilha da VPC a ser excluída e, então, escolha Delete (Excluir).
   c. Na caixa de diálogo de confirmação Delete stack (Excluir pilha), escolha Delete stack (Excluir pilha).

Excluir cluster (AWS CLI)

1. Liste todos os serviços em execução no cluster.

```
kubectl get svc --all-namespaces
```

2. Exclua todos os serviços que têm um valor `EXTERNAL-IP` associado. Esses serviços são liderados por um balanceador de carga do Elastic Load Balancing, e você deverá excluí-los no Kubernetes para permitir que o balanceador de carga e os recursos associados sejam liberados corretamente. Substitua *`service-name`* pelo nome de cada serviço listado, conforme descrito.

```
kubectl delete svc service-name
```

3. Exclua todos os grupos de nós e perfis do Fargate.
   a. Liste os grupos de nós no cluster com o comando a seguir.

```
aws eks list-nodegroups --cluster-name my-cluster
```

> **Note**
>
> Os grupos de nós listados são somente grupos de nós gerenciados .

   b. Exclua cada grupo de nós com o comando a seguir. Exclua todos os grupos de nós no cluster.

```
aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
```

   c. Liste os perfis do Fargate no cluster com o comando a seguir.

```
aws eks list-fargate-profiles --cluster-name my-cluster
```

d. Exclua cada perfil do Fargate com o comando a seguir. Exclua todos os perfis do Fargate no cluster.

```
aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --
cluster-name my-cluster
```

4. Exclua todas as pilhas do nó autogerenciado do AWS CloudFormation.

   a. Liste suas pilhas disponíveis no AWS CloudFormation com o seguinte comando. Encontre o nome do modelo do nó no resultado.

   ```
   aws cloudformation list-stacks --query "StackSummaries[].StackName"
   ```

   b. Exclua a pilha de nós com o comando a seguir, substituindo *node-stack* pelo nome da sua pilha de nós. Exclua todas as pilhas de nós autogerenciados do cluster.

   ```
   aws cloudformation delete-stack --stack-name node-stack
   ```

5. Exclua o cluster com o comando a seguir, substituindo *my-cluster* pelo nome do cluster.

```
aws eks delete-cluster --name my-cluster
```

6. (Opcional) Exclua a pilha do VPC AWS CloudFormation.

   a. Liste suas pilhas disponíveis no AWS CloudFormation com o seguinte comando. Encontre o nome do modelo da VPC na saída resultante.

   ```
   aws cloudformation list-stacks --query "StackSummaries[].StackName"
   ```

   b. Exclua a pilha da VPC com o comando a seguir, substituindo *my-vpc-stack* pelo nome da pilha da VPC.

   ```
   aws cloudformation delete-stack --stack-name my-vpc-stack
   ```

Proteção dos clusters do EKS contra exclusão acidental

A exclusão acidental de um cluster do EKS pode prejudicar as operações do cluster do Kubernetes.

Agora é possível proteger clusters do EKS contra exclusão acidental. Se você habilitar a proteção contra exclusão em um cluster, primeiro será necessário desabilitar a proteção contra exclusão antes de poder excluir o cluster.

O objetivo da proteção contra exclusão é o de evitar acidentes. Você deve restringir cuidadosamente quem está autorizado a excluir clusters.

Se você tentar excluir um cluster ativo com a proteção de exclusão ativada, você receberá uma `InvalidRequestException`.

> **Important**
>
> Se você habilitar a proteção contra exclusão em um cluster, deverá ter ambas as permissões do IAM UpdateClusterConfig e DeleteCluster para primeiramente remover a proteção contra exclusão e, finalmente, excluir o cluster.

> **Note**
>
> Se o estado do cluster for criando, falhando ou excluindo, será possível excluir o cluster mesmo se a proteção contra exclusão estiver ativada.

Para habilitar a proteção contra exclusão para um cluster existente

Você só pode executar isso em um cluster no status ativo.

```
aws eks update-cluster-config --name <cluster-name> --region <aws-region> --deletion-protection
```

Para desabilitar a proteção contra exclusão para um cluster existente

```
aws eks update-cluster-config --name <cluster-name> --region <aws-region> --no-deletion-protection
```

Endpoint do servidor de API do cluster

Este tópico ajuda a habilitar o acesso privado ao endpoint do servidor de API do Kubernetes do cluster do Amazon EKS e limitar ou desabilitar completamente o acesso público pela internet.

Quando você cria um cluster, o Amazon EKS cria um endpoint para o servidor gerenciado de API do Kubernetes usado para se comunicar com o cluster (usando as ferramentas de gerenciamento do Kubernetes, como `kubectl`). Por padrão, esse endpoint do servidor de API é público para a internet, e o acesso ao servidor de API é protegido usando uma combinação do AWS Identity and Access Management (IAM) e do Role Based Access Control (RBAC) nativo do Kubernetes. Esse endpoint é conhecido como o endpoint de cluster público. Também há um endpoint de cluster privado. Para obter mais informações sobre o endpoint privado do cluster, consulte a seção apresentada a seguir the section called “Endpoint privado do cluster”.

Formato de endpoint de cluster `IPv6`

O EKS cria um endpoint exclusivo de pilha dupla no seguinte formato para novos clusters `IPv6` criados após outubro de 2024. Um cluster IPv6 é um cluster que você seleciona `IPv6` na configuração da família de IP (`ipFamily`) do cluster.

Example

AWS

Endpoint público/privado do cluster do EKS: `eks-cluster.`*`region`*`.api.aws`

AWS GovCloud (US)

Endpoint público/privado do cluster do EKS: `eks-cluster.`*`region`*`.api.aws`

Amazon Web Services in China

Endpoint público/privado do cluster do EKS: `eks-cluster.`*`region`*`.api.amazonwebservices.com.cn`

> **Note**
>
> O endpoint de cluster de pilha dupla foi introduzido em outubro de 2024. Para obter mais informações sobre clusters `IPv6`, consulte the section called “IPv6”. Clusters criados antes de outubro de 2024 devem usar o seguinte formato de endpoint.

Formato de endpoint de cluster `IPv4`

O EKS cria um endpoint exclusivo no formato a seguir para cada cluster que seleciona `IPv4` na configuração da família de IP (ipFamily) do cluster:

Example

AWS

Endpoint público/privado do cluster EKS `eks-cluster.region.eks.amazonaws.com`

AWS GovCloud (US)

Endpoint público/privado do cluster EKS `eks-cluster.region.eks.amazonaws.com`

Amazon Web Services in China

Endpoint público/privado do cluster EKS `eks-cluster.region.amazonwebservices.com.cn`

> **Note**
>
> Antes de outubro de 2024, os clusters `IPv6` também usavam esse formato de endpoint. Para eles, tanto o endpoint público quanto o privado têm apenas endereços `IPv4` resolvidos a partir desse endpoint.

Endpoint privado do cluster

Você pode habilitar o acesso privado ao servidor de API do Kubernetes para que todas as comunicações entre os nós e o servidor de API fiquem na VPC. Você pode limitar os endereços IP que podem acessar o servidor de API pela Internet ou desativar completamente o acesso à Internet para o servidor de API.

> **Note**
>
> Como esse endpoint é para o servidor de API do Kubernetes e não um endpoint tradicional do AWS PrivateLink para se comunicar com uma API da AWS, ele não aparece como um endpoint no console da Amazon VPC.

Ao ativar o acesso privado ao endpoint para o cluster, o Amazon EKS cria uma zona hospedada privada do Route 53 em seu nome e a associa à VPC do cluster. Essa zona hospedada privada é gerenciada pelo Amazon EKS e não aparece nos recursos do Route 53 da sua conta. Para que a zona hospedada privada roteie o tráfego adequadamente para o seu servidor da API, a VPC deve

ter `enableDnsHostnames` e `enableDnsSupport` definidos como `true`, e o conjunto de opções DHCP para sua VPC deve incluir `AmazonProvidedDNS` na lista de servidores de nome de domínio. Para obter mais informações, consulte Updating DNS support for your VPC (Atualizar o suporte de DNS para VPC) no Manual do usuário da Amazon VPC.

É possível definir os requisitos de acesso ao endpoint do servidor de API ao criar um cluster, e você pode atualizar o acesso ao endpoint do servidor de API para um cluster a qualquer momento.

Modificar o acesso ao endpoint do cluster

Use os procedimentos desta seção para modificar o acesso ao endpoint para um cluster existente. A tabela a seguir mostra as combinações compatíveis de acesso ao endpoint do servidor de API e seus comportamentos associados.

Acesso público ao endpoint	Acesso privado ao endpoint	Comportamento
Habilitado	Desabilitado	• Esse é o comportamento padrão para novos clusters do Amazon EKS. • As solicitações de API do Kubernetes que são originadas na VPC do cluster (como um nó para a comunicação do ambiente de gerenciamento) saem da VPC, mas não da rede da Amazon. • O servidor de API do cluster é acessível pela internet. Como opção, é possível usar a lista de CIDRs de acesso público para controlar o acesso ao endpoint público por meio de uma lista de blocos CIDR. Se você limitar o acesso a blocos CIDR específicos, recomendamos que também habilite o endpoint privado ou garanta que os blocos CIDR especificados incluam os endereços dos quais os nós e os pods do Fargate (caso você os utilize) usam para acessar o endpoint público.
Habilitado	Habilitado	• As solicitações de API do Kubernetes na VPC do cluster (como um nó para a comunicação do ambiente de gerenciamento) usam o endpoint da VPC privado.

Acesso público ao endpoint	Acesso privado ao endpoint	Comportamento
		Você usa as regras no grupo de segurança do cluster para controlar o acesso ao endpoint privado. • O servidor de API do cluster é acessível pela internet. Como opção, é possível usar a lista de CIDRs de acesso público para controlar o acesso ao endpoint público por meio de uma lista de blocos CIDR. • Se você estiver usando nós híbridos com o cluster do Amazon EKS, não é recomendado ter o acesso aos endpoints públicos e privados habilitados simultane amente. Como os nós híbridos estão sendo executado s fora da VPC, eles resolverão o endpoint do cluster para os endereços IP públicos. É recomendável empregar apenas um tipo de acesso (seja por endpoint público ou privado) para clusters com nós híbridos.

Acesso público ao endpoint	Acesso privado ao endpoint	Comportamento
Desabilitado	Habilitado	• Todo o tráfego para o servidor de API do cluster deve vir da VPC do cluster ou de uma rede conectada. • Não há acesso público ao servidor de API pela internet. Todos os comandos `kubectl` devem vir da VPC ou de uma rede conectada. Para ver as opções de conectividade, consulte the section called “Acessar um servidor de API somente privado”. • Você usa as regras no grupo de segurança do cluster para controlar o acesso ao endpoint privado. Lembre-se de que os blocos CIDR de acesso público não têm impacto no endpoint privado. • O endpoint do servidor de API do cluster é resolvido por servidores DNS públicos para um endereço IP privado da VPC. No passado, o endpoint só podia ser resolvido a partir da VPC. Se o endpoint não for resolvido para um endereço IP privado na VPC para um cluster existente, será possível: • Ativar o acesso público e desativá-lo novamente. Só é necessário fazer isso uma vez para um cluster e o endpoint será resolvido para um endereço IP privado desse ponto em diante. • Atualizar o cluster.

Controles de acesso ao endpoint

Lembre-se de que cada um dos métodos apresentados a seguir para controle de acesso ao endpoint afeta somente o respectivo endpoint.

Security group de cluster

O grupo de segurança do cluster controla dois tipos de conexões, as conexões com a API do kubelet e o endpoint privado. As conexões com a API do `kubelet` são usadas nos comandos `kubectl attach`, `kubectl cp`, `kubectl exec`, `kubectl logs` e `kubectl port-forward`. O grupo de segurança do cluster não tem impacto sobre o endpoint público.

Blocos CIDR de acesso público

Os CIDRs de acesso público controlam o acesso ao endpoint público por meio de uma lista de blocos CIDR. Lembre-se de que os blocos CIDR de acesso público não têm impacto no endpoint privado. Os CIDRs de acesso público comportam-se de forma diferente nos clusters `IPv6` e clusters `IPv4`, dependendo da data em que foram criados, conforme descrito a seguir:

Blocos CIDR no endpoint público (cluster **`IPv6`**)

Você pode adicionar blocos CIDR `IPv6` e `IPv4` ao endpoint público de um cluster `IPv6`, porque o endpoint público é de pilha dupla. Isso só se aplica a novos clusters com o `ipFamily` definido como `IPv6` que você criou em outubro de 2024 ou posteriormente. Você pode identificar esses clusters pelo novo nome de domínio do endpoint `api.aws`.

Blocos CIDR no endpoint público (cluster **`IPv4`**)

Você pode adicionar blocos CIDR `IPv4` ao endpoint público de um cluster `IPv4`. Você não pode adicionar blocos CIDR `IPv6` ao endpoint público de um cluster `IPv4`. Se você tentar, o EKS retornará a seguinte mensagem de erro: `The following CIDRs are invalid in publicAccessCidrs`

Blocos CIDR no endpoint público (cluster **`IPv6`** criado antes de outubro de 2024)

Você pode adicionar blocos CIDR `IPv4` ao endpoint público dos clusters `IPv6` antigos que você criou antes de outubro de 2024. Você pode identificar esses clusters pelo endpoint `eks.amazonaws.com`. Você não pode adicionar blocos CIDR `IPv6` ao endpoint público desses clusters `IPv6` antigos que você criou antes de outubro de 2024. Se você tentar, o EKS retornará a seguinte mensagem de erro: `The following CIDRs are invalid in publicAccessCidrs`

Acessar um servidor de API somente privado

Se tiver desabilitado o acesso público do endpoint do servidor de API do Kubernetes do cluster, você só poderá acessar o servidor de API pela VPC ou por uma rede conectada. Veja a seguir algumas maneiras possíveis de acessar o endpoint do servidor de API do Kubernetes:

Rede conectada

Conecte a sua rede à VPC com um gateway de trânsito da AWS ou outra opção de conectividade e depois use um computador na rede conectada. É necessário garantir que o grupo de segurança do plano de controle do Amazon EKS tenha regras para permitir o tráfego de entrada na porta 443 da rede conectada.

Bastion host do Amazon EC2

Você pode executar uma instância do Amazon EC2 em uma sub-rede pública na VPC do cluster e fazer login via SSH nessa instância para executar os comandos do `kubectl`. Para obter mais informações, consulte Bastion hosts do Linux na AWS. É necessário garantir que o grupo de segurança do plano de controle do Amazon EKS tenha regras para permitir o tráfego de entrada na porta 443 do seu bastion host. Para obter mais informações, consulte the section called “Requisitos para grupos de segurança”.

Quando você configurar `kubectl` para o bastion host, use credenciais da AWS que já estiverem mapeadas para a configuração do RBAC do cluster ou adicione a entidade principal do IAM que o bastion usará para a configuração do RBAC antes de remover o acesso público ao endpoint. Para obter mais informações, consulte the section called “Acesso à API do Kubernetes” e the section called “Acesso negado ou não autorizado (`kubectl`)”.

IDE do AWS Cloud9

AWS O Cloud9 é um ambiente de desenvolvimento integrado (IDE) baseado em nuvem que permite que você escreva, execute e depure seu código com apenas um navegador. Você pode criar um IDE do AWS Cloud9 na VPC do seu cluster e usar o IDE para se comunicar com o cluster. Para obter mais informações, consulte Criação de um ambiente em AWS Cloud9. É necessário garantir que o grupo de segurança do plano de controle do Amazon EKS contenha regras para permitir o tráfego de entrada na porta 443 do seu grupo de segurança IDE. Para obter mais informações, consulte the section called “Requisitos para grupos de segurança”.

Quando você configurar o `kubectl` para o seu AWS Cloud9 IDE, certifique-se de usar as credenciais do AWS que já estão mapeadas na configuração RBAC do cluster ou adicione a entidade principal do IAM que o IDE usará à configuração RBAC antes de remover o acesso público ao endpoint. Para obter mais informações, consulte the section called “Acesso à API do Kubernetes” e the section called “Acesso negado ou não autorizado (`kubectl`)”.

Editar esta página no GitHub

Configurar o acesso à rede para o endpoint do servidor de API do cluster

Você pode modificar o acesso ao endpoint do servidor de API do cluster usando o Console de gerenciamento da AWS ou a AWS CLI nas seções a seguir.

Configure o acesso ao endpoint - AWS console

1. Abra o console do Amazon EKS.
2. Selecione o nome do cluster para exibir as informações dele.
3. Escolha a guia Redes e depois Gerenciar acesso do endpoint.
4. Em Acesso privado: escolha se deseja habilitar ou desabilitar o acesso privado para o endpoint do servidor de API do Kubernetes do cluster. Se você habilitar o acesso privado, as solicitações de API do Kubernetes originadas na VPC do cluster usarão o endpoint da VPC privado. É necessário habilitar o acesso privado para desabilitar o acesso público.
5. Em Acesso público: escolha se deseja habilitar ou desabilitar o acesso público para o endpoint do servidor de API do Kubernetes do cluster. Se você desabilitar o acesso público, o servidor de API do Kubernetes do cluster só poderá receber solicitações da VPC do cluster.
6. (Opcional) Se você habilitou o Acesso público, poderá especificar quais endereços da Internet podem se comunicar com o endpoint público. Selecione Advanced Settings (Configurações avançadas). Insira um bloco CIDR, como *`203.0.113.5/32`*. O bloco não pode incluir endereços reservados. É possível inserir blocos adicionais selecionando Add Source (Adicionar origem). Há um número máximo de blocos CIDR que você pode especificar. Para obter mais informações, consulte the section called “Cotas de serviço”. Se você não especificar nenhum bloco, então o endpoint do servidor de API público receberá solicitações de todos os endereços IP tanto para `IPv4` (`0.0.0.0/0`) como para `IPv6` (`::/0`) para o cluster `IPv6` de pilha dupla. Se você restringir o acesso ao endpoint público usando blocos CIDR, recomendamos habilitar também o acesso ao endpoint privado para que os nós e os Pods do Fargate (se você usá-los) possam se comunicar com o cluster. Sem o endpoint privado habilitado, suas origens CIDR de endpoint de acesso público devem incluir as origens de saída de sua VPC. Por exemplo, se você tiver um nó em uma sub-rede privada que se comunica com a Internet por meio de um gateway NAT, será necessário adicionar o endereço IP de saída do gateway NAT como parte de um bloco CIDR na lista de permissões no endpoint público.
7. Selecione Update (Atualizar) para concluir.

Configurar o acesso ao endpoint - AWS CLI

Conclua as etapas a seguir usando a versão AWS CLI `1.27.160` ou posterior. É possível verificar sua versão atual com `aws --version`. Para instalar ou atualizar a AWS CLI, consulte Instalar a AWS CLI.

1. Atualize o acesso ao endpoint do servidor de API do cluster com o seguinte comando da CLI AWS. Substitua o nome do cluster e os valores desejados de acesso ao endpoint. Se você definir `endpointPublicAccess=true`, também poderá inserir um único bloco CIDR ou uma lista de blocos CIDR separados por vírgulas para `publicAccessCidrs`. Os blocos não podem incluir endereços reservados. Se você especificar blocos CIDR, o endpoint do servidor de API público só receberá solicitações dos blocos listados. Há um número máximo de blocos CIDR que você pode especificar. Para obter mais informações, consulte the section called “Cotas de serviço”. Se você restringir o acesso ao endpoint público usando blocos CIDR, é recomendável habilitar também o acesso ao endpoint privado para que os nós e os Pods do Fargate (se você usá-los) possam se comunicar com o cluster. Sem o endpoint privado habilitado, suas origens CIDR de endpoint de acesso público devem incluir as origens de saída de sua VPC. Por exemplo, se você tiver um nó em uma sub-rede privada que se comunica com a Internet por meio de um gateway NAT, será necessário adicionar o endereço IP de saída do gateway NAT como parte de um bloco CIDR na lista de permissões no endpoint público. Se você não especificar nenhum bloco CIDR, então o endpoint do servidor de API público receberá solicitações de todos os endereços IP (0.0.0.0/0) e do `IPv6` (`::/0`) para o cluster `IPv6` de pilha dupla.

   > **Note**
   >
   > O comando a seguir permite acesso privado e acesso público a partir de um único endereço IP para o endpoint do servidor de API. Substitua *`203.0.113.5/32`* por um único bloco CIDR ou uma lista de blocos CIDR separados por vírgulas aos quais você deseja restringir o acesso à rede.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --resources-vpc-config
 endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

   Veja abaixo um exemplo de saída.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\"203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

2. Monitore o status da atualização do acesso ao endpoint com o comando a seguir, usando o nome do cluster e o ID da atualização retornado pelo comando anterior. Sua atualização estará concluída quando o status for exibido como `Successful`.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000
```

Veja abaixo um exemplo de saída.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "Successful",
        "type": "EndpointAccessUpdate",
        "params": [
```

```
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\"203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

Editar esta página no GitHub

Implantar nós Windows em clusters EKS

Saiba como habilitar e gerenciar o suporte do Windows para o cluster do Amazon EKS para executar contêineres do Windows junto com contêineres do Linux.

Considerações

Antes de implantar nós do Windows, esteja ciente das seguintes considerações:

- O Modo Automático do EKS não é compatível com nós do Windows
- É possível usar rede de hosts nos nós do Windows usando `HostProcess` Pods. Para obter mais informações, consulte Create a Windows HostProcessPod na documentação do Kubernetes.
- Os clusters do Amazon EKS devem conter um ou mais nós do Linux ou do Fargate para executar pods do sistema de núcleo que só são executados no Linux, como o CoreDNS.
- Os logs de eventos do `kubelet` e `kube-proxy` são redirecionados para o log de eventos do `EKS Windows` e são definidos com um limite de 200 MB.
- Você não pode usar Atribuir grupos de segurança a pods individuais com pods em execução nos nós do Windows.
- Você não pode usar redes personalizadas com nós do Windows.

- Você não pode usar `IPv6` com nós do Windows.
- Os nós o Windows oferecem suporte para uma interface de rede elástica por nó. Por padrão, o número de pods que podem ser executados por nó do Windows é igual ao número de endereços IP disponíveis por interface de rede elástica para o tipo de instância do nó, menos um. Para obter mais informações, consulte Endereços IP por interface de rede por tipo de instância no Guia do usuário do Amazon EC2.
- Em um cluster do Amazon EKS, um único serviço com um balanceador de carga pode ser compatível com até 1.024 pods de backend. Cada pod tem seu próprio endereço IP exclusivo. O limite anterior de 64 pods não é mais o caso, depois de uma atualização do Windows Server a partir da compilação do sistema operacional 17763.2746.
- Contêineres do Windows não são compatíveis com os pods do Amazon EKS no Fargate.
- Você não pode usar os Amazon EKS Hybrid Nodes com o Windows como sistema operacional para o host.
- Não é possível recuperar logs do pod `vpc-resource-controller`. Antes, era possível implantar o controlador no plano de dados.
- Há um período de resfriamento antes de um endereço `IPv4` ser atribuído a um novo pod. Isso evita que o tráfego flua para um pod antigo com o mesmo endereço `IPv4` por causa de regras `kube-proxy` obsoletas.
- A fonte do controlador é gerenciada no GitHub. Para contribuir ou arquivar problemas sobre o controlador, acesse o projeto no GitHub.
- Ao especificar um ID de AMI personalizado para grupos de nós gerenciados do Windows, adicione `eks:kube-proxy-windows` ao mapa de configuração do AWS IAM Authenticator. Para obter mais informações, consulte the section called “Limites e condições ao especificar uma ID de AMI”.
- Se preservar os endereços IPv4 disponíveis for crucial para sua sub-rede, consulte IP Address Management em Windows Networking no Guia de práticas recomendadas do EKS para obter orientação.
- Considerações sobre as entradas de acesso do EKS
  - As entradas de acesso para uso com os nós do Windows precisam do tipo `EC2_WINDOWS`. Para obter mais informações, consulte the section called “Criar entradas de acesso”.

    Para criar uma entrada de acesso para um nó do Windows:

    ```
    aws eks create-access-entry --cluster-name my-cluster --principal-arn
     arn:aws:iam::111122223333:role/<role-name> --type EC2_Windows
    ```

Pré-requisitos

- Um cluster existente.
- Seu cluster deve ter pelo menos um (recomendamos pelo menos dois) nó do Linux ou pod do Fargate para executar o CoreDNS. Se você habilitar o suporte legado do Windows, deverá usar um nó do Linux (não é possível usar um pod do Fargate) para executar o CoreDNS.
- Perfil do IAM existente do cluster do Amazon EKS.

Habilitar o suporte ao Windows

1. Se não houver nós do Amazon Linux no cluster e você usar grupos de segurança para pods, vá para a próxima etapa. Caso contrário, confirme se a política gerenciada `AmazonEKSVPCResourceController` está anexada à função do cluster. Substitua *`eksClusterRole`* com o nome da função do cluster.

   ```
   aws iam list-attached-role-policies --role-name eksClusterRole
   ```

   Veja abaixo um exemplo de saída.

   ```
   {
       "AttachedPolicies": [
           {
               "PolicyName": "AmazonEKSClusterPolicy",
               "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
           },
           {
               "PolicyName": "AmazonEKSVPCResourceController",
               "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"
           }
       ]
   }
   ```

   Se a política estiver anexada, como está na saída anterior, pule a próxima etapa.

2. Anexe a política gerenciada AmazonEKSVPCResourceController ao seu perfil do IAM de cluster do Amazon EKS. Substitua *`eksClusterRole`* com o nome da função do cluster.

   ```
   aws iam attach-role-policy \
     --role-name eksClusterRole \
   ```

```
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
```

3. Atualize o ConfigMap da CNI da VPC para habilitar o IPAM do Windows. Observe que, se a CNI da VPC estiver instalada no seu cluster usando um chart do Helm ou como um complemento do Amazon EKS, você talvez não consiga modificar diretamente o ConfigMap. Para obter informações sobre como configurar os complementos do Amazon EKS, consulte the section called "Campos que você pode personalizar".

   a. Crie um arquivo chamado *vpc-resource-controller-configmap.yaml* com o conteúdo a seguir.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: amazon-vpc-cni
     namespace: kube-system
   data:
     enable-windows-ipam: "true"
   ```

   b. Aplicar o `ConfigMap` ao seu cluster.

   ```
   kubectl apply -f vpc-resource-controller-configmap.yaml
   ```

4. Se o cluster tiver o modo de autenticação definido para habilitar o configmap `aws-auth`:

   - Verifique se seu `ConfigMap` do `aws-auth` contém um mapeamento para o perfil de instância do nó do Windows para incluir o grupo de permissões `eks:kube-proxy-windows` do RBAC. O comando a seguir pode ser executado para verificar.

   ```
   kubectl get configmap aws-auth -n kube-system -o yaml
   ```

   Veja abaixo um exemplo de saída.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: aws-auth
     namespace: kube-system
   data:
     mapRoles: |
       - groups:
         - system:bootstrappers
   ```

```
      - system:nodes
      - eks:kube-proxy-windows # This group is required for Windows DNS resolution
 to work
      rolearn: arn:aws:iam::111122223333:role/eksNodeRole
      username: system:node:{{EC2PrivateDNSName}}
[...]
```

Você deverá ver `eks:kube-proxy-windows` listado em grupos. Se o grupo não for especificado, será necessário atualizar seu `ConfigMap` ou criá-lo para incluir o grupo necessário. Para obter mais informações sobre o `ConfigMap` do `aws-auth`, consulte the section called “Como aplicar o `ConfigMap` `aws-auth` ao seu cluster”.

5. Se o cluster tiver o modo de autenticação definido para desabilitar o configmap `aws-auth`, você poderá usar as entradas de acesso do EKS. Crie um perfil de nó para uso com instâncias do Windows, e o EKS criará automaticamente uma entrada de acesso do tipo `EC2_WINDOWS`.

Implantar pods do Windows

Para implantar pods no cluster, será necessário especificar o sistema operacional que eles usam, se você estiver executando uma mistura de tipos de nós.

Para pods do Linux, use o texto do seletor de nó a seguir em seus manifestos.

```
nodeSelector:
        kubernetes.io/os: linux
        kubernetes.io/arch: amd64
```

Para pods do Windows, use o texto do seletor de nó a seguir em seus manifestos.

```
nodeSelector:
        kubernetes.io/os: windows
        kubernetes.io/arch: amd64
```

Você pode implantar uma aplicação de exemplo para ver os seletores de nós em uso.

Compatibilidade com uma maior densidade de pods em nós do Windows

No Amazon EKS, cada pod recebe um endereço `IPv4` da VPC. Como resultado, o número de pods que podem ser implantados em um nó é limitado pelos endereços IP disponíveis, mesmo que haja

recursos suficientes para executar mais pods no nó. Como somente uma interface de rede elástica é aceita por cada nó do Windows, por padrão, o número máximo de endereços IP disponíveis em um nó do Windows é igual a:

```
Number of private IPv4 addresses for each interface on the node - 1
```

Um endereço IP é usado como endereço IP principal da interface de rede, e portanto, não pode ser alocado a pods.

É possível permitir uma maior densidade de pods em nós do Windows ao habilitar a delegação de prefixo de IP. Esse recurso permite atribuir um prefixo `IPv4 /28` à interface de rede primária, em vez de atribuir endereços `IPv4` secundários. A atribuição de um prefixo IP aumenta o número máximo de endereços `IPv4` disponíveis no nó para:

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) *
 16
```

Com esse número significativamente maior de endereços IP disponíveis, eles não limitarão sua capacidade de escalar o número de pods em seus nós. Para obter mais informações, consulte the section called “Aumento dos endereços IP”.

Desativar o suporte ao Windows

1. Se o cluster contiver nós do Amazon Linux e você usar grupos de segurança para pods com eles, ignore esta etapa.

   Remova a política do IAM `AmazonVPCResourceController` gerenciada da função do cluster. Substitua *`eksClusterRole`* pelo nome do perfil do cluster.

   ```
   aws iam detach-role-policy \
       --role-name eksClusterRole \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
   ```

2. Desabilite o Windows IPAM no ConfigMap `amazon-vpc-cni`.

   ```
   kubectl patch configmap/amazon-vpc-cni \
                       -n kube-system \
                       --type merge \
   ```

```
    -p '{"data":{"enable-windows-ipam":"false"}}'
```

Implementar clusters privados com acesso limitado à internet

Este tópico descreve como implantar um cluster do Amazon EKS que está implantado na nuvem AWS, mas não tem acesso à Internet de saída. Se você tiver um cluster local em AWS Outposts, consulte the section called “Nós”, em vez deste tópico.

Se você não estiver familiarizado com a rede do Amazon EKS, consulte De-mystifying cluster networking for Amazon EKS worker nodes. Se o seu cluster não tiver acesso de saída à Internet, ele deverá atender aos seguintes requisitos:

Requisitos de arquitetura do cluster

- O cluster deve extrair imagens de um registro de contêiner que esteja na VPC. Você pode criar um Amazon Elastic Container Registry na VPC e copiar para ele as imagens de contêiner para que os nós as extrai dali. Para obter mais informações, consulte the section called “Copiar uma imagem para um repositório”.
- O cluster deve ter acesso privado ao endpoint habilitado. Isso é necessário para que os nós sejam registrados no endpoint do cluster. O acesso público ao endpoint é opcional. Para obter mais informações, consulte the section called “Acesso ao endpoint do cluster”.

Requisitos do nó

- Os nós autogerenciados do Windows e do Linux devem incluir os seguintes argumentos de bootstrap antes de serem executados. Esses argumentos ignoram a introspecção do Amazon EKS e não exigem acesso à API do Amazon EKS de dento da VPC.

  a. Determine o valor do endpoint do cluster com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

  ```
  aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
  ```

  Veja abaixo um exemplo de saída.

  ```
  https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
  ```

b. Determine o valor da autoridade de certificação do cluster com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --
output text
```

A saída retornada é uma string longa.

c. Substitua os valores de `apiServerEndpoint` e `certificateAuthority` no objeto NodeConfig pelos valores retornados na saída dos comandos anteriores. Para obter mais informações sobre como especificar argumentos de inicialização ao executar nós gerenciados pelo cliente do Amazon Linux 2023, consulte the section called “Amazon Linux” e the section called “Windows”.

- Para nós do Linux:

```
---
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: [.replaceable]https://
EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
    certificateAuthority: [.replaceable]Y2VydGlmaWNhdGVBdXRob3JpdHk=
    ...
```

Para obter argumentos adicionais, consulte o script de bootstrap no GitHub.

- Para nós do Windows:

> **Note**
>
> Se você estiver usando um CIDR de serviço personalizado, será necessário especificá-lo usando o parâmetro `-ServiceCIDR`. Do contrário, a resolução de DNS para pods no cluster falhará.

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

Para obter argumentos adicionais, consulte the section called “Parâmetros de configuração do script de bootstrap”.

- O `ConfigMap` da `aws-auth` do cluster deve ser criado de dentro da VPC. Para obter mais informações sobre como criar e adicionar entradas ao `ConfigMap` `aws-auth`, insira `eksctl create iamidentitymapping --help` em seu terminal. Se o `ConfigMap` não existir no seu servidor, `eksctl` será criado quando você usar o comando para adicionar um mapeamento de identidade.

Requisitos do Pod

- Identidade de Pods: os pods configurados com a Identidade de Pods do EKS adquirem credenciais da API Auth do EKS. Se não houver acesso de saída à Internet, será necessário criar e usar um endpoint da VPC para a API Auth do EKS: `com.amazonaws.region-code.eks-auth`. Para obter mais informações sobre endpoints da VPC do EKS e o EKS Auth, consulte the section called “ AWS PrivateLink”.
- IRSA: pods configurados com perfis do IAM para contas de serviço adquirem credenciais de uma chamada de API do AWS Security Token Service (AWS STS). Se não houver acesso de saída à Internet, você deverá criar e usar um endpoint AWS STS VPC em sua VPC. A maioria dos SDKs do AWS `v1` usa o endpoint global AWS STS por padrão (`sts.amazonaws.com`), que não usa o endpoint AWS STS VPC. Para usar o endpoint AWS STS VPC, talvez seja necessário configurar o SDK para usar o endpoint regional AWS STS (`sts.`*`region-code`*`.amazonaws.com`). Para obter mais informações, consulte the section called “Endpoints do STS”.
- As sub-redes da VPC do cluster devem ter um endpoint de interface da VPC para todos os serviços da AWS que os pods precisam acessar. Para obter mais informações, consulte Acessar um serviço da AWS usando um endpoint da VPC de interface. Alguns serviços e endpoints

comumente usados estão listados na tabela a seguir. Para obter uma lista completa de endpoints, consulte AWS services that integrate with AWS PrivateLink no AWS PrivateLink Guide.

Recomendamos que você habilite nomes de DNS privados para seus endpoints da VPC, para que as workloads possam continuar usando endpoints de serviço da AWS públicos sem problemas.

Serviço	Endpoint
Amazon EC2	com.amazonaws.*region-code* .ec2
Amazon Elastic Container Registry (para extração de imagens de contêineres)	com.amazonaws.*region-code* .ecr.api, com.amazonaws.*region-code* .ecr.dkr e com.amazonaws.*region-code* .s3
Application Load Balancers e Network Load Balancers da Amazon	com.amazonaws.*region-code* .elasticl oadbalancing
(Opcional) AWS X-Ray (necessário para o rastreamento enviado ao AWS X-Ray)	com.amazonaws.*cregion-code* .xray
(Opcional) Amazon SSM (obrigatório para o atendente SSM para tarefas de gerenciam ento de nós. Alternativa para SSH)	com.amazonaws.*region-code* .ssm
Amazon CloudWatch Logs (necessário para logs de nós e pods enviados para o Amazon CloudWatch Logs)	com.amazonaws.*region-code* .logs
Serviço de token de segurança da AWS (necessário ao usar perfis de IAM para contas de serviço)	com.amazonaws.*region-code* .sts
Amazon EKS Auth (obrigatório ao usar associações de Identidade de Pods)	com.amazonaws.*region-code* .eks-auth
Amazon EKS	com.amazonaws.*region-code* .eks

- Todos os nós autogerenciados devem ser implantados em sub-redes que tenham os endpoints de interface da VPC necessários. Se você criar um grupo de nós gerenciados, o grupo de segurança de endpoint de interface da VPC deverá aceitar o CIDR para as sub-redes ou será necessário

adicionar o grupo de segurança do nó criado ao grupo de segurança do endpoint de interface da VPC.

- Armazenamento do EFS: se os seus pods usarem volumes do Amazon EFS, antes de implantar a ação indicada em Armazene um sistema de arquivos elástico com o Amazon EFS, o arquivo kustomization.yaml do driver deverá ser alterado para definir as imagens do contêiner para usar a mesma região da AWS que o cluster do Amazon EKS.
- O Route 53 não é compatível com o AWS PrivateLink. Você não pode gerenciar registros DNS do Route 53 em um cluster privado do Amazon EKS. Isso afeta o external-dns do Kubernetes.
- Se você usa a AMI otimizada para EKS, deve habilitar o endpoint do `ec2` na tabela acima. Como alternativa, você pode definir manualmente o nome DNS do nó. A AMI otimizada usa APIs do EC2 para definir o nome DNS do nó automaticamente.
- Você pode usar o AWS Load Balancer Control ler para implantar AWS Application Load Balancers (ALB) e Network Load Balancers em seu cluster privado. Ao implantá-lo, você deve usar sinalizadores de linha de comando para definir `enable-shield`, `enable-waf` e `enable-wafv2` como falsos. Detecção de certificados com nomes de host de objetos de ingresso não é compatível. Isso ocorre porque o controlador precisa acessar o AWS Certificate Manager, que não tem um endpoint de interface VPC.

  O controlador é compatível com balanceadores de carga de rede com destinos IP, que são necessários para uso com Fargate. Para obter mais informações, consulte the section called “Balanceador de carga da aplicação” e the section called “Criar um balanceador de carga da rede”.
- Há suporte para o Cluster Autoscaler. Ao implantar pods do Cluster Autoscaler, certifique-se de que a linha de comandos inclua `--aws-use-static-instance-list=true`. Para obter mais informações, consulte Usar lista de instâncias estáticas no GitHub. A VPC do nó de processamento também deve incluir o endpoint da VPC AWS STS e o endpoint VPC de escalabilidade automática.
- Alguns produtos de software de contêiner utilizam chamadas de API que acessam o serviço AWS Marketplace para monitorar a utilização. Como os clusters privados não permitem essas chamadas, esses tipos de contêiner não podem ser usados nesses clusters.

Escale a computação em cluster com o Karpenter e o Cluster Autoscaler

A escalação automática é uma função que aumenta ou reduz automaticamente a escala dos recursos para atender a alterações de demanda. É uma função importante do Kubernetes cuja execução manual exigiria extensos recursos humanos.

Modo Automático do EKS

O Modo Automático do Amazon EKS escala automaticamente os recursos computacionais do cluster. Se um pod não se adaptar aos nós existentes, o Modo Automático do EKS criará um novo. O Modo Automático do EKS também consolida workloads e exclui nós. O Modo Automático do EKS é baseado no Karpenter.

Para obter mais informações, consulte:

- Modo Automático do EKS
- the section called “Criar grupo de nós”
- the section called “Implantar workload de ampliação”

Soluções adicionais

O Amazon EKS é compatível com dois produtos de ajuste de escala automático:

Karpenter

O Karpenter é um autoscaler de clusters do Kubernetes flexível e de alta performance que ajuda a melhorar a disponibilidade das aplicações e a eficiência do cluster. O Karpenter inicia recursos computacionais do tamanho certo (por exemplo, instâncias do Amazon EC2) em resposta à alteração da carga da aplicação em menos de um minuto. Com a integração do Kubernetes à AWS, o Karpenter pode provisionar recursos computacionais just-in-time que atendam aos requisitos de sua workload com precisão. O Karpenter provisiona automaticamente novos recursos de computação com base nos requisitos específicos das workloads de cluster. Isso inclui requisitos de computação, armazenamento, aceleração e agendamento. O Amazon EKS oferece suporte a clusters que usam o Karpenter, mas o Karpenter funciona com qualquer cluster do Kubernetes compatível. Para obter mais informações, consulte a documentação do Karpenter.

> **Important**
>
> O Karpenter é um software open-source que os clientes da AWS são responsáveis por instalar, configurar e gerenciar em clusters do Kubernetes. A AWS fornece suporte técnico quando o Karpenter é executado sem modificações usando uma versão compatível em clusters do Amazon EKS. É essencial que os clientes mantenham a disponibilidade e a segurança do controlador Karpenter, bem como os procedimentos de teste adequados ao atualizá-lo ou ao atualizar o cluster do Kubernetes no qual ele está sendo executado, assim como qualquer outro software gerenciado pelo cliente. Não há um acordo de serviço (SLA) da AWS para o Karpenter, e os clientes são responsáveis por garantir que as instâncias do EC2 iniciadas pelo Karpenter atendam aos seus requisitos comerciais.

Autoscaler do cluster

O Cluster Autoscaler do Kubernetes ajusta automaticamente o número de nós no cluster quando os pods falharem ou forem reagendados em outros nós. O Cluster Autoscaler usa grupos do Auto Scaling. Para obter mais informações, consulte Autoscaler do cluster na AWS.

Saiba mais sobre a mudança de zona do Controlador de Recuperação de Aplicações da Amazon (ARC, na sigla em inglês) no Amazon EKS

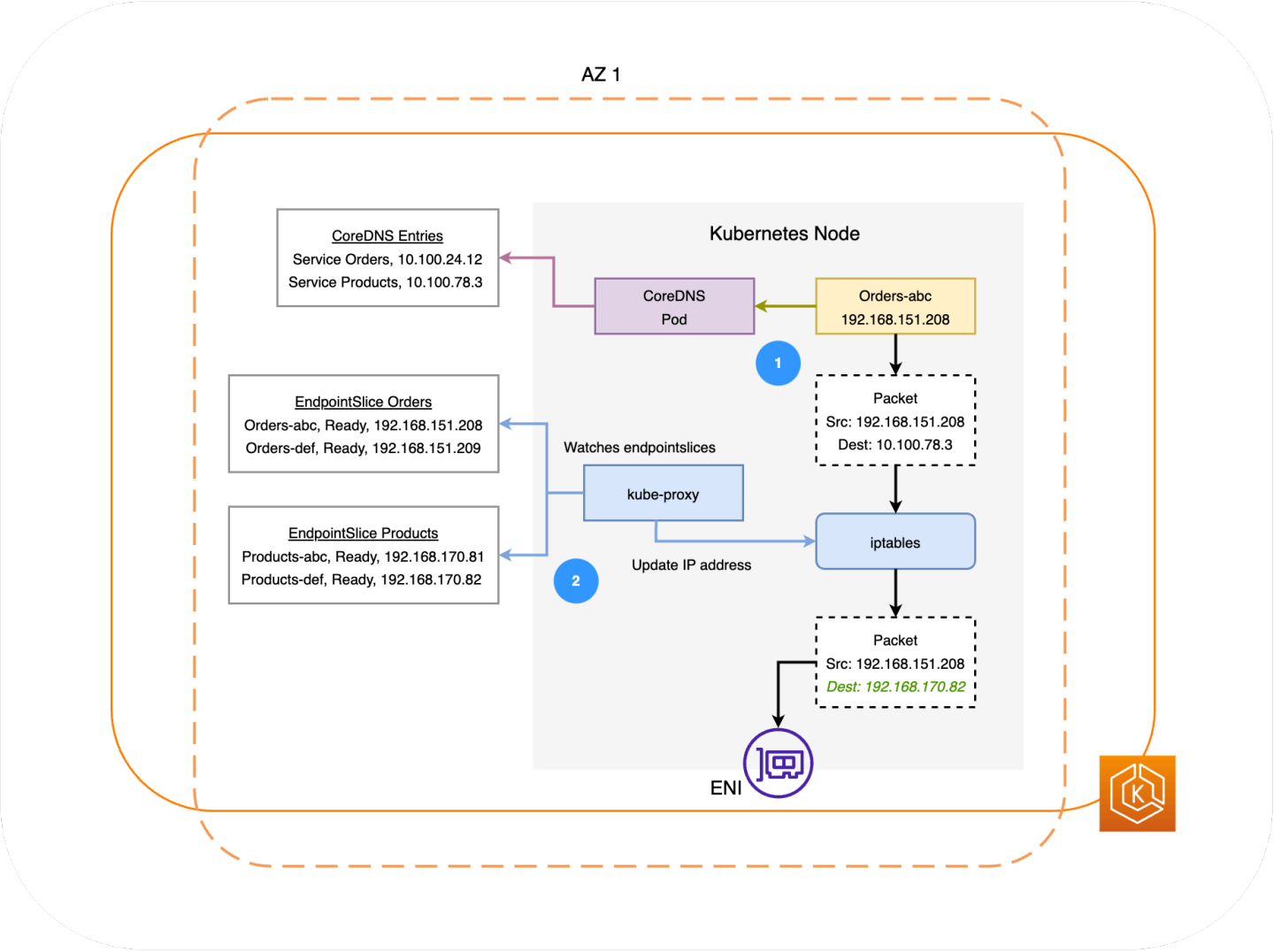
O Kubernetes tem recursos nativos que permitem que você torne suas aplicações mais resilientes a eventos como a degradação da integridade ou a deficiência de uma zona de disponibilidade (AZ). Ao executar as workloads em um cluster do Amazon EKS, é possível aumentar ainda mais a tolerância a falhas do ambiente e a recuperação da aplicação usando a mudança de zona do Controlador de Recuperação de Aplicações da Amazon (ARC, na sigla em inglês) ou a mudança de zona automática. A mudança de zona do ARC é uma medida temporária que possibilita desviar o tráfego de um recurso de uma AZ comprometida até a expiração ou o cancelamento da mudança de zona. É possível prolongar a mudança de zona, caso seja necessário.

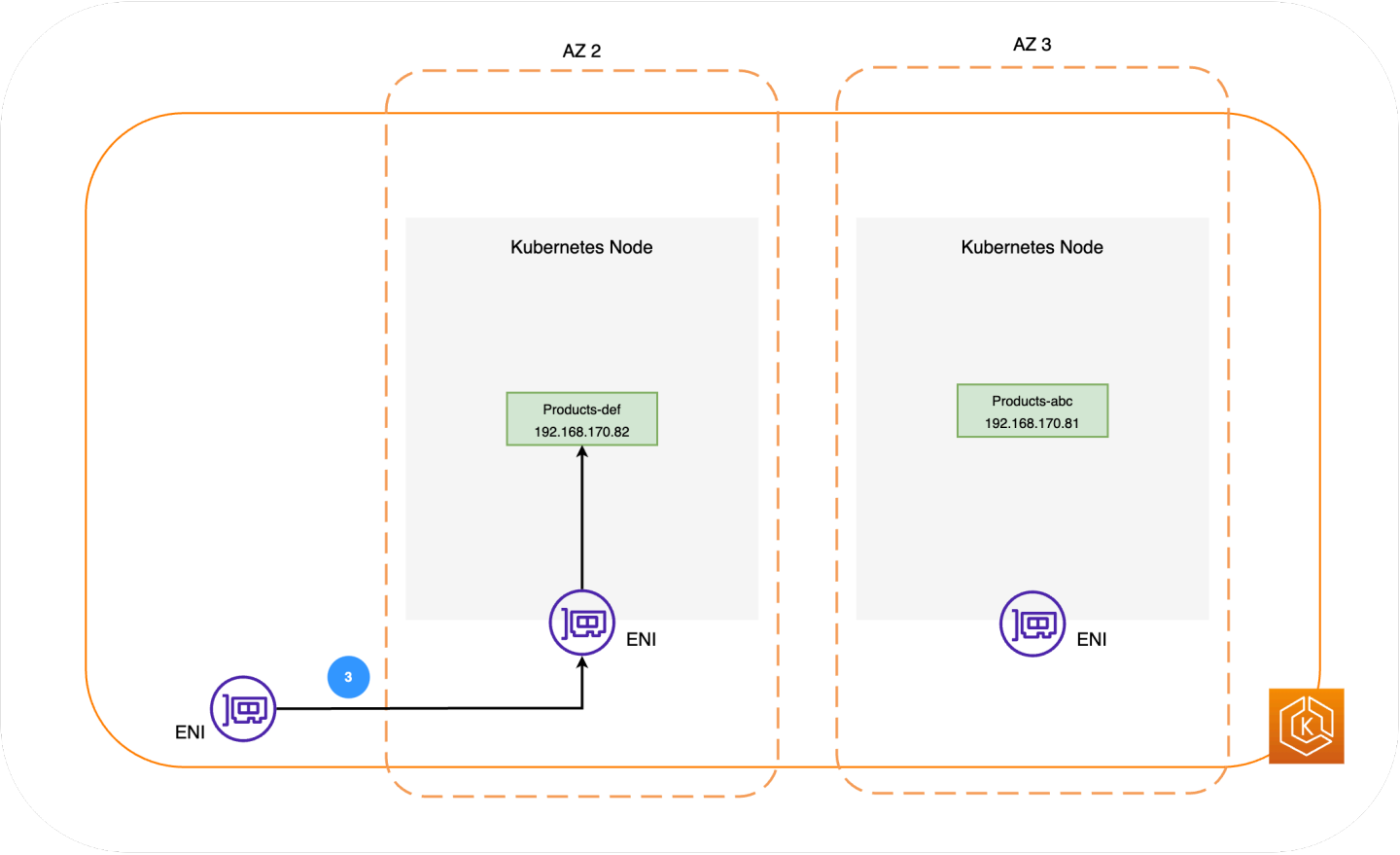
Você pode iniciar uma mudança de zona em um cluster do EKS ou permitir que a AWS realize o deslocamento do tráfego por você ao habilitar a mudança de zona automática. Essa mudança atualiza o fluxo do tráfego de rede leste-oeste em seu cluster para considerar apenas os endpoints

de rede para Pods em execução em nós de processamento em AZs íntegras. Além disso, qualquer ALB ou NLB que lide com o tráfego de entrada para aplicações no seu cluster do EKS encaminhará automaticamente o tráfego para destinos nas AZs íntegras. Para os clientes que buscam as metas de disponibilidade mais altas, no caso de uma AZ ser prejudicada, pode ser importante poder desviar todo o tráfego da AZ afetada até que ela se recupere. Para isso, você também pode habilitar um ALB ou NLB com mudança de zona do ARC.

Noções básicas sobre o fluxo de tráfego de rede leste-oeste entre pods

O diagrama a seguir ilustra dois exemplos de workloads: Pedidos e Produtos. O objetivo desse exemplo é mostrar como as workloads e os pods em diferentes AZs se comunicam.





1. Para que o serviço Orders estabeleça comunicação com o Products, ele precisa primeiro resolver o nome de DNS do serviço de destino. O Orders se comunica com o CoreDNS para buscar o endereço IP virtual (IP do cluster) desse serviço. Após o Orders resolver o nome do serviço Products, ele envia o tráfego para esse endereço IP de destino.
2. O kube-proxy é executado em cada nó do cluster e monitora continuamente os EndpointSlices para os serviços. Quando um serviço é criado, um EndpointSlice é criado e gerenciado em segundo plano pelo controlador de EndpointSlice. Cada EndpointSlice conta com uma lista ou com uma tabela de endpoints que contém um subconjunto de endereços de pods, juntamente com os nós em que eles estão sendo executados. O kube-proxy configura regras de roteamento para cada um desses endpoints de Pod usando `iptables` nos nós. O kube-proxy também é responsável por uma forma básica de balanceamento de carga, redirecionando o tráfego destinado ao endereço IP do cluster de um serviço para que seja enviado diretamente ao endereço IP de um pod. O kube-proxy realiza esse processo ao reescrever o endereço IP de destino na conexão de saída.
3. Em seguida, os pacotes de rede são enviados ao pod do Products na AZ 2 por meio das ENIs nos respectivos nós, como ilustrado no diagrama anterior.

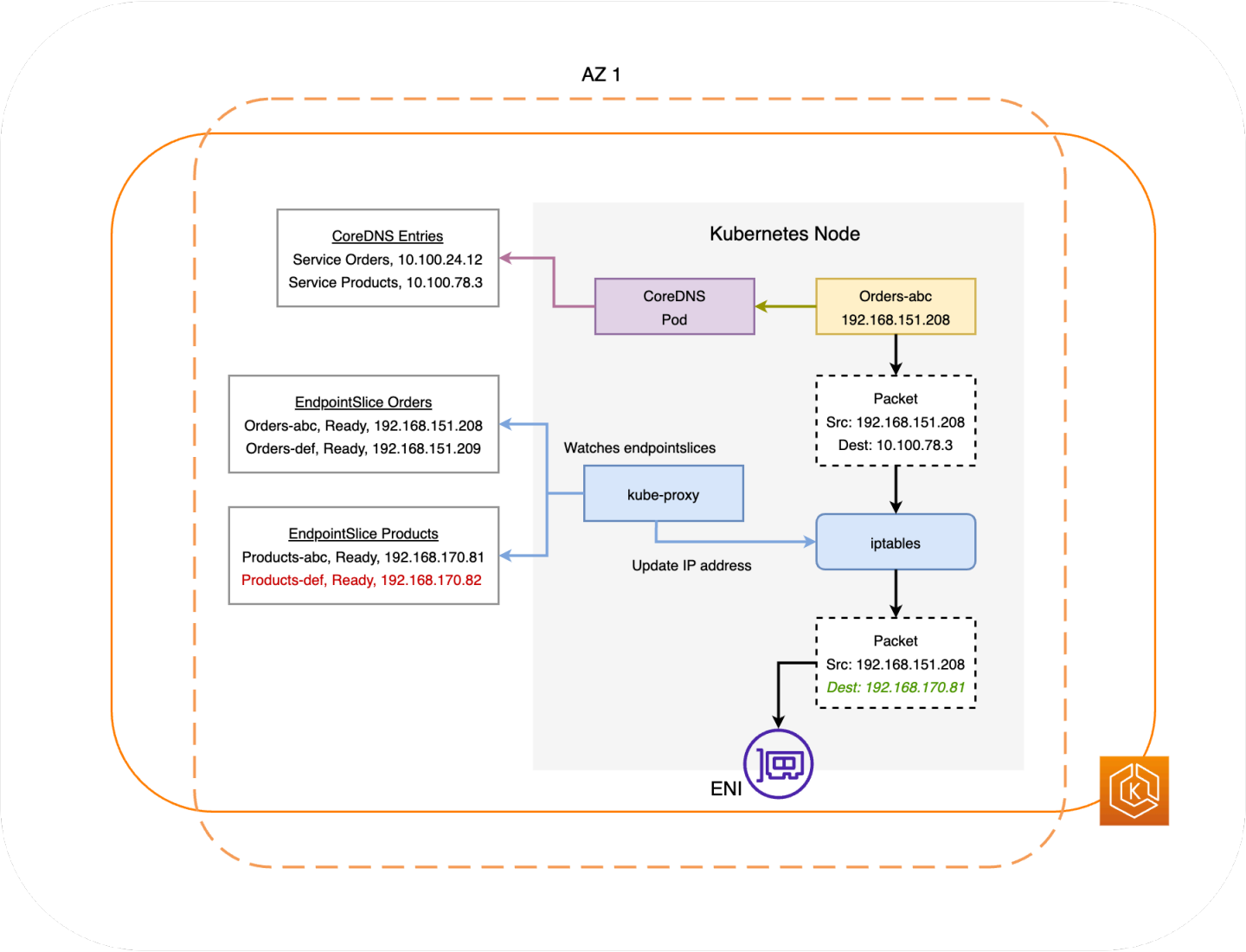
Noções básicas sobre a mudança de zona do ARC no Amazon EKS

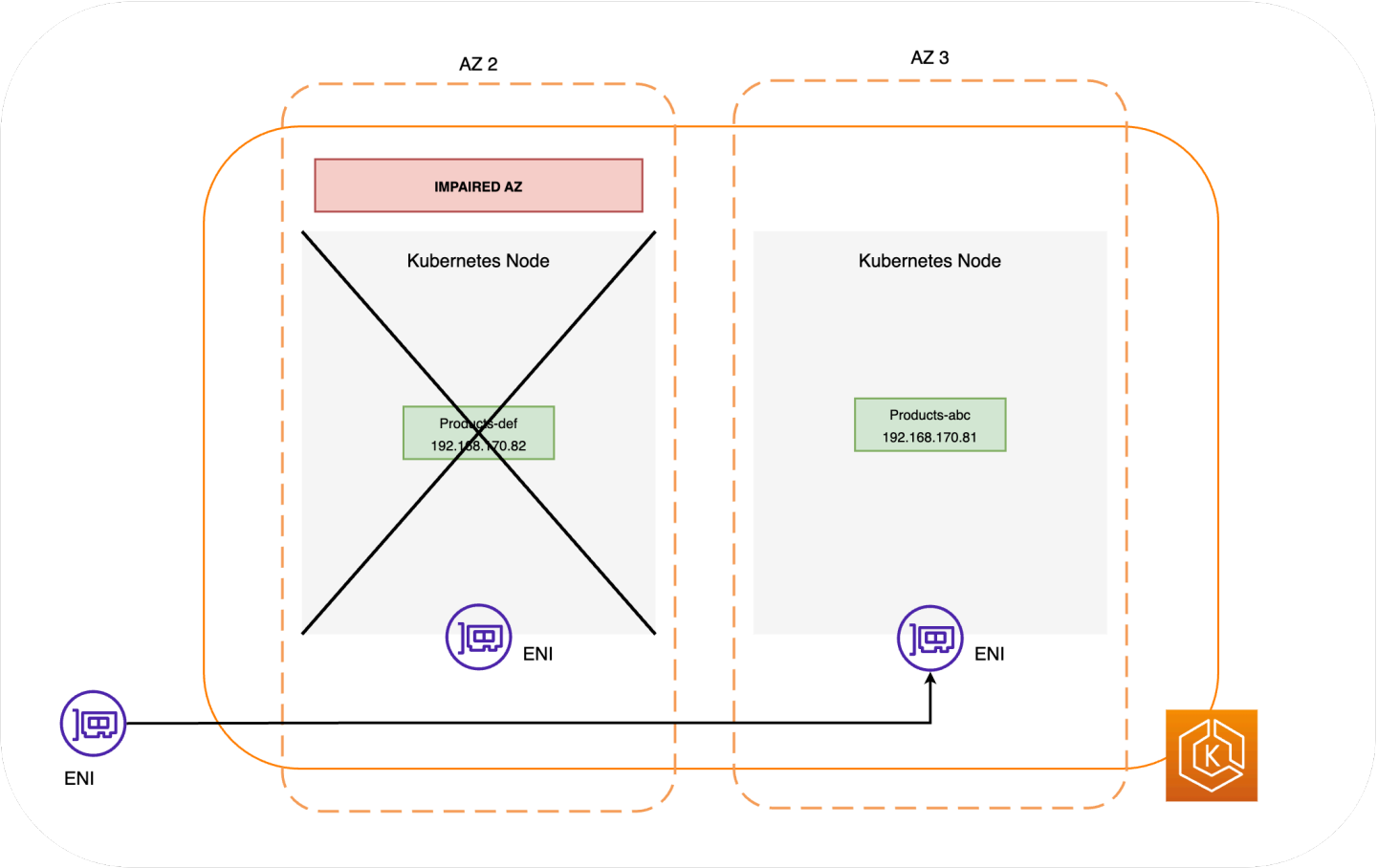
Se houver um comprometimento de uma AZ em seu ambiente, você poderá iniciar uma mudança de zona para o ambiente do cluster do EKS. Como alternativa, é possível permitir que a AWS gerencie o deslocamento do tráfego para você com a mudança de zona automática. Com a mudança de zona automática, a AWS monitora a integridade geral das AZs e responde a um possível comprometimento de AZ mudando automaticamente o tráfego para fora da AZ comprometida no ambiente de cluster.

Após habilitar a mudança de zona com o ARC em seu cluster do Amazon EKS, você pode iniciar uma mudança de zona ou habilitar a mudança de zona automática pelo console do ARC, pela AWS CLI ou pelas APIs de mudança de zona e de mudança de zona automática. Durante uma mudança de zona do EKS, as seguintes ações são realizadas automaticamente:

- Todos os nós da AZ comprometida serão isolados. Isso impede que o Agendador do Kubernetes agende novos pods em nós na AZ não íntegra.
- Se você estiver usando grupos de nós gerenciados, o rebalanceamento de zonas de disponibilidade será suspenso e o grupo do Auto Scaling será atualizado para garantir que novos nós do plano de dados do EKS sejam iniciados apenas em AZs íntegras.
- Os nós na AZ não íntegra não serão encerrados, e os pods não são removidos desses nós. Isso garante que, quando uma mudança de zona expirar ou for cancelada, o tráfego possa ser retornado com segurança à AZ para obter capacidade total.
- O controlador de EndpointSlice identifica todos os endpoints de Pods na AZ comprometida e os remove dos EndpointSlices correspondentes. Isso garantirá que somente os endpoints do pod em AZs íntegras sejam direcionados para receber tráfego de rede. Quando uma mudança de zona for cancelada ou expirar, o controlador de EndpointSlice atualizará os EndpointSlices para incluir os endpoints da AZ restaurada.

Os diagramas a seguir fornecem uma visão geral de alto nível sobre como a mudança de zona do EKS garante que apenas endpoints de pod íntegros sejam selecionados em seu ambiente de cluster.


AZ 1
CoreDNS Entries
Service Orders, 10.100.24.12
Service Products, 10.100.78.3
Kubernetes Node
CoreDNS
Pod
Orders-abc
192.168.151.208
Packet
Src: 192.168.151.208
Dest: 10.100.78.3
EndpointSlice Orders
Orders-abc, Ready, 192.168.151.208
Orders-def, Ready, 192.168.151.209
Watches endpointslices
kube-proxy
iptables
EndpointSlice Products
Products-abc, Ready, 192.168.170.81
Products-def, Ready, 192.168.170.82
Update IP address
Packet
Src: 192.168.151.208
Dest: 192.168.170.81
ENI



Requisitos de mudança de zona do EKS

Para garantir o funcionamento da mudança de zona no EKS, é necessário configurar previamente o ambiente do cluster para que ele seja resiliente ao comprometimento de uma AZ. A seguir, apresentamos uma lista de opções de configuração que ajudam a garantir a resiliência.

- Provisionar os nós de processamento do seu cluster em várias AZs
- Provisionamento de capacidade de computação suficiente para atender à remoção de uma única AZ.
- Escalabilidade prévia dos pods em todas as AZs, incluindo o CoreDNS.
- Distribuição de várias réplicas de pods entre todas as AZs, auxiliando na garantia de que você ainda terá capacidade de escalabilidade suficiente após a mudança de zona de uma AZ individual.
- Colocalização de pods interdependentes ou relacionados na mesma AZ.
- Verificação se o ambiente de cluster opera conforme previsto na ausência de uma AZ, ao iniciar manualmente uma mudança de zona de uma AZ. Como alternativa, você pode habilitar a mudança de zona automática e contar com as execuções de teste de mudança de zona automática. Realizar

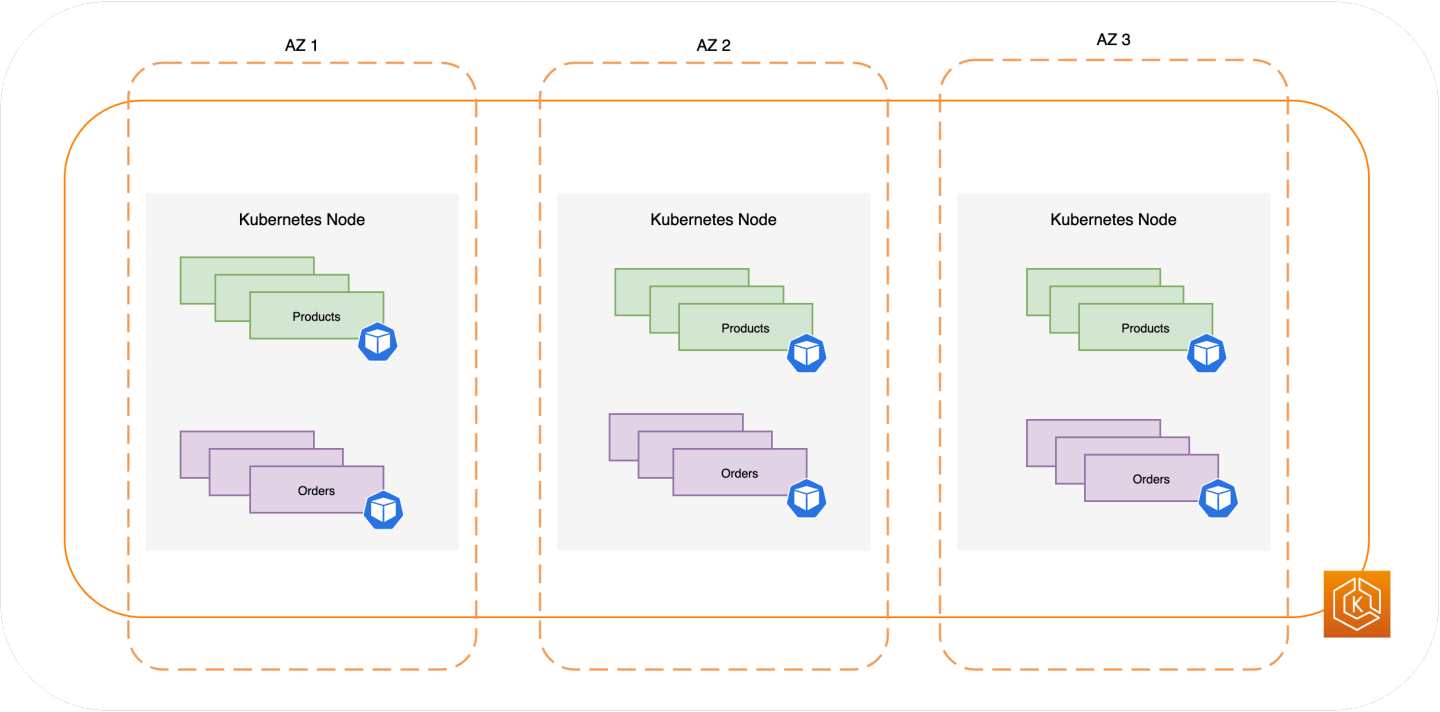
testes com mudanças de zona manuais ou de prática não é obrigatório para que a mudança de zona funcione no EKS, mas é uma prática fortemente recomendada.

Provisionamento de nós de processamento do EKS em várias zonas de disponibilidade

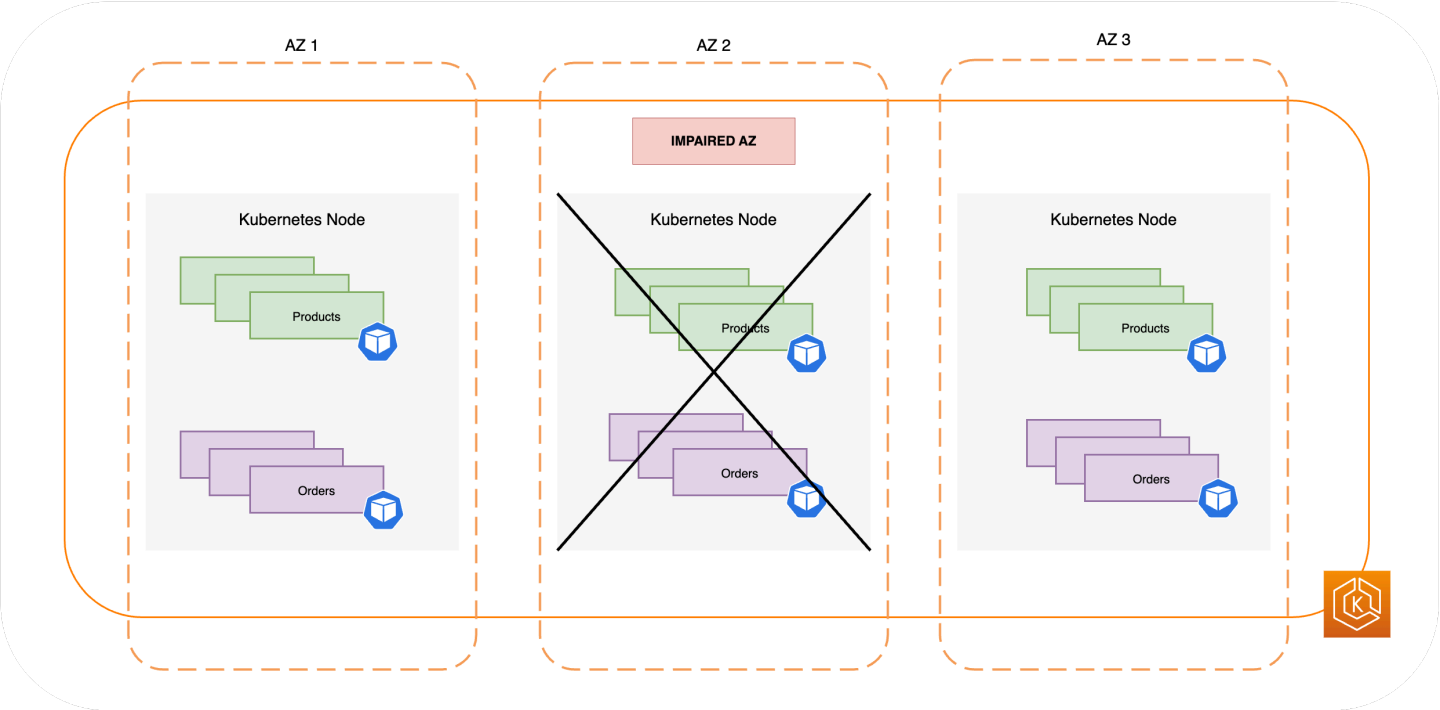
As regiões da AWS contam com diversos locais distintos que abrigam data centers físicos, conhecidos como zonas de disponibilidade (AZs). As AZs são projetadas para serem fisicamente isoladas umas das outras a fim de evitar impactos simultâneos que possam afetar toda uma região. Ao realizar o provisionamento de um cluster do EKS, a recomendação é implantar os nós de processamento por diversas AZs em uma região. Essa prática contribui para que o ambiente de cluster seja mais resiliente em caso do comprometimento de uma única AZ, permitindo que você mantenha a alta disponibilidade das aplicações em execução nas demais AZs. Ao iniciar uma mudança de zona para fora da AZ impactada, a rede interna do cluster do ambiente do EKS é atualizada automaticamente para usar apenas AZs íntegras, ajudando a manter a alta disponibilidade do cluster.

Garantir que você tenha uma configuração multi-AZ no ambiente do EKS aumenta a confiabilidade geral do sistema. Contudo, ambientes multi-AZ influenciam o processamento e a transferência de dados das aplicações, resultando em um impacto nos custos de rede do ambiente. Especificamente, o tráfego de saída frequente entre zonas (ou seja, o tráfego distribuído entre AZs) pode ter um grande impacto nos custos relacionados à rede. É possível aplicar estratégias diferentes para controlar a quantidade de tráfego entre zonas entre os pods no cluster do EKS e reduzir os custos associados. Para obter mais informações sobre como otimizar os custos de rede ao executar ambientes do EKS de alta disponibilidade, consulte estas práticas recomendadas.

O diagrama apresentado a seguir ilustra um ambiente do EKS de alta disponibilidade com três AZs íntegras.



O diagrama abaixo demonstra como um ambiente do EKS com três AZs é resiliente ao comprometimento de uma das zonas, mantendo a alta disponibilidade por meio das duas AZs íntegras remanescentes.



Provisionamento de capacidade de computação suficiente para resistir à remoção de uma única zona de disponibilidade

A fim de otimizar a utilização de recursos e os custos da infraestrutura de computação no plano de dados do EKS, recomenda-se garantir a escalabilidade da capacidade de computação para que ela esteja alinhada às necessidades das workloads. No entanto, se todos os seus nós de processamento estiverem com capacidade máxima, você dependerá da adição de novos nós de processamento ao plano de dados do EKS antes que novos pods da workload possam ser agendados. Ao executar workloads críticas, geralmente é uma boa prática operar com capacidade redundante ativa para lidar com cenários como aumentos repentinos de carga e problemas de integridade dos nós. Ao optar pelo uso da mudança de zona, você está planejando a remoção da capacidade de uma zona de disponibilidade completa no caso do comprometimento. Isso implica ajustar a capacidade redundante de computação para que haja escalabilidade suficiente para suportar a carga, mesmo que uma das AZs não esteja ativa.

Ao escalar os recursos de computação, a inclusão de novos nós no plano de dados do EKS não é instantânea e requer um tempo de provisionamento. Isso pode ter implicações na performance em tempo real e na disponibilidade das aplicações, especialmente no caso do comprometimento da zona. O ambiente do EKS deve conseguir suportar a carga de trabalho caso uma AZ fique indisponível, sem comprometer a experiência dos seus usuários finais ou clientes. Isso significa minimizar ou eliminar o atraso entre o momento em que um novo pod é necessário e o momento em que ele é efetivamente agendado em um nó de processamento.

Além disso, em casos de comprometimento de uma zona, o objetivo deve ser evitar limitações de capacidade de computação que possam impedir que novos nós sejam integrados ao plano de dados do EKS nas zonas de disponibilidade que estiverem operando normalmente.

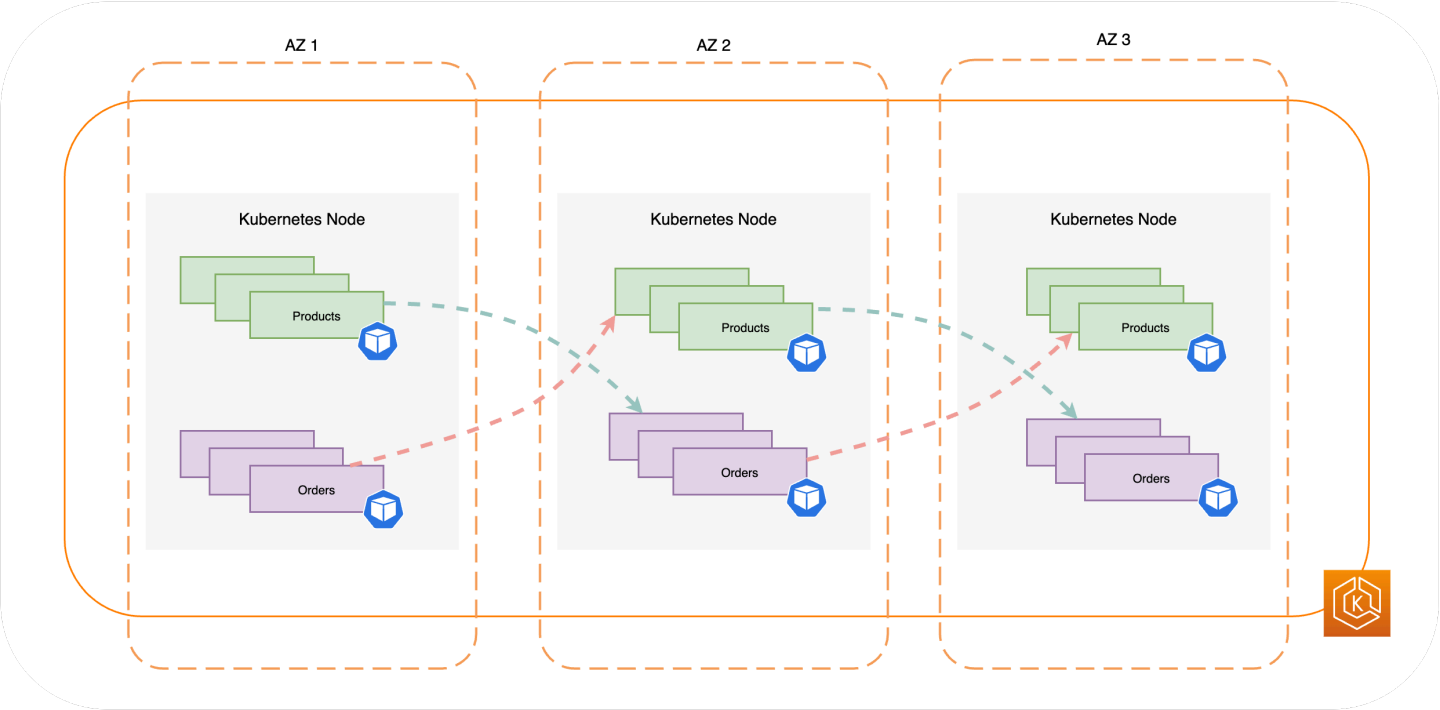
Para reduzir o risco desses potenciais impactos negativos, recomendamos que você provisione capacidade de computação excedente em parte dos nós de processamento em cada uma das AZs. Ao fazer isso, o Agendador do Kubernetes dispõe de capacidade disponível antecipadamente para novas alocações de pods, o que é especialmente importante quando você perde uma das AZs no ambiente.

Execução e distribuição de diversas réplicas de pods entre as zonas de disponibilidade

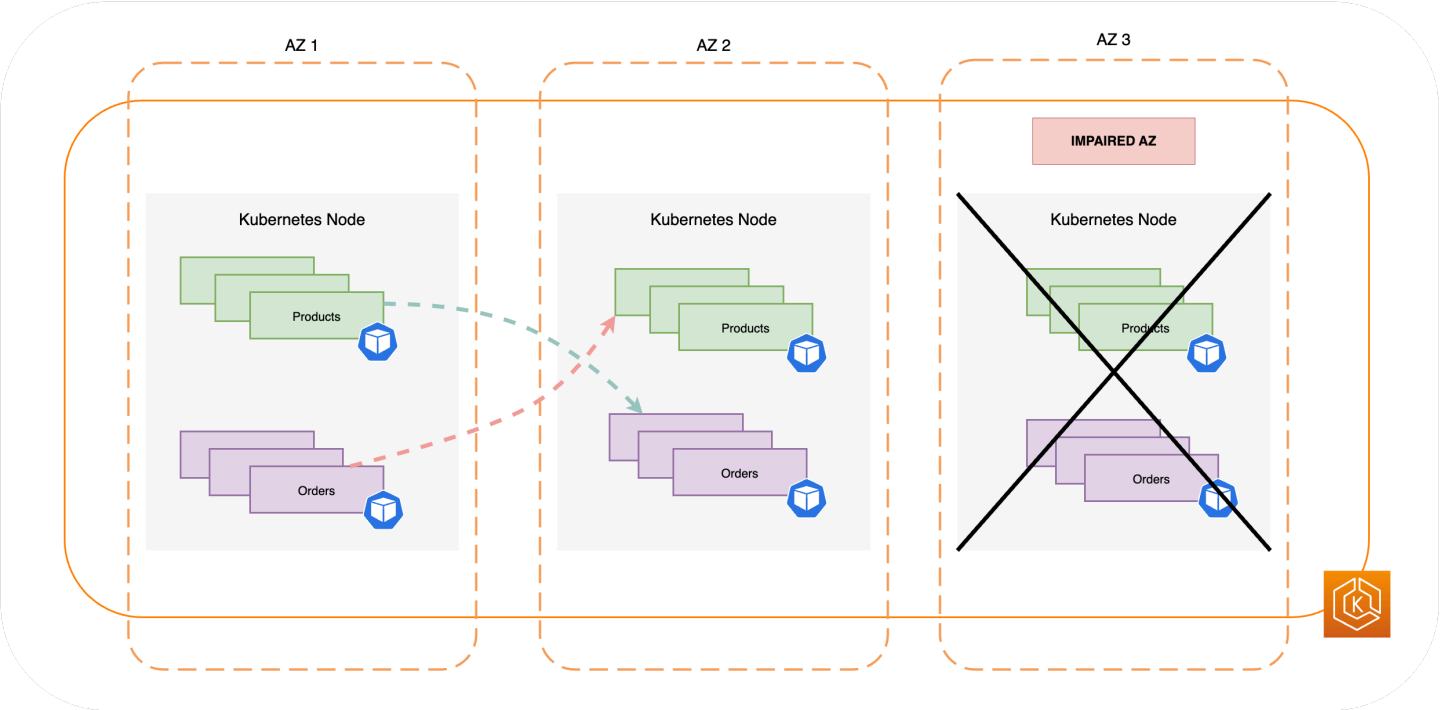
O Kubernetes permite que você pré-escale suas workloads executando várias instâncias (réplicas de pod) de uma única aplicação. Executar diversas réplicas de pods para uma aplicação elimina

pontos únicos de falha e aumenta a performance geral ao reduzir a sobrecarga de recursos em uma única réplica. No entanto, para garantir alta disponibilidade e tolerância a falhas aprimorada para as aplicações, recomendamos que você execute várias réplicas da aplicação e distribua essas réplicas por diferentes domínios de falha, também chamados de domínios de topologia. Os domínios de falha neste cenário são as zonas de disponibilidade. Ao utilizar restrições de distribuição de topologia, você pode configurar as aplicações para terem estabilidade estática já estabelecida. Assim, quando houver o comprometimento de uma AZ, seu ambiente terá réplicas suficientes em AZs íntegras para lidar imediatamente com quaisquer picos ou aumentos repentinos de tráfego.

O diagrama apresentado a seguir ilustra um ambiente do EKS com fluxo de tráfego leste-oeste quando todas as AZs estão íntegras.



O diagrama abaixo demonstra um ambiente do EKS com fluxo de tráfego leste-oeste em um cenário de falha em uma única AZ e início da mudança de zona por parte do usuário.



O trecho de código a seguir é um exemplo de como configurar a workload com várias réplicas no Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders
spec:
  replicas: 9
  selector:
    matchLabels:
      app:orders
  template:
    metadata:
      labels:
        app: orders
        tier: backend
    spec:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: "topology.kubernetes.io/zone"
        whenUnsatisfiable: ScheduleAnyway
        labelSelector:
          matchLabels:
```

```
            app: orders
```

Mais importante ainda, você deve executar várias réplicas do software do servidor de DNS (CoreDNS/kube-dns) e aplicar restrições de distribuição de topologia semelhantes, caso elas não estejam configuradas por padrão. Essa prática ajuda a assegurar que, se houver um comprometimento de uma única AZ, você tenha pods de DNS suficientes em AZs íntegras para continuar processando as solicitações de descoberta de serviços para outros pods em comunicação no cluster. O complemento do CoreDNS para EKS conta com configurações padrão para os pods do CoreDNS que garantem que, se houver nós disponíveis em diversas AZs, eles sejam distribuídos entre as zonas de disponibilidade do cluster. Se desejar, você pode substituir essas definições padrão por configurações personalizadas.

Ao instalar o CoreDNS usando o Helm, é possível atualizar o valor `replicaCount` no arquivo values.yaml para assegurar a existência de réplicas suficientes em cada AZ. Além disso, para garantir que essas réplicas sejam distribuídas entre as diferentes AZs do ambiente de cluster, certifique-se de atualizar a propriedade `topologySpreadConstraints` no mesmo arquivo `values.yaml`. O trecho de código a seguir ilustra como configurar o CoreDNS para este propósito.

CoreDNS Helm values.yaml

```
replicaCount: 6
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

Caso ocorra o comprometimento de uma AZ, é possível absorver o aumento de carga nos pods do CoreDNS ao usar um sistema de ajuste de escala automático para o CoreDNS. A quantidade de instâncias de DNS necessária dependerá do volume de workloads em execução no cluster. O CoreDNS é limitado pela CPU, o que permite que ele seja escalado com base no uso de processamento, por meio do autoescalador de pod horizontal (HPA, na sigla em inglês). A seguir, apresentamos um exemplo que você pode modificar para atender às suas necessidades.

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: coredns
```

```
  namespace: default
spec:
  maxReplicas: 20
  minReplicas: 2
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: coredns
  targetCPUUtilizationPercentage: 50
```

Como alternativa, o EKS pode gerenciar o ajuste de escala automático da implantação do CoreDNS na versão do complemento do CoreDNS para o EKS. Esse autoscaler do CoreDNS monitora continuamente o estado do cluster, incluindo o número de nós e núcleos de CPU. Com base nessas informações, o controlador ajusta dinamicamente o número de réplicas da implantação do CoreDNS em um cluster do EKS.

Para habilitar a configuração de ajuste de escala automático no complemento do CoreDNS para EKS, use a seguinte definição de configuração:

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

Você também pode usar o DNS NodeLocal ou o autoscaler proporcional ao cluster para dimensionar o CoreDNS. Para obter mais informações, consulte Scaling CoreDNS horizontally.

Colocalização de pods interdependentes na mesma zona de disponibilidade

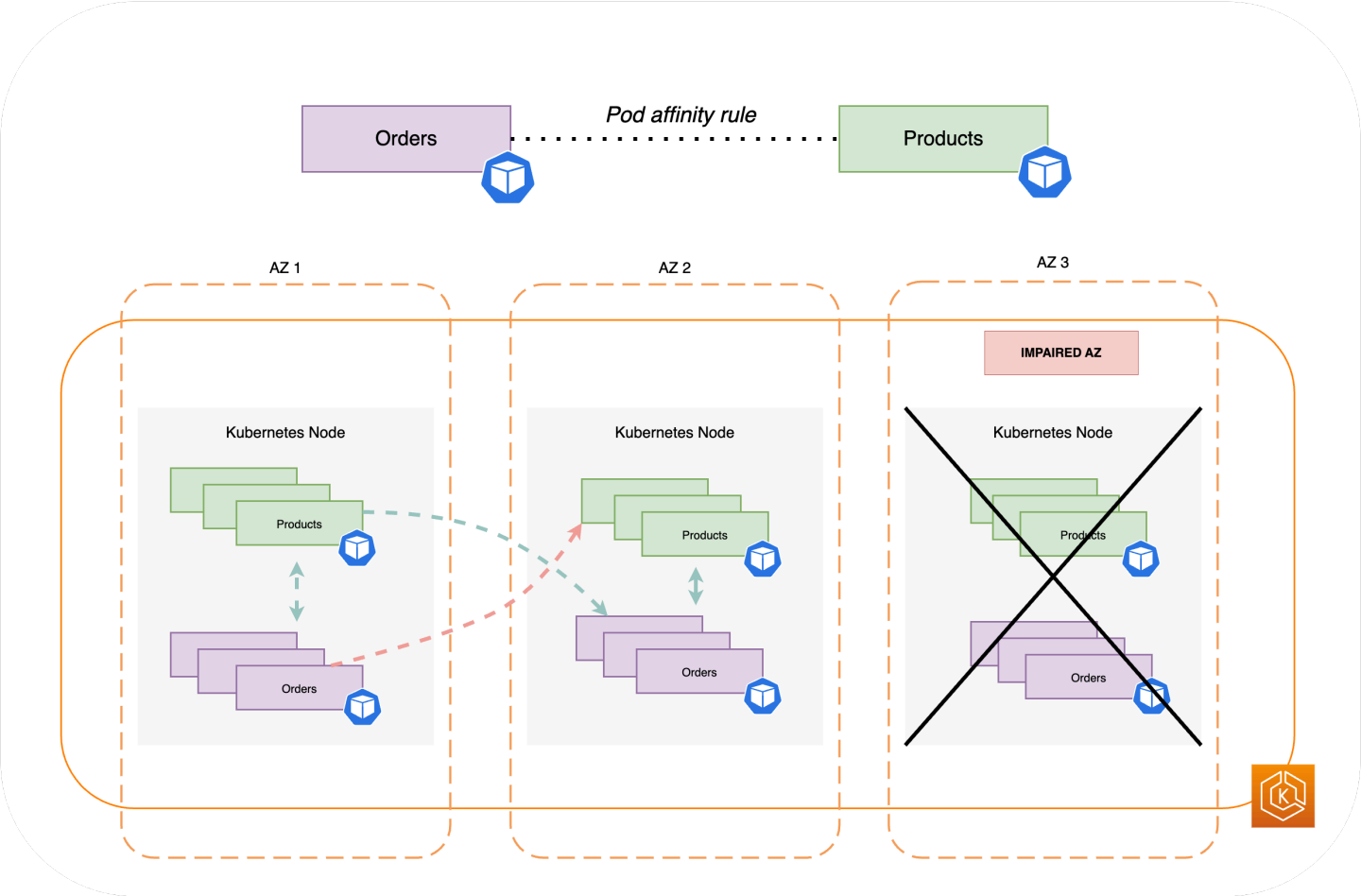
Geralmente, as aplicações têm workloads distintas que precisam se comunicar entre si para concluir um processo de ponta a ponta com êxito. Caso essas aplicações distintas estejam distribuídas em várias AZs e não sejam alocadas na mesma zona, um problema em apenas uma dessas AZs poderá comprometer todo o processo de ponta a ponta. Por exemplo, se a Aplicação A tiver várias réplicas nas zonas de disponibilidade 1 e 2, mas a Aplicação B tiver todas as suas réplicas na zona de disponibilidade 3, a perda da zona 3 afetará os processos de ponta a ponta entre as duas workloads, Aplicação A e Aplicação B. Se você combinar restrições de distribuição de topologia com afinidade de pods, poderá aumentar a resiliência da aplicação distribuindo os pods por todas as zonas de disponibilidade. Além disso, essa definição estabelece uma conexão entre determinados pods para assegurar que sejam alocados no mesmo local.

Com regras de afinidade de pod, você pode definir relações entre workloads para influenciar o comportamento do Kubernetes Scheduler, de modo que ele coloque os pods no mesmo nó de processamento ou na mesma AZ. Você também pode configurar o quanto as restrições de agendamento devem ser rigorosas.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: products
  namespace: ecommerce
  labels:
    app.kubernetes.io/version: "0.1.6"

    spec:
      serviceAccountName: graphql-service-account
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - orders
            topologyKey: "kubernetes.io/hostname"
```

O diagrama apresentado a seguir mostra vários pods que foram alocados no mesmo local, no mesmo nó, com as regras de afinidade de pods.



Teste para validar se o ambiente de cluster consegue lidar com a perda de uma AZ

Após concluir os requisitos descritos nas seções anteriores, a próxima etapa consiste em testar se você conta com capacidade de computação e de workload suficientes para lidar com a perda de uma AZ. Isso pode ser feito ao iniciar manualmente uma mudança de zona no EKS. Como alternativa, você pode habilitar a mudança de zona automática e configurar execuções práticas, que também verificam se as aplicações funcionam conforme o esperado com uma AZ a menos no ambiente do cluster.

Perguntas frequentes

Por que devo usar esse recurso?

Ao usar a mudança de zona do ARC ou a mudança de zona automática no cluster do EKS, você pode manter melhor a disponibilidade das aplicações Kubernetes automatizando o processo de recuperação rápida de deslocamento do tráfego de rede no cluster para longe de uma AZ afetada.

Com o ARC, é possível evitar etapas extensas e complexas que podem resultar em um período de recuperação prolongado durante eventos de AZ com falhas.

Como esse recurso funciona com outros serviços do AWS?

O EKS se integra ao ARC, que fornece a interface principal para você realizar operações de recuperação na AWS. Para garantir que o tráfego dentro do cluster seja direcionado corretamente para fora de uma AZ comprometida, o EKS faz alterações na lista de endpoints de rede dos pods que estão em execução no plano de dados do Kubernetes. Se você estiver usando o Elastic Load Balancing para direcionar tráfego externo para o cluster, é possível registrar os balanceadores de carga no ARC e iniciar uma mudança de zona neles para evitar que o tráfego seja enviado para a AZ degradada. Além disso, a mudança de zona é compatível com grupos do Amazon EC2 Auto Scaling criados por grupos de nós gerenciados do EKS. Para impedir que uma AZ degradada seja usada para criar novos pods do Kubernetes ou para iniciar novos nós, o EKS exclui a AZ comprometida dos grupos do Auto Scaling.

Como esse recurso é diferente das proteções padrão do Kubernetes?

Esse recurso funciona em conjunto com diversas proteções nativas do Kubernetes que ajudam na resiliência das aplicações dos clientes. Você pode configurar a prontidão do pod e as sondas de atividade que decidem quando um pod deve receber tráfego. Se essas sondas falharem, o Kubernetes retira esses pods da lista de destinos de um serviço, e o tráfego não é mais direcionado ao pod. Embora isso seja útil, não é simples para os clientes configurar essas verificações de integridade de modo que elas falhem garantidamente quando uma AZ estiver degradada. O recurso de mudança de zona do ARC oferece uma camada adicional de segurança que ajuda a isolar completamente uma AZ degradada quando as proteções nativas do Kubernetes não foram suficientes. Além disso, a mudança de zona fornece um método simples para verificar a prontidão operacional e a resiliência da arquitetura.

A AWS pode iniciar uma mudança de zona em meu nome?

Sim, se você quiser uma maneira totalmente automatizada de usar a mudança de zona do ARC, poderá habilitar a mudança de zona automática do ARC. Com a mudança de zona automática, você pode contar com a AWS para monitorar a integridade das AZs do cluster do EKS e iniciar automaticamente uma mudança de zona quando o comprometimento de uma AZ for detectado.

O que acontecerá se eu usar esse recurso e meus nós de processamento e workloads não forem pré-escalonados?

Caso não tenha escalado os recursos antecipadamente e precise provisionar nós ou pods adicionais durante uma mudança de zona, a recuperação do sistema poderá sofrer atrasos. O processo de adição de novos nós ao plano de dados do Kubernetes demora algum tempo, o que pode afetar a performance em tempo real e a disponibilidade das aplicações, especialmente quando há o comprometimento de uma AZ. Além disso, durante o comprometimento de uma AZ, pode haver uma restrição de capacidade de computação que impeça a inclusão de nós recém-requeridos nas AZs que estão operando normalmente.

Se as workloads não estiverem com recursos escalados antecipadamente e distribuídos por todas as AZs do cluster, um comprometimento de zona poderá afetar a disponibilidade de uma aplicação que esteja sendo executada apenas em nós de processamento na AZ impactada. Para reduzir o risco de uma interrupção completa da disponibilidade da sua aplicação, o EKS tem uma proteção contra falhas para que o tráfego seja enviado aos endpoints do Pod em uma zona afetada se essa workload tiver todos os seus endpoints na AZ não íntegra. Contudo, é altamente recomendável escalar recursos antecipadamente e distribuir aplicações entre todas as AZs, garantindo que o serviço mantenha a disponibilidade se houver falha em uma das zonas.

Como isso funciona caso eu esteja executando uma aplicação com estado?

Se você estiver executando uma aplicação com estado, deverá avaliar a tolerância a falhas com base no caso de uso e na arquitetura. Caso use um padrão ou uma arquitetura com ativo e em espera, pode haver situações em que o recurso ativo esteja em uma AZ comprometida. No nível da aplicação, se o recurso em espera não for devidamente ativado, você poderá enfrentar problemas com a aplicação. Você também pode enfrentar problemas quando novos pods do Kubernetes forem iniciados em AZs íntegras, já que esses pods não conseguirão se conectar aos volumes persistentes vinculados à AZ comprometida.

Esse recurso funciona com o Karpenter?

No momento, o suporte ao Karpenter não está disponível com a mudança de zona do ARC e a mudança de zona automática no EKS. Caso uma AZ esteja comprometida, é possível ajustar a configuração pertinente do NodePool do Karpenter ao remover a AZ que não está íntegra, garantindo que novos nós de processamento sejam iniciados somente nas demais AZs.

Esse recurso funciona com o EKS Fargate?

Esse recurso não funciona com o EKS Fargate. Por padrão, quando o EKS Fargate reconhecer um evento de integridade de zona, os pods preferirão ser executados nas outras AZs.

O ambiente de gerenciamento do Kubernetes gerenciado pelo EKS será afetado?

Não, por padrão, o Amazon EKS executa e dimensiona o ambiente de gerenciamento do Kubernetes em várias AZs para garantir alta disponibilidade. A mudança de zona e a mudança de zona automática do ARC operam somente no plano de dados do Kubernetes.

Há algum custo associado a esse novo recurso?

Você pode usar a mudança de zona do ARC e a mudança de zona automática no seu cluster do EKS sem custo adicional. No entanto, você continuará pagando pelas instâncias provisionadas. Portanto, recomendamos que realize a escala antecipada de recursos para o plano de dados do Kubernetes antes de utilizá-los. Você deve considerar um equilíbrio entre o custo e a disponibilidade da aplicação.

Recursos adicionais

- Como funciona uma mudança de zona
- Práticas recomendadas para mudanças de zona no ARC
- Recursos e cenários suportados para deslocamento zonal e deslocamento automático zonal
- Operar workloads resilientes no Amazon EKS
- Elimine o atraso no dimensionamento de nós do Kubernetes com prioridade de pods e o superprovisionamento
- Dimensione pods do CoreDNS para alto tráfego de DNS

Habilitação da mudança de zona do EKS para evitar zonas de disponibilidade comprometidas

O Amazon Application Recovery Controller (ARC) ajuda você a gerenciar e coordenar a recuperação de aplicações nas zonas de disponibilidade (AZs) e funciona com muitos serviços, incluindo o Amazon EKS. Com o suporte do EKS para a mudança de zona do ARC, é possível deslocar o tráfego de rede no cluster para longe de uma AZ com problemas. Você também pode autorizar o AWS a monitorar a integridade das suas AZs e desviar temporariamente o tráfego de rede de uma AZ não íntegra em seu nome.

Como usar a mudança de zona do EKS:

1. Habilite seu cluster do EKS com o Amazon Application Recovery Controller (ARC). Isso é feito no nível do cluster usando o Console do Amazon EKS, a AWS CLI, o CloudFormation ou o eksctl.

2. Uma vez habilitado, você pode gerenciar mudanças de zona ou mudanças de zona automáticas usando o console do ARC, a AWS CLI ou as APIs de mudança de zona e de mudança de zona automática.

Observe que, depois de registrar um cluster EKS no ARC, ainda é necessário configurar o ARC. Por exemplo, é possível utilizar o console do ARC para configurar a mudança de zona automática.

Para obter informações mais detalhadas sobre o funcionamento da mudança de zona do EKS e como estruturar as workloads para lidar com zonas de disponibilidade comprometidas, consulte the section called “Saiba mais sobre a mudança de zona”.

Considerações

- O modo automático do EKS não é compatível com o Controlador de Recuperação de Aplicações da Amazon, a mudança de zona e a mudança de zona automática.
- Recomendamos esperar pelo menos 60 segundos entre as operações de mudança de zona para garantir o processamento adequado de cada solicitação.

  Ao tentar realizar mudanças de zona em uma sucessão rápida (em até 60 segundos uma da outra), o serviço do Amazon EKS pode não processar adequadamente todas as solicitações de mudança. Isso se deve ao mecanismo de sondagem atual que atualiza o estado zonal do cluster. Se você precisar realizar várias mudanças de zona, certifique-se de que haja um tempo adequado entre as operações para que o sistema processe cada mudança.

O que é o Amazon Application Recovery Controller?

O Amazon Application Recovery Controller (ARC) ajuda você a se preparar e a realizar uma recuperação mais rápida dos aplicações executados em AWS. A mudança de zona permite que você se recupere rapidamente de deficiências na zona de disponibilidade (AZ), movendo temporariamente o tráfego de um recurso compatível de uma AZ para AZs íntegras na região da AWS.

Saiba mais sobre o Amazon Application Recovery Controller (ARC)

O que é mudança de zona?

A mudança de zona é um recurso do ARC que permite mover o tráfego de um recurso, como um cluster do EKS ou um Elastic Load Balancer, para fora de uma zona de disponibilidade em uma região da AWS para atenuar rapidamente um problema e recuperar rapidamente a aplicação.

Você pode optar por mudar o tráfego, por exemplo, porque uma implantação ruim está causando problemas de latência ou porque a Availability Zone está afetada. Uma mudança de zona não exige nenhuma etapa de configuração prévia.

Saiba mais sobre a mudança de zona do ARC.

O que é mudança de zona automática?

A mudança de zona automática é um recurso do ARC que você pode habilitar para autorizar a AWS a desviar o tráfego de uma AZ para recursos compatíveis, em seu nome, para AZs saudáveis na região da AWS. A AWS inicia uma transferência automática quando a telemetria interna indica que há uma deficiência em uma AZ em uma região que pode afetar potencialmente os clientes. A telemetria interna incorpora métricas de várias fontes, incluindo a rede AWS e os serviços Amazon EC2 e Elastic Load Balancing.

A AWS encerra as mudanças automáticas quando os indicadores mostram que não há mais um problema ou um problema em potencial.

Saiba mais sobre a mudança de zona automática do ARC.

O que o EKS faz durante uma mudança automática?

O EKS atualiza as configurações de rede para evitar o direcionamento do tráfego para AZs com problemas. Além disso, se você estiver usando Grupos de nós gerenciados, o EKS só iniciará novos nós nas AZs íntegras durante uma mudança de zona. Quando a mudança expirar ou for cancelada, as configurações de rede serão restauradas para incluir a AZ que foi detectada anteriormente como não íntegra.

Saiba mais sobre a mudança de zona do EKS.

Registre o cluster EKS com o Amazon Application Recovery Controller (ARC) (console do AWS)

1. Encontre o nome e a região do cluster EKS que você deseja registrar no ARC.
2. Navegue até o console do EKS nessa região e selecione seu cluster.
3. Na página de informações do cluster, selecione a guia Overview (Visão geral).
4. No título Mudança de zona, selecione o botão Gerenciar.
5. Selecione habilitar ou desabilitar para a mudança de zona do EKS.

Agora, seu cluster EKS está registrado no ARC.

Para que a AWS identifique e evite zonas de disponibilidade comprometidas, é necessário configurar a mudança de zona automática do ARC. Por exemplo, você pode fazer isso no console do ARC.

Próximas etapas

- Saiba como habilitar a mudança de zona automática.
- Saiba como iniciar uma mudança de zona manualmente.

Saiba como o controle de acesso funciona no Amazon EKS

Saiba como gerenciar o acesso ao seu cluster do Amazon EKS. O uso do Amazon EKS exige o conhecimento de como o Kubernetes e o AWS Identity and Access Management (AWS IAM) lidam com o controle de acesso.

Esta seção inclui:

the section called “Acesso à API do Kubernetes” : saiba como permitir que aplicações ou usuários se autentiquem na API do Kubernetes. É possível usar entradas de acesso, o aws-auth ConfigMap ou um provedor de OIDC externo.

the section called “Acessar recursos do cluster” : saiba como configurar o Console de gerenciamento da AWS para se comunicar com o cluster do Amazon EKS. Use o console para visualizar os recursos do Kubernetes no cluster, como namespaces, nós e pods.

the section called “Acessar meu cluster com o kubectl” : saiba como configurar o kubectl para se comunicar com o cluster do Amazon EKS. Use a AWS CLI para criar um arquivo kubeconfig.

the section called “Acesso da workload à AWS ” : saiba como associar uma conta de serviço do Kubernetes a perfis do AWS IAM. Você pode usar a Identidade de Pods ou perfis do IAM para contas de serviço (IRSA).

Tarefas comuns

- Conceda aos desenvolvedores acesso à API do Kubernetes. Visualize os recursos do Kubernetes no Console de gerenciamento da AWS.
  - Solução: use entradas de acesso para associar permissões RBAC do Kubernetes a usuários ou perfis do AWS IAM.
- Configure o kubectl para se comunicar com um cluster do Amazon EKS usando credenciais da AWS.
  - Solução: usar a AWS CLI para criar um arquivo kubeconfig.
- Use um provedor de identidade externo, como o Ping Identity, para autenticar usuários na API do Kubernetes.
  - Solução: vincule um provedor de OIDC externo.
- Conceda às workloads no cluster do Kubernetes a capacidade de chamar APIs da AWS.

- Solução: use a Identidade de Pods para associar um perfil do AWS IAM a uma conta de serviço do Kubernetes.

Contexto

- Saiba como as contas de serviço do Kubernetes funcionam.
- Analise o modelo de controle de acesso baseado em perfil (RBAC) do Kubernetes
- Para obter mais informações sobre como gerenciar o acesso a recursos da AWS, consulte o Guia do usuário do AWS IAM. Como alternativa, faça um treinamento introdutório gratuito sobre o uso do AWS IAM.

Considerações para o Modo Automático do EKS

O Modo Automático do EKS se integra à Identidade de Pods do EKS e às entradas de acesso ao EKS.

- O Modo Automático do EKS usa entradas de acesso para conceder permissões ao Kubernetes ao ambiente de gerenciamento do EKS. Por exemplo, as políticas de acesso permitem que o Modo Automático do EKS leia informações sobre endpoints e serviços de rede.
  - Você não pode desabilitar as entradas de acesso em um cluster do Modo Automático do EKS.
  - Opcionalmente, você pode habilitar o `ConfigMap` `aws-auth`.
  - As entradas de acesso para o Modo Automático do EKS são configuradas automaticamente. Você pode visualizar essas entradas de acesso, mas não pode modificá-las.
  - Caso use um NodeClass para criar um perfil do IAM de nós personalizado, você precisará criar uma entrada de acesso para o perfil usando a política de acesso AmazonEKSAutoNodePolicy.
- Caso queira conceder permissões de workloads para os serviços da AWS, use a Identidade de Pods do EKS.
  - Não é necessário instalar o agente da Identidade de Pods nos clusters do Modo Automático do EKS.

Conceder aos usuários e perfis do IAM acesso às APIs do Kubernetes

O cluster tem um endpoint de API do Kubernetes. O Kubectl usa essa API. Você pode se autenticar nessa API usando dois tipos de identidades:

- Uma entidade principal (perfil ou usuário) do AWS Identity and Access Management (IAM): esse tipo requer autenticação no IAM. Os usuários podem fazer login na AWS como um usuário do IAM ou com uma identidade federada ao usar credenciais fornecidas por meio de uma fonte de identidades. Os usuários só poderão fazer login com uma identidade federada se o administrador tiver configurado a federação de identidades anteriormente usando perfis do IAM. Quando os usuários acessam a AWS usando a federação, estão indiretamente assumindo um perfil. Quando os usuários usam esse tipo de identidade, você:
  - Pode atribuir permissões do Kubernetes a eles para que possam trabalhar com objetos do Kubernetes no cluster. Para obter mais informações sobre como atribuir permissões às entidades principais do IAM para que elas possam acessar objetos do Kubernetes no seu cluster, consulte the section called “Entradas de acesso”.
  - Pode atribuir a eles permissões de IAM para que possam trabalhar com seu cluster do Amazon EKS e seus recursos usando a API do Amazon EKS, AWS CLI, AWS CloudFormation, Console de gerenciamento da AWS ou `eksctl`. Para obter mais informações, consulte Ações definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização do serviço.
  - Os nós se juntam ao seu cluster assumindo um perfil do IAM. A capacidade de acessar seu cluster usando princípios de IAM é fornecida pelo AWS IAM Authenticator for Kubernetes, que é executado no ambiente de gerenciamento do Amazon EKS.
- Um usuário em seu próprio provedor OpenID Connect (OIDC): este tipo requer autenticação em seu provedor OIDC. Para obter mais informações sobre como configurar o seu próprio provedor OIDC com o cluster do Amazon EKS, consulte the section called “Vincular um provedor OIDC”. Quando os usuários usam esse tipo de identidade, você:
  - Pode atribuir permissões do Kubernetes a eles para que possam trabalhar com objetos do Kubernetes no cluster.
  - Não é possível atribuir a eles permissões de IAM para que possam trabalhar com seu cluster do Amazon EKS e seus recursos usando a API do Amazon EKS, AWS CLI, AWS CloudFormation, Console de gerenciamento da AWS ou `eksctl`.

Você pode usar os dois tipos de identidades com seu cluster. Não é possível desabilitar o método de autenticação do IAM. O método de autenticação OIDC é opcional.

Associe identidades do IAM a permissões do Kubernetes

O Autenticador AWS IAM para Kubernetes está instalado no ambiente de gerenciamento do seu cluster. Ele habilita as entidades principais (perfis e usuários) do AWS Identity and Access Management (IAM) que você permite acessar os recursos do Kubernetes no seu cluster. É possível permitir que entidades principais do IAM acessem objetos do Kubernetes no seu cluster usando um dos seguintes métodos:

- Criação de entradas de acesso: se o cluster for igual ou posterior à versão da plataforma listada na seção Pré-requisitos da versão do Kubernetes do cluster, recomendamos que você use essa opção.

  Use entradas de acesso para gerenciar as permissões do Kubernetes das entidades principais do IAM de fora do cluster. Você pode adicionar e gerenciar o acesso ao cluster usando a API do EKS, a interface de linha de comando AWS, os SDKs AWS, o AWS CloudFormation e o Console de gerenciamento da AWS. Isso significa que você pode gerenciar usuários com as mesmas ferramentas com as quais criou o cluster.

  Para começar, siga as instruções em Alterar o modo de autenticação para usar entradas de acesso e, em seguida, Migrar entradas existentes do ConfigMap aws-auth para entradas de acesso.
- Adicionar entradas ao **`ConfigMap aws-auth`**: se a versão da plataforma do cluster for anterior à versão listada na seção Pré-requisitos, você deverá usar esta opção. Se a versão da plataforma do cluster for igual ou posterior à versão da plataforma listada na seção Pré-requisitos da versão do Kubernetes do cluster e você tiver adicionado entradas ao `ConfigMap`, recomendamos que migre essas entradas para entradas de acesso. No entanto, você não pode migrar entradas que o Amazon EKS adicionou ao `ConfigMap`, como entradas para perfis do IAM usadas com grupos de nós gerenciados ou perfis do Fargate. Para obter mais informações, consulte the section called “Acesso à API do Kubernetes”.
  - Caso precise usar a opção do `aws-auth ConfigMap`, você poderá adicionar entradas ao `ConfigMap` usando o comando `eksctl create iamidentitymapping`. Para obter mais informações, consulte Gerenciar usuários e perfis do IAM na documentação do `eksctl`.

Definir modo de autenticação do cluster

Cada cluster tem um modo de autenticação. O modo de autenticação determina quais métodos você pode usar para permitir que as entidades principais do IAM acessem os objetos do Kubernetes no seu cluster. Existem três modos de autenticação.

> **Important**
>
> Depois que o método de entrada de acesso estiver habilitado, ele não poderá ser desabilitado.
> Se o método `ConfigMap` não for habilitado durante a criação do cluster, ele não poderá ser habilitado posteriormente. Todos os clusters criados antes da introdução das entradas de acesso têm o método `ConfigMap` habilitado.
> Caso esteja usando nós híbridos com o cluster, você deverá usar os modos de autenticação de `API` ou `API_AND_CONFIG_MAP` do cluster.

O **`aws-auth` `ConfigMap`** dentro do cluster

Este é o modo de autenticação original para clusters do Amazon EKS. A entidade principal do IAM que criou o cluster é o usuário inicial que pode acessar o cluster usando `kubectl`. O usuário inicial deve adicionar outros usuários à lista no `aws-auth` `ConfigMap` e atribuir permissões que afetem os outros usuários dentro do cluster. Os outros usuários não podem gerenciar nem remover o usuário inicial, pois não há uma entrada no `ConfigMap` para gerenciar.

Tanto as entradas do **`ConfigMap`** quanto as de acesso

Com este modo de autenticação, é possível usar os dois métodos para adicionar entidades principais do IAM ao cluster. Observe que cada método armazena entradas separadas; por exemplo, se você adicionar uma entrada de acesso pela AWS CLI, o `aws-auth` `ConfigMap` não será atualizado.

Somente entradas de acesso

Com esse modo de autenticação, você pode usar a API do EKS, a AWS Command Line Interface, SDKs AWS, o AWS CloudFormation e o Console de gerenciamento da AWS para gerenciar o acesso ao cluster para as entidades principais do IAM.

Cada entrada de acesso tem um tipo e você pode usar a combinação de um escopo de acesso para limitar a entidade principal a um namespace específico e uma política de acesso para definir

políticas de permissões reutilizáveis pré-configuradas. Como alternativa, você pode usar o tipo STANDARD e grupos de RBAC do Kubernetes para atribuir permissões personalizadas.

Modos de autenticação	Métodos
Somente `ConfigMap` (`CONFIG_MAP`)	`aws-auth` `ConfigMap`
API do EKS e `ConfigMap` (`API_AND_CONFIG_MAP`)	entradas de acesso na API do EKS, na interface de linha de comando AWS, nos SDKs da AWS, no AWS CloudFormation e no Console de gerenciamento da AWS e `aws-auth` `ConfigMap`
Somente API do EKS (`API`)	entradas de acesso na API do EKS, na interface de linha de comando AWS, nos SDKs da AWS, no AWS CloudFormation e no Console de gerenciamento da AWS

> **Note**
>
> O Modo Automático do Amazon EKS requer entradas de acesso.

Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS

Esta seção foi criada para mostrar como gerenciar o acesso de entidades principais do IAM aos clusters do Kubernetes no Amazon Elastic Kubernetes Service (EKS) usando entradas e políticas de acesso. Você encontrará detalhes sobre como alterar os modos de autenticação, migrar a partir de entradas legadas ConfigMap do `aws-auth`, criar, atualizar e excluir entradas de acesso, associar políticas a entradas, revisar permissões de políticas predefinidas e os principais pré-requisitos e considerações para o gerenciamento seguro de acesso.

Visão geral

As entradas de acesso do EKS são a maneira mais eficiente de conceder aos usuários o acesso à API do Kubernetes. Por exemplo, você pode usar as entradas de acesso para conceder aos desenvolvedores acesso para usar o kubectl. De maneira geral, uma entrada de acesso do EKS associa um conjunto de permissões do Kubernetes com uma identidade do IAM, como um perfil do IAM. Por exemplo, um desenvolvedor pode assumir um perfil do IAM e usá-lo para se autenticar em um cluster do EKS.

Atributos

- Autenticação e autorização centralizadas: controla o acesso aos clusters do Kubernetes diretamente por meio das APIs do Amazon EKS, eliminando a necessidade de alternar entre as APIs da AWS e as do Kubernetes para obter permissões de usuário.
- Gerencimento granular de permissões: usa entradas e políticas de acesso para definir permissões granulares para as entidades principais do AWS IAM, incluindo a capacidade de modificar ou de revogar o acesso de administrador de cluster do criador.
- Ferramenta de integração de IaC: oferece suporte a ferramentas de infraestrutura como código como o AWS CloudFormation, o Terraform e o AWS CDK para definir configurações de acesso durante a criação do cluster.
- Recuperação de erros de configuração: permite restaurar o acesso ao cluster por meio da API do Amazon EKS, sem acesso direto à API do Kubernetes.
- Sobrecarga reduzida e segurança aprimorada: centraliza as operações para reduzir a sobrecarga ao aproveitar as funcionalidades do AWS IAM, como o registro em log da auditoria do CloudTrail e a autenticação multifator.

Como anexar permissões

Você pode conceder permissões do Kubernetes às entradas de acesso de duas maneiras:

- Use uma política de acesso. As políticas de acesso são modelos de permissões do Kubernetes definidos previamente e mantidos pela AWS. Para obter mais informações, consulte the section called “Analisar as políticas de acesso”.
- Referencie um grupo do Kubernetes. Se você associar uma identidade do IAM a um grupo do Kubernetes, será possível criar recursos do Kubernetes que concedem permissões ao grupo. Para obter mais informações, consulte Usar a autorização de RBAC na documentação do Kubernetes.

Considerações

Ao habilitar entradas de acesso ao EKS em clusters existentes, lembre-se do seguinte:

- Comportamento de cluster legado: para clusters criados antes da introdução de entradas de acesso (aqueles com versões iniciais da plataforma anteriores às especificadas nos Requisitos de versão da plataforma), o EKS cria automaticamente uma entrada de acesso refletindo permissões preexistentes. Essa entrada inclui a identidade do IAM que criou originalmente o cluster e as permissões administrativas concedidas a essa identidade durante a criação do cluster.
- Manipulação do ConfigMap do **`aws-auth`** legado: se seu cluster depende do ConfigMap do `aws-auth` legado para gerenciamento de acesso, somente a entrada de acesso do criador original do cluster é criada automaticamente ao habilitar as entradas de acesso. Perfis ou permissões adicionais adicionadas ao ConfigMap (por exemplo, perfis do IAM personalizados para desenvolvedores ou serviços) não são migradas automaticamente. Para resolver isso, crie manualmente as entradas de acesso correspondentes.

Conceitos básicos

1. Determine a identidade do IAM e a política de acesso que você deseja usar.
   - the section called “Analisar as políticas de acesso”
2. Habilite as entradas de acesso do EKS no seu cluster. Confirme se você tem uma versão da plataforma compatível.
   - the section called “Modo de autenticação”
3. Crie uma entrada de acesso que associe uma identidade do IAM com permissões do Kubernetes.
   - the section called “Criar entradas de acesso”
4. Faça a autenticação do cluster com a identidade do IAM.
   - the section called “Configurar a AWS CLI”
   - the section called “Configuração do kubectl e eksctl”

Associar políticas de acesso a entradas de acesso

Você pode atribuir uma ou mais políticas de acesso para acessar entradas do tipo `STANDARD`. O Amazon EKS concede automaticamente aos outros tipos de entradas de acesso as permissões necessárias para funcionar adequadamente em seu cluster. As políticas de acesso do Amazon EKS incluem permissões do Kubernetes, não permissões do IAM. Antes de associar uma política

de acesso a uma entrada de acesso, familiarize-se com as permissões do Kubernetes incluídas em cada política de acesso. Para ter mais informações, consulte the section called “Analisar as políticas de acesso”. Se nenhuma das políticas de acesso atender aos seus requisitos, não associe uma política de acesso a uma entrada de acesso. Em vez disso, especifique um ou mais nomes de grupos para a entrada de acesso e crie e gerencie objetos de controle de acesso baseado em perfil do Kubernetes. Para ter mais informações, consulte the section called “Criar entradas de acesso”.

- Uma entrada de acesso existente. Para criar uma, consulte the section called “Criar entradas de acesso”.
- Um perfil ou usuário do AWS Identity and Access Management com as seguintes permissões: `ListAccessEntries`, `DescribeAccessEntry`, `UpdateAccessEntry`, `ListAccessPolicies`, `AssociateAccessPolicy`, e `DisassociateAccessPolicy`. Para obter mais informações, consulte Actions defined by Amazon Elastic Kubernetes Service na Referência de autorização do serviço.

Antes de associar políticas de acesso às entradas de acesso, considere os seguintes requisitos:

- Você pode associar várias políticas de acesso a cada entrada de acesso, mas só pode associar cada política a uma entrada de acesso uma vez. Se você associar várias políticas de acesso, a entidade principal do IAM da entrada de acesso tem todas as permissões incluídas em todas as políticas de acesso associadas.
- Você pode definir o escopo de uma política de acesso para todos os recursos em um cluster ou especificando o nome de um ou mais namespaces do Kubernetes. Você pode usar caracteres curinga para o nome de um namespace. Por exemplo, se você quiser definir o escopo de uma política de acesso para todos os namespaces que começam com `dev-`, você pode especificar `dev-*` como um nome de namespace. Verifique se os namespaces existem no cluster e se a ortografia corresponde ao nome real do namespace no cluster. O Amazon EKS não confirma a ortografia nem a existência dos namespaces no seu cluster.
- Você pode alterar o escopo de acesso de uma política de acesso depois de associá-la a uma entrada de acesso. Caso tenha definido o escopo da política de acesso para namespaces do Kubernetes, você pode adicionar e remover namespaces da associação, conforme necessário.
- Se você associar uma política de acesso a uma entrada de acesso que também tenha nomes de grupos especificados, a entidade principal do IAM terá todas as permissões em todas as políticas de acesso associadas. Ela também tem todas as permissões em qualquer objeto `Role` ou `ClusterRole` do Kubernetes especificado em qualquer objeto `Role` ou `RoleBinding` do Kubernetes que especifica os nomes dos grupos.

- Caso execute o comando `kubectl auth can-i --list`, você não verá nenhuma permissão do Kubernetes atribuída pelas políticas de acesso associadas a uma entrada de acesso para a entidade principal do IAM que você está usando ao executar o comando. O comando só mostrará as permissões do Kubernetes se você as tiver concedido em objetos `Role` ou `ClusterRole` do Kubernetes vinculados aos nomes de grupo ou ao nome de usuário que você especificou para uma entrada de acesso.
- Se você representar um usuário ou grupo do Kubernetes ao interagir com objetos do Kubernetes no cluster, como usar o comando `kubectl` com `--as` *`username`* ou `--as-group` *`group-name`* , você estará forçando o uso da autorização RBAC do Kubernetes. Como resultado, a entidade principal do IAM não tem permissões atribuídas por nenhuma política de acesso associada à entrada de acesso. As únicas permissões do Kubernetes que o usuário ou grupo que a entidade principal do IAM está representando tem são as permissões do Kubernetes que você concedeu a eles em objetos `Role` ou `ClusterRole` do Kubernetes que você vinculou aos nomes do grupo ou do usuário. Para que a entidade principal do IAM tenha as permissões nas políticas de acesso associadas, não represente um usuário ou grupo do Kubernetes. A entidade principal do IAM ainda terá todas as permissões que você concedeu a ela nos objetos `Role` ou `ClusterRole` do Kubernetes que você vinculou aos nomes dos grupos ou do usuário que você especificou para a entrada de acesso. Para obter mais informações, consulte Personificação de usuário na documentação do Kubernetes.

Você pode associar uma política de acesso a uma entrada de acesso usando o Console de gerenciamento da AWS ou a AWS CLI.

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Escolha o nome do cluster que tem uma entrada de acesso à qual você deseja associar uma política de acesso.
3. Escolha a guia Acesso.
4. Se o tipo da entrada de acesso for Padrão, você poderá associar ou desassociar as políticas de acesso do Amazon EKS. Se o tipo da sua entrada de acesso for diferente de Padrão, essa opção não estará disponível.
5. Escolha Associar política de acesso.
6. Em Nome da política, selecione a política com as permissões que você deseja que a entidade principal do IAM tenha.. Para visualizar as permissões incluídas em cada política, consulte the section called “Analisar as políticas de acesso”.

7. Em Escopo de acesso, escolha um escopo de acesso. Se você escolher Cluster, as permissões na política de acesso serão concedidas à entidade principal do IAM para recursos em todos os namespaces do Kubernetes. Se você escolher Namespace do Kubernetes, poderá então escolher Adicionar novo namespace. No campo Namespace exibido, você pode inserir o nome de um namespace do Kubernetes no cluster. Se quiser que a entidade principal do IAM tenha as permissões em vários namespaces, você pode inserir vários namespaces.
8. Escolha Adicionar política de acesso.

AWS CLI

1. Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar e Configuração rápida com aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
2. Visualize as políticas de acesso disponíveis.

```
aws eks list-access-policies --output table
```

Veja um exemplo de saída abaixo.

```
-------------------------------------------------------------------------------------------
|                                      ListAccessPolicies                                 |
                  |
+-----------------------------------------------------------------------------------------
+
||                                        accessPolicies                                   |
                 ||
|+-------------------------------------------------------------------
+------------------------------+|
||                                arn                                 |
 name             ||
|+-------------------------------------------------------------------
+------------------------------+|
```

```
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSAdminPolicy        |
 AmazonEKSAdminPolicy         ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy |
 AmazonEKSClusterAdminPolicy  ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSEditPolicy         |
 AmazonEKSEditPolicy          ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSViewPolicy         |
 AmazonEKSViewPolicy          ||
|+-----------------------------------------------------------------------
+------------------------------+|
```

Para visualizar as permissões incluídas em cada política, consulte the section called “Analisar as políticas de acesso”.

3. Visualize suas entradas de acesso existentes. Substitua *my-cluster* pelo nome do cluster.

```
aws eks list-access-entries --cluster-name my-cluster
```

Veja um exemplo de saída abaixo.

```
{
    "accessEntries": [
        "arn:aws:iam::111122223333:role/my-role",
        "arn:aws:iam::111122223333:user/my-user"
    ]
}
```

4. Associe uma política de acesso a uma entrada de acesso. O exemplo a seguir associa a política de acesso do `AmazonEKSViewPolicy` a uma entrada de acesso. Sempre que o perfil do IAM *my-role* tentar acessar objetos do Kubernetes no cluster, o Amazon EKS autorizará o perfil a usar as permissões na política para acessar os objetos do Kubernetes somente nos namespaces *my-namespace1* e *my-namespace2* do Kubernetes. Substitua *my-cluster* pelo nome do cluster, *111122223333* pelo ID da conta AWS e *my-role* pelo nome do perfil do IAM para o qual você deseja que o Amazon EKS autorize o acesso aos objetos do cluster do Kubernetes.

```
aws eks associate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --policy-arn
 arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy
```

Se você quiser que a entidade principal do IAM tenha as permissões em todo o cluster, substitua `type=namespace,namespaces=my-namespace1,my-namespace2` por `type=cluster`. Se você quiser associar várias políticas de acesso à entrada de acesso, execute o comando várias vezes, cada uma com uma política de acesso exclusiva. Cada política de acesso associada tem seu próprio escopo.

> **Note**
>
> Se você quiser alterar posteriormente o escopo de uma política de acesso associada, execute o comando anterior novamente com o novo escopo. Por exemplo, se você quisesse remover *my-namespace2*, executaria o comando novamente usando somente `type=namespace,namespaces=my-namespace1` . Se você quisesse alterar o escopo de `namespace` para `cluster`, executaria o comando novamente usando `type=cluster`, removendo `type=namespace,namespaces=my-namespace1,my-namespace2` .

5. Determine quais políticas de acesso estão associadas a uma entrada de acesso.

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

Veja um exemplo de saída abaixo.

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws:iam::111122223333",
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws:eks::aws:cluster-access-policy/
AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster",
                "namespaces": []
            },
            "associatedAt": "2023-04-17T15:25:21.675000-04:00",
            "modifiedAt": "2023-04-17T15:25:21.675000-04:00"
        },
        {
```

```
            "policyArn": "arn:aws:eks::aws:cluster-access-policy/
AmazonEKSAdminPolicy",
            "accessScope": {
                "type": "namespace",
                "namespaces": [
                    "my-namespace1",
                    "my-namespace2"
                ]
            },
            "associatedAt": "2023-04-17T15:02:06.511000-04:00",
            "modifiedAt": "2023-04-17T15:02:06.511000-04:00"
        }
    ]
}
```

No exemplo anterior, a entidade principal do IAM para essa entrada de acesso tem permissões de visualização em todos os namespaces no cluster e permissões de administrador em dois namespaces do Kubernetes.

6. Desassocie uma política de acesso de uma entrada de acesso. Neste exemplo, a política `AmazonEKSAdminPolicy` é desassociada de uma entrada de acesso. No entanto, a entidade principal do IAM retém as permissões na política de acesso `AmazonEKSViewPolicy` para objetos nos namespaces *`my-namespace1`* e *`my-namespace2`* porque essa política de acesso não está dissociada da entrada de acesso.

```
aws eks disassociate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
    --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy
```

Para listar as políticas de acesso disponíveis, consulte the section called “Analisar as políticas de acesso”.

Migrando entradas existentes **`aws-auth ConfigMap`** para entradas de acesso

Se você adicionou entradas ao `ConfigMap` `aws-auth` no seu cluster, recomendamos que você crie entradas de acesso para as entradas existentes no seu `ConfigMap` `aws-auth`. Depois de criar as entradas de acesso, você pode remover as entradas do seu `ConfigMap`. Você não pode associar políticas de acesso às entradas no `ConfigMap` `aws-auth`. Se você quiser associar políticas de acesso às entidades principais do IAM, crie entradas de acesso.

> **Important**
>
> - Quando um cluster está no modo de autenticação `API_AND_CONFIGMAP` e há um mapeamento para o mesmo perfil do IAM tanto no `ConfigMap` `aws-auth` como nas entradas de acesso, o perfil usará o mapeamento da entrada de acesso para autenticação. As entradas de acesso têm precedência sobre as entradas do `ConfigMap` do mesmo principal do IAM.
> - Antes de remover as entradas do `ConfigMap` `aws-auth` existentes que foram criadas pelo Amazon EKS para um grupo de nós gerenciados ou um perfil do Fargate no cluster, verifique se as entradas de acesso corretas para esses recursos específicos existem no cluster do Amazon EKS. Se você remover as entradas que o Amazon EKS criou no `ConfigMap` sem ter as entradas de acesso equivalentes, seu cluster não funcionará corretamente.

Pré-requisitos

- Familiaridade com entradas de acesso e políticas de acesso. Para obter mais informações, consulte the section called “Entradas de acesso” e the section called “Associar políticas de acesso”.
- Um cluster existente com uma versão da plataforma igual ou posterior às versões listadas nos pré-requisitos do tópico the section called “Entradas de acesso”.
- Versão `0.215.0` ou posterior da ferramenta de linha de comando da `eksctl` instalada no seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
- Permissões do Kubernetes para modificar o `ConfigMap` `aws-auth` no namespace do `kube-system`.
- Um perfil ou usuário do AWS Identity and Access Management com as seguintes permissões: `CreateAccessEntry` e `ListAccessEntries`. Para obter mais informações, consulte Ações definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização do serviço.

`eksctl`

1. Veja as entradas existentes em seu `aws-auth ConfigMap`. Substitua *`my-cluster`* pelo nome do cluster.

```
eksctl get iamidentitymapping --cluster my-cluster
```

Veja abaixo um exemplo de saída.

```
ARN
           USERNAME                                GROUPS
                     ACCOUNT
arn:aws:iam::111122223333:role/EKS-my-cluster-Admins
           Admins                                  system:masters
arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers
           my-namespace-Viewers                    Viewers
arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1
           system:node:{{EC2PrivateDNSName}}       system:bootstrappers,system:nodes
arn:aws:iam::111122223333:user/my-user
           my-user
arn:aws:iam::111122223333:role/EKS-my-cluster-fargateprofile1
                              system:node:{{SessionName}}
 system:bootstrappers,system:nodes,system:node-proxier
arn:aws:iam::111122223333:role/EKS-my-cluster-managed-ng
           system:node:{{EC2PrivateDNSName}}       system:bootstrappers,system:nodes
```

2. the section called “Criar entradas de acesso” para qualquer uma das entradas do `ConfigMap` que você criou retornadas na saída anterior. Ao criar as entradas de acesso, verifique se você especificou os mesmos valores para`ARN`, `USERNAME`, `GROUPS` e `ACCOUNT` retornados em sua saída. No exemplo de saída, você criaria entradas de acesso para todas as entradas, exceto as duas últimas, já que essas entradas foram criadas pelo Amazon EKS para um perfil do Fargate e um grupo de nós gerenciados.
3. Exclua as entradas do `ConfigMap` para todas as entradas de acesso que você criou. Se você não excluir a entrada do `ConfigMap`, as configurações da entrada de acesso do ARN principal do IAM substituirão a entrada do `ConfigMap`. Substitua *`111122223333`* pelo seu ID de conta AWS e *`EKS-my-cluster-my-namespace-Viewers`* pelo nome do perfil na entrada em seu `ConfigMap`. Se a entrada que você está removendo for para um usuário do IAM, em vez de um perfil do IAM, substitua `role` por `user` e *`EKS-my-cluster-my-namespace-Viewers`* pelo nome do usuário.

```
eksctl delete iamidentitymapping --arn arn:aws:iam::111122223333:role/EKS-my-cluster-
my-namespace-Viewers --cluster my-cluster
```

Analisar permissões da política de acesso

As políticas de acesso incluem `rules` que contêm `verbs` (permissões) e `resources` do Kubernetes. As políticas de acesso não incluem permissões ou recursos do IAM. Semelhantes aos objetos `Role` e `ClusterRole` do Kubernetes, as políticas de acesso incluem apenas `rules` de `allow`. Não é possível modificar o conteúdo de uma política de acesso. Você não pode criar suas próprias políticas de acesso. Se as permissões nas políticas de acesso não atenderem às suas necessidades, crie objetos RBAC do Kubernetes e especifique nomes de grupos para as entradas de acesso. Para obter mais informações, consulte the section called “Criar entradas de acesso”. As permissões contidas nas políticas de acesso são semelhantes às permissões nos perfis de clusters do Kubernetes voltados para o usuário. Para obter mais informações, consulte User-facing roles na documentação do Kubernetes.

Listagem de todas as políticas

Use qualquer uma das políticas de acesso listadas nesta página ou recupere uma lista de todas as políticas de acesso disponíveis usando a AWS CLI:

```
aws eks list-access-policies
```

A saída esperada deve ser semelhante a esta (abreviada por brevidade):

```
{
    "accessPolicies": [
        {
            "name": "AmazonAIOpsAssistantPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonAIOpsAssistantPolicy"
        },
        {
            "name": "AmazonARCRegionSwitchScalingPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/
AmazonARCRegionSwitchScalingPolicy"
        },
        {
            "name": "AmazonEKSAdminPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy"
        },
        {
            "name": "AmazonEKSAdminViewPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy"
        },
```

```
        {
            "name": "AmazonEKSAutoNodePolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy"
        }
        // Additional policies omitted
    ]
}
```

AmazonEKSAdminPolicy

Essa política de acesso inclui permissões que concedem à entidade principal do IAM a maioria das permissões aos recursos. Quando associada a uma entrada de acesso, seu escopo de acesso geralmente é um ou mais namespaces do Kubernetes. Se você quiser que uma entidade principal do IAM tenha acesso de administrador a todos os recursos do seu cluster, associe a política de acesso the section called “AmazonEKSClusterAdminPolicy” à sua entrada de acesso.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy`

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/	get, list, watch

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
	scale , statefulsets/status	
authorization.k8s.io	localsubjectaccessreviews	create
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
rbac.authorization.k8s.io	rolebindings , roles	create, delete, deletecollection , get, list, patch, update, watch

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get,list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update
	serviceaccounts	impersonate

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
	`bindings`, `events`, `limitranges` , `namespaces/status` , `pods/log`, `pods/status` , `replicationcontrollers/status` , `resourcequotas` , `resourcequotas/status`	`get`, `list`, `watch`
	`namespaces`	`get`,`list`, `watch`

AmazonEKSClusterAdminPolicy

Essa política de acesso inclui permissões que concedem ao administrador principal do IAM acesso a um cluster. Quando associada a uma entrada de acesso, seu escopo de acesso geralmente é o cluster, em vez de um namespace do Kubernetes. Se você quiser que uma entidade principal do IAM tenha um escopo administrativo mais limitado, considere associar a política de acesso the section called “AmazonEKSAdminPolicy” à sua entrada de acesso.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy`

Grupos de APIs Kubernetes	Kubernetes nonResourceURLs	Recursos Kubernetes	Verbos (permissões) Kubernetes
*		*	*
	*		*

AmazonEKSAdminViewPolicy

Essa política de acesso inclui permissões que concedem a uma entidade principal do IAM acesso de listagem/visualização a todos os recursos em um cluster. Observe que isso inclui Segredos do Kubernetes.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy`

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
*	*	get, list, watch

AmazonEKSEditPolicy

Essa política de acesso inclui permissões para que uma entidade principal do IAM edite a maioria dos recursos do Kubernetes.

ARN – arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizonta	get, list, watch

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
	lpodautoscalers/status	
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	namespaces	get, list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	serviceaccounts	impersonate
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch

AmazonEKSViewPolicy

Essa política de acesso inclui permissões para que uma entidade principal do IAM visualize a maioria dos recursos do Kubernetes.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy`

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status	get, list, watch

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
	, replicationcontrollers/scale	
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch
	namespaces	get, list, watch

AmazonEKSSecretReaderPolicy

Esta política de acesso contém permissões que autorizam uma entidade principal do IAM a realizar a leitura do Kubernetes Secrets.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSSecretReaderPolicy`

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
	`secrets`	`get`, `list`, `watch`

AmazonEKSAutoNodePolicy

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy`

Esta política inclui as permissões abaixo, que permitem aos componentes do Amazon EKS concluir as seguintes tarefas:

- `kube-proxy`: monitore endpoints e serviços de rede e gerencie eventos relacionados. Isso habilita a funcionalidade de proxy de rede em todo o cluster.
- `ipamd`: gerencie recursos de rede da AWS VPC e interfaces de rede de contêineres (CNI). Isso permite que o daemon de gerenciamento de endereços IP lide com a rede de pods.
- `coredns`: acesse recursos de descoberta de serviços, como endpoints e serviços. Isso permite a resolução de DNS no cluster.
- `ebs-csi-driver`: trabalhe com recursos relacionados ao armazenamento de volumes do Amazon EBS. Isso permite o provisionamento dinâmico e a anexação de volumes persistentes.
- `neuron`: monitore nós e pods de dispositivos do AWS Neuron. Isso permite o gerenciamento dos aceleradores AWS Inferentia e Trainium.
- `node-monitoring-agent`: acesse diagnósticos e eventos de nós. Isso permite o monitoramento da integridade do cluster e a coleta de diagnósticos.

Cada componente usa uma conta de serviço dedicada e está restrito somente às permissões necessárias para sua função específica.

Se você especificar manualmente um perfil do IAM de nó em uma NodeClass, é necessário criar uma entrada de acesso que associe o novo perfil do IAM de nó a esta política de acesso.

AmazonEKSBlockStoragePolicy

> **Note**
>
> Essa política é designada somente para perfis vinculados a serviços da AWS e não pode ser usado com perfis gerenciados pelo cliente.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSBlockStoragePolicy`

Esta política inclui permissões que permitem que o Amazon EKS gerencie recursos de eleição do líder e coordenação de operações de armazenamento:

- `coordination.k8s.io`: crie e gerencie objetos de concessão para a eleição do líder. Isso permite que os componentes de armazenamento do EKS coordenem suas atividades em todo o cluster por meio de um mecanismo de eleição do líder.

A política tem como escopo os recursos de concessão específicos usados pelos componentes de armazenamento do EKS para evitar o acesso conflitante a outros recursos de coordenação no cluster.

O Amazon EKS cria automaticamente uma entrada de acesso com essa política de acesso do perfil do IAM do cluster quando o Modo Automático está habilitado, garantindo que as permissões necessárias estejam em vigor para que o recurso de armazenamento em blocos funcione adequadamente.

AmazonEKSLoadBalancingPolicy

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSLoadBalancingPolicy`

Esta política inclui permissões que possibilitam ao Amazon EKS gerenciar recursos de eleição do líder para balanceamento de carga:

- `coordination.k8s.io`: crie e gerencie objetos de concessão para a eleição do líder. Isso permite que os componentes de balanceamento de carga do EKS coordenem atividades em várias réplicas ao eleger um líder.

A política tem como escopo específico os recursos de concessão de balanceamento de carga para garantir a coordenação adequada e, ao mesmo tempo, impedir o acesso a outros recursos de concessão no cluster.

O Amazon EKS cria automaticamente uma entrada de acesso com essa política de acesso para o perfil do IAM do cluster quando o Modo Automático está habilitado, garantindo que as permissões necessárias estejam em vigor para que o recurso de rede funcione adequadamente.

AmazonEKSNetworkingPolicy

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSNetworkingPolicy`

Esta política inclui permissões que possibilitam ao Amazon EKS gerenciar recursos de eleição do líder para redes:

- `coordination.k8s.io`: crie e gerencie objetos de concessão para a eleição do líder. Isso permite que os componentes de rede do EKS coordenem as atividades de alocação de endereços IP ao eleger um líder.

A política tem como escopo específico os recursos de concessão de rede para garantir a coordenação adequada e, ao mesmo tempo, impedir o acesso a outros recursos de concessão no cluster.

O Amazon EKS cria automaticamente uma entrada de acesso com essa política de acesso para o perfil do IAM do cluster quando o Modo Automático está habilitado, garantindo que as permissões necessárias estejam em vigor para que o recurso de rede funcione adequadamente.

AmazonEKSComputePolicy

> **Note**
>
> Essa política é designada somente para perfis vinculados a serviços da AWS e não pode ser usado com perfis gerenciados pelo cliente.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSComputePolicy`

Esta política inclui permissões que possibilitam ao Amazon EKS gerenciar recursos de eleição do líder para operações de computação:

- `coordination.k8s.io`: crie e gerencie objetos de concessão para a eleição do líder. Isso permite que os componentes de computação do EKS coordenem as atividades de escalabilidade de nós ao eleger um líder.

A política tem como escopo específico os recursos de concessão de gerenciamento de computação, permitindo ao mesmo tempo acesso básico de leitura (`get`, `watch`) a todos os recursos de concessão no cluster.

O Amazon EKS cria automaticamente uma entrada de acesso com essa política de acesso para o perfil do IAM do cluster quando o Modo Automático está habilitado, garantindo que as permissões necessárias estejam em vigor para que o recurso de rede funcione adequadamente.

AmazonEKSBlockStorageClusterPolicy

> **Note**
>
> Essa política é designada somente para perfis vinculados a serviços da AWS e não pode ser usado com perfis gerenciados pelo cliente.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSBlockStorageClusterPolicy`

Esta política concede as permissões necessárias para a capacidade de armazenamento em blocos do Modo Automático do Amazon EKS. Ela permite o gerenciamento eficiente dos recursos de armazenamento em blocos nos clusters do Amazon EKS. A política inclui as seguintes permissões:

Gerenciamento do driver CSI:

- Criar, ler, atualizar e excluir drivers CSI, especificamente para armazenamento em blocos.

Gerenciamento de volumes:

- Listar, observar, criar, atualizar, aplicar patch e excluir volumes persistentes.
- Listar, observar e atualizar reivindicações de volumes persistentes.
- Aplicar patch aos status de reivindicação de volumes persistentes.

Interação entre nós e pods:

- Ler as informações do nó e do pod.
- Gerenciar eventos relacionados às operações de armazenamento.

Classes e atributos de armazenamento:

- Leia classes de armazenamento e nós CSI.
- Leia as classes de atributos de volume.

Anexos de volume:

- Liste, observe e modifique anexos de volume e seus status.

Operações de snapshots:

- Gerencie snapshots de volume, conteúdo de snapshots e classes de snapshots.
- Lide com operações de snapshots de grupos de volumes e recursos relacionados.

Essa política foi projetada para ser compatível com o gerenciamento abrangente de armazenamento em blocos nos clusters do Amazon EKS em execução no Modo Automático. Ela combina permissões de várias operações, incluindo provisionamento, anexação, redimensionamento e criação de snapshots de volumes de armazenamento em blocos.

O Amazon EKS cria automaticamente uma entrada de acesso com essa política de acesso do perfil do IAM do cluster quando o Modo Automático está habilitado, garantindo que as permissões necessárias estejam em vigor para que o recurso de armazenamento em blocos funcione adequadamente.

AmazonEKSComputeClusterPolicy

> **Note**
>
> Essa política é designada somente para perfis vinculados a serviços da AWS e não pode ser usado com perfis gerenciados pelo cliente.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSComputeClusterPolicy`

Esta política concede as permissões necessárias para a capacidade de gerenciamento computacional do Modo Automático do Amazon EKS. Ela permite orquestração e escalabilidade eficientes de recursos computacionais nos clusters do Amazon EKS. A política inclui as seguintes permissões:

Gerenciamento de nós:

- Criar, ler, atualizar, excluir e gerenciar o status de NodePools e NodeClaims.
- Gerenciar NodeClasses, incluindo criação, modificação e exclusão.

Gerenciamento de recursos e programação:

- Acesso de leitura a pods, nós, volumes persistentes, reivindicações de volumes persistentes, controladores de replicação e namespaces.
- Acesso de leitura a classes de armazenamento, nós CSI e anexos de volumes.
- Listar e observar implantações, conjuntos daemon, conjuntos de réplicas e conjuntos com estado.
- Ler orçamentos de interrupção de pods.

Manipulação de eventos:

- Criar, ler e gerenciar eventos de clusters.

Desprovisionamento de nós e remoção de pods:

- Atualizar, aplicar patch e excluir nós.
- Criar remoções de pods e excluí-los quando necessário.

Gerenciamento de definição de recursos personalizados (CRD):

- Criar novos CRDs.
- Gerencie CRDs específicos relacionados ao gerenciamento de nós (NodeClasses, NodePools, NodeClaims e NodeDiagnostics).

Esta política foi projetada para ser compatível com o gerenciamento computacional abrangente nos clusters do Amazon EKS em execução no Modo Automático. Ela combina permissões para

várias operações, incluindo provisionamento de nós, programação, escalabilidade e otimização de recursos.

O Amazon EKS cria automaticamente uma entrada de acesso com essa política de acesso do perfil do IAM do cluster quando o Modo Automático está habilitado, garantindo que as permissões necessárias estejam em vigor para que a capacidade de gerenciamento de computação funcione adequadamente.

AmazonEKSLoadBalancingClusterPolicy

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSLoadBalancingClusterPolicy`

Esta política concede as permissões necessárias para a capacidade de balanceamento de carga do Modo Automático do Amazon EKS. Ele permite o gerenciamento e a configuração eficientes dos recursos de balanceamento de carga nos clusters do Amazon EKS. A política inclui as seguintes permissões:

Gerenciamento de eventos e recursos:

- Criar e aplicar patch a eventos.
- Acesso de leitura a pods, nós, endpoints e namespaces.
- Atualizar os status dos pods.

Gerenciamento de serviços e entradas:

- Gerenciamento completo dos serviços e seus status.
- Controle abrangente das entradas e seus status.
- Acesso de leitura às fatias de endpoint e às classes de entrada.

Vinculações de grupos de destino:

- Criar e modificar vinculações de grupos de destino e seus status.
- Acesso de leitura aos parâmetros da classe de entrada.

Gerenciamento de definição de recursos personalizados (CRD):

- Criar e ler todos os CRDs.

- Gerenciamento específico dos CRDs targetgroupbindings.eks.amazonaws.com e ingressclassparams.eks.amazonaws.com.

Configuração do webhook:

- Criar e ler configurações de webhook de mutação e validação.
- Gerenciar a configuração eks-load-balancing-webhook.

Esta política foi projetada para ser compatível com o gerenciamento abrangente do balanceamento de carga nos clusters do Amazon EKS em execução no Modo Automático. Ela combina permissões para várias operações, incluindo exposição de serviços, roteamento de entradas e integração com serviços de balanceamento de carga da AWS.

O Amazon EKS cria automaticamente uma entrada de acesso com essa política de acesso do perfil do IAM do cluster quando o Modo Automático está habilitado, garantindo que as permissões necessárias estejam em vigor para que a capacidade de balanceamento de carga funcione adequadamente.

AmazonEKSNetworkingClusterPolicy

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSNetworkingClusterPolicy`

AmazonEKSNetworkingClusterPolicy

Esta política concede as permissões necessárias para a capacidade de rede do Modo Automático do Amazon EKS. Ele permite o gerenciamento e a configuração eficientes dos recursos de rede nos clusters do Amazon EKS. A política inclui as seguintes permissões:

Gerenciamento de nós e pods:

- Ler o acesso aos NodeClasses e seus status.
- Ler o acesso aos NodeClaims e seus status.
- Acesso de leitura aos pods.

Gerenciamento de nós CNI:

- Permissões de CNINodes e seus status, incluindo criar, ler, atualizar, excluir e aplicar patch.

Gerenciamento de definição de recursos personalizados (CRD):

- Criar e ler todos os CRDs.
- Gerenciamento específico (atualização, aplicação de patch, exclusão) do CRD cninodes.eks.amazonaws.com.

Gerenciamento de eventos:

- Criar e aplicar patch a eventos.

Esta política foi projetada para ser compatível com o gerenciamento abrangente de redes nos clusters do Amazon EKS em execução no Modo Automático. Ela combina permissões para várias operações, incluindo configuração de rede de nós, gerenciamento de CNI (interface de rede de contêineres) e manipulação de recursos personalizados relacionados.

A política permite que os componentes de rede interajam com recursos relacionados a nós, gerenciem configurações de nós específicas da CNI e manipulem recursos personalizados essenciais para operações de rede no cluster.

O Amazon EKS cria automaticamente uma entrada de acesso com essa política de acesso para o perfil do IAM do cluster quando o Modo Automático está habilitado, garantindo que as permissões necessárias estejam em vigor para que o recurso de rede funcione adequadamente.

AmazonEKSHybridPolicy

> **Note**
>
> Essa política é designada somente para perfis vinculados a serviços da AWS e não pode ser usado com perfis gerenciados pelo cliente.

Esta política de acesso inclui permissões que concedem ao EKS acesso aos nós de um cluster. Quando associada a uma entrada de acesso, seu escopo de acesso geralmente é o cluster, em vez de um namespace do Kubernetes. Esta política é usada pelo Amazon EKS Hybrid Nodes.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSHybridPolicy`

Grupos de APIs Kubernetes	Kubernetes nonResourceURLs	Recursos Kubernetes	Verbos (permissões) Kubernetes
*		nodes	list

AmazonEKSClusterInsightsPolicy

> **Note**
>
> Essa política é designada somente para perfis vinculados a serviços da AWS e não pode ser usado com perfis gerenciados pelo cliente.

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterInsightsPolicy`

Esta política concede permissões somente leitura para a funcionalidade Amazon EKS Cluster Insights. A política inclui as seguintes permissões:

Acesso ao nó: - Listar e visualizar os nós do cluster - Ler as informações de status do nó

Acesso ao DaemonSet: - Acesso de leitura à configuração do kube-proxy

Esta política é gerenciada automaticamente pelo serviço do EKS para o Cluster Insights. Para obter mais informações, consulte the section called “Insights de cluster”.

AWSBackupFullAccessPolicyForBackup

ARN – `arn:aws:eks::aws:cluster-access-policy/AWSBackupFullAccessPolicyForBackup`

AWSBackupFullAccessPolicyForBackup

Esta política concede as permissões necessárias para que o AWS Backup gerencie e crie backups do cluster do EKS. Esta política inclui as seguintes permissões:

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
*	*	list, get

AWSBackupFullAccessPolicyForRestore

ARN – `arn:aws:eks::aws:cluster-access-policy/AWSBackupFullAccessPolicyForRestore`

AWSBackupFullAccessPolicyForRestore

Esta política concede as permissões necessárias para que o AWS Backup gerencie e restaure backups do cluster do EKS. Esta política inclui as seguintes permissões:

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
`*`	`*`	`list`, `get`, `create`

AmazonEKSACKPolicy

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSACKPolicy`

Esta política concede as permissões necessárias para que a funcionalidade AWS Controllers for Kubernetes (ACK) gerencie recursos da AWS usando o Kubernetes. A política inclui as seguintes permissões:

Gerenciamento de recursos personalizados do ACK:

- Permite acesso completo a todos os recursos personalizados dos serviços do ACK em mais de 50 serviços da AWS, como S3, RDS, DynamoDB, Lambda, EC2, entre outros.
- Possibilita a criação, a leitura, a atualização e a exclusão de definições de recursos personalizados do ACK.

Acesso a namespaces:

- Permite acesso de leitura a namespaces para organização de recursos.

Eleição do líder:

- Possibilita a criação e a leitura de concessões de coordenação para eleição de líder.
- Realiza a atualização e a exclusão de concessões específicas do controlador de serviço do ACK.

Gerenciamento de eventos:

- Possibilita a criação e a aplicação de patches em eventos relacionados às operações do ACK.

Esta política foi projetada para oferecer suporte ao gerenciamento abrangente de recursos da AWS por meio das APIs do Kubernetes. O Amazon EKS cria automaticamente uma entrada de acesso com a presente política de acesso para o perfil do IAM da funcionalidade que você fornece quando a funcionalidade do ACK é criada.

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
	`namespaces`	`get`, `watch`, `list`
`services.k8s.aws` , `acm.services.k8s.aws` , `acmpca.services.k8s.aws` , `apigateway.services.k8s.aws` , `apigatewayv2.services.k8s.aws` , `applicationautoscaling.services.k8s.aws` , `athena.services.k8s.aws` , `bedrock.services.k8s.aws` , `bedrockagent.services.k8s.aws` , `bedrockagentcorecontrol.services.k8s.aws` , `cloudfront.services.k8s.aws` , `cloudtrail.services.k8s.aws` , `cloudwatch.services.k8s.aws` , `cloudwatchlogs.services.k8s`	*	*

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
.aws , codeartifact.services.k8s.aws , cognitoidentityprovider.services.k8s.aws , documentdb.services.k8s.aws , dynamodb.services.k8s.aws , ec2.services.k8s.aws , ecr.services.k8s.aws , ecrpublic.services.k8s.aws , ecs.services.k8s.aws , efs.services.k8s.aws , eks.services.k8s.aws , elasticache.services.k8s.aws , elbv2.services.k8s.aws , emrcontainers.services.k8s.aws , eventbridge.services.k8s.aws , iam.services.k8s.aws , kafka.services.k8s.aws , keyspaces.services.k8s.aws , kinesis.services.k8s.aws , kms.services.k8s.aws , lambda.services.k8s.aws , memorydb.services.k8s.aws , mq.services.k8s.aw		

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
s , networkfirewall.services.k8s.aws , opensearchservice.services.k8s.aws , organizations.services.k8s.aws , pipes.services.k8s.aws , prometheusservice.services.k8s.aws , ram.services.k8s.aws , rds.services.k8s.aws , recyclebin.services.k8s.aws , route53.services.k8s.aws , route53resolver.services.k8s.aws , s3.services.k8s.aws , s3control.services.k8s.aws , sagemaker.services.k8s.aws , secretsmanager.services.k8s.aws , ses.services.k8s.aws , sfn.services.k8s.aws , sns.services.k8s.aws , sqs.services.k8s.aws , ssm.services.k8s.aws , wafv2.services.k8s.aws		
coordination.k8s.io	leases	create, get, list, watch

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
`coordination.k8s.io`	`leases` (somente concessões específicas do controlador de serviços do ACK)	`delete`, `update`, `patch`
	`events`	`create`, `patch`
`apiextensions.k8s.io`	`customresourcedefinitions`	`*`

AmazonEKSArgoCDClusterPolicy

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSArgoCDClusterPolicy`

Esta política concede permissões em nível de cluster necessárias para que a funcionalidade do Argo CD descubra recursos e gerencie objetos com escopo de cluster. A política inclui as seguintes permissões:

Gerenciamento de namespaces:

- Permite a criação, a leitura, a atualização e a exclusão de namespaces para o gerenciamento de namespaces de aplicações.

Gerenciamento da definição de recursos personalizados:

- Gerenciamento de CRDs específicas do Argo CD (Applications, AppProjects e ApplicationSets).

Descoberta de API:

- Possibilita acesso de leitura aos endpoints da API do Kubernetes para a descoberta de recursos.

Esta política foi projetada para fornecer suporte às operações do Argo CD em nível de cluster, incluindo o gerenciamento de namespaces e a instalação de CRDs. O Amazon EKS cria automaticamente uma entrada de acesso com a presente política de acesso para o perfil do IAM da funcionalidade que você fornece quando a funcionalidade do Argo CD é criada.

Grupos de APIs Kubernetes	Kubernetes nonResourceURLs	Recursos Kubernetes	Verbos (permissões) Kubernetes
		`namespaces`	`create`, `get`, `update`, `patch`, `delete`
`apiextensions.k8s.io`		`customresourcedefinitions`	`create`
`apiextensions.k8s.io`		`customresourcedefinitions` (Somente CRDs do Argo CD)	`get`, `update`, `patch`, `delete`
	`/api`, `/api/*`, `/apis`, `/apis/*`		`get`

AmazonEKSArgoCDPolicy

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSArgoCDPolicy`

Esta política concede permissões em nível de namespace necessárias para que a funcionalidade do Argo CD realize a implantação e o gerenciamento de aplicações. A política inclui as seguintes permissões:

Gerenciamento de segredos:

- Permite acesso completo aos segredos usados para as credenciais do Git e aos segredos do cluster.

Acesso ao ConfigMap:

- Possibilita acesso de leitura a ConfigMaps para o envio de alertas caso os clientes tentem usar ConfigMaps do Argo CD que não são compatíveis.

Gerenciamento de eventos:

- Permite acesso de leitura e criação de eventos com o objetivo de monitorar o ciclo de vida das aplicações.

Gerenciamento de recursos do Argo CD:

- Possibilita acesso completo a Applications, ApplicationSets e AppProjects.
- Permite gerenciamento de finalizadores e do status dos recursos do Argo CD.

Esta política foi projetada para fornecer suporte às operações do Argo CD em nível de namespace, incluindo a implantação e o gerenciamento de aplicações. O Amazon EKS cria automaticamente uma entrada de acesso com a presente política de acesso para o perfil do IAM da funcionalidade que você fornece quando a funcionalidade do Argo CD é criada, com escopo no namespace do Argo CD.

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
	`secrets`	`*`
	`configmaps`	`get`, `list`, `watch`
	`events`	`get`, `list`, `watch`, `patch`, `create`
`argoproj.io`	`applications` , `applications/finalizers` , `applications/status` , `applicationsets` , `applicationsets/finalizers` , `applicationsets/status` , `appprojects` , `appprojects/finalizers` , `appprojects/status`	`*`

AmazonEKSKROPolicy

ARN – `arn:aws:eks::aws:cluster-access-policy/AmazonEKSKROPolicy`

Esta política concede as permissões necessárias para que a funcionalidade do kro (Kube Resource Orchestrator) crie e gerencie APIs personalizadas do Kubernetes. A política inclui as seguintes permissões:

Gerenciamento de recursos do kro:

- Permite acesso completo a todos os recursos do kro, abrangendo ResourceGraphDefinitions e instâncias de recursos personalizados.

Gerenciamento da definição de recursos personalizados:

- Possibilita a criação, a leitura, a atualização e a exclusão de CRDs para APIs personalizadas e definidas por ResourceGraphDefinitions.

Eleição do líder:

- Possibilita a criação e a leitura de concessões de coordenação para eleição de líder.
- Permite a atualização e a exclusão da concessão do controlador do kro.

Gerenciamento de eventos:

- Possibilita a criação e a aplicação de patches em eventos relacionados às operações do kro.

Esta política foi projetada para oferecer suporte à composição abrangente de recursos e ao gerenciamento de APIs personalizadas por meio do kro. O Amazon EKS cria automaticamente uma entrada de acesso com a presente política de acesso para o perfil do IAM da funcionalidade que você fornece quando a funcionalidade do kro é criada.

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
`kro.run`	*	*

Grupos de APIs Kubernetes	Recursos Kubernetes	Verbos (permissões) Kubernetes
`apiextensions.k8s.io`	`customresourcedefinitions`	`*`
`coordination.k8s.io`	`leases`	`create`, `get`, `list`, `watch`
`coordination.k8s.io`	`leases` (somente para concessão de controladores do kro)	`delete`, `update`, `patch`
	`events`	`create`, `patch`

Atualizações de política de acesso

Visualize detalhes sobre as atualizações das políticas de acesso, desde que elas foram introduzidas. Para receber alertas automáticos sobre alterações nesta página, assine o feed RSS em Histórico do documento.

Alteração	Descrição	Data
Adição de políticas para as funcionalidades do EKS	Publicação das políticas `AmazonEKSACKPolicy`, `AmazonEKSArgoCDClusterPolicy`, `AmazonEKSArgoCDPolicy` e `AmazonEKSKROPolicy` para o gerenciamento das funcionalidades do EKS	22 de novembro de 2025
Adicionar `AmazonEKSSecretReaderPolicy`	Adição de uma nova política para acesso somente leitura a segredos	6 de novembro de 2025

Alteração	Descrição	Data
Adicionar política para o EKS Cluster Insights	Publicar `AmazonEKSClusterInsightsPolicy`	2 de dezembro de 2024
Adicionar políticas do Amazon EKS Hybrid	Publicar `AmazonEKSHybridPolicy`	2 de dezembro de 2024
Adicionar políticas do Modo Automático do Amazon EKS	Estas políticas de acesso dão ao perfil do IAM do cluster e ao perfil do IAM do nó permissão para chamar as APIs do Kubernetes. A AWS as usa para automatizar tarefas de rotina de recursos de armazenamento, computação e rede.	2 de dezembro de 2024
Adicionar `AmazonEKSAdminViewPolicy`	Adicione uma nova política para acesso de visualização expandido, incluindo recursos como Segredos.	23 de abril de 2024
Políticas de acesso introduzidas.	O Amazon EKS introduziu políticas de acesso.	29 de maio de 2023

Alterar o modo de autenticação para usar entradas de acesso

Para começar a usar entradas de acesso, você deve alterar o modo de autenticação do cluster para os modos `API_AND_CONFIG_MAP` ou `API`. Essa ação adiciona a API para entradas de acesso.

Console do AWS

1. Abra o console do Amazon EKS.
2. Escolha o nome do cluster em que você deseja criar uma entrada de acesso.
3. Escolha a guia Acesso.

4. O Modo de autenticação mostra o modo de autenticação atual do cluster. Se o modo indicar API do EKS, já será possível adicionar entradas de acesso e pular as etapas restantes.
5. Selecione Gerenciar acesso.
6. Em Modo de autenticação de cluster, selecione um modo com a API do EKS. Observe que você não pode alterar o modo de autenticação de volta para um modo que remova as entradas de acesso e a API do EKS.
7. Escolha Salvar alterações. O Amazon EKS começa a atualizar o cluster, o status do cluster muda para Updating e a alteração é registrada na guia Histórico de atualizações.
8. Aguarde até que o status do cluster retorne para Active. Quando o cluster estiver Active, será possível seguir as etapas em the section called “Criar entradas de acesso” para adicionar acesso ao cluster das entidades principais do IAM.

AWS CLI

1. Instale a AWS CLI, conforme descrito em Instalar no Guia do usuário da AWS Command Line Interface.
2. Execute o comando a seguir. Substitua *my-cluster* pelo nome do cluster. Se você quiser deshabilitar o método `ConfigMap` permanentemente, substitua `API_AND_CONFIG_MAP` por `API`.

   O Amazon EKS começa a atualizar o cluster, o status do cluster muda para UPDATING e a alteração é registrada em aws eks list-updates .

   ```
   aws eks update-cluster-config --name my-cluster --access-config
    authenticationMode=API_AND_CONFIG_MAP
   ```
3. Aguarde até que o status do cluster retorne para Active. Quando o cluster estiver Active, será possível seguir as etapas em the section called “Criar entradas de acesso” para adicionar acesso ao cluster das entidades principais do IAM.

Versão da plataforma necessária

Para usar as entradas de acesso, o cluster dove ter uma versão da plataforma igual ou posterior à versão listada na tabela apresentada a seguir ou uma versão do Kubernetes posterior às versões listadas na tabela. Caso sua versão do Kubernetes não esteja listada, todas as versões da plataforma serão compatíveis com as entradas de acesso.

Versão do Kubernetes	Versão da plataforma
Não listado	Todos compatíveis
1.30	eks.2
1.29	eks.1
1.28	eks.6

Para obter mais informações, consulte Versões da plataforma.

Criar entradas de acesso

Antes de criar entradas de acesso, considere o seguinte:

- Um modo de autenticação definido corretamente. Consulte the section called “Modo de autenticação”.
- Uma entrada de acesso inclui o nome do recurso da Amazon (ARN) de uma, e somente uma, entidade principal do IAM existente. Uma entidade principal do IAM não pode ser incluída em mais de uma entrada de acesso. Considerações adicionais para o ARN especificado por você:
  - As práticas recomendadas do IAM sugerem que você acesse o cluster usando perfis do IAM que têm credenciais de curto prazo, em vez de usuários do IAM que têm credenciais de longo prazo. Para obter mais informações, consulte Exigir que os usuários humanos usem a federação com um provedor de identidade para acessar a AWS usando credenciais temporárias no Guia do usuário do IAM.
  - Se o ARN for para um perfil do IAM, ele pode incluir um caminho. Os ARNs nas entradas `ConfigMap aws-auth` não podem incluir um caminho. Por exemplo, seu ARN pode ser `arn:aws:iam::<111122223333>:role/<development/apps/my-role>` ou `arn:aws:iam::<111122223333>:role/<my-role>`.
  - Se o tipo da entrada de acesso for diferente de `STANDARD` (consulte a próxima consideração sobre tipos), o ARN deverá estar na mesma conta AWS em que seu cluster está. Se o tipo for `STANDARD`, o ARN poderá estar na mesma conta AWS ou em uma conta diferente da conta em que seu cluster está.
  - Não é possível alterar a entidade principal do IAM depois que a entrada de acesso é criada.

  - Se você alguma vez você excluir a entidade principal do IAM com esse ARN, a entrada de acesso não será excluída automaticamente. Recomendamos que você exclua a entrada de acesso com um ARN para uma entidade principal do IAM que você exclua. Se você não excluir a entrada de acesso e nunca recriar a entidade principal do IAM, mesmo que ela tenha o mesmo ARN, a entrada de acesso não funcionará. Isso ocorre porque, embora o ARN seja o mesmo para a entidade principal do IAM recriada, o endereço `roleID` ou `userID` (você pode ver isso com o comando `aws sts get-caller-identity` da AWS CLI) é diferente para a entidade principal do IAM recriada e para a entidade IAM original. Mesmo que você não veja o `roleID` ou `userID` da entidade principal do IAM para uma entrada de acesso, o Amazon EKS o armazena com a entrada de acesso.
- Cada entrada de acesso tem um tipo. O tipo da entrada de acesso depende do tipo de recurso ao qual está associada, e não define as permissões. Se você não especificar um tipo, o Amazon EKS definirá automaticamente o tipo como `STANDARD`.
  - `EC2_LINUX`: para um perfil do IAM usado com nós autogerenciados do Linux ou Bottlerocket
  - `EC2_WINDOWS`: para um perfil do IAM usado com nós autogerenciados do Windows
  - `FARGATE_LINUX`: para um perfil do IAM usado com o AWS Fargate (Fargate)
  - `HYBRID_LINUX`: para um perfil do IAM usado com nós híbridos
  - `STANDARD`: tipo padrão se nenhum for especificado
  - `EC2`: para classes de nós personalizadas do Modo Automático do EKS Para obter mais informações, consulte the section called “Criar entrada de acesso de classe de nós”.
  - Não é possível alterar o tipo depois que a entrada de acesso é criada.
- Não é necessário criar uma entrada de acesso para um perfil do IAM usado em um grupo de nós gerenciados ou em um perfil do Fargate. O EKS criará entradas de acesso (se habilitado), ou atualizará o mapa de configuração de autenticação (caso as entradas de acesso não estejam disponíveis).
- Se o tipo da entrada de acesso for `STANDARD`, será possível especificar um nome de usuário para a entrada de acesso. Se você não especificar um valor para o nome de usuário, o Amazon EKS definirá um dos valores a seguir para você, dependendo do tipo de entrada de acesso e de se a entidade principal do IAM que você especificou é um perfil do IAM ou um usuário do IAM. A menos que você tenha um motivo específico para determinar seu próprio nome de usuário, recomendamos que não especifique um e deixe o Amazon EKS gerá-lo automaticamente para você. Se você especificar seu próprio nome de usuário:
  - Não pode começar com `system:`, `eks:`, `aws:`, `amazon:` ou `iam:`.

- Se o nome de usuário for para um perfil do IAM, recomendamos que você adicione `{{SessionName}}` ou `{{SessionNameRaw}}` ao final do seu nome de usuário. Se você adicionar `{{SessionName}}` ou `{{SessionNameRaw}}` ao seu nome de usuário, ele deverá incluir dois pontos antes de {{SessionName}}. Quando esse perfil é assumido, o nome da sessão do AWS STS especificado ao assumir o perfil é automaticamente transmitido para o cluster e aparecerá nos logs do CloudTrail. Por exemplo, não é possível ter um nome de usuário `john{{SessionName}}`. O nome de usuário teria que ser `:john{{SessionName}}` ou `jo:hn{{SessionName}}`. Os dois pontos precisam vir antes de `{{SessionName}}`. O nome de usuário gerado pelo Amazon EKS na tabela a seguir inclui um ARN. Como um ARN inclui dois pontos, ele atende a esse requisito. Os dois pontos não serão necessários se você não incluir `{{SessionName}}` no seu nome de usuário. Vale ressaltar que em `{{SessionName}}`, o caractere especial “@” é substituído por “-” no nome da sessão. `{{SessionNameRaw}}` mantém todos os caracteres especiais no nome da sessão.

Tipo de entidade principal do IAM	Tipo	Valor do nome de usuário que o Amazon EKS define automaticamente
Usuário	`STANDARD`	O ARN do usuário. Example: `arn:aws:iam::<111122223333>:user/<my-user>`

Tipo de entidade principal do IAM	Tipo	Valor do nome de usuário que o Amazon EKS define automaticamente
Função	`STANDARD`	O STS ARN da função quando ela é assumida. O Amazon EKS anexa `{{SessionName}}` à função. Example: `arn:aws:sts::<111122223333>:assumed-role/<my-role>/{{SessionName}}` Se o ARN da função que você especificou contiver um caminho, o Amazon EKS o removerá no nome de usuário gerado.
Função	`EC2_LINUX` ou `EC2_Windows`	`system:node:{{EC2PrivateDNSName}}`
Função	`FARGATE_LINUX`	`system:node:{{SessionName}}`
Função	`HYBRID_LINUX`	`system:node:{{SessionName}}`

Você poderá alterar o nome de usuário depois que a entrada de acesso for criada.

- Se o tipo de entrada de acesso for `STANDARD` e você quiser usar a autorização RBAC do Kubernetes RBAC, poderá adicionar um ou mais nomes de grupos à entrada de acesso. Depois de criar uma entrada de acesso, você pode adicionar e remover nomes de grupos. Para que a entidade principal do IAM tenha acesso aos objetos do Kubernetes em seu cluster, você deve criar e gerenciar objetos de autorização baseada em perfil (RBAC) do Kubernetes. Crie objetos

`RoleBinding` ou `ClusterRoleBinding` do Kubernetes no cluster que especifiquem o nome do grupo como um `subject` para `kind: Group`. O Kubernetes autoriza o acesso da entidade principal do IAM a quaisquer objetos do cluster que você tenha especificado em um objeto `Role` ou `ClusterRole` do Kubernetes, que você também tenha especificado na `roleRef` da associação. Se você especificar nomes de grupo, convém familiarizar-se com os objetos de autorização com base em perfil (RBAC) do Kubernetes. Para mais informações, consulte Usando a autorização de RBAC na documentação do Kubernetes.

> **Important**
>
> O Amazon EKS não confirma que nenhum objeto RBAC do Kubernetes existente no seu cluster inclua algum dos nomes de grupos que você especificar. Por exemplo, se você criar uma entrada de acesso para um grupo que ainda não existe, o EKS criará o grupo em vez de retornar um erro.

Em vez de, ou além de, o Kubernetes autorizar o acesso da entidade principal do IAM aos objetos do Kubernetes no cluster, você pode associar as políticas de acesso do Amazon EKS a uma entrada de acesso. O Amazon EKS autoriza as entidades principais do IAM a acessar objetos do Kubernetes no cluster com as permissões na política de acesso. Você pode definir o escopo das permissões de uma política de acesso aos namespaces do Kubernetes que você especificar. O uso de políticas de acesso não exige que você gerencie objetos RBAC do Kubernetes. Para obter mais informações, consulte the section called “Associar políticas de acesso”.

- Se você criar uma entrada de acesso com o tipo `EC2_LINUX` ou `EC2_Windows`, a entidade principal do IAM que criou a entrada de acesso deverá ter a permissão `iam:PassRole`. Para obter mais informações, consulte Conceder permissões ao usuário para passar uma função a um produto da AWS no Guia do usuário do IAM.
- Semelhante ao comportamento padrão do IAM, a criação e as atualizações de entradas de acesso acabam sendo consistentes e podem levar alguns segundos para serem efetivadas após o retorno bem-sucedido da chamada inicial de API. Suas aplicações devem ser projetadas para levar em conta esses possíveis atrasos. Recomendamos que você não inclua criações ou atualizações de entrada de acesso nos caminhos de código críticos de alta disponibilidade da sua aplicação. Em vez disso, faça alterações do em uma rotina de inicialização ou de configuração separada que você executa com menos frequência. Além disso, certifique-se de verificar se as alterações foram propagadas antes que os fluxos de trabalho de produção dependam delas.

- As entradas de acesso não oferecem suporte para perfis vinculados ao serviço. Não é possível criar entradas de acesso nas quais o ARN da entidade principal corresponde a um perfil vinculado ao serviço. É possível identificar perfis vinculados ao serviço pelo ARN deles, que está no formato `arn:aws:iam::*:role/aws-service-role/*`.

Você pode criar uma entrada de acesso usando o Console de gerenciamento da AWS ou a AWS CLI.

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Escolha o nome do cluster em que você deseja criar uma entrada de acesso.
3. Escolha a guia Acesso.
4. Selecione Criar entrada de acesso.
5. Para a entidade principal do IAM, selecione um perfil ou usuário do IAM existente. As práticas recomendadas do IAM sugerem que você acesse o cluster usando perfis do IAM que têm credenciais de curto prazo, em vez de usuários do IAM que têm credenciais de longo prazo. Para obter mais informações, consulte Exigir que os usuários humanos usem a federação com um provedor de identidade para acessar a AWS usando credenciais temporárias no Guia do usuário do IAM.
6. Em Tipo, se a entrada de acesso for para a função de nó usada para nós autogerenciados do Amazon EC2, selecione EC2 Linux ou EC2 Windows. Caso contrário, aceite o padrão (Padrão).
7. Se o Tipo escolhido for Padrão e você quiser especificar um nome de usuário, insira o nome de usuário.
8. Se o Tipo escolhido for Padrão e você quiser usar a autorização RBAC do Kubernetes para a entidade principal do IAM, especifique um ou mais nomes para Grupos. Se você não especificar nenhum nome de grupo e quiser usar a autorização do Amazon EKS, poderá associar uma política de acesso em uma etapa posterior ou após a criação da entrada de acesso.
9. (Opcional) Para Tags, atribua rótulos à entrada de acesso. Por exemplo, para facilitar a localização de todos os recursos com a mesma tag.
10. Escolha Próximo.
11. Na página Adicionar política de acesso, se o tipo escolhido for Padrão e você quiser que o Amazon EKS autorize a entidade principal do IAM a ter permissões para os objetos do Kubernetes em seu cluster, execute as etapas a seguir. Caso contrário, escolha Next.

a. Em Nome da política, escolha uma política de acesso. Você não pode ver as permissões das políticas de acesso, mas elas incluem permissões semelhantes às dos objetos do `ClusterRole` do Kubernetes voltados para o usuário. Para obter mais informações, consulte User-facing roles na documentação do Kubernetes.

b. Escolha uma das seguintes opções:

- Cluster: escolha esta opção se quiser que o Amazon EKS autorize a entidade principal do IAM a ter as permissões na política de acesso para todos os objetos do Kubernetes no cluster.
- Namespace do Kubernetes: escolha esta opção se quiser que o Amazon EKS autorize a entidade principal do IAM a ter as permissões na política de acesso para todos os objetos do Kubernetes em um namespace do Kubernetes específico no cluster. Em Namespace, insira o nome do namespace do Kubernetes no cluster. Se quiser adicionar mais namespaces, escolha Adicionar novo namespace e insira o nome do namespace.

c. Se você quiser adicionar mais políticas, escolha Adicionar política. Você pode definir o escopo de cada política de forma diferente, mas cada política só pode ser adicionada uma vez.

d. Escolha Próximo.

12 Revise a configuração da entrada de acesso. Se algo parecer errado, escolha Anterior para voltar às etapas anteriores e corrigir o erro. Se a configuração estiver correta, escolha Criar.

AWS CLI

1. Instale a AWS CLI, conforme descrito em Instalar, no Guia do usuário da AWS Command Line Interface.
2. Para criar uma entrada de acesso Você pode usar qualquer um dos exemplos a seguir para criar entradas de acesso:
   - Crie uma entrada de acesso para um grupo de nós autogerenciado do Amazon EC2 Linux. Substitua *`my-cluster`* pelo nome do cluster, *`111122223333`* pelo ID da conta da AWS e *`EKS-my-cluster-self-managed-ng-1`* pelo nome do perfil do IAM do nó. Caso seu grupo de nós seja um grupo de nós do Windows, substitua *`EC2_LINUX`* por `EC2_Windows`.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1 --type EC2_LINUX
```

Você não pode usar a opção `--kubernetes-groups` ao especificar um tipo diferente de `STANDARD`. Você não pode associar uma política de acesso a essa entrada de acesso porque seu tipo é um valor diferente de `STANDARD`.

- Crie uma entrada de acesso que permita um perfil do IAM que não é usado para um grupo de nós autogerenciados do Amazon EC2, com o qual você deseja que o Kubernetes autorize o acesso ao seu cluster. Substitua *`my-cluster`* pelo nome do cluster, *`111122223333`* pela ID da conta AWS e *`my-role`* pelo nome do perfil do IAM. Substitua *`Visualizadores`* pelo nome de um grupo que você especificou em um objeto `RoleBinding` ou `ClusterRoleBinding` do Kubernetes no cluster.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role --type STANDARD --user Viewers --
kubernetes-groups Viewers
```

- Crie uma entrada de acesso que permita que um usuário do IAM se autentique no seu cluster. Esse exemplo é fornecido porque isso é possível, embora as práticas recomendadas do IAM sugiram acessar seu cluster usando perfis do IAM que têm credenciais de curto prazo, em vez de usuários do IAM que têm credenciais de longo prazo. Para obter mais informações, consulte Exigir que os usuários humanos usem a federação com um provedor de identidade para acessar a AWS usando credenciais temporárias no Guia do usuário do IAM.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:user/my-user --type STANDARD --username my-user
```

Se quiser que esse usuário tenha mais acesso ao cluster do que as permissões nos perfis de descoberta da API do Kubernetes, você precisará associar uma política de acesso à entrada de acesso, já que a opção `--kubernetes-groups` não é usada. Para obter mais informações, consulte the section called “Associar políticas de acesso” e API discovery roles na documentação do Kubernetes.

Atualizar entradas de acesso

Você pode atualizar uma entrada de acesso usando o Console de gerenciamento da AWS ou a AWS CLI.

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Escolha o nome do cluster em que você deseja criar uma entrada de acesso.
3. Escolha a guia Acesso.
4. Selecione a entrada de acesso que você deseja atualizar.
5. Escolha Editar.
6. Em Nome de usuário, você pode alterar o valor existente.
7. Em Grupos, você pode remover nomes de grupos existentes ou adicionar novos nomes de grupos. Se os seguintes nomes de grupos existirem, não os remova: system:nodes ou system:bootstrappers. A remoção desses grupos pode fazer com que seu cluster funcione incorretamente. Se você não especificar nenhum nome de grupo e quiser usar a autorização do Amazon EKS, associe uma política de acesso em uma etapa posterior.
8. Em Tags, você pode atribuir rótulos à entrada de acesso. Por exemplo, para facilitar a localização de todos os recursos com a mesma tag. Você também pode remover as tags existentes.
9. Escolha Salvar alterações.
10. Se você quiser associar uma política de acesso à entrada, consulte the section called “Associar políticas de acesso”.

AWS CLI

1. Instale a AWS CLI, conforme descrito em Instalar, no Guia do usuário da AWS Command Line Interface.
2. Para atualizar uma entrada de acesso, substitua *my-cluster* pelo nome do seu cluster, *111122223333* pelo ID da sua conta AWS e *EKS-my-cluster-my-namespace-Viewers* pelo nome de um perfil do IAM.

```
aws eks update-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --kubernetes-
groups Viewers
```

Você não poderá usar a opção `--kubernetes-groups` se o tipo da entrada de acesso for um valor diferente de `STANDARD`. Você também não pode associar uma política de acesso a uma entrada de acesso com um tipo diferente de `STANDARD`.

Excluir entradas de acesso

Se você descobrir que excluiu uma entrada de acesso por engano, sempre poderá recriá-la. Se a entrada de acesso que você está excluindo estiver associada a alguma política de acesso, as associações serão excluídas automaticamente. Você não precisa desassociar as políticas de acesso de uma entrada de acesso antes de excluir a entrada de acesso.

Você pode excluir uma entrada de acesso usando o Console de gerenciamento da AWS ou a AWS CLI.

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Escolha o nome do cluster do qual você deseja excluir uma entrada de acesso.
3. Escolha a guia Acesso.
4. Na lista Entradas de acesso, escolha a entrada de acesso que você deseja excluir.
5. Escolha Delete.
6. Na caixa de diálogo de confirmação, escolha Excluir.

AWS CLI

1. Instale a AWS CLI, conforme descrito em Instalar, no Guia do usuário da AWS Command Line Interface.
2. Para excluir uma entrada de acesso Substitua *my-cluster* pelo nome do cluster, *111122223333* pela ID da conta AWS e *my-role* pelo nome do perfil do IAM que você não deseja mais que tenha acesso ao cluster.

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

Definir um nome de usuário personalizado para entradas de acesso ao EKS

Ao criar entradas de acesso para o Amazon EKS, é possível usar o nome de usuário gerado automaticamente ou especificar um personalizado. Esta página explica as duas alternativas e orienta sobre a configuração de um nome de usuário personalizado.

Visão geral

O nome de usuário em uma entrada de acesso é usado para identificar a entidade principal do IAM em logs e trilhas de auditoria do Kubernetes. Por padrão, o Amazon EKS gera um nome de usuário com base no ARN da identidade do IAM, mas também é possível especificar um nome de usuário personalizado, se necessário.

Geração de nome de usuário padrão

Se você não especificar um valor como nome de usuário, o Amazon EKS definirá um de maneira automática com base na identidade do IAM:

- Para usuários do IAM:
  - O EKS define o nome de usuário do Kubernetes como o ARN do usuário do IAM
  - Exemplo:

    ```
    {arn-aws}iam::<111122223333>:user/<my-user>
    ```
- Para perfis do IAM:
  - O EKS define o nome de usuário do Kubernetes como o ARN do perfil do IAM
  - O STS ARN do perfil quando ele é assumido. O Amazon EKS anexa `{{SessionName}}` ao perfil. Se o ARN do perfil que você especificou contiver um caminho, o Amazon EKS o removerá no nome de usuário gerado.
  - Exemplo:

    ```
    {arn-aws}sts::<111122223333>:assumed-role/<my-role>/{{SessionName}}
    ```

A menos que você tenha um motivo específico para determinar seu próprio nome de usuário, recomendamos que não especifique um e deixe o Amazon EKS gerá-lo automaticamente.

Configuração de um nome de usuário personalizado

Ao criar uma entrada de acesso, você pode especificar um nome de usuário personalizado usando o parâmetro `--username`:

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD --username <custom-username>
```

Requisitos para nomes de usuário personalizados

Se você especificar um nome de usuário personalizado:

- O nome de usuário não pode começar com `system:`, `eks:`, `aws:`, `amazon:`, ou `iam:`.
- Se o nome de usuário for para um perfil do IAM, recomendamos que você adicione `{{SessionName}}` ou `{{SessionNameRaw}}` ao final do seu nome de usuário.
  - Se você adicionar `{{SessionName}}` ou `{{SessionNameRaw}}` ao seu nome de usuário, ele deverá incluir dois pontos antes de {{SessionName}}.

Criar uma entrada de acesso para um usuário ou perfil do IAM usando uma política de acesso e a AWS CLI

Crie entradas de acesso do Amazon EKS que usam políticas de acesso gerenciadas pela AWS do EKS para conceder permissões padronizadas às identidades do IAM para acessar e gerenciar clusters do Kubernetes.

Visão geral

As entradas de acesso no Amazon EKS definem como as identidades do IAM (usuários e perfis) acessam e interagem com seus clusters do Kubernetes. Ao criar entradas de acesso com as políticas de acesso do EKS, você pode:

- Conceda permissão a usuários ou funções específicas do IAM para acessar seu cluster do EKS
- Controle as permissões usando políticas de acesso do EKS gerenciadas pela AWS que fornecem conjuntos de permissões padronizados e predefinidos
- Permissões de escopo para namespaces específicos ou em todo o cluster
- Simplifique o gerenciamento de acesso sem modificar o `aws-auth` ConfigMap ou criar recursos RBAC do Kubernetes
- Use uma abordagem integrada com a AWS para o controle de acesso do Kubernetes que abrange casos de uso comuns, mantendo as melhores práticas de segurança

Essa abordagem é recomendada para a maioria dos casos de uso porque fornece permissões gerenciadas pela AWS e padronizadas sem exigir a configuração manual do Kubernetes RBAC. As políticas de acesso do EKS eliminam a necessidade de configurar manualmente os recursos RBAC do Kubernetes e oferecem conjuntos de permissões predefinidos que abrangem casos de uso comuns.

Pré-requisitos

- O modo de autenticação do seu cluster deve ser configurado para permitir entradas de acesso. Para obter mais informações, consulte the section called “Modo de autenticação”.
- Instale e configure a AWS CLI, conforme descrito em Instalar, no Guia do usuário da AWS Command Line Interface.

Etapa 1: definir a entrada de acesso

1. Encontre o ARN da identidade do IAM, como um usuário ou um perfil, ao qual você deseja conceder permissões.
   - Cada identidade do IAM pode ter somente uma entrada de acesso ao EKS.
2. Determine se você deseja que as permissões da política de acesso do Amazon EKS se apliquem somente a um namespace específico do Kubernetes ou a todo o cluster.
   - Se você quiser limitar as permissões a um namespace específico, anote o nome do namespace.
3. Selecione a política de acesso do EKS que você deseja para a identidade do IAM. Essa política fornece permissões no cluster. Anote o ARN da política.
   - Para obter uma lista de políticas, consulte as políticas de acesso disponíveis.
4. Determine se o nome de usuário gerado automaticamente é apropriado para a entrada de acesso ou se você precisa especificar manualmente um nome de usuário.
   - A AWS gera automaticamente esse valor com base na identidade do IAM. Você pode definir um nome de usuário personalizado. Isso é visível nos logs do Kubernetes.
   - Para obter mais informações, consulte the section called “Definir um nome de usuário personalizado”.

Etapa 2: criar uma entrada de acesso

Depois de planejar a entrada de acesso, use a AWS CLI para criá-la.

Os exemplos a seguir abordam a maior parte dos casos de uso. Veja a referência da CLI para todas as opções de configuração.

Você anexará a política de acesso na próxima etapa.

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD
```

Etapa 3: associar a política de acesso

O comando difere com base em se você deseja que a política seja limitada a um namespace específico do Kubernetes.

Você precisa do ARN da política de acesso. Analise as políticas de acesso disponíveis.

Criar uma política sem escopo de namespace

```
aws eks associate-access-policy --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --policy-arn <access-policy-arn>
```

Criar com escopo de namespace

```
aws eks associate-access-policy --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --policy-arn
 <access-policy-arn>
```

Próximas etapas

- Crie um kubeconfig para usar o kubectl com uma identidade do IAM

Criar uma entrada de acesso usando grupos do Kubernetes com a AWS CLI

Crie entradas de acesso do Amazon EKS que usam grupos do Kubernetes para fins de autorização e exigem a configuração manual do RBAC.

> **Note**
>
> Para a maioria dos casos de uso, recomendamos usar as políticas de acesso do EKS em vez da abordagem de grupos do Kubernetes descrita nesta página. As políticas de acesso do EKS fornecem uma maneira mais simples e integrada com a AWS para gerenciar o acesso sem exigir a configuração manual do RBAC. Use a abordagem de grupos do Kubernetes somente quando precisar de um controle mais granular do que o oferecido pelas políticas de acesso do EKS.

Visão geral

As entradas de acesso definem como as identidades do IAM (usuários e perfis) acessam seus clusters do Kubernetes. A abordagem de grupos do Kubernetes concede aos usuários ou perfis do IAM permissão para acessar seu cluster EKS por meio de grupos RBAC padrão do Kubernetes. Esse método requer a criação e o gerenciamento de recursos RBAC do Kubernetes (Roles, RoleBindings, ClusterRoles e ClusterRoleBindings) e é recomendado quando você precisa de conjuntos de permissões altamente personalizados, requisitos de autorização complexos ou deseja manter padrões de controle de acesso consistentes em ambientes híbridos do Kubernetes.

Este tópico não aborda a criação de entradas de acesso para identidades do IAM usadas para instâncias do Amazon EC2 para integrar clusters do EKS.

Pré-requisitos

- O modo de autenticação do seu cluster deve ser configurado para permitir entradas de acesso. Para obter mais informações, consulte the section called “Modo de autenticação”.
- Instale e configure a AWS CLI, conforme descrito em Instalar, no Guia do usuário da AWS Command Line Interface.
- É recomendável familiarizar-se com o Kubernetes RBAC. Para obter mais informações, consulte Usar a autorização de RBAC na documentação do Kubernetes.

Etapa 1: definir a entrada de acesso

1. Encontre o ARN da identidade do IAM, como um usuário ou um perfil, ao qual você deseja conceder permissões.
   - Cada identidade do IAM pode ter somente uma entrada de acesso ao EKS.
2. Determine quais grupos do Kubernetes você deseja associar a essa identidade do IAM.
   - Você precisará criar ou usar os recursos do Kubernetes existentes `Role`/`ClusterRole` e `RoleBinding`/`ClusterRoleBinding` que façam referência a esses grupos.
3. Determine se o nome de usuário gerado automaticamente é apropriado para a entrada de acesso ou se você precisa especificar manualmente um nome de usuário.
   - A AWS gera automaticamente esse valor com base na identidade do IAM. Você pode definir um nome de usuário personalizado. Isso é visível nos logs do Kubernetes.
   - Para obter mais informações, consulte the section called “Definir um nome de usuário personalizado”.

Etapa 2: criar uma entrada de acesso para grupos do Kubernetes

Depois de planejar a entrada de acesso, use a AWS CLI para criá-la com os grupos apropriados do Kubernetes.

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD --kubernetes-groups <groups>
```

Substitua:

- `<cluster-name>` com o nome do seu cluster do EKS
- `<iam-identity-arn>` com o ARN do perfil ou usuário do IAM
- `<groups>` com uma lista separada por vírgulas de grupos do Kubernetes (por exemplo, "system:developers,system:readers")

Veja a referência da CLI para todas as opções de configuração.

Etapa 3: configurar recursos do RBAC

Para que a entidade principal do IAM tenha acesso aos objetos do Kubernetes em seu cluster, você deve criar e gerenciar objetos de controle de acesso com base em perfil (RBAC) do Kubernetes:

1. Crie objetos `Role` ou `ClusterRole` do Kubernetes que definam as permissões.
2. Crie objetos `RoleBinding` ou `ClusterRoleBinding` do Kubernetes no cluster que especifiquem o nome do grupo como um `subject` para `kind: Group`.

Para obter informações detalhadas sobre como configurar grupos e permissões no Kubernetes, consulte Como usar a autorização RBAC na documentação do Kubernetes.

Próximas etapas

- Crie um kubeconfig para usar o kubectl com uma identidade do IAM

Conceder aos usuários do IAM acesso ao Kubernetes com um ConfigMap

> **Important**
>
> O `aws-auth ConfigMap` está obsoleto. Para conhecer o método recomendado para gerenciar o acesso a APIs do Kubernetes, consulte the section called “Entradas de acesso”.

O acesso ao seu cluster usando entidades principais do IAM é ativado pelo AWS IAM Authenticator for Kubernetes, que é executado no ambiente de gerenciamento do Amazon EKS. O autenticador obtém suas informações de configuração do `ConfigMap aws-auth`. Para todas as configurações do `ConfigMap aws-auth`, consulte Formato de configuração completo no GitHub.

Adicione um usuário do IAM ao cluster do Amazon EKS

Quando você cria um cluster do Amazon EKS, a entidade principal do IAM que cria o cluster, recebe automaticamente permissões `system:masters` na configuração de controle de acesso baseado em perfil (RBAC) no ambiente de gerenciamento do Amazon EKS. Como essa entidade principal não é exibida em nenhuma configuração visível, mantenha o controle de qual entidade principal criou o cluster originalmente. Para conceder a outras entidades principais do IAM a capacidade de interagir com o cluster, edite o `aws-auth ConfigMap` no Kubernetes e crie um `rolebinding` ou `clusterrolebinding` do Kubernetes com o nome de um `group` especificado no `aws-auth ConfigMap`.

> **Note**
>
> Para obter mais informações sobre a configuração de controle de acesso baseado em perfil (RBAC) do Kubernetes, consulte Using RBAC Authorization na documentação do Kubernetes.

1. Determine quais credenciais o `kubectl` está usando para acessar o cluster. Em seu computador, você pode ver quais credenciais o `kubectl` usa com o comando a seguir. Substitua *`~/.kube/config`* pelo caminho do arquivo `kubeconfig` se você não usar o caminho padrão.

   ```
   cat ~/.kube/config
   ```

   Veja abaixo um exemplo de saída.

```
[...]
contexts:
- context:
    cluster: my-cluster.region-code.eksctl.io
    user: admin@my-cluster.region-code.eksctl.io
  name: admin@my-cluster.region-code.eksctl.io
current-context: admin@my-cluster.region-code.eksctl.io
[...]
```

No exemplo de saída anterior, as credenciais de um usuário chamado *admin* são configuradas para um cluster chamado *my-cluster*. Se for o usuário que criou o cluster, ele já terá acesso ao cluster. Se não for o usuário que criou o cluster, será necessário concluir as etapas restantes para habilitar o acesso ao cluster para os outras entidades principais do IAM. As melhores práticas do IAM recomendam que você conceda permissões para perfis e não para usuários. É possível ver quais outras entidades principais atualmente têm acesso ao cluster com o seguinte comando:

```
kubectl describe -n kube-system configmap/aws-auth
```

Veja um exemplo de saída abaixo.

```
Name:         aws-auth
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws:iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}


BinaryData
====

Events:  <none>
```

O exemplo anterior é um `aws-auth` `ConfigMap` padrão. Apenas a função de instância de nó tem acesso ao cluster.

2. Verifique se você já tem `roles` e `rolebindings` ou `clusterroles` e `clusterrolebindings` do Kubernetes para os quais é possível mapear as entidades principais do IAM. Para saber mais sobre esses recursos, consulte Usar a autorização de RBAC, na documentação do Kubernetes.

   a. Visualize suas `roles` ou `clusterroles` existentes do Kubernetes. `Roles` têm o escopo definido para um `namespace`, mas `clusterroles` têm o escopo definido para o cluster.

   ```
   kubectl get roles -A
   ```

   ```
   kubectl get clusterroles
   ```

   b. Visualize os detalhes de qualquer `role` ou `clusterrole` retornada no resultado anterior e confirme que ela tem as permissões (`rules`) que você quer que as entidades principais do IAM tenham no cluster.

   Substitua *`role-name`* por um nome de `role` retornado na saída do comando anterior. Substitua *`kube-system`* pelo namespace da `role`.

   ```
   kubectl describe role role-name -n kube-system
   ```

   Substitua *`cluster-role-name`* por um nome de `clusterrole` retornado na saída do comando anterior.

   ```
   kubectl describe clusterrole cluster-role-name
   ```

   c. Visualize suas `rolebindings` ou `clusterrolebindings` existentes do Kubernetes. `Rolebindings` têm o escopo definido para um `namespace`, mas `clusterrolebindings` têm o escopo definido para o cluster.

   ```
   kubectl get rolebindings -A
   ```

   ```
   kubectl get clusterrolebindings
   ```

   d. Visualize os detalhes de qualquer `rolebinding` ou `clusterrolebinding` e confirme que ele tem uma `role` ou `clusterrole` da etapa anterior listada como `roleRef` e um nome de grupo listado para `subjects`.

Substitua *role-binding-name* por um nome de `rolebinding` retornado na saída do comando anterior. Substitua *kube-system* pelo `namespace` da `rolebinding`.

```
kubectl describe rolebinding role-binding-name -n kube-system
```

Veja abaixo um exemplo de saída.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eks-console-dashboard-restricted-access-role-binding
  namespace: default
subjects:
- kind: Group
  name: eks-console-dashboard-restricted-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: eks-console-dashboard-restricted-access-role
  apiGroup: rbac.authorization.k8s.io
```

Substitua *cluster-role-binding-name* por um nome de `clusterrolebinding` retornado na saída do comando anterior.

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

Veja abaixo um exemplo de saída.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
- kind: Group
  name: eks-console-dashboard-full-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: eks-console-dashboard-full-access-clusterrole
```

```
  apiGroup: rbac.authorization.k8s.io
```

3. Edite o `ConfigMap` `aws-auth`. É possível usar uma ferramenta como o `eksctl` para atualizar o `ConfigMap`, ou você pode atualizá-lo manualmente via edição.

> **Important**
>
> Convém usar `eksctl` ou outra ferramenta para editar o `ConfigMap`. Para obter informações sobre outras ferramentas que você pode usar, consulte Usar ferramentas para fazer alterações no aws-authConfigMap nos guias de práticas recomendadas do Amazon EKS. Um `aws-auth` `ConfigMap` formatado incorretamente pode fazer com que você perca o acesso ao cluster.

- Veja as etapas para editar o configmap com eksctl.
- Veja as etapas para editar o configmap manualmente.

Editar o configmap com eksctl

1. Você precisa da versão `0.215.0` ou posterior da ferramenta de linha de comando da `eksctl` instalada no seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
2. Visualize os mapeamentos atuais no `ConfigMap`. Substitua *`my-cluster`* pelo nome do cluster. Substitua *`region-code`* pela região da AWS em que seu cluster se encontra.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

   Veja um exemplo de saída abaixo.

```
ARN
            USERNAME                                GROUPS
 ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

3. Adicione um mapeamento referente a uma função. Substitua *`my-role`* pelo nome da sua função. Substitua *`eks-console-dashboard-full-access-group`* pelo nome do grupo especificado

no objeto `RoleBinding` ou `ClusterRoleBinding` do Kubernetes. Substitua *`111122223333`* pelo ID da sua conta. Você pode substituir *`admin`* por qualquer nome que escolher.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws:iam::111122223333:role/my-role --username admin --group eks-
console-dashboard-full-access-group \
    --no-duplicate-arns
```

> **Important**
>
> O ARN da função não pode incluir um caminho, como `role/my-team/developers/my-role`. O formato do ARN deve ser `arn:aws:iam::`*`111122223333`*`:role/`*`my-role`* . Neste exemplo, `my-team/developers/` precisa ser removido.

Veja um exemplo de saída abaixo.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "{arn-aws}iam::111122223333:role/my-role" to
 auth ConfigMap
```

4. Adicione um mapeamento referente a um usuário. As melhores práticas do IAM recomendam que você conceda permissões para perfis e não para usuários. Substitua *`my-user`* pelo seu nome de usuário. Substitua *`eks-console-dashboard-restricted-access-group`* pelo nome do grupo especificado na `RoleBinding` ou `ClusterRoleBinding` do Kubernetes. Substitua *`111122223333`* pelo ID da sua conta. Você pode substituir *`my-user`* por qualquer nome que escolher.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user --username my-user --group eks-
console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

Veja um exemplo de saída abaixo.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws:iam::111122223333:user/my-user" to
 auth ConfigMap
```

5. Visualize os mapeamentos no `ConfigMap` novamente.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

Veja abaixo um exemplo de saída.

```
ARN
           USERNAME                                GROUPS
      ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:role/admin
           my-role                                 eks-console-dashboard-full-access-
group
arn:aws:iam::111122223333:user/my-user
            my-user                                 eks-console-dashboard-restricted-
access-group
```

Editar o configmap manualmente

1. Abra o `ConfigMap` para edição.

```
kubectl edit -n kube-system configmap/aws-auth
```

> **Note**
>
> Se você receber um erro informando "`Error from server (NotFound): configmaps "aws-auth" not found`", use o procedimento em Aplicar o ConfigMap aws-auth ao seu cluster para aplicar o estoque `ConfigMap`.

2. Adicione as entidades principais do IAM ao `ConfigMap`. Um grupo do IAM não é uma entidade principal do IAM, portanto, não pode ser adicionado ao `ConfigMap`.
   - Para adicionar uma função do IAM (por exemplo, para usuários federados): adicione os detalhes da função à seção `mapRoles` do `ConfigMap` em `data`. Adicione essa seção se ela ainda não existir no arquivo. Cada entrada oferece suporte aos seguintes parâmetros:
     - rolearn: o ARN da função do IAM a ser adicionada. Esse valor não pode incluir um caminho. Por exemplo, não é possível especificar um ARN como

`arn:aws:iam::111122223333:role/my-team/developers/role-name` . O ARN precisa ser `arn:aws:iam::111122223333:role/role-name` .

- username: o nome do usuário no Kubernetes a ser mapeado para a função do IAM.
- groups: o grupo ou a lista de grupos do Kubernetes para o qual mapear a função. O grupo pode ser padrão ou um grupo especificado em um `clusterrolebinding` ou `rolebinding`. Para obter mais informações, consulte Default roles and role bindings na documentação do Kubernetes.

- Para adicionar um usuário do IAM: as melhores práticas do IAM recomendam que você conceda permissões para perfis e não para usuários. Adicione os detalhes do usuário à seção `mapUsers` do `ConfigMap` em `data`. Adicione essa seção se ela ainda não existir no arquivo. Cada entrada oferece suporte aos seguintes parâmetros:
  - userarn: o ARN do usuário do IAM a ser adicionado.
  - username: o nome do usuário no Kubernetes a ser mapeado para o usuário do IAM.
  - groups: o grupo, ou a lista de grupos, do Kubernetes para o qual mapear o usuário. O grupo pode ser padrão ou um grupo especificado em um `clusterrolebinding` ou `rolebinding`. Para obter mais informações, consulte Default roles and role bindings na documentação do Kubernetes.

3. Por exemplo, o seguinte bloco YAML contém:
   - Uma seção `mapRoles` que mapeia a instância do nó do IAM para grupos do Kubernetes, para que os nós possam se registrar no cluster e o perfil do IAM `my-console-viewer-role` que é mapeado para um grupo do Kubernetes que pode visualizar todos os recursos do Kubernetes para todos os clusters. Para acessar uma lista das permissões de grupos do IAM e do Kubernetes necessárias para o perfil do IAM `my-console-viewer-role`, consulte the section called “Permissões obrigatórias”.
   - Uma seção `mapUsers` que mapeia o usuário do IAM `admin` da conta da AWS padrão para o grupo do Kubernetes `system:masters` e o usuário `my-user` de uma conta da AWS diferente mapeada para um grupo do Kubernetes que pode visualizar recursos do Kubernetes de um namespace específico. Para acessar uma lista das permissões de grupos do IAM e do Kubernetes necessárias para o usuário do IAM `my-user`, consulte the section called “Permissões obrigatórias”.

   Adicione ou remova linhas conforme necessário e substitua todos os valores de exemplo por seus próprios valores.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
```

```
# and an empty file will abort the edit. If an error occurs while saving this file
 will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::111122223333:role/my-role
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
    - groups:
      - system:masters
      userarn: arn:aws:iam::111122223333:user/admin
      username: admin
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws:iam::444455556666:user/my-user
      username: my-user
```

4. Salve o arquivo e saia do seu editor de texto.

Como aplicar o **`ConfigMap aws-auth`** ao seu cluster

O `ConfigMap aws-auth` é criado e aplicado automaticamente ao cluster ao criar um grupo de nós gerenciados ou ao criar um grupo de nós usando `eksctl`. Ele é criado inicialmente para permitir que os nós ingressem no cluster, mas você também pode usar esse `ConfigMap` para adicionar acesso com controle de acesso baseado em função (RBAC) às entidades principais do IAM. Se já tiver iniciado os nós autogerenciados, mas ainda não tiver aplicado o `ConfigMap aws-auth` ao seu cluster, você poderá fazer isso usando o procedimento a seguir.

1. Verifique se você já aplicou o `ConfigMap aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

Se você receber um erro informando "`Error from server (NotFound): configmaps "aws-auth" not found`", prossiga com as etapas a seguir para aplicar o estoque `ConfigMap`.

2. Faça download, edite e aplique o mapa de configuração do autenticador da AWS.

   a. Faça download do mapa de configuração.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-auth-cm.yaml
   ```

   b. No arquivo `aws-auth-cm.yaml`, defina o `rolearn` como o nome de recurso da Amazon (ARN) da função do IAM associada aos seus nós. Você pode fazer isso com um editor de texto ou substituindo *`my-node-instance-role`* e executando o seguinte comando:

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
   ```

   Não modifique outras linhas do arquivo.

   > **Important**
   >
   > O ARN da função não pode incluir um caminho, como `role/my-team/developers/my-role`. O formato do ARN deve ser `arn:aws:iam::`*`111122223333`*`:role/`*`my-role`* . Neste exemplo, `my-team/developers/` precisa ser removido.

   Você pode inspecionar as saídas da pilha do AWS CloudFormation para seus grupos de nós e procurar os seguintes valores:

   - InstanceRoleARN: para grupos de nós que foram criados com `eksctl`
   - NodeInstanceRole - Para grupos de nós que foram criados com os modelos do Amazon EKS fornecidos pelo AWS CloudFormation no Console de gerenciamento da AWS

   c. Aplique a configuração. Esse comando pode demorar alguns minutos para ser concluído.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

> **Note**
>
> Se você receber qualquer erro de autorização ou de tipo de recurso, consulte the section called “Acesso negado ou não autorizado (`kubectl`)” no tópico de solução de problemas.

3. Observe o status de seus nós e aguarde até que eles atinjam o status `Ready`.

```
kubectl get nodes --watch
```

Insira `Ctrl+C` para retornar a um prompt de shell.

Conceder aos usuários acesso ao Kubernetes com um provedor OIDC externo

O Amazon EKS oferece suporte ao uso de provedores de identidade OpenID Connect (OIDC) como um método de autenticação de usuários no cluster. Provedores de identidade OIDC podem ser usados com ou como alternativa ao AWS Identity and Access Management (IAM). Para obter mais informações sobre como usar a IAM, consulte the section called “Acesso à API do Kubernetes”. Depois de configurar a autenticação do cluster, você pode criar a `roles` e `clusterroles` do Kubernetes para atribuir permissões às funções e, em seguida, vincular as funções às identidades usando o Kubernetes `rolebindings` e `clusterrolebindings`. Para obter mais informações, consulte Usar a autorização de RBAC na documentação do Kubernetes.

- Você pode associar um provedor de identidade OIDC ao cluster.
- O Kubernetes não fornece um provedor de identidade OIDC. Você pode usar um provedor de identidade OIDC público existente ou executar seu próprio provedor de identidade. Para uma lista de provedores certificados, consulte OpenID Certification (Certificação OpenID) no site OpenID.
- O URL do emissor do provedor de identidade OIDC deve ser acessível publicamente, para que o Amazon EKS possa descobrir as chaves de assinatura. O Amazon EKS não é compatível com provedores de identidade OIDC com certificados autoassinados.
- Não é possível desabilitar a autenticação IAM em seu cluster, porque ela ainda é necessária para juntar nós a um cluster.

- Um cluster do Amazon EKS ainda deve ser criado por uma entidade principal do IAM da AWS em vez de um usuário do provedor de identidade do OIDC. Isso ocorre porque o criador do cluster interage com as APIs do Amazon EKS, em vez das APIs do Kubernetes.
- Os usuários autenticados pelo provedor de identidade OIDC serão listados no log de auditoria do cluster se os logs do CloudWatch estiverem ativados para o ambiente de gerenciamento. Para obter mais informações, consulte the section called “Habilitar ou desabilitar os logs do ambiente de gerenciamento”.
- Não é possível fazer login no Console de gerenciamento da AWS com uma conta de um provedor OIDC. Você só pode the section called “Acessar recursos do cluster” fazendo login no Console de gerenciamento da AWS com uma conta do AWS Identity and Access Management.

Associe um provedor de identidade OIDC

Antes de associar um provedor de identidade OIDC ao cluster, você precisa das seguintes informações do seu provedor:

URL do emissor

URL do provedor de identidade do OIDC que permite que o servidor de API descubra as chaves de assinatura públicas para verificar tokens. O URL deve começar com `https://` e deve corresponder à reivindicação `iss` nos tokens de ID do OIDC do provedor. De acordo com o padrão OIDC, os componentes de caminho são permitidos, mas os parâmetros de consulta não são. Normalmente, o URL consiste apenas em um nome de host, como `https://server.example.org` ou `https://example.com` . Este URL deve apontar para o nível abaixo de `.well-known/openid-configuration` e deve ser acessível ao público pela Internet.

ID do cliente (também conhecido como público)

O ID da aplicação do cliente que faz solicitações de autenticação ao provedor de identidades OIDC.

Você pode associar um provedor de identidade usando o `eksctl` ou o Console de gerenciamento da AWS.

Associar um provedor de identidade usando o eksctl

1. Crie um arquivo denominado `associate-identity-provider.yaml` com os conteúdos a seguir. Substitua os valores de exemplo pelos seus próprios. Os valores na seção

`identityProviders` são obtidos do seu provedor de identidade OIDC. Os valores são necessários apenas para as configurações `name`, `type`, `issuerUrl` e `clientId` em `identityProviders`.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: your-region-code

identityProviders:
  - name: my-provider
    type: oidc
    issuerUrl: https://example.com
    clientId: kubernetes
    usernameClaim: email
    usernamePrefix: my-username-prefix
    groupsClaim: my-claim
    groupsPrefix: my-groups-prefix
    requiredClaims:
      string: string
    tags:
      env: dev
```

> **Important**
>
> Não especifique o `system:`, ou qualquer parte dessa string, para `groupsPrefix` ou `usernamePrefix`.

2. Crie o provedor.

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. Para usar o `kubectl` para trabalhar com o cluster e o provedor de identidade OIDC, consulte Utilizando kubectl na documentação do Kubernetes.

Associe um provedor de identidade usando o console AWS

1. Abra o console do Amazon EKS.
2. Selecione seu cluster e, em seguida, selecione a guia Acesso.
3. Na seção Provedores de identidade OIDC, selecione* Provedor de identidade do associado*.
4. Na página Associate OIDC Identity Provider (Provedor de identidade OIDC), insira ou selecione as opções a seguir e escolha Associate (Associar).
   - Em Name (Nome), insira um nome exclusivo para o provedor.
   - Para Issuer URL (URL do emissor), insira a URL do seu provedor. Este URL deve ser acessado pela Internet.
   - Em ID do cliente, insira o ID do cliente do provedor de identidade OIDC (também conhecido como público).
   - Para o Username claim (Reivindicação do usuário), insira a reivindicação a ser usada como nome de usuário.
   - Para o Username claim (Reivindicação do usuário), insira a reivindicação a ser usada como grupo de usuários.
   - (Opcional) Selecione Advanced options (Opções avançadas) e insira ou selecione as informações a seguir.
     - Username prefix (Prefixo do usuário): insira um prefixo para preceder as reivindicações de nome de usuário. O prefixo precede as reivindicações de nome de usuário para evitar conflitos com nomes existentes. Se você não fornecer um valor e o nome de usuário for um valor diferente de `email`, o prefixo assume como padrão o valor para a URL do emissor. Você pode usar o valor `-` para desativar todos os prefixos. Não especifique o `system:`, ou qualquer parte dessa string.
     - Groups prefix (Prefixo de grupos): insira um prefixo para preceder as reivindicações de grupos. O prefixo precede as reivindicações de nome de usuário para evitar conflitos com nomes existentes (como `system: groups`). Por exemplo, o valor `oidc:` cria nomes de grupos como `oidc:engineering` e `oidc:infra`. Não especifique o `system:`, ou qualquer parte dessa string.
     - Required claims (Reivindicações necessárias): selecione Add claim (adicionar reivindicação) e insira um ou mais pares de valor de chave que descrevam as reivindicações necessárias no token de ID do cliente. Os pares descrevem as reivindicações obrigatórias no token de ID. Se definido, cada reivindicação é verificada para estar presente no token de ID com um valor correspondente.

a. Para usar o `kubectl` para trabalhar com o cluster e o provedor de identidade OIDC, consulte Utilizando kubectl na documentação do Kubernetes.

Exemplo de política do IAM

Se você quiser impedir que um provedor de identidade OIDC seja associado a um cluster, crie e associe a seguinte política do IAM às contas do IAM dos seus administradores do Amazon EKS. Para obter mais informações, consulte Criação de políticas do IAM e Adição de permissões de identidade do IAM no Guia do usuário do IAM e Ações na Referência de autorização do serviço.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "denyOIDC",
            "Effect": "Deny",
            "Action": [
                "eks:AssociateIdentityProviderConfig"
            ],
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/*"

        },
        {
            "Sid": "eksAdmin",
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

A política de exemplo a seguir permite a associação do provedor de identidade OIDC se o `clientID` for `kubernetes` e a `issuerUrl` for `https://cognito-idp.us-west-2amazonaws.com/*` .

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowCognitoOnly",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotLikeIfExists": {
                    "eks:issuerUrl": "https://cognito-idp.us-west-2.amazonaws.com/*"
                }
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
                    "eks:clientId": "kubernetes"
                }
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}
```

Desassociar um provedor de identidade OIDC do cluster

Se você desassociar um provedor de identidade OIDC do cluster, os usuários incluídos no provedor não poderão mais acessar o cluster. No entanto, você ainda pode acessar o cluster com as entidades principais do IAM.

1. Abra o console do Amazon EKS.
2. Na seção OIDC Identity Providers (Provedores de identidade OIDC), selecione Disassociate (Desassociar), insira o nome do provedor de identidade e selecione `Disassociate`.

Visualize os recursos do Kubernetes na seção Console de gerenciamento da AWS

É possível visualizar os recursos do Kubernetes implantados no seu cluster com o Console de gerenciamento da AWS. Não é possível visualizar os recursos do Kubernetes com a AWS CLI ou o eksctl. Para visualizar os recursos do Kubernetes usando uma ferramenta de linha de comando, use kubectl.

> **Note**
>
> Para visualizar a guia Recursos e a seção Nós na guia Computação no Console de gerenciamento da AWS, a entidade principal do IAM que você está usando deve ter permissões específicas do IAM e do Kubernetes. Para obter mais informações, consulte the section called “Permissões obrigatórias”.

1. Abra o console do Amazon EKS.
2. Na lista Clusters, escolha o cluster que contém os recursos do Kubernetes que você deseja visualizar.
3. Selecione a guia Recursos.
4. Selecione um grupo Resource type (Tipo de recurso) cujos recursos você deseja visualizar, como Workloads. Você visualizará uma lista de tipos de recursos nesse grupo.
5. Selecione um tipo de recurso, como Deployments (Implantações), no grupo Workloads. Você verá uma descrição do tipo de recurso, um link para a documentação do Kubernetes para obter mais informações sobre esse tipo, bem como uma lista de recursos desse tipo implantados no cluster. Se essa lista estiver vazia, significa que não há recursos desse tipo implantados no cluster.
6. Selecione um recurso para exibir mais informações sobre ele. Teste os exemplos a seguir:
   - Selecione o grupo Workloads, escolha o tipo de recurso Deployments (Implantações) e depois selecione o recurso coredns. Por padrão, ao selecionar um recurso, você está em Structured view (Visualização estruturada). Para alguns tipos de recursos, há uma seção Pods em Structured view (Visualização estruturada). Esta seção lista os pods gerenciados pela workload. Você pode selecionar qualquer pod listado para visualizar as informações sobre o pod. Nem todos os tipos de recursos mostram informações em Structured view (Visualização estruturada). Se a opção Raw view (Visualização bruta) for selecionada no canto superior direito da página

do recurso, será possível ver a resposta JSON completa da API do Kubernetes para esse recurso.

- Selecione o grupo Cluster e depois o tipo de recurso Nodes (Nós). Você verá uma lista com todos os nós do seu cluster. Esses nós podem ser de qualquer Tipo de nó do Amazon EKS. Esta é a mesma lista visível na seção Nodes (Nós) ao selecionar a guia Compute (Computação) do seu cluster. Escolha um recurso de nó da lista. Em Structured view (Visualização estruturada), também é possível ver uma seção Pods. Esta seção mostra todos os pods em execução no nó.

Permissões obrigatórias

Para visualizar a guia Recursos e a seção Nós na guia Computação no Console de gerenciamento da AWS, a entidade principal do IAM que você está usando deve ter permissões mínimas específicas do IAM e do Kubernetes. Conclua as seguintes etapas para atribuir as permissões necessárias às entidades principais do IAM.

1. Verifique se `eks:AccessKubernetesApi` e as outras permissões necessárias do IAM para visualizar os recursos do Kubernetes estão atribuídas à entidade principal do IAM que você está usando. Para obter mais informações sobre como editar permissões para uma entidade principal do IAM, consulte Controle de acesso das entidades principais no Guia do usuário do IAM. Para saber mais sobre como editar as permissões de um perfil, consulte Modificar a política de permissões de um perfil (console), no Guia do usuário do IAM.

   O exemplo de política a seguir inclui as permissões necessárias para uma entidade principal visualizar os recursos do Kubernetes de todos os clusters da conta. Substitua *111122223333* pelo ID de sua conta da AWS.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:ListFargateProfiles",
                "eks:DescribeNodegroup",
                "eks:ListNodegroups",
                "eks:ListUpdates",
                "eks:AccessKubernetesApi",
                "eks:ListAddons",
```

```
                "eks:DescribeCluster",
                "eks:DescribeAddonVersions",
                "eks:ListClusters",
                "eks:ListIdentityProviderConfigs",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:GetParameter",
            "Resource": "arn:aws:ssm:*:111122223333:parameter/*"
        }
    ]
}
```

Para visualizar nós em clusters conectados, o perfil do IAM do conector do Amazon EKS deve ser capaz de representar a entidade principal no cluster. Isso permite que o Amazon EKS Connector mapeie a entidade principal para um usuário do Kubernetes.

2. Crie uma `rolebinding` ou `clusterrolebinding` do Kubernetes que esteja vinculada a uma `role` ou `clusterrole` Kubernetes que tenha as permissões necessárias para visualizar os recursos do Kubernetes. Para saber mais sobre funções e associações de função do Kubernets, consulte Usar a autorização do RBAC, na documentação do Kubernetes. É possível aplicar um dos seguintes manifestos ao cluster que cria um `role` e `rolebinding` ou um `clusterrole` e `clusterrolebinding` com as permissões necessárias do Kubernetes:

   Visualizar recursos do Kubernetes em todos os namespaces

   - O nome do grupo no arquivo é `eks-console-dashboard-full-access-group`. Aplique o manifesto ao seu cluster com o seguinte comando:

     ```
     kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
     ```

   Visualizar recursos do Kubernetes em um namespace específico

   - O namespace neste arquivo é `default`. O nome do grupo no arquivo é `eks-console-dashboard-restricted-access-group`. Aplique o manifesto ao seu cluster com o seguinte comando:

     ```
     kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
     ```

Se for necessário modificar o nome do grupo Kubernetes, o namespace, as permissões ou qualquer outra configuração no arquivo, baixe esse arquivo e edite-o antes de aplicá-lo ao seu cluster:

a. Baixe o arquivo com um dos seguintes comandos:

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
```

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
```

b. Edite esse arquivo conforme for necessário.

c. Aplique o manifesto ao seu cluster com um dos seguintes comandos:

```
kubectl apply -f eks-console-full-access.yaml
```

```
kubectl apply -f eks-console-restricted-access.yaml
```

3. Mapeie a entidade principal do IAM para o usuário ou grupo do Kubernetes no `ConfigMap` `aws-auth`. É possível usar uma ferramenta como o `eksctl` para atualizar o `ConfigMap`, ou você pode atualizá-lo manualmente via edição.

> **Important**
>
> Convém usar `eksctl` ou outra ferramenta para editar o `ConfigMap`. Para obter informações sobre outras ferramentas que você pode usar, consulte Usar ferramentas para fazer alterações no aws-authConfigMap nos guias de práticas recomendadas do Amazon EKS. Um `aws-auth` `ConfigMap` formatado incorretamente pode fazer com que você perca o acesso ao cluster.

Editar com o eksctl

1. Você precisa da versão `0.215.0` ou posterior da ferramenta de linha de comando da `eksctl` instalada no seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

2. Visualize os mapeamentos atuais no `ConfigMap`. Substitua *`my-cluster`* pelo nome do cluster. Substitua *`region-code`* pela região da AWS em que seu cluster se encontra.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

Veja um exemplo de saída abaixo.

```
ARN
            USERNAME                                GROUPS
 ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

3. Adicione um mapeamento referente a uma função. Esse exemplo supõe que você tenha anexado as permissões do IAM na primeira etapa a um perfil denominado *`my-console-viewer-role`*. Substitua *`111122223333`* pelo ID da sua conta.

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:role/my-console-viewer-role \
    --group eks-console-dashboard-full-access-group \
    --no-duplicate-arns
```

> **Important**
>
> O ARN da função não pode incluir um caminho, como `role/my-team/developers/my-role`. O formato do ARN deve ser `arn:aws:iam::`*`111122223333`*`:role/`*`my-role`* . Neste exemplo, `my-team/developers/` precisa ser removido.

Veja um exemplo de saída abaixo.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "arn:aws:iam::111122223333:role/my-console-
viewer-role" to auth ConfigMap
```

4. Adicione um mapeamento referente a um usuário. As melhores práticas do IAM recomendam que você conceda permissões para perfis e não para usuários. Esse exemplo supõe que você tenha anexado as permissões do IAM na primeira etapa a um usuário denominado *my-user*. Substitua *111122223333* pelo ID da sua conta.

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user \
    --group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

   Veja abaixo um exemplo de saída.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws:iam::111122223333:user/my-user" to
 auth ConfigMap
```

5. Visualize os mapeamentos no `ConfigMap` novamente.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

   Veja abaixo um exemplo de saída.

```
ARN
           USERNAME                                GROUPS
      ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:role/my-console-viewer-role
                                                    eks-console-dashboard-full-access-
group
arn:aws:iam::111122223333:user/my-user
                                                     eks-console-dashboard-restricted-
access-group
```

Editar o ConfigMap manualmente

Para obter mais informações sobre como adicionar usuários ou perfis do IAM ao `aws-auth` `ConfigMap`, consulte the section called “Adicione um usuário do IAM ao cluster do Amazon EKS”.

1. Abra o `ConfigMap` de `aws-auth` para edição.

```
kubectl edit -n kube-system configmap/aws-auth
```

2. Adicione os mapeamentos ao `aws-auth` `ConfigMap`, mas não substitua os mapeamentos existentes. O seguinte exemplo adiciona mapeamentos entre as entidades principais do IAM com permissões adicionadas na primeira etapa e os grupos do Kubernetes criados na etapa anterior:
   - O perfil *`my-console-viewer-role`* e `eks-console-dashboard-full-access-group`.
   - O usuário *`my-user`* `eks-console-dashboard-restricted-access-group`.

   Esses exemplos supõem que você tenha anexado as permissões do IAM da primeira etapa a um perfil denominado *`my-console-viewer-role`* e a um usuário denominado *`my-user`*. Substitua *`111122223333`* pelo ID de sua conta da AWS.

```
apiVersion: v1
data:
mapRoles: |
  - groups:
    - eks-console-dashboard-full-access-group
    rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
    username: my-console-viewer-role
mapUsers: |
  - groups:
    - eks-console-dashboard-restricted-access-group
    userarn: arn:aws:iam::111122223333:user/my-user
    username: my-user
```

> **Important**
>
> O ARN da função não pode incluir um caminho, como `role/my-team/developers/my-console-viewer-role`. O formato do ARN deve ser `arn:aws:iam::`*`111122223333`*`:role/`*`my-console-viewer-role`* . Neste exemplo, `my-team/developers/` precisa ser removido.

3. Salve o arquivo e saia do seu editor de texto.

Conecte o kubectl a um cluster EKS criando um arquivo kubeconfig

Neste tópico, você cria um arquivo `kubeconfig` para o cluster (ou atualiza um existente).

A ferramenta da linha de comando `kubectl` usa informações de configuração em arquivos `kubeconfig` para se comunicar com o servidor de API de um cluster. Para obter mais informações, consulte Organizando o acesso ao cluster usando arquivos kubeconfig na documentação do Kubernetes.

O Amazon EKS usa o comando `aws eks get-token` com `kubectl` para autenticação de cluster. Por padrão, a AWS CLI usa as mesmas credenciais que são retornadas com o seguinte comando:

```
aws sts get-caller-identity
```

- Um cluster existente do Amazon EKS. Para implantar, consulte Conceitos básicos.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar e Configuração rápida com aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- Um usuário ou perfil do IAM com permissão para usar a ação de API `eks:DescribeCluster` para o cluster que você especificar. Para ter mais informações, consulte the section called “Políticas baseadas em identidade”. Se você usar uma identidade do seu próprio provedor OpenID Connect para acessar o cluster, consulte Utilizando kubectl na documentação do Kubernetes para criar ou atualizar o arquivo `kube config`.

Criar arquivo **kubeconfig** automaticamente

- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar e Configuração rápida com aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- Permissão para usar a ação de API `eks:DescribeCluster` para o cluster que você especificar. Para ter mais informações, consulte the section called “Políticas baseadas em identidade”.

1. Crie um arquivo `kubeconfig` para o cluster. Substitua *`region-code`* pela região AWS em que seu cluster se encontra e substitua *`my-cluster`* pelo nome do seu cluster.

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   Por padrão, o arquivo de configuração resultante é criado no caminho padrão do `kubeconfig` (`.kube`) em seu diretório inicial ou mesclado com um `config` existente no local. Você pode especificar outro caminho com a opção `--kubeconfig`.

   Você pode especificar um ARN de função do IAM com a opção `--role-arn` para uso na autenticação quando você emitir comandos `kubectl`. Caso contrário, será usada a entidade principal do IAM em sua cadeia de credenciais padrão da AWS CLI ou SDK. Você pode visualizar sua identidade padrão AWS CLI ou SDK executando o comando `aws sts get-caller-identity`.

   Para obter todas as opções disponíveis, execute o comando `aws eks update-kubeconfig help` ou consulte update-kubeconfig em AWS CLI Command Reference.

2. Teste a configuração.

   ```
   kubectl get svc
   ```

   Veja um exemplo de saída abaixo.

   ```
   NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   ```

```
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

Se você receber qualquer erro de autorização ou de tipo de recurso, consulte the section called "Acesso negado ou não autorizado (`kubectl`)" no tópico de solução de problemas.

Conceder às workloads do Kubernetes acesso a AWS usando contas de serviço do Kubernetes

A Kubernetes service account provides an identity for processes that run in a Pod. For more information see Gerenciar contas de serviço in the Kubernetes documentation. If your Pod needs access to AWS services, you can map the service account to an AWS Identity and Access Management identity to grant that access. For more information, see the section called "Credenciais com IRSA" or the section called "Identidade de Pods".

Tokens de contas de serviço

O recurso BoundServiceAccountTokenVolume é habilitado por padrão nas versões do Kubernetes. Esse recurso melhora a segurança dos tokens de contas de serviço, permitindo que workloads em execução no Kubernetes solicitem tokens da Web JSON que são público-alvo, hora e chave vinculados. Tokens de contas de serviço têm uma expiração de uma hora. Em versões anteriores do Kubernetes, os tokens não tinham expiração. Isso significa que os clientes que dependem desses tokens precisam atualizá-los em até uma hora. Os seguintes exemplos de SDKs cliente Kubernetes atualizam tokens automaticamente no período de tempo necessário:

- Versão do Go `0.15.7` e posteriores
- Versão do Python `12.0.0` e posteriores
- Versão Java `9.0.0` e posterior
- Versão JavaScript `0.10.3` e posterior
- Ramificação Ruby `master`
- Versão do Haskell `0.3.0.0`
- Versão `7.0.5` e mais recente do C#

Se sua workload estiver usando uma versão anterior do cliente, você deverá atualizá-la. Para possibilitar uma migração suave de clientes para os tokens de conta de serviço com limite de tempo mais recentes, o Kubernetes adiciona um período de expiração estendido ao token da conta de

serviço além do tempo padrão de uma hora. Para clusters do Amazon EKS, o período de expiração prolongado equivale a 90 dias. O servidor de API do Kubernetes do cluster de Amazon EKS rejeita solicitações com tokens com mais de 90 dias. É recomendável verificar as aplicações e suas dependências para garantir que os SDKs de cliente do Kubernetes sejam iguais ou posteriores às versões listadas anteriormente.

Quando o servidor de API recebe solicitações com tokens com mais de uma hora, ele anota o evento de logs de auditoria da API com `annotations.authentication.k8s.io/stale-token`. O valor da anotação é parecido com o exemplo a seguir:

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold:
 4185802.
```

Se o cluster tiver registro em log do ambiente de gerenciamento habilitado, as anotações estarão nos logs de auditoria. Você pode usar a seguinte consulta do CloudWatch Log Insights para identificar todos os pods no cluster de Amazon EKS que estejam usando tokens obsoletos:

```
fields @timestamp
|filter @logStream like /kube-apiserver-audit/
|filter @message like /seconds after warning threshold/
|parse @message "subject: *, seconds after warning threshold:*\"" as subject,
 elapsedtime
```

O `subject` refere-se à conta de serviço que o pod usou. O `elapsedtime` indica o tempo decorrido (em segundos) após a leitura do token mais recente. As solicitações para o servidor de API são negadas quando o `elapsedtime` excede 90 dias (7.776.000 segundos). É necessário atualizar de forma proativa o SDK do cliente do Kubernetes das aplicações para usar uma das versões listadas anteriormente que atualizam o token de modo automático. Se o token da conta de serviço usado estiver se aproximando dos 90 dias e você não tiver tempo suficiente para atualizar as versões do SDK do cliente antes da expiração desse token, então será possível encerrar os pods existentes e criar novos. Isso provoca a reformulação do token da conta de serviço, fornecendo mais 90 dias para você atualizar os SDKs da versão do cliente.

Se o pod fizer parte de uma implantação, a maneira sugerida de encerrar pods e, ao mesmo tempo, manter a alta disponibilidade é executar uma implantação com o comando a seguir. Substitua *`my-deployment`* pelo nome da sua implantação.

```
kubectl rollout restart deployment/my-deployment
```

Complementos de cluster

Os complementos de cluster a seguir foram atualizados para usar os SDKs de cliente do Kubernetes que redefinem os tokens de contas de serviço automaticamente. Convém garantir que as versões listadas ou posteriores estejam instaladas no seu cluster.

- Versão `1.8.0` e mais recente do plug-in CNI da Amazon VPC para Kubernetes ou do auxiliar de métricas. Para verificar sua versão atual ou atualizá-la, consulte the section called “CNI da Amazon VPC” e cni-metrics-helper.
- Versão `1.8.4` e mais recente do CoreDNS. Para conferir sua versão atual ou atualizá-la, consulte the section called “CoreDNS”.
- Versão `2.0.0` e mais recente do AWS Load Balancer Controller. Para conferir sua versão atual ou atualizá-la, consulte the section called “ AWS Load Balancer Controller”.
- Uma Versão atual do `kube-proxy`. Para conferir sua versão atual ou atualizá-la, consulte the section called “kube-proxy”.
- AWS para a versão Fluent Bit `2.25.0` ou posterior. Para atualizar sua versão atual, consulte Releases, no GitHub
- Versão de imagem do Fluentd 1.14.6-1.2 ou posterior e plugin de filtro do Fluentd para metadados do Kubernetes versão 2.11.1 ou posterior.

Concessão de permissões do AWS Identity and Access Management para workloads em clusters do Amazon Elastic Kubernetes Service

O Amazon EKS oferece duas maneiras de conceder permissões de gerenciamento de acesso e identidade AWS a workloads executadas em clusters do Amazon EKS: Funções de IAM para contas de serviço e identidades de pod do EKS.

Perfis do IAM para contas de serviço

> Os perfis do IAM para contas de serviço (IRSA) configuram aplicações Kubernetes em execução na AWS com permissões minuciosas do IAM para acessar vários outros recursos da AWS, como buckets do Amazon S3, tabelas do Amazon DynamoDB e muito mais. É possível executar várias aplicações juntas no mesmo cluster de Amazon EKS e garantir que cada aplicação tenha apenas o conjunto mínimo de permissões de que precisa. O IRSA foi criado para ser compatível com várias opções de implementação do Kubernetes com suporte da AWS, como o Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service na AWS e clusters autogerenciados do

Kubernetes em instâncias do Amazon EC2. Assim, o IRSA foi criado usando um serviço da AWS básico, como o IAM, e não dependia diretamente do serviço Amazon EKS e da API do EKS. Para obter mais informações, consulte the section called “Credenciais com IRSA”.

EKS Pod Identities

O EKS Pod Identity oferece aos administradores de clusters um fluxo de trabalho simplificado para autenticar aplicações para acessar vários outros recursos da AWS, como buckets do Amazon S3, tabelas do Amazon DynamoDB e muito mais. O EKS Pod Identity é somente para EKS e, como resultado, simplifica como os administradores de cluster podem configurar aplicações Kubernetes para obter permissões do IAM. Essas permissões agora podem ser facilmente configuradas com menos etapas diretamente no Console de gerenciamento da AWS, na API do EKS e na AWS CLI, e não há nenhuma ação a ser executada dentro do cluster em nenhum objeto do Kubernetes. Os administradores de cluster não precisam alternar entre os serviços EKS e IAM nem usar operações privilegiadas do IAM para configurar as permissões exigidas por suas aplicações. Os perfis do IAM agora podem ser usados em vários clusters sem a necessidade de atualizar a política de confiança do perfil ao criar novos clusters. As credenciais do IAM fornecidas pelo EKS Pod Identity incluem tags de sessão de perfil, com atributos como nome do cluster, namespace e nome da conta de serviço. As tags de sessão de perfil permitem que os administradores criem um único perfil que pode funcionar em várias contas de serviço, permitindo o acesso aos recursos da AWS com base em tags correspondentes. Para obter mais informações, consulte the section called “Identidade de Pods”.

Comparação entre o EKS Pod Identity e o IRSA

Em alto nível, tanto o EKS Pod Identity quanto o IRSA permitem conceder permissões do IAM para aplicações executadas em clusters do Kubernetes. No entanto, eles são fundamentalmente diferentes na forma como você os configura, nos limites aceitos e nos recursos ativados. A seguir, comparamos algumas das principais características de ambas as soluções.

> **Note**
>
> A AWS recomenda usar as Identidades de Pod do EKS para conceder acesso aos recursos da AWS dos seus pods sempre que possível. Para obter mais informações, consulte the section called “Identidade de Pods”.

Atributo	EKS Pod Identity	IRSA
Extensibilidade de perfis	É necessário configurar cada perfil uma vez para estabelecer confiança com a recém-introduzida entidade principal do serviço Amazon EKS `pods.eks.amazonaws.com`. Após essa etapa única, não será necessário atualizar a política de confiança do perfil sempre que ela for usada em um novo cluster.	Você precisará atualizar a política de confiança do perfil do IAM com o novo endpoint do provedor OIDC do cluster de EKS sempre que quiser usar o perfil em um novo cluster.
Escalabilidade de clusters	O EKS Pod Identity não exige que os usuários configurem o provedor OIDC do IAM, portanto, esse limite não se aplica.	Cada cluster de EKS tem um URL do emissor do OpenID Connect (OIDC) associado a ele. Para usar o IRSA, é necessário criar um provedor OpenID Connect exclusivo para cada cluster de EKS no IAM. O IAM tem um limite global padrão de cem provedores OIDC para cada conta da AWS. Se você planeja ter mais de cem clusters do EKS para cada conta da AWS com o IRSA, você atingirá o limite do provedor OIDC do IAM.
Escalabilidade de funções	O EKS Pod Identity não exige que os usuários definam uma relação de confiança entre o perfil do IAM e a conta de serviço na política	No IRSA, é necessário definir a relação de confiança entre um perfil do IAM e uma conta de serviço na política de confiança do perfil. Por

Atributo	EKS Pod Identity	IRSA
	de confiança, portanto, esse limite não se aplica.	padrão, a duração do tamanho da política de confiança é `2048`. Isso significa que você pode definir quatro relações de confiança normalmen te em uma única política de confiança. Embora seja possível aumentar o limite de duração da política de confiança, em geral você tem um limite máximo de 8 relações de confiança em uma única política de confiança.
Uso da cota da API do STS	A Identidade de Pods do EKS simplifica a entrega de credenciais da AWS aos seus pods e não exige que seu código faça chamadas diretamente com o AWS Security Token Service (STS). O serviço EKS lida com a atribuição de perfis e fornece credenciais para aplicações escritas usando o AWS SDK em seus pods sem que eles se comuniquem com o AWS STS ou usem a cota da API STS.	No IRSA, aplicações escritas usando o AWS SDK em seus pods usam tokens para chamar a API `AssumeRol eWithWebIdentity` no AWS Security Token Service (STS). Dependendo da lógica do seu código no AWS SDK, é possível que seu código faça chamadas desnecessárias para o AWS STS e receba erros de limitação.

Atributo	EKS Pod Identity	IRSA
Reutilização de perfis	AWS As credenciais temporárias do STS fornecidas pelo EKS Pod Identity incluem tags de sessão de perfil, como nome do cluster, namespace, nome da conta de serviço. As tags de sessão de perfil permitem que os administradores criem um único perfil do IAM que pode ser usado com várias contas de serviço, com diferentes permissões efetivas, permitindo acesso a recursos da AWS com base nas tags anexadas a elas. Isso também é chamado controle de acesso por atributo (ABAC). Para obter mais informações, consulte the section called “Conceder acesso aos pods”.	AWS Não há suporte para tags de sessão STS. Você pode reutilizar um perfil entre clusters, mas cada pod recebe todas as permissões do perfil.
Ambientes compatíveis	O EKS Pod Identity só está disponível no Amazon EKS.	O IRSA pode ser usado como o Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service na AWS e clusters autogerenciados do Kubernetes em instâncias do Amazon EC2.

Atributo	EKS Pod Identity	IRSA
Versões do EKS compatíveis	Todas as versões de cluster de EKS compatíveis. Para obter números de versão específicos, consulte the section called “Versões de cluster do EKS Pod Identity”.	Todas as versões de cluster de EKS compatíveis.

Perfis do IAM para contas de serviço

Os aplicações nos contêineres de um Pod podem usar um SDK AWS ou a CLI AWS para fazer solicitações de API para serviços AWS usando permissões de gerenciamento de identidade e acesso (IAM) AWS. As aplicações devem assinar suas solicitações de API da AWS com as credenciais da AWS. Os perfis do IAM para contas de serviço (IRSA) permitem gerenciar credenciais para suas aplicações, de modo semelhante à forma como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2. Em vez de criar e distribuir as credenciais da AWS para os contêineres ou usar o perfil da instância do Amazon EC2, você pode associar um perfil do IAM a uma conta de serviço do Kubernetes e configurar os pods para usar a conta de serviço. Não é possível usar perfis do IAM para contas de serviço com clusters locais para o Amazon EKS no AWS Outposts.

Os perfis do IAM para contas de serviço oferecem as seguintes vantagens:

- Privilégio mínimo: você pode definir o escopo das permissões do IAM para uma conta de serviço, e somente os pods que usarem essa conta de serviço terão acesso a essas permissões. Esse recurso também elimina a necessidade de soluções de terceiros, como o `kiam` ou o `kube2iam`.
- Isolamento de credenciais: quando o acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 estiver restrito, um contêiner de pods só poderá recuperar as credenciais do perfil do IAM que estiver associado à conta de serviço que o contêiner usa. Um contêiner nunca tem acesso a credenciais usadas por outros contêineres em outros pods. Se o IMDS não estiver restrito, os contêineres do pod também terão acesso ao perfil do IAM do nó do Amazon EKS, e os contêineres poderão obter acesso às credenciais dos perfis do IAM de outros pods no mesmo nó. Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

> **Note**
>
> Os pods configurados com `hostNetwork: true` sempre terão acesso ao IMDS, mas a CLI e os SDKs da AWS usarão as credenciais do IRSA quando habilitados.

- Auditabilidade: o log de acesso e de eventos está disponível no AWS CloudTrail para ajudar a garantir a auditoria retrospectiva.

> **Important**
>
> Os contêineres não são um limite de segurança, e o uso dos perfis do IAM para contas de serviço não muda isso. Os pods atribuídos ao mesmo nó compartilharão um kernel e possivelmente outros recursos, dependendo da configuração do pod. Embora os pods executados em nós separados sejam isolados na camada de computação, existem aplicações de nós que têm permissões adicionais na API do Kubernetes além do escopo de uma instância individual. Alguns exemplos são `kubelet`, `kube-proxy`, drivers de armazenamento CSI ou suas próprias aplicações do Kubernetes.

Habilite perfis do IAM para contas de serviço concluindo os seguintes procedimentos:

1. Criar um provedor OIDC do IAM para o cluster: você só deve concluir esse procedimento uma vez para cada cluster.

   > **Note**
   >
   > Se você habilitou o endpoint da VPC do EKS, o endpoint de serviço de OIDC do EKS não poderá ser acessado de dentro dessa VPC. Consequentemente, suas operações, como a criação de um provedor de OIDC `eksctl` dentro da VPC, não funcionarão e resultarão em um tempo limite ao tentar solicitar `https://oidc.eks.region.amazonaws.com`. Segue um exemplo de mensagem de erro:
   >
   > ```
   > server cant find oidc.eks.region.amazonaws.com: NXDOMAIN
   > ```
   >
   > Para concluir essa etapa, você pode executar o comando fora da VPC, por exemplo, no AWS CloudShell ou em um computador conectado à Internet. Como alternativa, você

pode criar um resolvedor condicional de horizonte segmentado na VPC, como o Route 53 Resolver, para usar um resolvedor diferente para o URL do emissor do OIDC e não usar o DNS da VPC para ela. Para ver um exemplo de encaminhamento condicional em CoreDNS, consulte Amazon EKS feature request no GitHub.

2. Atribuir perfis do IAM a contas de serviço do Kubernetes: conclua este procedimento para cada conjunto exclusivo de permissões que você deseja que uma aplicação tenha.
3. Configurar pods para usar uma conta de serviço do Kubernetes: conclua este procedimento para cada pod que precisa de acesso aos serviços da AWS.
4. Usar IRSA com o AWS SDK: verifique se a workload usa um AWS SDK de uma versão compatível e se a workload usa a cadeia de credenciais padrão.

Informações de contexto sobre IAM, Kubernetes e OpenID Connect (OIDC)

Em 2014, o AWS Identity and Access Management adicionou suporte para identidades federadas usando o OpenID Connect (OIDC). Esse recurso permite que você autentique chamadas de API da AWS com provedores de identidade compatíveis e receba um token de web JSON (JWT) OIDC válido. Você pode passar esse token para a operação da API AWS STS `AssumeRoleWithWebIdentity` e receber credenciais de perfil temporária do IAM. Você pode usar essas credenciais para interagir com qualquer serviço do AWS, inclusive o Amazon S3 e o DynamoDB.

Cada token JWT é assinado por um par de chaves de assinatura. As chaves são servidas no provedor OIDC gerenciado pelo Amazon EKS e a chave privada é trocada a cada 7 dias. O Amazon EKS mantém as chaves públicas até a sua expiração. Se você conectar clientes OIDC externos, esteja ciente de que precisará atualizar as chaves de assinatura antes que a chave pública expire. Saiba como Buscar chaves de assinatura para validar tokens OIDC.

O Kubernetes usa há muito tempo contas de serviço como seu próprio sistema de identidade interno. Os pods podem ser autenticados com o servidor de API do Kubernetes usando um token montado automaticamente (que era um JWT não OIDC) que somente o servidor de API do Kubernetes podia validar. Esses tokens de conta de serviço legados não expiram, e a rotação da chave de assinatura é um processo difícil. Na versão `1.12` do Kubernetes, foi adicionado suporte para um novo recurso `ProjectedServiceAccountToken`. Esse recurso é um JSON web token do OIDC que também contém a identidade da conta de serviço e é compatível com um público configurável.

O Amazon EKS hospeda um endpoint público de descoberta do OIDC em cada cluster que contém as chaves de assinatura para JSON web tokens `ProjectedServiceAccountToken`, de modo que sistemas externos, como o IAM, possam validar e aceitar os tokens do OIDC emitidos pelo Kubernetes.

Criar um provedor de identidade OIDC do IAM para o cluster

O cluster tem um URL do emissor do OpenID Connect (OIDC) associado a ele. Para usar os perfis do AWS Identity and Access Management (IAM) para contas de serviço, deve haver um provedor OIDC do IAM para o URL do emissor do OIDC do cluster.

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Conceitos básicos.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Um arquivo `kubectl config` existente que contém a configuração do seu cluster. Para criar um arquivo `kubectl config`, consulte the section called “Acessar meu cluster com o kubectl”.

Você pode criar um provedor OIDC do IAM para o cluster usando o `eksctl` ou o Console de gerenciamento da AWS.

Criar provedor OIDC (eksctl)

1. Versão `0.215.0` ou posterior da ferramenta de linha de comando da `eksctl` instalada no seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
2. Determine o ID do emissor do OIDC para o cluster.

   Recupere o ID do emissor do OIDC do cluster e armazene-o em uma variável. Substitua `<my-cluster>` pelos seus próprios valores.

   ```
   cluster_name=<my-cluster>
   oidc_id=$(aws eks describe-cluster --name $cluster_name --query
    "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
   echo $oidc_id
   ```

3. Determine se um provedor OIDC do IAM com o ID do emissor do cluster já está em sua conta.

   ```
   aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
   ```

   Caso retorne um resultado, significa que você já tem um provedor OIDC do IAM para o cluster e pode pular a próxima etapa. Se nenhum resultado for retornado, você deverá criar um provedor OIDC do IAM do seu cluster.

4. Crie o provedor de identidade OIDC do IAM para o cluster com o comando a seguir.

   ```
   eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
   ```

   > **Note**
   >
   > Se você habilitou o endpoint da VPC do EKS, o endpoint de serviço de OIDC do EKS não poderá ser acessado de dentro dessa VPC. Consequentemente, suas operações, como a criação de um provedor de OIDC com o `eksctl` na VPC, não funcionarão e resultarão em um tempo limite. Segue um exemplo de mensagem de erro:
   >
   > ```
   > ** server cant find oidc.eks.<region-code>.amazonaws.com: NXDOMAIN
   > ```

Para concluir essa etapa, você pode executar o comando fora da VPC, por exemplo, no AWS CloudShell ou em um computador conectado à Internet. Como alternativa, você pode criar um resolvedor condicional de horizonte segmentado na VPC, como o Route 53 Resolver, para usar um resolvedor diferente para o URL do emissor do OIDC e não usar o DNS da VPC para ela. Para ver um exemplo de encaminhamento condicional em CoreDNS, consulte Amazon EKS feature request no GitHub.

Criar provedor OIDC (console do AWS)

1. Abra o console do Amazon EKS.
2. No painel esquerdo, selecione Clusters e, em seguida, selecione o nome do cluster na página Clusters.
3. Na seção Details (Detalhes) da guia Overview (Visão geral), observe o valor de OpenID Connect provider URL (URL do provedor OpenID Connect).
4. Abra o console do IAM em https://console.aws.amazon.com/iam/.
5. No painel de navegação à esquerda, escolha Identity Providers (Provedores de identidade) em Access management (Gerenciamento de acesso). Se um fornecedor listado corresponder à URL do cluster, então você já tem um provedor para o cluster. Se um provedor não estiver listado que corresponda à URL do seu cluster, você deverá criar um.
6. Para criar um provedor, selecione Add provider (Adicionar provedor).
7. Em Tipo de provedor, selecione OpenID Connect.
8. Em URL do provedor, insira o URL do provedor OIDC do cluster.
9. Em Público, insira `sts.amazonaws.com`.
10. (Opcional) Adicione qualquer tag, por exemplo, uma tag para identificar qual cluster destina-se a esse provedor.
11. Escolha Add provider (Adicionar provedor).

Próxima etapa: the section called “Atribuir perfil do IAM”

Atribuir perfis do IAM às contas de serviço do Kubernetes

Este tópico aborda como configurar uma conta de serviço do Kubernetes para assumir um perfil do AWS Identity and Access Management (IAM). Qualquer pod que esteja configurado para usar a conta de serviço pode acessar qualquer serviço da AWS que o perfil tenha permissão para acessar.

Pré-requisitos

- Um cluster existente. Se você não tiver um, poderá criá-lo seguindo um dos guias em Conceitos básicos.
- Um provedor do IAM OpenID Connect (OIDC) para o cluster. Para saber se você já tem um ou como criar um, consulte the section called “Provedor OIDC do IAM”.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Um arquivo `kubectl config` existente que contém a configuração do seu cluster. Para criar um arquivo `kubectl config`, consulte the section called “Acessar meu cluster com o kubectl”.

Etapa 1: criar a política do IAM

Se você quiser associar uma política do IAM existente ao seu perfil do IAM, vá para a próxima etapa.

1. Crie uma política do IAM. É possível criar a sua própria política ou copiar uma política gerenciada da AWS que já conceda algumas das permissões de que você precisa e a personalizar de acordo com seus requisitos específicos. Para obter mais informações, consulte Criar políticas do IAM no Guia do usuário do IAM.
2. Crie um arquivo que inclua as permissões para os serviços da AWS que você deseja que os pods acessem. Para obter uma lista de todas as ações de todos os serviços do AWS, consulte a Referência de autorização de serviço.

   É possível executar o comando a seguir para criar um arquivo de política de exemplo que permita acesso somente leitura a um bucket do Amazon S3. Você pode, opcionalmente, armazenar

informações de configuração ou um script de bootstrap nesse bucket, e os contêineres no pod podem ler o arquivo do bucket e carregá-lo na aplicação. Se você quiser criar esse exemplo de política, copie o conteúdo a seguir para o seu dispositivo. Substitua *my-pod-secrets-bucket* pelo nome do seu bucket e execute o comando.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

3. Crie a política do IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

Etapa 2: criar e associar o perfil do IAM

Crie um perfil do IAM e associe-o a uma conta de serviço do Kubernetes. Você pode usar o `eksctl` ou a AWS CLI.

Criar e associar um perfil (eksctl)

Esse comando `eksctl` cria uma conta de serviço do Kubernetes no namespace especificado, cria um perfil do IAM (se não existir) com o nome especificado, anexa um ARN de política do IAM existente ao perfil e anota a conta de serviço com o ARN do perfil do IAM. Certifique-se de substituir os valores de espaço reservado de exemplo neste comando por seus valores específicos. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

```
eksctl create iamserviceaccount --name my-service-account --namespace default --cluster
 my-cluster --role-name my-role \
    --attach-policy-arn arn:aws:iam::111122223333:policy/my-policy --approve
```

> **Important**
>
> Se a função ou a conta de serviço já existir, o comando anterior poderá falhar. O `eksctl` tem opções diferentes que você pode oferecer nessas situações. Para obter mais informações, execute `eksctl create iamserviceaccount --help`.

Criar e associar um perfil (AWS CLI)

Se você já tem um conta de serviço do Kubernetes na qual deseja assumir um perfil do IAM, você pode ignorar esta etapa.

1. Crie uma conta de serviço do Kubernetes. Copie o conteúdo a seguir para o seu dispositivo. Substitua *my-service-account* pelo nome desejado e *default* (padrão) por um namespace diferente, se necessário. Se você alterar o *padrão*, o namespace já deve existir.

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

2. Defina o ID da conta da AWS como uma variável de ambiente com o comando a seguir.

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

3. Defina o provedor de identidade OIDC do cluster como uma variável de ambiente com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --
query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
```

4. Defina variáveis para o namespace e o nome da conta de serviço. Substitua *my-service-account* pela conta de serviço do Kubernetes que você deseja que assuma o perfil. Substitua *default* (padrão) pelo namespace da conta de serviço.

```
export namespace=default
```

```
export service_account=my-service-account
```

5. Execute o seguinte comando para criar um arquivo de política de confiança para o perfil do IAM. Se você quiser permitir que todas as contas de serviço em um namespace usem o perfil, copie o conteúdo a seguir para o seu dispositivo. Substitua *`StringEquals`* por `StringLike` e substitua *`$service_account`* por `*`. É possível adicionar várias entradas nas condições `StringEquals` ou `StringLike` abaixo para permitir que várias contas de serviço ou namespaces assumam o perfil. Para permitir que perfis de uma conta da AWS diferente da conta em que seu cluster está assumam o perfil, consulte the section called “IAM entre contas” para obter mais informações.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789012:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
      }
    }
  ]
}
```

6. Crie a função. Substitua *`my-role`* por um nome para seu perfil do IAM e *`my-role-description`* por uma descrição para seu perfil.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

7. Associe uma política do IAM ao seu perfil. Substitua *`my-role`* pelo nome do seu perfil do IAM e *`my-policy`* pelo nome de uma política existente que você criou.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:iam::
$account_id:policy/my-policy
```

8. Anote sua conta de serviço com o nome do recurso da Amazon (ARN) do perfil do IAM que você deseja que a conta de serviço assuma. Substitua *my-role* pelo nome do seu perfil do IAM existente. Suponha que você tenha permitido que um perfil de uma conta AWS diferente da conta na qual seu cluster está assumisse o perfil em uma etapa anterior. Em seguida, certifique-se de especificar a conta AWS e o perfil da outra conta. Para obter mais informações, consulte the section called “IAM entre contas”.

```
kubectl annotate serviceaccount -n $namespace $service_account eks.amazonaws.com/
role-arn=arn:aws:iam::$account_id:role/my-role
```

9. (Opcional) Configure o endpoint do AWS Security Token Service para uma conta de serviço. A AWS recomenda o uso de um endpoint regional do AWS STS em vez do endpoint global. Isso reduz a latência, fornece redundância integrada e aumenta a validade do token da sessão.

Etapa 3: confirmar a configuração

1. Confirme se a política de confiança do perfil do IAM está configurada corretamente.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

   Veja abaixo um exemplo de saída.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-service-
account",
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
                }
```

```
            }
        }
    ]
}
```

2. Confirme se a política que você anexou ao seu perfil em uma etapa anterior está vinculada ao perfil.

```
aws iam list-attached-role-policies --role-name my-role --query
 "AttachedPolicies[].PolicyArn" --output text
```

Veja um exemplo de saída abaixo.

```
arn:aws:iam::111122223333:policy/my-policy
```

3. Defina uma variável para armazenar o nome do recurso da Amazon (ARN) da política que deseja usar. Substitua *my-policy* pelo nome da política para a qual deseja confirmar permissões.

```
export policy_arn=arn:aws:iam::111122223333:policy/my-policy
```

4. Visualize a versão padrão da política.

```
aws iam get-policy --policy-arn $policy_arn
```

Veja abaixo um exemplo de saída.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws:iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. Confira o conteúdo da política para garantir que ela inclua todas as permissões de que seu pod precisa. Se necessário, substitua*1* no comando a seguir pela versão retornada na saída anterior.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

Veja abaixo um exemplo de saída.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Se você criou a política de exemplo em uma etapa anterior, sua saída será a mesma. Se você criou uma política diferente, então o *exemplo* de conteúdo é diferente.

6. Verifique se a conta do serviço do Kubernetes está anotada com o perfil.

```
kubectl describe serviceaccount my-service-account -n default
```

Veja abaixo um exemplo de saída.

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/my-
role
Image pull secrets:  <none>
Mountable secrets:   my-service-account-token-qqjfl
Tokens:              my-service-account-token-qqjfl
[...]
```

Próximas etapas

- the section called “Atribuir ao pod”

Configurar pods para usar uma conta de serviço do Kubernetes

Se um pod precisar acessar os serviços da AWS, você deverá configurá-lo para usar uma conta de serviço do Kubernetes. A conta de serviço deve estar associada a um perfil do AWS Identity and Access Management (IAM) que tenha permissões para acessar os serviços da AWS.

- Um cluster existente. Se não tiver um, você poderá criá-lo usando um dos guias em Conceitos básicos.
- Um provedor do IAM OpenID Connect (OIDC) para o cluster. Para saber se você já tem um ou como criar um, consulte the section called “Provedor OIDC do IAM”.
- Uma conta de serviço existente do Kubernetes que é associada a um perfil do IAM. A conta de serviço deve ser anotada com o nome do recurso da Amazon (ARN) do perfil do IAM. O perfil deve ter uma política do IAM associada que contenha as permissões que você deseja que os pods tenham para usar os serviços da AWS. Para obter mais informações sobre como criar a conta de serviço e como configurá-la, consulte the section called “Atribuir perfil do IAM”.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Um arquivo `kubectl config` existente que contém a configuração do seu cluster. Para criar um arquivo `kubectl config`, consulte the section called “Acessar meu cluster com o kubectl”.

  1. Use o comando a seguir para criar um manifesto de implantação com o qual é possível implantar um pod para confirmar a configuração. Substitua os valores de exemplo por seus próprios valores.

```
cat >my-deployment.yaml <<EOF
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Implante o manifesto no seu cluster.

```
kubectl apply -f my-deployment.yaml
```

3. Verifique se as variáveis de ambiente necessárias existem para o pod.

   a. Visualize os pods que foram implantados na implantação da etapa anterior.

   ```
   kubectl get pods | grep my-app
   ```

   Veja abaixo um exemplo de saída.

   ```
   my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
   ```

   b. Visualize o ARN do perfil do IAM que o pod está usando.

   ```
   kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
   ```

   Veja abaixo um exemplo de saída.

   ```
   AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/my-role
   ```

O ARN do perfil deve corresponder ao ARN do perfil com o qual você anotou a conta de serviço existente. Para saber mais sobre como fazer anotações na conta de serviço, consulte the section called “Atribuir perfil do IAM”.

c. Verifique se o pod tem um arquivo de token de identidade da web montado.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_WEB_IDENTITY_TOKEN_FILE:
```

Veja abaixo um exemplo de saída.

```
AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/serviceaccount/
token
```

O `kubelet` solicita e armazena o token em nome do pod. Por padrão, o `kubelet` atualiza o token se ele for mais antigo do que 80% do seu tempo de vida total, ou se o token tiver mais de 24 horas. Você pode modificar a duração da expiração de qualquer conta, exceto a conta de serviço padrão, usando as configurações na especificação do pod. Para obter mais informações, consulte Service Account Token Volume Projection (Projeção do volume de tokens da conta de serviço) na documentação do Kubernetes.

O Webhook de Identidade de Pods do Amazon EKS no cluster observa os pods que usam uma conta de serviço com a seguinte anotação:

```
eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/my-role
```

O webhook aplica as variáveis de ambiente anteriores a esses pods. Seu cluster não precisa usar o webhook para configurar as variáveis de ambiente e montagens de arquivos do token. Você pode configurar manualmente os pods para ter essas variáveis de ambiente. As versões compatíveis do SDK da AWS procuram essas variáveis de ambiente primeiro no fornecedor da cadeia de credenciais. As credenciais de perfil são usadas para os pods que atendem a esses critérios.

4. Verifique se os pods podem interagir com os serviços da AWS usando as permissões que você atribuiu na política do IAM anexada ao seu perfil.

> **Note**
>
> Quando um pod usa as credenciais da AWS de um perfil do IAM associado a uma conta de serviço, a AWS CLI ou outros SDKs nos contêineres desse pod usam as credenciais fornecidas por esse perfil. Se você não restringir o acesso às credenciais fornecidas ao perfil do IAM do nó do Amazon EKS, o pod ainda terá acesso a elas. Para obter mais informações, consulte Restringir o acesso ao perfil de instância atribuído ao nó de processamento.

Se os pods não conseguirem interagir com os serviços conforme o esperado, conclua as etapas a seguir para verificar se tudo está configurado corretamente.

a. Verifique se os pods usam uma versão do AWS SDK que permita assumir um perfil do IAM por meio de um arquivo de token de identidade da web do OpenID Connect. Para obter mais informações, consulte the section called “SDKs compatíveis”.

b. Confirme se a implantação está usando a conta de serviço.

```
kubectl describe deployment my-app | grep "Service Account"
```

Veja abaixo um exemplo de saída.

```
Service Account:  my-service-account
```

c. Se os pods ainda não conseguirem acessar os serviços, revise as etapas descritas em Atribuir perfis do IAM a contas de serviço do Kubernetes para verificar se seu perfil e conta de serviço estão configurados corretamente.

Configurar o endpoint do AWS Security Token Service para uma conta de serviço

Se estiver usando uma conta de serviço do Kubernetes com perfis de IAM para contas de serviço, você poderá configurar o tipo de endpoint do AWS Security Token Service usado pela conta de serviço.

A AWS recomenda usar os endpoints regionais do AWS STS em vez do endpoint global. Isso reduz a latência, fornece redundância integrada e aumenta a validade do token da sessão. O AWS Security Token Service deve estar ativo na região da AWS onde o pod está sendo executado. Além disso,

sua aplicação deve ter redundância integrada para uma região da AWS diferente no caso de uma falha do serviço na região da AWS. Para obter mais informações, consulte Gerenciar o AWS STS em uma região da AWS, no Guia do usuário do IAM.

- Um cluster existente. Se não tiver um, você poderá criá-lo usando um dos guias em Conceitos básicos.
- Um provedor de OIDC do IAM existente para o cluster. Para obter mais informações, consulte the section called “Provedor OIDC do IAM”.
- Uma conta de serviço existente do Kubernetes configurada para uso com o recurso Amazon EKS IAM para contas de serviço.

Todos os exemplos a seguir usam a conta de serviço aws-node do Kubernetes usada pelo plug-in CNI da Amazon VPC. É possível substituir os *valores de exemplo* pelas suas próprias contas de serviço, pods, namespaces e outros recursos.

1. Selecione um pod que use uma conta de serviço para a qual você deseja alterar o endpoint. Determine em qual região da AWS o pod é executado. Substitua *aws-node-6mfgv* pelo nome do pod e *kube-system* pelo namespace do seu pod.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
   ```

   Veja abaixo um exemplo de saída.

   ```
   ip-192-168-79-166.us-west-2/192.168.79.166
   ```

   Na saída anterior, o pod está sendo executado em um nó na região us-west-2 da AWS.
2. Determine o tipo de endpoint que a conta de serviço do pod estará usando.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
   ```

   Veja abaixo um exemplo de saída.

   ```
   AWS_STS_REGIONAL_ENDPOINTS: regional
   ```

   Se o endpoint atual for global, `global` será retornado na saída. Se nenhuma saída for retornada, significa que tipo de endpoint padrão está em uso e não foi substituído.

3. Se a versão do cluster ou plataforma for a mesma ou posterior à da tabela, você poderá alterar o tipo de endpoint utilizado pela conta de serviço do tipo padrão para um tipo diferente, usando um dos comandos a seguir. Substitua *aws-node* pelo nome da conta de serviço e *aws-node* pelo namespace da sua conta de serviço.

   - Se o tipo de endpoint padrão ou atual for global e você quiser modificá-lo para regional:

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
     ```

     Se você estiver usando perfis do IAM para contas de serviço para gerar URLs do S3 pré-assinados na aplicação em execução em contêineres de pods, o formato do URL para endpoints regionais será semelhante ao seguinte exemplo:

     ```
     https://bucket.s3.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
     ```

   - Se o tipo de endpoint padrão ou atual for global, e você quiser modificá-lo para regional:

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
     ```

     Se a sua aplicação estiver fazendo solicitações explicitamente para endpoints globais do AWS STS e você não substituir o comportamento padrão de usar endpoints regionais em clusters do Amazon EKS, as solicitações falharão com um erro. Para obter mais informações, consulte the section called “Os contêineres do pod recebem o seguinte erro: `An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`”.

     Se você estiver usando perfis do IAM para contas de serviço para gerar URLs do S3 pré-assinados na sua aplicação em execução em contêineres de pods, o formato do URL para endpoints globais será semelhante ao seguinte exemplo:

     ```
     https://bucket.s3.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
     ```

   Se você tem uma automação que espera o URL pré-assinado em um determinado formato ou se a sua aplicação ou dependências downstream que usam URLs pré-assinados têm expectativas

para a região da AWS visada, faça as alterações necessárias para usar o endpoint do AWS STS apropriado.

4. Exclua e recrie os pods existentes associados à conta de serviço para aplicar as variáveis de credenciais do ambiente. O webhook de mutação não as aplica aos pods que já estão em execução. É possível substituir *pods*, *kube-system* e *-l k8s-app=aws-node* pelas informações dos pods para os quais você definiu sua anotação.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

5. Verifique se todos os pods foram reiniciados.

```
kubectl get Pods -n kube-system -l k8s-app=aws-node
```

6. Visualize as variáveis de ambiente de um dos pods. Verifique se o valor `AWS_STS_REGIONAL_ENDPOINTS` é o que você o definiu em uma etapa anterior.

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

Veja abaixo um exemplo de saída.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Autenticar em outra conta com IRSA

É possível configurar permissões entre contas do IAM criando um provedor de identidade do cluster de outra conta ou usando operações `AssumeRole` encadeadas. Nos exemplos a seguir, a Conta A tem um cluster do Amazon EKS compatível com perfis do IAM para contas de serviço. Os pods em execução nesse cluster devem assumir as permissões do IAM da Conta B.

Example Criar um provedor de identidade de cluster de outra conta

Example

Neste exemplo, a Conta A fornece à Conta B o URL do emissor de OpenID Connect (OIDC) de seu cluster. A conta B segue as instruções em Criar um provedor OIDC do IAM para o cluster e the section called “Atribuir perfil do IAM” usando o URL do emissor do OIDC do cluster da Conta A. Em seguida, um administrador de cluster anota a conta de serviço no cluster da Conta A para usar o perfil da Conta B (*444455556666*).

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::444455556666:role/account-b-role
```

Example Usar operações **`AssumeRole`** encadeadas

Example

Neste exemplo, a Conta B cria uma política do IAM com as permissões a serem concedidas aos pods no cluster da Conta A. A Conta B (*444455556666*) anexa essa política a um perfil do IAM com uma relação de confiança que concede permissões `AssumeRole` à Conta A (*111122223333*).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

A conta A cria um perfil com uma política de confiança que obtém credenciais do provedor de identidade criado com o endereço do emissor de OIDC do cluster.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
```

```
        "StringEquals": {
          "oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

A Conta A anexa uma política a essa função com as seguintes permissões para assumir a função criada pela Conta B.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::444455556666:role/account-b-role"
        }
    ]
}
```

A aplicação cria o código para que os pods assumam que o perfil da Conta B usa dois perfis: `account_b_role` e `account_a_role`. O perfil `account_b_role` usa o perfil `account_a_role` como origem. Para a CLI AWS, o arquivo `~/.aws/config` é semelhante ao seguinte.

```
[profile account_b_role]
source_profile = account_a_role
role_arn=arn:aws:iam::444455556666:role/account-b-role

[profile account_a_role]
web_identity_token_file = /var/run/secrets/eks.amazonaws.com/serviceaccount/token
role_arn=arn:aws:iam::111122223333:role/account-a-role
```

Para especificar perfis encadeados para outras SDKs da AWS, consulte a documentação do SDK em uso. Para obter mais informações, consulte Ferramentas para criar na AWS.

Usar o IRSA com o AWS SDK

Usar as credenciais

Para usar as credenciais dos perfis do IAM para contas de serviço (IRSA), seu código pode usar qualquer SDK da AWS para criar um cliente para um serviço da AWS com um SDK e, por padrão, o SDK pesquisa em uma cadeia de locais as credenciais do AWS Identity and Access Management a serem usadas. As credenciais do perfis do IAM para contas de serviço serão usadas se você não especificar um provedor de credenciais ao criar o cliente ou ao inicializar o SDK.

Isso funciona porque os perfis do IAM para contas de serviço foram adicionados como uma etapa na cadeia de credenciais padrão. Se suas workloads usam no momento credenciais mais antigas na cadeia de credenciais, essas credenciais continuarão sendo usadas mesmo se você configurar perfis do IAM para contas de serviço para a mesma workload.

O SDK troca automaticamente o token OIDC da conta de serviço por credenciais temporárias do AWS Security Token Service usando a ação `AssumeRoleWithWebIdentity`. O Amazon EKS e essa ação do SDK continuam alternando as credenciais temporárias, renovando-as antes que elas expirem.

Ao usar perfis do IAM para contas de serviço, os contêineres nos pods devem usar uma versão do AWS SDK que permita assumir um perfil do IAM por meio de um arquivo de token de identidade da web do OpenID Connect. Certifique-se de usar as seguintes versões, ou versões posteriores, do AWS SDK:

- Java (versão 2): 2.10.11
- Java: 1.12.782
- AWS SDK para Go v1: 1.23.13
- AWS SDK para Go v2: suporte para todas as versões
- Python (Boto3): 1.9.220
- Python (botocore): 1.12.200
- AWS CLI – 1.16.232
- Node: 2.525.0 e 3.27.0
- Ruby: 3.58.0
- C++: 1.7.174
- .NET: 3.3.659.1 - Você também deve incluir o `AWSSDK.SecurityToken`.
- PHP: 3.110.7

Muitos complementos populares do Kubernetes, como o Cluster Autoscaler, o Roteamento do tráfego da internet com o AWS Load Balancer Controller e o plug-in CNI da Amazon VPC para Kubernetes, são compatíveis com perfis do IAM para contas de serviço.

Para garantir que você esteja usando um SDK compatível, siga as instruções de instalação do SDK de sua preferência em Ferramentas para desenvolver na AWS quando estiver criando seus contêineres.

Considerações

Java

Quando usar o Java, você deve incluir o `sts` no caminho da classe. Para obter mais informações, consulte WebIdentityTokenFileCredentialsProvider nos documentos SDK do Java.

Obter as chaves de assinatura para validar os tokens OIDC

O Kubernetes emite um `ProjectedServiceAccountToken` para cada conta de serviço do Kubernetes. Este token é um token OIDC, que também é um tipo de JSON web token (JWT). O Amazon EKS hospeda um endpoint público do OIDC em cada cluster que contém as chaves de assinatura para o token de modo que sistemas externos possam validá-lo.

Para validar um `ProjectedServiceAccountToken`, você precisa buscar as chaves públicas de assinatura do OIDC, também denominadas JSON Web Key Set (JWKS). Use essas chaves em sua aplicação para validar o token. Por exemplo, você pode usar a biblioteca PyJWT Python para validar tokens usando essas chaves. Para obter mais informações sobre o `ProjectedServiceAccountToken`, consulte o the section called “Informações de contexto sobre IAM, Kubernetes e OpenID Connect (OIDC)”.

Pré-requisitos

- Um provedor AWS Identity and Access Management (IAM) OpenID Connect (OIDC) para o seu cluster. Para determinar se você tem ou para criar uma, consulte the section called “Provedor OIDC do IAM”.
- AWS CLI- Uma ferramenta de linha de comando para trabalhar com os serviços AWS, incluindo o Amazon EKS. Para obter mais informações, consulte Instalação no Guia do Usuário da Interface de Linha de Comando AWS. Depois de instalar a AWS CLI, recomendamos que você também a configure. Para saber mais, consulte Configuração rápida com aws configure, no Guia do usuário da AWS Command Line Interface.

Procedimento

1. Recupere o URL do OIDC para o cluster do Amazon EKS usando a AWS CLI.

```
$ aws eks describe-cluster --name my-cluster --query 'cluster.identity.oidc.issuer'
"https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE"
```

2. Recupere a chave de assinatura pública usando o curl ou uma ferramenta similar. O resultado é um JSON Web Key Set (JWKS).

> **Important**
>
> O Amazon EKS controla a utilização das chamadas para o endpoint do OIDC. Você deve armazenar em cache a chave pública de assinatura. Respeite o cabeçalho `cache-control` incluído na resposta.

> **Important**
>
> O Amazon EKS alterna a chave de assinatura do OIDC a cada sete dias.

```
$ curl https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE/keys
{"keys":
[{"kty":"RSA","kid":"2284XXXX4a40","use":"sig","alg":"RS256","n":"wklbXXXXMVfQ","e":"AQAB"}]
```

Saiba como a Identidade de Pods do EKS concede aos pods acesso aos serviços da AWS

Os aplicações nos contêineres de um Pod podem usar um SDK AWS ou a CLI AWS para fazer solicitações de API para serviços AWS usando permissões de gerenciamento de identidade e acesso (IAM) AWS. As aplicações devem assinar suas solicitações de API da AWS com as credenciais da AWS.

Os EKS Pod Identities permitem gerenciar credenciais para suas aplicações de forma semelhante a como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2. Em vez de criar e distribuir as credenciais da AWS para os contêineres ou usar o perfil da instância

do Amazon EC2, você pode associar um perfil do IAM a uma conta de serviço do Kubernetes e configurar os pods para usar a conta de serviço.

Cada associação ao EKS Pod Identity mapeia um perfil a uma conta de serviço em um namespace no cluster especificado. Se você tiver a mesma aplicação em vários clusters, poderá fazer associações idênticas em cada cluster sem modificar a política de confiança do perfil.

Se um pod usa uma conta de serviço que tem uma associação, o Amazon EKS define variáveis de ambiente nos contêineres do pod. As variáveis de ambiente configuram os SDKs da AWS, incluindo a AWS CLI, para usar as credenciais de Identidade de pods do EKS.

Benefícios dos EKS Pod Identities

Os EKS Pod Identities fornecem os seguintes benefícios:

- Privilégio mínimo: você pode definir o escopo das permissões do IAM para uma conta de serviço, e somente os pods que usarem essa conta de serviço terão acesso a essas permissões. Esse recurso também elimina a necessidade de soluções de terceiros, como o `kiam` ou o `kube2iam`.
- Isolamento de credenciais: quando o acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 estiver restrito, um contêiner de pods só poderá recuperar as credenciais do perfil do IAM que estiver associado à conta de serviço que o contêiner usa. Um contêiner nunca tem acesso a credenciais usadas por outros contêineres em outros pods. Se o IMDS não estiver restrito, os contêineres do pod também terão acesso ao perfil do IAM do nó do Amazon EKS, e os contêineres poderão obter acesso às credenciais dos perfis do IAM de outros pods no mesmo nó. Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

> **Note**
>
> Os pods configurados com `hostNetwork: true` sempre terão acesso ao IMDS, mas a CLI e os SDKs da AWS usarão as credenciais da Identidade de Pods quando ativados.

- Auditabilidade: o log de acesso e de eventos está disponível no AWS CloudTrail para ajudar a facilitar a auditoria retrospectiva.

> **Important**
>
> Os contêineres não são um limite de segurança, e o uso da Identidade de Pods não muda isso. Os pods atribuídos ao mesmo nó compartilharão um kernel e possivelmente outros recursos, dependendo da configuração do pod. Embora os pods executados em nós separados sejam isolados na camada de computação, existem aplicações de nós que têm permissões adicionais na API do Kubernetes além do escopo de uma instância individual. Alguns exemplos são `kubelet`, `kube-proxy`, drivers de armazenamento CSI ou suas próprias aplicações do Kubernetes.

O Identidade de Pods do EKS é um método mais simples do que the section called “Credenciais com IRSA”, pois esse método não usa provedores de identidade OIDC. O EKS Pod Identity oferece os seguintes aprimoramentos:

- Operações independentes: em muitas organizações, criar provedores de identidade OIDC é responsabilidade de equipes diferentes da administração dos clusters do Kubernetes. O EKS Pod Identity tem uma separação clara dos deveres em que toda a configuração das associações ao EKS Pod Identity é feita no Amazon EKS e toda a configuração das permissões do IAM é feita no IAM.
- Reutilização: o EKS Pod Identity usa uma única entidade principal do IAM em vez das entidades principais separadas para cada cluster que os perfis do IAM para contas de serviço utilizam. Seu administrador do IAM adiciona a seguinte entidade principal à política de confiança de qualquer perfil para torná-la utilizável pelo EKS Pod Identities.

  ```
  "Principal": {
      "Service": "pods.eks.amazonaws.com"
  }
  ```

- Escalabilidade: cada conjunto de credenciais temporárias é assumido pelo serviço de autenticação no EKS na Identidade de Pods do EKS, em vez de cada AWS SDK que você executa em cada pod. Em seguida, o Amazon EKS Pod Identity Agent executado em cada nó emite as credenciais para os SDKs. Assim, a carga é reduzida para uma vez para cada nó e não é duplicada em cada pod. Para obter mais detalhes do processo, consulte the section called “Como funciona”.

Para obter mais informações sobre como comparar as duas alternativas, consulte the section called “Acesso da workload à AWS ”.

Visão geral da configuração do EKS Pod Identities

Ative o EKS Pod Identities concluindo os seguintes procedimentos:

1. the section called “Configurar o atendente”: esse procedimento é realizado apenas uma vez para cada cluster. Você não precisará concluir esta etapa se o Modo Automático do EKS estiver habilitado no cluster.
2. the section called “Atribuir perfil do IAM”: conclua esse procedimento para cada conjunto exclusivo de permissões que você deseja que uma aplicação tenha.
3. the section called “Conta de serviço de pods”: conclua este procedimento para cada pod que precisa de acesso aos serviços da AWS.
4. the section called “SDKs compatíveis”: confirme que a workload usa um AWS SDK de uma versão compatível e que essa workload usa a cadeia de credenciais padrão.

Limites

- São aceitas até 5 mil associações de Identidades de Pods do EKS por cluster para mapear perfis do IAM para contas de serviço do Kubernetes.

Considerações

- Associação de perfil do IAM: cada conta de serviço do Kubernetes em um cluster pode ser associada a um perfil do IAM da mesma conta da AWS que o cluster. Para alterar o perfil, edite a associação de Identidades de Pods do EKS. Para acesso entre contas, delegue o acesso ao perfil usando perfis do IAM. Para obter mais informações, consulte Delegar acesso entre contas da AWS usando perfis do IAM no Guia do usuário do IAM.
- Agente da Identidade de Pods do EKS: o agente da Identidade de Pods é necessário para usar a Identidade de Pods do EKS. O agente é executado como um `DaemonSet` do Kubernetes em nós de cluster, fornecendo credenciais somente para pods no mesmo nó. Ele usa o `hostNetwork` do nó, ocupando a porta `80` e a `2703` no endereço local do link (`169.254.170.23` para IPv4, `[fd00:ec2::23]` para IPv6). Se o IPv6 estiver desabilitado em seu cluster, desabilite o IPv6 para o agente da Identidade de Pods. Para saber mais, consulte Desabilitar o IPv6 no agente da Identidade de Pods do EKS.
- Consistência eventual: as associações de Identidades de Pods do EKS acabam sendo consistentes, com possíveis atrasos de vários segundos após as chamadas de API. Evite criar ou atualizar associações em caminhos de código críticos e de alta disponibilidade. Em vez disso,

execute essas ações em rotinas de inicialização ou configuração separadas e menos frequentes. Para saber mais, consulte Security Groups Per Pod no EKS Best Practices Guide.

- Considerações sobre proxy e grupos de segurança: para pods que usam um proxy, adicione `169.254.170.23` (IPv4) e `[fd00:ec2::23]` (IPv6) às variáveis de ambiente `no_proxy/NO_PROXY` para evitar falhas nas solicitações ao agente da Identidade de Pods do EKS. Se estiver usando grupos de segurança para pods com a CNI da AWS VPC, defina a sinalização `ENABLE_POD_ENI` como “true” e a sinalização `POD_SECURITY_GROUP_ENFORCING_MODE` como “standard”. Para saber mais, consulte Atribuir grupos de segurança a pods individuais.

Versões de cluster do EKS Pod Identity

Para usar a Identidade de Pods do EKS, o cluster deve ter uma versão da plataforma igual ou posterior à versão listada na tabela a seguir, ou uma versão do Kubernetes posterior às versões listadas na tabela. Para encontrar a versão sugerida do agente da Identidade de Pods do EKS para uma versão do Kubernetes, consulte the section called “Verificação da compatibilidade”.

Versão do Kubernetes	Versão da plataforma
Versões do Kubernetes não listadas	Todas as versões compatíveis da plataforma
`1.28`	`eks.4`

Restrições do EKS Pod Identity

Os EKS Pod Identities estão disponíveis em:

- Versões de cluster do Amazon EKS listadas no tópico the section called “Versões de cluster do EKS Pod Identity” anterior.
- Nós de processamento no cluster que são instâncias do Linux do Amazon EC2.

Os EKS Pod Identities não estão disponíveis em:

- AWS Outposts.
- Amazon EKS Anywhere
- Clusters do Kubernetes que você cria e executa no Amazon EC2. Os componentes do EKS Pod Identity estão disponíveis somente no Amazon EKS.

Não é possível usar o EKS Pod Identities com:

- Pods executados em qualquer lugar, exceto instâncias do Linux do Amazon EC2. Não há suporte para pods Linux e Windows executados em AWS Fargate (Fargate). Não há suporte a pods executados em instâncias do Windows do Amazon EC2.

Como a Identidade de Pods do EKS funciona

As associações do Amazon EKS Pod Identity permitem gerenciar credenciais para suas aplicações de forma semelhante a como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2.

O Amazon EKS Pod Identity fornece credenciais para suas workloads com uma API EKS Auth adicional e um pod de agente que é executado em cada nó.

Em seus complementos, como os complementos do Amazon EKS e controladores autogerenciados, operadores e outros complementos, o autor precisa atualizar o software para usar os AWS SDKs mais recentes. Para ver a lista de compatibilidade entre o EKS Pod Identity e os complementos produzidos pelo Amazon EKS, consulte a seção the section called “Restrições do EKS Pod Identity” anterior.

Usar o EKS Pod Identities em seu código

É possível usar os AWS SDKs no código para acessar os serviços da AWS. Você escreve um código para criar um cliente para um serviço AWS com um SDK e, por padrão, o SDK pesquisa em uma cadeia de locais as credenciais do AWS Identity and Access Management a serem usadas. Depois que as credenciais válidas são encontradas, a pesquisa é interrompida. Para obter mais informações sobre os locais padrão usados, consulte a Cadeia de provedores de credenciais no Guia de referência de SDKs e ferramentas da AWS.

As identidades do EKS Pod foram adicionadas ao Provedor de credenciais do container, o qual é pesquisado em uma etapa na cadeia de credenciais padrão. Se atualmente suas workloads usam credenciais mais antigas na cadeia de credenciais, elas continuarão sendo usadas mesmo se você configurar uma associação do EKS Pod Identity para a mesma workload. Dessa forma, você pode migrar com segurança de outros tipos de credenciais criando a associação antes de remover as credenciais antigas.

O provedor de credenciais do contêiner fornece credenciais temporárias de um agente executado em cada nó. No Amazon EKS, o agente é o Amazon EKS Pod Identity Agent e, no Amazon Elastic

Container Service, o agente é o `amazon-ecs-agent`. Os SDKs usam variáveis de ambiente para localizar os agentes ao qual se conectarão.

Por outro lado, as perfis de IAM para contas de serviço fornecem um token de identidade da Web que o SDK AWS deve trocar com o Serviço de token de segurança AWS usando `AssumeRoleWithWebIdentity`.

Como a Identidade de Pods do EKS funciona com um Pod

1. Quando o Amazon EKS inicia um novo pod que usa uma conta de serviço com uma associação da Identidade de Pods do EKS, o cluster adiciona o seguinte conteúdo ao manifesto de pods:

```
    env:
    - name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
      value: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-
identity-token"
    - name: AWS_CONTAINER_CREDENTIALS_FULL_URI
      value: "http://169.254.170.23/v1/credentials"
    volumeMounts:
    - mountPath: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/"
      name: eks-pod-identity-token
  volumes:
  - name: eks-pod-identity-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          audience: pods.eks.amazonaws.com
          expirationSeconds: 86400 # 24 hours
          path: eks-pod-identity-token
```

2. O Kubernetes seleciona em qual nó o pod será executado. Em seguida, o Amazon EKS Pod Identity Agent no nó usa a ação AssumeRoleForPodIdentity para recuperar credenciais temporárias da API de autenticação do EKS.
3. O EKS Pod Identity Agent disponibiliza essas credenciais para os AWS SDKs que você executa em seus contêineres.
4. Você usa o SDK em sua aplicação sem especificar um provedor de credenciais para usar a cadeia de credenciais padrão. Ou você especifica o provedor de credenciais do contêiner. Para obter mais informações sobre os locais padrão usados, consulte a Cadeia de provedores de credenciais no Guia de referência de SDKs e ferramentas da AWS.

5. O SDK usa as variáveis de ambiente para se conectar ao EKS Pod Identity Agent e recuperar as credenciais.

> **Note**
>
> Se suas workloads usam no momento credenciais mais antigas na cadeia de credenciais, essas credenciais continuarão sendo usadas mesmo se você configurar uma associação do EKS Pod Identity para a mesma workload.

Configurar o atndente do Identidade de Pods do Amazon EKS

As associações de Identidade de Pods do Amazon EKS permitem gerenciar credenciais para suas aplicações de forma semelhante a como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2.

A Identidade de Pods do Amazon EKS fornece credenciais para suas workloads com uma API EKS Auth adicional e um pod de atendente executado em cada nó.

> **Tip**
>
> Você não precisa instalar o atendente da Identidade de Pods nos clusters do Modo automático do EKS. Essa capacidade é integrada ao Modo automático do EKS.

Considerações

- Por padrão, o atendente de Identidade de Pods do EKS é pré-instalado nos clusters do EKS em modo automático. Para saber mais, consulte Modo Automático do EKS.
- Por padrão, o atendente de Identidade de Pods do EKS escuta em um endereço `IPv4` e `IPv6` para os pods para solicitar credenciais. O atendente usa o endereço IP de loopback (localhost) `169.254.170.23` para `IPv4` e o endereço IP de localhost `[fd00:ec2::23]` para `IPv6`.
- Se você desabilitar endereços `IPv6` ou de alguma forma impedir endereços IP `IPv6` do localhost, o atendente não poderá iniciar. Para iniciar o atendente em nós que não podem usar `IPv6`, siga as etapas em the section called “Desabilitar o IPv6” para desabilitar a configuração de `IPv6`.

Criar o atendente de Identidade de Pods do Amazon EKS

Pré-requisitos do atendente

- Um cluster existente do Amazon EKS. Para implantar, consulte Conceitos básicos. A versão do cluster e a versão da plataforma devem ser iguais ou posteriores às versões listadas em Versões de clusters da Identidade de Pods do EKS.
- O perfil de nó tem permissões para o atendente realizar a ação `AssumeRoleForPodIdentity` na API de autenticação do EKS. Você pode usar a política gerenciada da AWS: AmazonEKSWorkerNodePolicy ou adicionar uma política personalizada semelhante à seguinte:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity"
            ],
            "Resource": "*"
        }
    ]
}
```

  Esta ação pode ser limitada por tags para restringir quais funções podem ser assumidas pelos pods que usam o atendente.
- Os nós podem acessar e baixar imagens do Amazon ECR. A imagem de contêiner do complemento está nos registros listados em Visualizar registros de imagem de contêiner da Amazon para complementos do Amazon EKS.

  Observe que você pode alterar o local da imagem e fornecer `imagePullSecrets` para complementos EKS nas definições de configuração opcionais em Console de gerenciamento da AWS e em `--configuration-values` na CLI de AWS.
- Os nós podem acessar a API de autenticação do Amazon EKS. Para clusters privados, é necessário o endpoint `eks-auth` em AWS PrivateLink.

Configurar o atendente com o console do AWS

1. Abra o console do Amazon EKS.

2. No painel de navegação esquerdo, selecione Clusters e depois o nome do cluster para o qual você deseja configurar o complemento do atendente de Identidade de Pods do EKS.
3. Escolha a guia Add-ons (Complementos).
4. Escolha Obter mais complementos.
5. Selecione a caixa no canto superior direito da caixa do complemento do atentente de Identidade de Pods do EKS e escolha Editar.
6. Na página Configurar complementos selecionados, selecione qualquer versão na lista suspensa Versão.
7. (Opcional) Expanda Configurações opcionais para inserir configurações adicionais. Por exemplo, é possível indicar um local alternativo para a imagem do contêiner e `ImagePullSecrets`. O esquema JSON com chaves aceitas é mostrado no Esquema de configuração do complemento.

   Insira as chaves e os valores de configuração em Valores de configuração.
8. Escolha Próximo.
9. Confirme que os pods do atentente de Identidade de Pods do EKS estão em execução no cluster.

   ```
   kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
   ```

   Veja abaixo um exemplo de saída.

   ```
   eks-pod-identity-agent-gmqp7                                          1/1     Running
      1 (24h ago)   24h
   eks-pod-identity-agent-prnsh                                          1/1     Running
      1 (24h ago)   24h
   ```

   Agora é possível usar associações da Identidade de Pods do EKS em seu cluster. Para obter mais informações, consulte the section called “Atribuir perfil do IAM”.

Configurar o atendente com a AWS CLI

1. Execute o seguinte comando da AWS CLI. Substitua o `my-cluster` pelo nome do cluster.

   ```
   aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --
   addon-version v1.0.0-eksbuild.1
   ```

> **Note**
>
> O atentente de Identidade de Pods do EKS não usa o `service-account-role-arn` para perfis do IAM para contas de serviço. Você deve fornecer ao atentente de Identidade de Pods do EKS as permissões no perfil de nó.

2. Confirme que os pods do atentente de Identidade de Pods do EKS estão em execução no cluster.

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

Veja abaixo um exemplo de saída.

```
eks-pod-identity-agent-gmqp7                                          1/1     Running
   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1     Running
   1 (24h ago)   24h
```

Agora é possível usar associações da Identidade de Pods do EKS em seu cluster. Para obter mais informações, consulte the section called “Atribuir perfil do IAM”.

Atribuir um perfil do IAM a uma conta de serviço do Kubernetes

Este tópico aborda como configurar uma conta de serviço do Kubernetes para assumir um perfil do AWS Identity and Access Management (IAM) com a Identidade de Pods do EKS. Qualquer pod que esteja configurado para usar a conta de serviço pode acessar qualquer serviço da AWS que o perfil tenha permissão para acessar.

Para criar uma associação de identidade de pod do EKS, há apenas uma única etapa; você cria a associação no EKS por meio de Console de gerenciamento da AWS, AWS CLI, AWS SDKs, AWS CloudFormation e outras ferramentas. Não há dados ou metadados sobre as associações dentro do cluster em nenhum objeto do Kubernetes, e você não adiciona anotações às contas de serviço.

Pré-requisitos

- Um cluster existente. Se você não tiver um, poderá criá-lo seguindo um dos guias em Conceitos básicos.
- A entidade principal do IAM que está criando a associação deve ter o `iam:PassRole`.

- A versão mais recente da AWS CLI instalada e configurada em seu dispositivo ou no AWS CloudShell. É possível verificar sua versão atual com `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar e Configuração rápida com aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Um arquivo `kubectl config` existente que contém a configuração do seu cluster. Para criar um arquivo `kubectl config`, consulte the section called “Acessar meu cluster com o kubectl”.

Criar uma associação de identidade de pod (console do AWS)

1. Abra o console do Amazon EKS.
2. No painel de navegação esquerdo, selecione Clusters e depois o nome do cluster para o qual você deseja configurar o complemento do atendente de Identidade de Pods do EKS.
3. Escolha a guia Acesso.
4. Nas Associações do Pod Identity, escolha Criar.
5. Para o Perfil do IAM, selecione o perfil do IAM com as permissões que você deseja que a workload tenha.

> **Note**
>
> A lista contém apenas perfis que têm a política de confiança a seguir que permite que o EKS Pod Identity as use.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`: a Identidade de Pods do EKS usa `AssumeRole` para assumir o perfil do IAM antes de passar as credenciais temporárias para os pods.

`sts:TagSession`: a Identidade de Pods do EKS usa `TagSession` para incluir etiquetas de sessão nas solicitações para o AWS STS.

É possível usar essas tags nas chaves de condição na política de confiança para restringir quais contas de serviço, namespaces e clusters podem usar esse perfil.

Para obter uma lista de chaves de condição do Amazon EKS, consulte Condições definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar a chave de condição, consulte Ações definidas pelo Amazon Elastic Kubernetes Service.

6. Em Namespace do Kubernetes, selecione o namespace do Kubernetes que contém a conta de serviço e a workload. Opcionalmente, é possível especificar um namespace por nome que não existe no cluster.
7. Em conta de serviço do Kubernetes, selecione a conta de serviço do Kubernetes a ser usada. O manifesto da workload do Kubernetes deve especificar essa conta de serviço. Opcionalmente, é possível especificar uma conta de serviço por nome que não existe no cluster.
8. (Opcional) Para as Tags, escolha Adicionar tag para adicionar metadados em um par de chave e valor. Essas tags são aplicadas à associação e podem ser usadas nas políticas do IAM.

   Repita essa etapa para adicionar várias tags.
9. Escolha Criar.

Criar uma associação de identidade de pod (AWS CLI)

1. Se você quiser associar uma política do IAM existente ao seu perfil do IAM, vá para a próxima etapa.

   Crie uma política do IAM. É possível criar a sua própria política ou copiar uma política gerenciada da AWS que já conceda algumas das permissões de que você precisa e a personalizar de acordo com seus requisitos específicos. Para obter mais informações, consulte Criar políticas do IAM no Guia do usuário do IAM.

   a. Crie um arquivo que inclua as permissões para os serviços da AWS que você deseja que os pods acessem. Para obter uma lista de todas as ações de todos os serviços do AWS, consulte a Referência de autorização de serviço.

      É possível executar o comando a seguir para criar um arquivo de política de exemplo que permita acesso somente leitura a um bucket do Amazon S3. Você pode, opcionalmente, armazenar informações de configuração ou um script de bootstrap nesse bucket, e os contêineres no pod podem ler o arquivo do bucket e carregá-lo na aplicação. Se você quiser criar esse exemplo de política, copie o conteúdo a seguir para o seu dispositivo. Substitua *my-pod-secrets-bucket* pelo nome do seu bucket e execute o comando.

      ```
      {
          "Version":"2012-10-17",
          "Statement": [
              {
                  "Effect": "Allow",
                  "Action": "s3:GetObject",
                  "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
              }
          ]
      }
      ```

   b. Crie a política do IAM.

      ```
      aws iam create-policy --policy-name my-policy --policy-document file://my-
      policy.json
      ```

2. Crie um perfil do IAM e associe-o a uma conta de serviço do Kubernetes.

   a. Se você já tem um conta de serviço do Kubernetes na qual deseja assumir um perfil do IAM, você pode ignorar esta etapa.

Crie uma conta de serviço do Kubernetes. Copie o conteúdo a seguir para o seu dispositivo. Substitua *my-service-account* pelo nome desejado e *default* (padrão) por um namespace diferente, se necessário. Se você alterar o *padrão*, o namespace já deve existir.

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

Execute o comando a seguir.

```
kubectl apply -f my-service-account.yaml
```

b. Execute o seguinte comando para criar um arquivo de política de confiança para o perfil do IAM.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

c. Crie a função. Substitua *my-role* por um nome para seu perfil do IAM e *my-role-description* por uma descrição para seu perfil.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://
trust-relationship.json --description "my-role-description"
```

d. Associe uma política do IAM ao seu perfil. Substitua *my-role* pelo nome do seu perfil do IAM e *my-policy* pelo nome de uma política existente que você criou.

```
aws iam attach-role-policy --role-name my-role --policy-
arn=arn:aws:iam::111122223333:policy/my-policy
```

> **Note**
>
> Ao contrário dos perfis do IAM para contas de serviço, o EKS Pod Identity não usa uma anotação na conta de serviço.

e. Execute o comando a seguir para criar a associação. Substitua `my-cluster` pelo nome do cluster, substitua *my-service-account* pelo nome desejado e *default* por um namespace diferente, se necessário.

```
aws eks create-pod-identity-association --cluster-name my-cluster --role-arn
 arn:aws:iam::111122223333:role/my-role --namespace default --service-account my-
service-account
```

Veja abaixo um exemplo de saída.

```
{
    "association": {
        "clusterName": "my-cluster",
        "namespace": "default",
        "serviceAccount": "my-service-account",
        "roleArn": "arn:aws:iam::111122223333:role/my-role",
        "associationArn": "arn:aws::111122223333:podidentityassociation/my-
cluster/a-abcdefghijklmnop1",
        "associationId": "a-abcdefghijklmnop1",
        "tags": {},
        "createdAt": 1700862734.922,
        "modifiedAt": 1700862734.922
    }
}
```

> **Note**
>
> É possível especificar um namespace e uma conta de serviço por nome que não existe no cluster. Você deve criar o namespace, a conta de serviço e a workload que usa a conta de serviço para que a associação ao EKS Pod Identity funcione.

Confirmar configuração

1. Confirme se a política de confiança do perfil do IAM está configurada corretamente.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

   Veja abaixo um exemplo de saída.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow EKS Auth service to assume this role for Pod Identities",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

2. Confirme se a política que você anexou ao seu perfil em uma etapa anterior está vinculada ao perfil.

```
aws iam list-attached-role-policies --role-name my-role --query
 'AttachedPolicies[].PolicyArn' --output text
```

   Veja um exemplo de saída abaixo.

```
                arn:aws:iam::111122223333:policy/my-policy
```

3. Defina uma variável para armazenar o nome do recurso da Amazon (ARN) da política que deseja usar. Substitua *my-policy* pelo nome da política para a qual deseja confirmar permissões.

```
export policy_arn=arn:aws:iam::111122223333:policy/my-policy
```

4. Visualize a versão padrão da política.

```
aws iam get-policy --policy-arn $policy_arn
```

   Veja abaixo um exemplo de saída.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws:iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. Confira o conteúdo da política para garantir que ela inclua todas as permissões de que seu pod precisa. Se necessário, substitua*1* no comando a seguir pela versão retornada na saída anterior.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

   Veja abaixo um exemplo de saída.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
```

```
    ]
}
```

Se você criou a política de exemplo em uma etapa anterior, sua saída será a mesma. Se você criou uma política diferente, então o *exemplo* de conteúdo é diferente.

Próximas etapas

the section called “Conta de serviço de pods”

Acessar Recursos da AWS usando os perfis do IAM de destino da Identidade de Pods do EKS

Ao executar aplicações no Amazon Elastic Kubernetes Service (Amazon EKS), talvez você precise acessar recursos da AWS que existem em diferentes contas da AWS. Este guia mostra como configurar o acesso entre contas usando a Identidade de Pods do EKS, que permite que seus pods do Kubernetes acessem outros recursos da AWS usando perfis de destino.

Pré-requisitos

Antes de começar, certifique-se de ter concluído as seguintes etapas:

- Configurar o agente da Identidade de Pods do Amazon EKS
- Criar perfil da Identidade de Pods do EKS

Como funciona

A Identidade de Pods permite que as aplicações em seu cluster do EKS acessem recursos da AWS em várias contas por meio de um processo chamado encadeamento de perfis.

Ao criar uma associação da Identidade de Pods, é possível fornecer dois perfis do IAM: um perfil da Identidade de Pods do EKS na mesma conta do cluster do EKS e um perfil do IAM de destino da conta que contém os recursos da AWS (como buckets do S3 ou bancos de dados RDS). O perfil da Identidade de Pods do EKS deve estar na conta do cluster do EKS devido aos requisitos do IAM PassRole, enquanto o perfil do IAM de destino pode estar em qualquer conta da AWS. O PassRole permite que uma entidade da AWS delegue a suposição de perfil a outro serviço. A Identidade de Pods do EKS usa o PassRole para conectar um perfil a uma conta de serviço do Kubernetes, exigindo que o perfil e a identidade que a transmitem estejam na mesma conta da AWS que o

cluster do EKS. Quando o pod da aplicação precisa acessar os recursos da AWS, ele solicita as credenciais da Identidade de Pods. Em seguida, a Identidade de Pods executa automaticamente duas suposições de perfil em sequência: primeiro, assumindo o perfil da Identidade de Pods do EKS e, em seguida, usando essas credenciais para assumir o perfil do IAM de destino. Esse processo fornece ao seu pod credenciais temporárias que têm as permissões definidas no perfil de destino, permitindo acesso seguro aos recursos em outras contas da AWS.

Considerações sobre armazenamento em cache

Devido aos mecanismos de armazenamento em cache, as atualizações de um perfil do IAM em uma associação existente da Identidade de Pods podem não entrar em vigor imediatamente nos pods em execução no cluster do EKS. O agente da Identidade de Pods armazena em cache as credenciais do IAM com base na configuração da associação no momento em que as credenciais são obtidas. Se a associação incluir apenas um perfil da Identidade de Pods do EKS e nenhum perfil do IAM de destino, a duração das credenciais em cache será de seis horas. Se a associação incluir o ARN do perfil da Identidade de Pods do EKS e um perfil do IAM de destino, as credenciais em cache vão durar 59 minutos. Modificar uma associação existente, como atualizar o ARN do perfil da Identidade de Pods do EKS ou adicionar um perfil do IAM de destino, não redefine o cache existente. Como resultado, o agente não reconhecerá as atualizações até que as credenciais em cache sejam atualizadas. Para aplicar as alterações antes, você pode recriar os pods existentes; caso contrário, você precisará esperar que o cache expire.

Etapa 1: criar e associar um perfil do IAM de destino

Nesta etapa, você estabelecerá uma cadeia de confiança segura criando e configurando um perfil do IAM de destino. Para fins de demonstração, criaremos um novo perfil do IAM de destino para estabelecer uma cadeia de confiança entre duas contas da AWS: o perfil da Identidade de Pods do EKS (por exemplo, `eks-pod-identity-primary-role`) na conta da AWS do cluster do EKS ganha permissão para assumir o perfil do IAM de destino (por exemplo, `eks-pod-identity-aws-resources`) em sua conta de destino, permitindo o acesso a recursos da AWS como buckets do Amazon S3.

Criar perfil do IAM de destino

1. Abra o console do Amazon IAM.
2. Na barra de navegação superior, verifique se você fez login na conta que contém os recursos da AWS (como buckets do S3 ou tabelas do DynamoDB) para o perfil do IAM de destino.
3. No painel de navegação à esquerda, selecione Perfis.

4. Escolha o botão Criar perfil e, em seguida, Conta da AWS em "Tipo de entidade confiável".
5. Escolha Outra conta da AWS, insira o número da sua conta da AWS (a conta em que seu perfil da Identidade de Pods do EKS existe) e escolha Avançar.
6. Adicione as políticas de permissão que você gostaria de associar ao perfil (por exemplo, AmazonS3FullAccess) e escolha Avançar.
7. Insira um nome de perfil, como `MyCustomIAMTargetRole`, e escolha Criar perfil.

Atualizar a política de confiança de um perfil do IAM de destino

1. Depois de criar o perfil, você retornará à lista de Perfis. Encontre e selecione o novo perfil que você criou na etapa anterior (por exemplo, `MyCustomIAMTargetRole`).
2. Selecione a guia Relações de confiança.
3. Clique em Editar política de confiança no lado direito.
4. No editor de políticas, substitua a política JSON padrão pela sua política de confiança. Substitua os valores do espaço reservado pelo nome do perfil e `111122223333` no ARN do perfil do IAM pelo ID da conta da AWS que hospeda o cluster do EKS. Opcionalmente, você também pode usar PrincipalTags na política de confiança do perfil para autorizar somente contas de serviço específicas de um determinado cluster e namespace a assumir seu perfil de destino. Por exemplo:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/eks-cluster-arn": "arn:aws:eks:us-
east-1:111122223333:cluster/example-cluster",
          "aws:RequestTag/kubernetes-namespace": "ExampleNameSpace",
          "aws:RequestTag/kubernetes-service-account": "ExampleServiceAccountName"
        },
```

```
        "ArnEquals": {
          "aws:PrincipalARN": "arn:aws:iam::111122223333:role/eks-pod-identity-primary-
role"
        }
      }
    }
  ]
}
```

A política acima permite que o perfil `eks-pod-identeity-primary-role` da conta da AWS 111122223333 com as tags de sessão de Identidade de Pods do EKS relevantes assumam esse perfil.

Se você desativou as tags de sessão na sua Identidade de Pods do EKS, a Identidade de Pods do EKS também define `sts:ExternalId` com informações sobre o cluster, namespace e conta de serviço de um pod ao assumir um perfil de destino.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "region/111122223333/cluster-name/namespace/service-
account-name"
        },
        "ArnEquals": {
          "aws:PrincipalARN": "arn:aws:iam::111122223333:role/eks-pod-identity-primary-
role"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:TagSession"
```

```
    }
  ]
}
```

A política acima ajuda a garantir que somente o cluster, namespace e conta de serviço esperados possam assumir o perfil de destino.

Atualizar a política de permissão para o perfil da Identidade de Pods do EKS

Nesta etapa, você atualizará a política de permissão do perfil da Identidade de Pods do EKS associada ao cluster do Amazon EKS adicionando o ARN do perfil do IAM de destino como um recurso.

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, selecione Clusters e depois o nome do seu cluster do EKS.
3. Escolha a guia Acesso.
4. Em Associações da Identidade de Pods, selecione seu perfil da Identidade de Pods do EKS.
5. Escolha Permissões, Adicionar permissões e Criar política em linha.
6. Escolha JSON no lado direito.
7. No editor de políticas, substitua a política JSON padrão pela sua política de permissão. Substitua o valor do espaço reservado pelo nome do perfil e `222233334444` no ARN do perfil do IAM pelo perfil do IAM de destino. Por exemplo:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Resource": "arn:aws:iam::222233334444:role/eks-pod-identity-aws-resources"
        }
    ]
}
```

Etapa 2: associar o perfil do IAM de destino a uma conta de serviço do Kubernetes

Nesta etapa, você criará uma associação entre o perfil do IAM de destino e a conta de serviço do Kubernetes no cluster do EKS.

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, selecione Clusters e depois o nome do cluster para o qual você deseja adicionar a associação.
3. Escolha a guia Acesso.
4. Nas Associações do Pod Identity, escolha Criar.
5. Escolha o perfil da Identidade de Pods do EKS em Perfil do IAM para que suas workloads assumam.
6. Escolha o perfil do IAM de destino em Perfil do IAM de destino que será assumido pelo perfil da Identidade de Pods do EKS.
7. No campo Namespace do Kubernetes, insira o nome do namespace em que você deseja criar a associação (por exemplo, `my-app-namespace`). Isso definirá onde a conta de serviço vai residir.
8. No campo Conta de serviço do Kubernetes, insira o nome da conta de serviço (por exemplo, `my-service-account`) que usará as credenciais do IAM. Isso vinculará o perfil do IAM à conta de serviço.
9. Escolha Criar para criar a associação.

Configurar pods para acessar serviços da AWS com contas de serviço

Se um pod precisar acessar os serviços da AWS, você deverá configurá-lo para usar uma conta de serviço do Kubernetes. A conta de serviço deve estar associada a um perfil do AWS Identity and Access Management (IAM) que tenha permissões para acessar os serviços da AWS.

- Um cluster existente. Se não tiver um, você poderá criá-lo usando um dos guias em Conceitos básicos.
- Uma associação existente entre a conta de serviço do Kubernetes e a Identidade de Pods do EKS que associa a conta de serviço a um perfil do IAM. O perfil deve ter uma política do IAM associada que contenha as permissões que você deseja que os pods tenham para usar os serviços da AWS. Para obter mais informações sobre como criar a conta de serviço e como configurá-la, consulte the section called “Atribuir perfil do IAM”.
- A versão mais recente da AWS CLI instalada e configurada em seu dispositivo ou no AWS CloudShell. É possível verificar sua versão atual com `aws --version | cut -d / -f2`

`| cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar e Configuração rápida com aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.

- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Um arquivo `kubectl config` existente que contém a configuração do seu cluster. Para criar um arquivo `kubectl config`, consulte the section called “Acessar meu cluster com o kubectl”.

  1. Use o comando a seguir para criar um manifesto de implantação com o qual é possível implantar um pod para confirmar a configuração. Substitua os valores de exemplo por seus próprios valores.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

  2. Implante o manifesto no seu cluster.

```
kubectl apply -f my-deployment.yaml
```

3. Verifique se as variáveis de ambiente necessárias existem para o pod.
   a. Visualize os pods que foram implantados na implantação da etapa anterior.

      ```
      kubectl get pods | grep my-app
      ```

      Veja abaixo um exemplo de saída.

      ```
      my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
      ```

   b. Verifique se o pod tem um arquivo de token de conta de serviço montado.

      ```
      kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
       AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
      ```

      Veja abaixo um exemplo de saída.

      ```
      AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:  /var/run/secrets/
      pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
      ```

4. Verifique se os pods podem interagir com os serviços da AWS usando as permissões que você atribuiu na política do IAM anexada ao seu perfil.

   > **Note**
   >
   > Quando um pod usa as credenciais da AWS de um perfil do IAM associado a uma conta de serviço, a AWS CLI ou outros SDKs nos contêineres desse pod usam as credenciais fornecidas por esse perfil. Se você não restringir o acesso às credenciais fornecidas ao perfil do IAM do nó do Amazon EKS, o pod ainda terá acesso a elas. Para obter mais informações, consulte Restringir o acesso ao perfil de instância atribuído ao nó de processamento.

   Se os pods não conseguirem interagir com os serviços conforme o esperado, conclua as etapas a seguir para verificar se tudo está configurado corretamente.

a. Verifique se os pods usam uma versão do AWS SDK que permita assumir um perfil do IAM por meio de uma associação da Identidade de Pods do EKS. Para obter mais informações, consulte the section called “SDKs compatíveis”.

b. Confirme se a implantação está usando a conta de serviço.

```
kubectl describe deployment my-app | grep "Service Account"
```

Veja abaixo um exemplo de saída.

```
Service Account:  my-service-account
```

Conceder acesso dos pods a recursos da AWS baseados em tags

O controle de acesso por atributo (ABAC) concede direitos aos usuários por meio de políticas que combinam os atributos. A Identidade de Pods do EKS anexa tags às credenciais temporárias de cada pod com atributos como nome do cluster, namespace e nome da conta de serviço. Essas tags de sessão de perfil permitem que os administradores criem um único perfil que pode funcionar em várias contas de serviço, permitindo o acesso aos recursos da AWS com base em tags correspondentes. Ao adicionar suporte a tags de sessão de perfil, você pode impor limites de segurança mais rígidos entre clusters e workloads dentro dos clusters enquanto reutilizam os mesmos perfis e políticas do IAM.

Exemplo de política com tags

Confira abaixo um exemplo de política do IAM que concede permissões `s3:GetObject` quando o objeto correspondente é marcado com o nome do cluster do EKS.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/eks-cluster-name": "${aws:PrincipalTag/eks-
cluster-name}"
                }
            }
        }
    ]
}
```

Habilitar ou desabilitar tags de sessão

A Identidade de Pods do EKS adiciona um conjunto predefinido de tags de sessão quando assume o perfil. Essas tags de sessão permitem que os administradores criem um único perfil que pode funcionar em vários recursos, permitindo o acesso aos recursos da AWS com base em tags correspondentes.

Habilitar tags de sessão

As tags de sessão são automaticamente habilitadas com a Identidade de Pods do EKS. Nenhuma ação é necessária de sua parte. Por padrão, a Identidade de Pods do EKS anexa um conjunto de tags predefinidas à sua sessão. Para referenciar essas tags nas políticas, use a sintaxe `${aws:PrincipalTag/` seguida pela chave da tag. Por exemplo, `${aws:PrincipalTag/kubernetes-namespace}`.

- `eks-cluster-arn`
- `eks-cluster-name`
- `kubernetes-namespace`
- `kubernetes-service-account`
- `kubernetes-pod-name`
- `kubernetes-pod-uid`

Desabilitar tags de sessão

A AWS compacta políticas de sessão em linha, ARNs de política gerenciada e tags de sessão em um formato binário compactado que tem um limite separado. Caso receba um erro `PackedPolicyTooLarge` indicando que o formato binário compactado excedeu o limite de tamanho, você pode tentar reduzir o tamanho desabilitando as tags de sessão adicionadas pela Identidade de Pods do EKS. Para desabilitar essas tags de sessão, siga estas etapas:

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, selecione Clusters e depois o nome do cluster que você deseja modificar.
3. Escolha a guia Acesso.
4. Nas Associações de Identidade de Pods, escolha o ID da associação que você gostaria de modificar em ID da associação e depois Editar.
5. Em Tags de sessão, escolha Desabilitar tags de sessão.
6. Escolha Salvar alterações.

Cópia entre contas

Todas as tags de sessão adicionadas pelo EKS Pod Identity são transitivas; as chaves e os valores das tags são passados para quaisquer ações `AssumeRole` usadas por suas workloads para trocar de perfil para outra conta. Você pode usar essas tags em políticas de outras contas para limitar o acesso em cenários entre contas. Para obter mais informações, consulte Encadeamento de funções com tags de sessão no Guia do usuário do IAM.

Tags personalizadas

O EKS Pod Identity não pode adicionar outras tags personalizadas à ação `AssumeRole` que ele mesmo executa. No entanto, as tags que você aplica ao perfil do IAM estão sempre disponíveis no mesmo formato: `${aws:PrincipalTag/` seguida pela chave, por exemplo, `${aws:PrincipalTag/MyCustomTag}`.

> **Note**
>
> As tags adicionadas à sessão por meio da solicitação `sts:AssumeRole` têm precedência em caso de conflito. Por exemplo, vamos supor que:

- O Amazon EKS adiciona uma chave `eks-cluster-name` e um valor `my-cluster` à sessão quando o EKS assume o perfil de cliente.
- Você adiciona uma tag `eks-cluster-name` ao perfil do IAM com o valor `my-own-cluster`.

Nesse caso, o primeiro tem precedência e o valor da tag `eks-cluster-name` será `my-cluster`.

Usar identidade de pods com o AWS SDK

Usar as credenciais da Identidade de Pods do EKS

Para usar as credenciais de uma associação de identidade de pod EKS, seu código pode usar qualquer SDK AWS para criar um cliente para um serviço AWS com um SDK e, por padrão, o SDK pesquisa em uma cadeia de locais as credenciais de gerenciamento de identidade e acesso AWS a serem usadas. As credenciais da Identidade de Pods do EKS serão usadas se você não especificar um provedor de credenciais ao criar o cliente ou ao inicializar o SDK.

Isso funciona porque as identidades de Pods do EKS foram adicionadas ao Provedor de credenciais do contêiner, o qual é pesquisado em uma etapa na cadeia de credenciais padrão. Se suas workloads usam no momento credenciais mais antigas na cadeia de credenciais, essas credenciais continuarão sendo usadas mesmo se você configurar uma associação da Identidade de Pods do EKS para a mesma workload.

Para obter mais informações sobre como as identidades do EKS Pod funcionam, consulte the section called “Como funciona”.

Ao usar Saiba como Identidade de Pods do EKS concede acesso a serviços da AWS para pods, os contêineres nos pods devem usar uma versão do AWS SDK que permita assumir um perfil do IAM do atendente da Identidade de Pods do EKS. Certifique-se de usar as seguintes versões, ou versões posteriores, do AWS SDK:

- Java (versão 2): 2.21.30
- Java: 1.12.746
- Go v1: v1.47.11
- Go v2: release-2023-11-14

- Python (Boto3): 1.34.41
- Python (botocore): 1.34.41
- AWS CLI: 1.30.0

  AWS CLI – 2.15.0
- JavaScript v2: 2.1550.0
- JavaScript v3: 3.458.0
- Kotlin: v1.0.1
- Ruby: 3.188.0
- Rust: release-2024-03-13
- C++: 1.11.263
- .NET: 3.7.734.0
- PowerShell: 4.1.502
- PHP: 3.289.0

Para garantir que você esteja usando um SDK compatível, siga as instruções de instalação do SDK de sua preferência em Ferramentas para desenvolver na AWS quando estiver criando seus contêineres.

Para obter uma lista de complementos compatíveis com Identidade de Pods do EKS, consulte the section called “Referência de suporte de Identidade de Pods”.

Desabilitar o **`IPv6`** no EKS Pod Identity Agent

Console de gerenciamento da AWS

1. Para desabilitar o `IPv6` no EKS Pod Identity Agent, adicione a configuração a seguir às Configurações opcionais do EKS Add-on.
   a. Abra o console do Amazon EKS.
   b. No painel de navegação esquerdo, selecione Clusters e depois o nome do cluster para o qual você deseja configurar o complemento.
   c. Escolha a guia Add-ons (Complementos).
   d. Selecione a caixa no canto superior direito da caixa do complemento do EKS Pod Identity Agent e escolha Editar.
   e. Na página Configurar o EKS Pod Identity Agent:

i. Selecione a Versão que você deseja usar. Recomendamos manter a mesma versão da etapa anterior e atualizar a versão e a configuração em ações separadas.

ii. Expanda Definições de configuração opcionais.

iii. Insira a chave JSON `"agent":` e o valor de um objeto JSON aninhado com uma chave `"additionalArgs":` em Valores de configuração. O texto resultante deve ser um objeto JSON válido. Se esse par de chave e valor for o único dado na caixa de texto, coloque-o entre colchetes `{ }`. O exemplo apresentado a seguir mostra que a política de rede está habilitada:

```
{
    "agent": {
        "additionalArgs": {
            "-b": "169.254.170.23"
        }
    }
}
```

Essa configuração define o endereço `IPv4` como o único usado pelo agente.

f. Para aplicar a nova configuração substituindo os pods do EKS Pod Identity Agent, escolha Salvar alterações.

O Amazon EKS aplica as alterações nos complementos do EKS usando uma distribuição do `DaemonSet` do Kubernetes para o agente da Identidade de Pods do EKS. É possível acompanhar o status da implantação no Histórico de atualizações do complemento no Console de gerenciamento da AWS e com `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`.

`kubectl rollout` oferece os seguintes comandos:

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Se a distribuição demorar muito, o Amazon EKS desfará a distribuição e uma mensagem com o tipo Atualização do complemento e o status Falha será adicionada ao Histórico de atualizações do complemento. Para investigar qualquer problema, comece com o histórico da distribuição e execute `kubectl logs` em um pod do EKS Pod Identity Agent para ver os logs do EKS Pod Identity Agent.

2. Se a nova entrada no Histórico de atualizações tiver o status de Êxito, a distribuição foi concluída e o complemento está usando a nova configuração em todos os pods do EKS Pod Identity Agent.

AWS CLI

1. Para desabilitar o `IPv6` no EKS Pod Identity Agent, adicione a seguinte configuração aos valores de configuração do EKS Add-on.

   Execute o seguinte comando da AWS CLI. Substitua `my-cluster` pelo nome do cluster e o ARN do perfil do IAM pelo perfil que você está usando.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent \
       --resolve-conflicts PRESERVE --configuration-values '{"agent":{"additionalArgs":
    { "-b": "169.254.170.23"}}}'
   ```

   Essa configuração define o endereço `IPv4` como o único usado pelo agente.

   O Amazon EKS aplica as alterações nos complementos do EKS usando uma distribuição do DaemonSet do Kubernetes para o agente da Identidade de Pods do EKS. É possível acompanhar o status da implantação no Histórico de atualizações do complemento no Console de gerenciamento da AWS e com `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`.

   `kubectl rollout` oferece os seguintes comandos:

   ```
   kubectl rollout

   history  -- View rollout history
   pause    -- Mark the provided resource as paused
   restart  -- Restart a resource
   resume   -- Resume a paused resource
   status   -- Show the status of the rollout
   undo     -- Undo a previous rollout
   ```

Se a distribuição demorar muito, o Amazon EKS desfará a distribuição e uma mensagem com o tipo Atualização do complemento e o status Falha será adicionada ao Histórico de atualizações do complemento. Para investigar qualquer problema, comece com o histórico da distribuição e execute `kubectl logs` em um pod do EKS Pod Identity Agent para ver os logs do EKS Pod Identity Agent.

Criar um perfil do IAM com a política de confiança exigida pela Identidade de Pods do EKS

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`

A Identidade de Pods do EKS usa `AssumeRole` para assumir o perfil do IAM antes de passar as credenciais temporárias para seus pods.

`sts:TagSession`

A Identidade de Pods do EKS usa `TagSession` para incluir tags de sessão nas solicitações para AWS STS.

Configuração de condições

É possível usar essas tags nas chaves de condição na política de confiança para restringir quais contas de serviço, namespaces e clusters podem usar esse perfil. Para ter acesso à lista de

tags de solicitação disponível com a Identidade de Pods, consulte the section called “Habilitar ou desabilitar tags de sessão”.

Por exemplo, você pode restringir quais pods podem assumir o papel de um perfil do IAM de Identidade de Pods para uma `ServiceAccount` e `Namespace` específicos com a seguinte Política de confiança com a adição de `Condition`:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/kubernetes-namespace": [
                        "Namespace"
                    ],
                    "aws:RequestTag/kubernetes-service-account": [
                        "ServiceAccount"
                    ]
                }
            }
        }
    ]
}
```

Para obter uma lista de chaves de condição do Amazon EKS, consulte Condições definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar a chave de condição, consulte Ações definidas pelo Amazon Elastic Kubernetes Service.

Gerenciar recursos computacionais usando nós

Um nó do Kubernetes é uma máquina que executa aplicações conteinerizadas. Todo nó tem os seguintes componentes:

- Runtime do contêiner - Software responsável pela execução dos contêineres.
- kubelet : garante que os contêineres estejam íntegros e em execução nos pods associados.
- kube-proxy : mantém as regras de rede que permitem a comunicação com os pods.

Para obter mais informações, consulte Nodes (Nós) na documentação do Kubernetes.

O cluster do Amazon EKS pode agendar pods em qualquer combinação de nós gerenciados do Modo Automático do EKS, nós autogerenciados, grupos de nós gerenciados do Amazon EKS, AWS Fargate e Amazon EKS Hybrid Nodes. Para saber mais sobre nós implantados em seu cluster, consulte the section called “Acessar recursos do cluster”.

> **Note**
>
> Excluindo os nós híbridos, os nós devem estar na mesma VPC que as sub-redes selecionadas no momento da criação do cluster. Porém, esses nós não precisam estar nas mesmas sub-redes.

Comparar opções de computação

A tabela a seguir fornece vários critérios para avaliar ao decidir quais opções atendem melhor às suas necessidades. Os nós autogerenciados são uma opção adicional que cumpre todos os critérios listados, mas requerem muito mais esforço manual para a manutenção. Para ter mais informações, consulte the section called “Nós autogerenciados”.

> **Note**
>
> O Bottlerocket tem algumas diferenças específicas em relação às informações gerais desta tabela. Para obter mais informações, consulte a documentação do Bottlerocket no GitHub.

Critérios	Grupos de nós gerenciados do EKS	Modo Automático do EKS	Amazon EKS Hybrid Nodes
Pode ser implantado no AWS Outposts	Não	Não	Não
Pode ser implantado em uma Zona local da AWS	Sim	Não	Não
Pode executar contêineres que exijam o Windows	Sim	Não	Não
Pode executar contêineres que exijam o Linux	Sim	Sim	Sim
Pode executar workloads que exijam o chip do Inferentia	Sim. Somente nós do Amazon Linux	Sim	Não
Pode executar workloads que exijam uma GPU	Sim. Somente nós do Amazon Linux	Sim	Sim
Pode executar workloads que exijam processadores ARM	Sim	Sim	Sim
Pode executar o AWS Bottlerocket	Sim	Sim	Não
Os pods compartilham recursos de CPU, memória, armazenamento e rede com outros pods.	Sim	Sim	Sim

Critérios	Grupos de nós gerenciados do EKS	Modo Automático do EKS	Amazon EKS Hybrid Nodes
Deve implantar e gerenciar instâncias do Amazon EC2	Sim	Não - saiba mais sobre instâncias gerenciadas pelo EC2	Sim - as máquinas físicas ou virtuais on-premises são gerenciadas por você com as ferramentas de sua escolha.
Deve proteger, manter e corrigir o sistema operacional das instâncias do Amazon EC2	Sim	Não	Sim - o sistema operacional em execução nas máquinas físicas ou virtuais são gerenciadas por você com as ferramentas de sua escolha.
Pode fornecer argumentos de bootstrap na implantação de um nó, como argumentos kubelet extras.	Sim - ao usar o `eksctl` ou um modelo de execução com uma AMI personalizada.	Não - use uma `NodeClass` para configurar nós	Sim - você pode personalizar os argumentos de bootstrap com o nodeadm. Consulte the section called “Nós híbridos nodeadm”.
Pode atribuir endereços IP a pods de um bloco CIDR diferente do endereço IP atribuído ao nó.	Sim. Usando um modelo de execução com uma AMI personalizada Para ter mais informações, consulte the section called “Modelos de execução”.	Não	Sim - consulte the section called “Configurar CNI”.

Critérios	Grupos de nós gerenciados do EKS	Modo Automático do EKS	Amazon EKS Hybrid Nodes
É possível executar o SSH no nó	Sim	Não - saiba como solucionar problemas de nós	Sim
Pode implantar sua própria AMI personalizada em nós	Sim. Usar um modelo de execução	Não	Sim
Pode implantar seu próprio CNI personalizado em nós	Sim. Usando um modelo de execução com uma AMI personalizada	Não	Sim

Critérios	Grupos de nós gerenciados do EKS	Modo Automático do EKS	Amazon EKS Hybrid Nodes
Você deve atualizar a AMI do nó por conta própria	Sim: se você implantou uma AMI otimizada do Amazon EKS, receberá uma notificação no console do Amazon EKS quando as atualizações estiverem disponíveis. É possível executar a atualização com um clique no console. Se você implantou uma AMI personalizada, não será notificado no console do Amazon EKS quando as atualizações estiverem disponíveis. Você deve executar a atualização por conta própria.	Não	Sim - o sistema operacional em execução nas máquinas físicas ou virtuais é gerenciado por você com as ferramentas de sua escolha. Consulte the section called “Preparar o sistema operacional”.

Critérios	Grupos de nós gerenciados do EKS	Modo Automático do EKS	Amazon EKS Hybrid Nodes
Você deve atualizar a versão de Kubernetes do nó por conta própria	Sim: se você implantou uma AMI otimizada do Amazon EKS, receberá uma notificação no console do Amazon EKS quando as atualizações estiverem disponíveis. É possível executar a atualização com um clique no console. Se você implantou uma AMI personalizada, não será notificado no console do Amazon EKS quando as atualizações estiverem disponíveis. Você deve executar a atualização por conta própria.	Não	Sim - você gerencia as atualizações de nós híbridos com as ferramentas de sua própria escolha ou com o `nodeadm`. Consulte the section called “Atualizar os nós híbridos”.
Pode usar armazenamento do Amazon EBS com pods	Sim	Sim, como um recurso integrado. Saiba como criar uma classe de armazenamento.	Não
Pode usar armazenamento do Amazon EFS com pods	Sim	Sim	Não

Critérios	Grupos de nós gerenciados do EKS	Modo Automático do EKS	Amazon EKS Hybrid Nodes
Pode usar armazenamento do Amazon FSx para Lustre com pods	Sim	Sim	Não
Pode usar o Network Load Balancer para serviços	Sim	Sim	Sim - deve usar o tipo de destino `ip`.
Pods podem ser executados em uma sub-rede pública	Sim	Sim	Não - os pods são executados no ambiente on-premises.
Pode atribuir diferentes grupos de segurança da VPC a pods individuais	Sim. Somente nós do Linux	Não	Não
Pode executar o Kubernetes DaemonSets	Sim	Sim	Sim
Compatível com `HostPort` e `HostNetwork` no manifesto do pod	Sim	Sim	Sim
Disponibilidade de regiões do AWS	Todas as regiões compatíveis com Amazon EKS	Todas as regiões compatíveis com Amazon EKS	Todas as regiões compatíveis com o Amazon EKS, exceto as regiões AWS GovCloud (EUA) e da China.

Critérios	Grupos de nós gerenciados do EKS	Modo Automático do EKS	Amazon EKS Hybrid Nodes
Pode executar contêineres em hosts dedicados do Amazon EC2	Sim	Não	Não
Preços	Custo da instância do Amazon EC2 que executa vários pods. Para obter mais informações, consulte Definição de preço do Amazon EC2.	Quando o Modo Automático do EKS está habilitado no cluster, você paga uma taxa separada, além das cobranças padrão da instância do EC2, pelas instâncias executadas usando o recurso de computação do Modo Automático. O valor varia de acordo com o tipo de instância executada e a região da AWS em que o cluster está localizado. Para obter mais informações, consulte Preços do Amazon EKS.	Custo de vCPU de nós híbridos por hora. Para obter mais informações, consulte Preços do Amazon EKS.

Simplificar o ciclo de vida dos nós com grupos de nós gerenciados

Os grupos de nós gerenciados do Amazon EKS automatizam o provisionamento e o gerenciamento do ciclo de vida dos nós (instâncias do Amazon EC2) para clusters de Kubernetes do Amazon EKS.

Com os grupos de nós gerenciados do Amazon EKS, não é necessário provisionar ou registrar separadamente as instâncias do Amazon EC2 que fornecem capacidade computacional para executar as aplicações do Kubernetes. Você pode criar, atualizar ou encerrar os nós para o cluster com uma única operação. As atualizações e terminações do nó esvaziam os nós automaticamente para garantir que as aplicações continuem disponíveis.

Todos os nós gerenciados são provisionados como parte de um grupo do Amazon EC2 Auto Scaling gerenciado pelo Amazon EKS para você. Todos os recursos, inclusive as instâncias e os grupos do Auto Scaling, são executados em sua conta da AWS. Cada grupo de nós é executado em várias zonas de disponibilidade definidas por você.

Os grupos de nós gerenciados também podem, opcionalmente, aproveitar o reparo automático de nós, que monitora continuamente a integridade dos nós. Ele reage automaticamente aos problemas detectados e substitui os nós quando possível. Isso ajuda na disponibilidade geral do cluster com o mínimo de intervenção manual. Para obter mais informações, consulte the section called “Integridade de nós”.

Você pode adicionar um grupo de nós gerenciados a clusters novos ou existentes usando o console do Amazon EKS, o `eksctl`, a AWS CLI, a API da AWS ou a infraestrutura como ferramentas de código, incluindo o AWS CloudFormation. Os nós iniciados como parte de um grupo de nós gerenciados são marcados automaticamente para detecção automática pelo Cluster Autoscaler do Kubernetes. É possível usar o grupo de nós para aplicar rótulos do Kubernetes aos nós e atualizá-los a qualquer momento.

Não há custos adicionais pelo uso dos grupos de nós gerenciados pelo Amazon EKS. Você só pagará pelos recursos da AWS que provisionar. Isso inclui as instâncias do Amazon EC2, os volumes do Amazon EBS, as horas de cluster do Amazon EKS e qualquer outra infraestrutura da AWS. Não há taxas mínimas nem compromissos antecipados.

Para começar a usar um novo cluster do Amazon EKS e um grupo de nós gerenciados, consulte the section called “Criar cluster (console e CLI)”.

Para adicionar um grupo de nós gerenciados a um cluster existente, consulte the section called “Criar”.

Conceitos dos grupos de nós gerenciados

- Os grupos de nós gerenciados do Amazon EKS criam e gerenciam instâncias do Amazon EC2 para você.

- Todos os nós gerenciados são provisionados como parte de um grupo do Amazon EC2 Auto Scaling gerenciado pelo Amazon EKS para você. Além disso, todos os recursos, inclusive as instâncias do Amazon EC2 e os grupos do Auto Scaling, são executados em sua conta da AWS.
- O grupo do Auto Scaling de um grupo de nós gerenciados abrange todas as sub-redes que você especificar ao criar o grupo.
- O Amazon EKS marca recursos do grupo de nós gerenciados para que eles sejam configurados para usar o Cluster Autoscaler do Kubernetes.

> **Important**
>
> Se estiver executando uma aplicação com estado em várias zonas de disponibilidade com suporte de volumes do Amazon EBS e usando o Cluster Autoscaler do Kubernetes, você deverá configurar vários grupos de nós, cada um com escopo de uma única zona de disponibilidade. Além disso, você deverá ativar o recurso `--balance-similar-node-groups`.

- Você pode usar um modelo de execução personalizado para obter um nível maior de flexibilidade e personalização ao implantar nós gerenciados. Por exemplo, é possível especificar argumentos `kubelet` extras e usar uma AMI personalizada. Para obter mais informações, consulte the section called “Modelos de execução”. Se você não usar um modelo de execução personalizado ao criar primeiro um grupo de nós gerenciados, um modelo de execução será gerado automaticamente. Não modifique manualmente esse modelo gerado automaticamente, ou ocorrerão erros.
- O Amazon EKS segue o modelo de responsabilidade compartilhada para CVEs e patches de segurança nos grupos de nós gerenciados. Quando os nós gerenciados executam uma AMI otimizada para o Amazon EKS, o Amazon EKS é responsável por criar versões da AMI com patches quando erros ou problemas forem relatados. Podemos publicar uma correção. No entanto, você é responsável por implantar essas versões de AMI com patches nos grupos de nós gerenciados. Quando os nós gerenciados executam uma AMI personalizada, você é responsável por criar versões da AMI com patches quando erros ou problemas forem relatados e, em seguida, implantar a AMI. Para obter mais informações, consulte the section called “Atualizar”.
- Os grupos de nós gerenciados do Amazon EKS podem ser executados em sub-redes públicas e privadas. Se você executar um grupo de nós gerenciados em uma sub-rede pública depois de 22 de abril de 2020, a sub-rede deverá ter o `MapPublicIpOnLaunch` definida como true para que as instâncias possam ingressar com êxito em um cluster. Se a sub-rede pública tiver sido criada usando `eksctl` ou os modelos do Amazon EKS vended AWS CloudFormation em ou após 26 de março de 2020, essa configuração já estará definida como verdadeira. Se as sub-redes

públicas foram criadas antes de 26 de março de 2020, será necessário alterar a configuração manualmente. Para obter mais informações, consulte Modificar o atributo de endereçamento IPv4 público para a sub-rede.

- Ao implantar um grupo de nós gerenciados em sub-redes privadas, certifique-se de que ele possa acessar o Amazon ECR para extrair imagens de contêiner. Você pode fazer isso conectando um gateway NAT à tabela de rotas da sub-rede ou adicionando os seguintes endpoints de VPC do AWS PrivateLink:
  - Interface do endpoint de API do Amazon ECR - `com.amazonaws.region-code.ecr.api`
  - Interface do endpoint de API do registro Docker do Amazon ECR - `com.amazonaws.region-code.ecr.dkr`
  - Endpoint de gateway do Amazon S3 - – `com.amazonaws.region-code.s3`

  Para conhecer alguns serviços e endpoints frequentemente, consulte the section called “Clusters privados”.
- Os grupos de nós gerenciados não podem ser implantados no AWS Outposts ou no AWS Wavelength. Os grupos de nós gerenciados podem ser criados nas Zonas locais da AWS. Para obter mais informações, consulte the section called “ Zonas locais da AWS”.
- Você pode criar vários grupos de nós gerenciados em um único cluster. Por exemplo, é possível criar um grupo de nós com a AMI padrão do Amazon Linux otimizada para o Amazon EKS para algumas workloads e outro grupo com a variante de GPU para workloads que requerem suporte para GPU.
- Se o grupo de nós gerenciados encontrar uma falha nas verificações de status para as instâncias do Amazon EC2, o Amazon EKS retornará um código de erro para ajudar você a diagnosticar o problema. Para obter mais informações, consulte the section called “Códigos de erro para o grupo de nós gerenciados”.
- O Amazon EKS adiciona rótulos do Kubernetes a instâncias de grupo de nós gerenciados. Esses rótulos fornecidos pelo Amazon EKS são prefixados com `eks.amazonaws.com`.
- O Amazon EKS drena automaticamente os nós usando a API do Kubernetes durante os encerramentos ou atualizações.
- Os orçamentos de interrupção do pod não são respeitados ao encerrar um nó com `AZRebalance` ou reduzir a contagem de nós desejada. Essas ações tentam remover os pods no nó. Mas se demorar mais de 15 minutos, o nó será encerrado, independentemente de todos os pods no nó serem encerrados. Para estender o período até que o nó seja encerrado, adicione um hook do ciclo de vida ao grupo do Auto Scaling. Para obter mais informações, consulte Adicionar ganchos de ciclo de vida, no Guia do usuário do Amazon EC2 Auto Scaling.

- Para executar o processo de drenagem corretamente após receber uma notificação de interrupção pontual ou uma notificação de rebalanceamento de capacidade, `CapacityRebalance` deve ser definido como `true`.
- A atualização de grupos de nós gerenciados respeita os orçamentos de interrupções de pods definidos para os pods. Para obter mais informações, consulte the section called “Atualizar detalhes de comportamento”.
- Não há custos adicionais para o uso dos grupos de nós gerenciados do Amazon EKS. Você paga apenas pelo recursos da AWS que provisionar.
- Se quiser criptografar volumes do Amazon EBS para os nós, você pode implantar os nós usando um modelo de execução. Para implantar nós gerenciados com volumes criptografados do Amazon EBS sem usar um modelo de execução, criptografe todos os novos volumes do Amazon EBS criados em sua conta. Para obter mais informações, consulte Criptografia por padrão no Guia do usuário do Amazon EC2.

Tipos de capacidade do grupo de nós gerenciados

Ao criar um grupo de nós gerenciados, você pode escolher o tipo de capacidade sob demanda ou spot. O Amazon EKS implanta um grupo de nós gerenciados com um grupo do Amazon EC2 Auto Scaling que contém apenas instâncias spot sob demanda ou apenas instâncias spot do Amazon EC2. Você pode agendar pods para aplicações tolerantes a falhas em grupos de nós gerenciados Spot e aplicações não tolerantes a falhas para grupos de nós Sob demanda em um único cluster do Kubernetes. Por padrão, um grupo de nós gerenciados implanta instâncias do Amazon EC2 sob demanda.

Sob demanda

Com o as instâncias sob demanda, você paga pela capacidade computacional por segundo, sem nenhum compromisso em longo prazo.

Por padrão, se você não especificar um Tipo de capacidade, o grupo de nós gerenciados será provisionado com instâncias sob demanda. Um grupo de nós gerenciados configura um grupo do Amazon EC2 Auto Scaling em seu nome com as seguintes configurações aplicadas:

- A estratégia de alocação para provisionar capacidade sob demanda é definida como `prioritized`. O grupo de nós gerenciados usa a ordem dos tipos de instâncias transferidas na API para determinar qual tipo de instância usar primeiro para atender à capacidade sob demanda. Por exemplo, você pode especificar três tipos de instância na seguinte ordem:

`c5.large`, `c4.large` e `c3.large`. Quando as instâncias sob demanda são iniciadas, o grupo de nós gerenciados atende à capacidade sob demanda, começando com o `c5.large`, depois o `c4.large` e, em seguida, o `c3.large`. Para obter mais informações, consulte Grupo do Amazon EC2 Auto Scaling) no Guia do usuário do Amazon EC2 Auto Scaling.

- O Amazon EKS adiciona o seguinte rótulo do Kubernetes a todos os nós do grupo de nós gerenciados que especifica o tipo de capacidade: `eks.amazonaws.com/capacityType: ON_DEMAND`. Você pode usar esse rótulo para agendar aplicações com estado ou intolerantes a falhas em nós sob demanda.

Spot

As instâncias spot do Amazon EC2 representam uma capacidade sobressalente do Amazon EC2 que oferecem descontos substanciais nos preços sob demanda. As instâncias spot do Amazon EC2 podem ser interrompidas com um aviso de interrupção de dois minutos, quando o EC2 precisar da capacidade de volta. Para obter mais informações, consulte Instâncias spot no Guia do usuário do Amazon EC2. Você pode configurar um grupo de nós gerenciados com as instâncias spot do Amazon EC2 para otimizar os custos dos nós de computação em execução no cluster do Amazon EKS.

Para usar instâncias spot dentro de um grupo de nós gerenciados, crie o grupo definindo o tipo de capacidade como `spot`. Um grupo de nós gerenciados configura um grupo do Amazon EC2 Auto Scaling em seu nome com as seguintes práticas recomendadas Spot aplicadas:

- Para garantir que os nós spot sejam provisionados nos pools de capacidade spot ideais, a estratégia de alocação é definida como uma das seguintes opções:
  - `price-capacity-optimized` (PCO): ao criar grupos de nós em um cluster com a versão `1.28` ou mais recente do Kubernetes, a estratégia de alocação é definida como `price-capacity-optimized`. No entanto, a estratégia de alocação não será alterada para grupos de nós já criados com `capacity-optimized` antes de os grupos de nós gerenciados pelo Amazon EKS começarem a oferecer suporte ao PCO.
  - `capacity-optimized` (CO): ao criar grupos de nós em um cluster com a versão `1.27` ou mais recente do Kubernetes, a estratégia de alocação é definida como `capacity-optimized`.

  Para aumentar o número de grupos de capacidade spot disponíveis para alocação de capacidade, configure um grupo de nós gerenciados para usar vários tipos de instância.
- O Rebalanceamento de capacidade spot do Amazon EC2 está habilitado para que o Amazon EKS possa drenar e reequilibrar os nós spot para minimizar a interrupção da aplicação quando um nó

spot correr um risco alto de interrupção. Para obter mais informações, consulte Rebalancear a capacidade do Amazon EC2 Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling.

- Quando um nó spot recebe uma recomendação de rebalanceamento, o Amazon EKS tenta iniciar automaticamente um novo nó spot de substituição.
- Se um aviso de interrupção do spot de dois minutos chegar antes que o nó spot de substituição esteja em um estado `Ready`, o Amazon EKS começa a drenar o nó spot que recebeu a recomendação de rebalanceamento. O Amazon EKS drena o nó da melhor forma possível. Como resultado, não há garantia de que o Amazon EKS aguardará que o nó de substituição se junte ao cluster antes de drenar o nó existente.
- Quando um nó spot de substituição inclui o bootstrap no estado `Ready` no Kubernetes, o Amazon EKS isola e drena o nó spot que recebeu a recomendação de rebalanceamento. Isolar o nó spot garante que o controlador de serviço não envie novas solicitações para esse nó spot. Ele também o remove da lista de nós spot íntegros e ativos. Drenar o nó spot garante que os pods em execução sejam removidos de forma adequada.

- O Amazon EKS adiciona o seguinte rótulo do Kubernetes a todos os nós do grupo de nós gerenciados que especifica o tipo de capacidade: `eks.amazonaws.com/capacityType: SPOT`. Você pode usar esse rótulo para agendar aplicações com estado ou intolerantes a falhas em nós spot.

> **Important**
>
> O EC2 emite um aviso de interrupção da instância spot dois minutos antes de encerrar a instância spot. No entanto, os pods presentes nos nós spot podem não dispor de toda a janela de dois minutos para realizar um encerramento sem interrupções. Após a emissão do aviso pelo EC2, há um atraso antes que o Amazon EKS comece a remoção dos pods. As remoções ocorrem de forma sequencial para proteger o servidor de APIs do Kubernetes, portanto, durante diversas recuperações simultâneas de instâncias spot, alguns pods podem receber avisos de remoção com atraso. Os pods podem ser encerrados abruptamente sem receber sinais de encerramento, especialmente em nós com alta densidade de pods, durante recuperações simultâneas ou ao usar períodos de carência prolongados para o encerramento. Para workloads spot, recomendamos projetar aplicações tolerantes a interrupções, empregando períodos de carência para o encerramento de 30 segundos ou menos, evitando hooks de interrupções antecipadas de longa duração e monitorando métricas de remoção de pods para compreender os períodos

reais de carência nos clusters. Para workloads que exigem a garantia de um encerramento sem interrupções, recomendamos o uso da capacidade sob demanda em vez disso.

Ao decidir se deseja implantar um grupo de nós com capacidade sob demanda ou spot, você deve considerar as seguintes condições:

- Convém usar instâncias spot para aplicações sem estado, com tolerância a falhas e flexíveis. Isso inclui workloads de treinamento em lote e de machine learning, ETLs de big data, como Apache Spark, aplicações de processamento de fila e endpoints de API sem estado. Como o spot é a capacidade sobressalente do Amazon EC2, que pode ser alterada ao longo do tempo, recomendamos usar a capacidade spot para workloads tolerantes a interrupções. Mais especificamente, a capacidade spot é adequada para workloads que podem tolerar períodos em que a capacidade necessária não está disponível.
- Recomendamos usar On-Demand para aplicações que sejam intolerantes a falhas. Isso inclui ferramentas de gerenciamento de clusters, como ferramentas operacionais e de monitoramento, implantações que exigem `StatefulSets` e aplicações com estado, como bancos de dados.
- Para maximizar a disponibilidade de suas aplicações ao usar instâncias spot, recomendamos que você configure um grupo de nós gerenciados spot para usar vários tipos de instância. Recomendamos aplicar as seguintes regras ao usar vários tipos de instância:
  - Dentro de um grupo de nós gerenciados, se você estiver usando o Cluster Autoscaler, recomendamos o uso de um conjunto flexível de tipos de instância com a mesma quantidade de vCPU e recursos de memória. Isso serve para garantir que os nós de seu cluster escalem conforme o esperado. Por exemplo, se você precisar de quatro vCPUs e oito GiB de memória, use `c3.xlarge`, `c4.xlarge`, `c5.xlarge`, `c5d.xlarge`, `c5a.xlarge`, `c5n.xlarge` ou outros tipos de instância semelhantes.
  - Para melhorar a disponibilidade das aplicações, recomendamos implantar vários grupos de nós gerenciados pelo spot. Para isso, cada grupo deverá usar um conjunto flexível de tipos de instância que tenham os mesmos recursos de vCPU e memória. Por exemplo, se você precisar de 4 vCPUs e 8 GiB de memória, recomendamos que você crie um grupo de nós gerenciados com `c3.xlarge`, `c4.xlarge`, `c5.xlarge`, `c5d.xlarge`, `c5a.xlarge`, `c5n.xlarge` ou outros tipos de instância semelhantes, e um segundo grupo de nós gerenciados com `m3.xlarge`, `m4.xlarge`, `m5.xlarge`, `m5d.xlarge`, `m5a.xlarge`, `m5n.xlarge` ou outros tipos de instância semelhantes.
  - Ao implantar o grupo de nós com o tipo de capacidade spot que estiver usando um modelo de execução personalizado, use a API para passar vários tipos de instância. Não passe um único

tipo de instância pelo modelo de execução. Para obter mais informações sobre como implantar um grupo de nós usando um modelo de execução, consulte the section called “Modelos de execução”.

Criar um grupo de nós gerenciados para seu cluster

Este tópico descreve como iniciar grupos de nós gerenciados do Amazon EKS com nós registrados no cluster do Amazon EKS. Depois que os nós ingressam no cluster, você pode implantar aplicações do Kubernetes neles.

Se esta for a primeira vez que você inicia um grupo de nós gerenciados do Amazon EKS, recomendamos que você siga um de nossos guias em Conceitos básicos. Estes guias fornecem orientações para a criação de um cluster do Amazon EKS com nós.

Important

- Os nós do Amazon EKS são instâncias padrão do Amazon EC2. Você é cobrado com base nos preços normais do Amazon EC2. Para obter mais informações, consulte Preço do Amazon EC2.
- Não é possível criar nós gerenciados em uma região da AWS onde haja AWS Outposts ou AWS Wavelength habilitados. Você pode criar nós autogerenciados em vez isso. Para obter mais informações, consulte the section called “Amazon Linux”, the section called “Windows” e the section called “Bottlerocket”. Você também pode criar um grupo de nós autogerenciado do Amazon Linux em um Outpost. Para obter mais informações, consulte the section called “Nós”.
- Se você não especificar um ID da AMI para o arquivo `bootstrap.sh` incluso no Linux ou Bottlerocket otimizado para Amazon EKS, os grupos de nós gerenciados vão impor um número máximo no valor de `maxPods`. Para instâncias com menos de 30 vCPUs, o número máximo é `110`. Para instâncias com mais de 30 vCPUs, o número máximo aumenta para `250`. Esses números são baseados nos limites de escalabilidade do Kubernetes e nas configurações recomendadas pelos testes internos da equipe de escalabilidade do Amazon EKS. Para obter mais informações, consulte a publicação no blog Plug-in do Amazon VPC CNI aumenta os limites de pods por nó.

- Um cluster existente do Amazon EKS. Para implantar, consulte the section called “Criar um cluster”.
- Um perfil do IAM existente para os nós usarem. Para criar uma, consulte the section called “Perfil do IAM do nó”. Se essa função não tiver nenhuma das políticas da VPC CNI, a função separada a seguir será necessária para os pods da VPC CNI.
- (Opcional, mas recomendado) O complemento plug-in CNI da Amazon VPC para Kubernetes configurado com o seu próprio perfil do IAM que tem a política do IAM necessária anexada a ele. Para obter mais informações, consulte the section called “Configuração para IRSA”.
- Familiaridade com as considerações listadas em Escolher um tipo de instância de nó ideal do Amazon EC2. Dependendo do tipo de instância que você escolher, pode ser que haja pré-requisitos adicionais para o seu cluster e VPC.
- Para adicionar um grupo de nós gerenciados do Windows, primeiro é necessário habilitar o suporte do Windows para o cluster. Para obter mais informações, consulte the section called “Habilitar o suporte ao Windows”.

É possível criar um grupo de nós gerenciados com uma das seguintes opções:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”

`eksctl`

Criar um grupo de nós gerenciados com o eksctl

Este procedimento exige a versão `eksctl` `0.215.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

1. (Opcional) Se a política gerenciada do IAM AmazonEKS_CNI_Policy estiver anexada ao perfil do IAM do nó do Amazon EKS, recomendamos atribuí-la a um perfil do IAM associado à conta de serviço do `aws-node` do Kubernetes. Para obter mais informações, consulte the section called “Configuração para IRSA”.

2. Crie o grupo de nós com ou sem o uso de um modelo de execução personalizado. A especificação manual de um modelo de lançamento permite uma maior personalização de um grupo de nós. Por exemplo, pode permitir a implantação de uma AMI personalizada ou fornecer argumentos para o script de `boostrap.sh` em uma AMI otimizada do Amazon EKS. Para obter uma lista completa de todas as opções e padrões disponíveis, insira o comando a seguir.

   ```
   eksctl create nodegroup --help
   ```

   No comando a seguir, substitua *`my-cluster`* pelo nome do seu cluster e *`my-mng`* pelo nome do seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.

   > **Important**
   >
   > Se você não usar um modelo de execução personalizado ao criar um grupo de nós gerenciados, não use um superiormente para o grupo de nós. Se você não especificou um modelo de execução personalizado, o sistema gerará automaticamente um modelo de execução que não é recomendável alterar manualmente. Modificar manualmente esse modelo de execução gerado poderá causar erros automaticamente.

Sem um modelo de inicialização

O `eksctl` cria um modelo de inicialização padrão do Amazon EC2 na sua conta e implanta o grupo de nós usando um modelo de inicialização que ele criou baseado nas opções que você especifica. Antes de especificar um valor para `--node-type`, consulte the section called “Tipos de instância do Amazon EC2”.

Substitua *`ami-family`* por uma palavra-chave permitida. Para obter mais informações, consulte Setting the node AMI Family (Configurar a família de AMIs de nós) na documentação do `eksctl`. Substitua *`my-key`* pelo nome do seu par de chaves ou chave pública do Amazon EC2. Essa chave é usada para SSH em seus nós depois que eles forem iniciados.

> **Note**
>
> No Windows, esse comando não habilita o SSH. Em vez disso, ele associa seu par de chaves do Amazon EC2 à instância e permite que você faça RDP na instância.

Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no Console de gerenciamento da AWS. Para obter mais informações sobre o Linux, consulte Amazon EC2 key pairs and Linux instances no Guia do usuário do Amazon EC2. Para obter informações sobre o Windows, consulte Amazon EC2 key pairs and Windows instances no Guia do usuário do Amazon EC2.

Recomendamos bloquear o acesso dos pods ao IMDS se as seguintes condições a forem verdadeiras:

- Você planeja atribuir perfis do IAM a todas as suas contas de serviço do Kubernetes para que os pods tenham apenas as permissões mínimas de que precisam.
- Nenhum pod no cluster exige acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 por outros motivos, como recuperação da região atual da AWS.

Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Se você quiser bloquear o acesso do pod ao IMDS, adicione a opção `--disable-pod-imds` ao comando a seguir.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --region region-code \
  --name my-mng \
  --node-ami-family ami-family \
  --node-type m5.large \
  --nodes 3 \
  --nodes-min 2 \
  --nodes-max 4 \
  --ssh-access \
  --ssh-public-key my-key
```

As instâncias podem, opcionalmente, atribuir um número significativamente maior de endereços IP aos pods, atribuir endereços IP aos pods de um bloco CIDR diferente do da instância e ser implantadas em um cluster sem acesso à internet. Para obter mais informações, consulte the section called “Aumento dos endereços IP”, the section called “Redes personalizadas”, e consulte the section called “Clusters privados” para obter opções adicionais a serem incluídas no comando anterior.

Os grupos de nós gerenciados calculam e aplicam um único valor para o número máximo de pods que podem ser executados em cada nó do grupo de nós, com base no tipo de instância. Se você criar um grupo de nós com diferentes tipos de instância, o menor valor calculado em todos os tipos

de instância será aplicado como o número máximo de pods que podem ser executados em cada tipo de instância no grupo de nós. Grupos de nós gerenciados calculam o valor usando o script referenciado em Pods máximos recomendados pelo Amazon EKS para cada tipo de instância do Amazon EC2.

Com um modelo de inicialização

O modelo de lançamento já deve existir e deve atender aos requisitos especificados em Fundamentos da configuração do modelo de lançamento. Recomendamos bloquear o acesso dos pods ao IMDS se as seguintes condições a forem verdadeiras:

- Você planeja atribuir perfis do IAM a todas as suas contas de serviço do Kubernetes para que os pods tenham apenas as permissões mínimas de que precisam.
- Nenhum pod no cluster exige acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 por outros motivos, como recuperação da região atual da AWS.

Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Se você quiser bloquear o acesso do pod ao IMDS, especifique as configurações necessárias no modelo de execução.

a. Copie o conteúdo a seguir para o seu dispositivo. Substitua os valores de exemplo e execute o comando modificado para criar o arquivo `eks-nodegroup.yaml`. Várias configurações que você especifica ao implantar sem um modelo de execução são movidas para o modelo de execução. Se você não especificar a `version`, a versão padrão do modelo será usada.

```
cat >eks-nodegroup.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
- name: my-mng
  launchTemplate:
    id: lt-id
    version: "1"
EOF
```

Para obter uma lista completa de configurações `eksctl` do arquivo de configurações, consulte Config file schema (Esquema do arquivo de configurações) na documentação do `eksctl` As instâncias podem, opcionalmente, atribuir um número significativamente maior de endereços IP aos pods, atribuir endereços IP aos pods de um bloco CIDR diferente do da instância e ser implantadas em um cluster sem acesso à internet de saída. Para obter mais informações, consulte the section called “Aumento dos endereços IP”, the section called “Redes personalizadas” e the section called “Clusters privados” para conferir as opções adicionais a serem adicionadas ao arquivo de configurações.

Se você não especificou um ID de AMI no modelo de execução, os grupos de nós gerenciados calculam e aplicam um único valor para o número máximo de pods que podem ser executados em cada nó do grupo de nós, com base no tipo de instância. Se você criar um grupo de nós com diferentes tipos de instância, o menor valor calculado em todos os tipos de instância será aplicado como o número máximo de pods que podem ser executados em cada tipo de instância no grupo de nós. Grupos de nós gerenciados calculam o valor usando o script referenciado em Pods máximos recomendados pelo Amazon EKS para cada tipo de instância do Amazon EC2.

Se você especificou um ID de AMI no modelo de execução, especifique o número máximo de pods que podem ser executados em cada nó do grupo de nós se estiver usando redes personalizadas ou quiser aumentar o número de endereços IP atribuídos à instância. Para obter mais informações, consulte the section called “O máximo de pods do Amazon EKS recomendado para cada tipo de instância do Amazon EC2”.

b. Implante o grupo de nós com o comando a seguir.

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

Console de gerenciamento da AWS

Crie um grupo de nós gerenciados usando o endereço Console de gerenciamento da AWS

1. Aguarde até que o status do cluster seja exibido como `ACTIVE`. Não é possível criar um grupo de nós gerenciados para um cluster que ainda não esteja `ACTIVE`.
2. Abra o console do Amazon EKS.
3. Escolha o nome do cluster em que você deseja criar um grupo de nós gerenciados.
4. Selecione a guia Compute (Computação).

5. Escolha Add Node Group (Adicionar grupo de nós).
6. Na página Configure node group (Configurar o grupo de nós) preencha os parâmetros adequadamente e escolha Next (Próximo).
   - Name (Nome) – Insira um nome exclusivo para o grupo de nós gerenciados. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.
   - Node IAM role (Perfil do IAM do nó): escolha a função da instância do nó a ser usada com o grupo de nós. Para obter mais informações, consulte the section called “Perfil do IAM do nó”.

   > ⚠ Important
   > - Não é possível usar a mesma função usada para criar clusters.
   > - Recomendamos usar um perfil que não esteja em uso atualmente por um grupo de nós autogerenciados. Caso contrário, planeje usar com um novo grupo de nós autogerenciados. Para obter mais informações, consulte the section called “Excluir”.

   - Usar: (opcional) escolha se deseja usar um modelo de execução existente. Selecione um Nome de modelo de inicialização. Depois, selecione Launch template version (Versão do modelo de inicialização). Se você não selecionar uma versão, o Amazon EKS usará a versão padrão do modelo. Os modelos de execução permitem uma maior personalização do grupo de nós, como, por exemplo, que você implante uma AMI personalizada, atribua um número significativamente maior de endereços IP a pods, atribua endereços IP a pods de um bloco CIDR diferente do da instância e implante nós em um cluster sem acesso à internet de saída. Para obter mais informações, consulte the section called “Aumento dos endereços IP”, the section called “Redes personalizadas” e the section called “Clusters privados”.

     O modelo de inicialização deve atender aos requisitos em Personalizar nós gerenciados com modelos de inicialização. Se você não usar seu próprio modelo de execução, a API do Amazon EKS criará um modelo de execução padrão do Amazon EC2 em sua conta e implantará o grupo de nós usando o modelo de execução padrão.

     Se você implementar perfis do IAM para contas de serviço, atribua as permissões necessárias diretamente a todos os pods que necessitam de acesso aos serviços da AWS, e nenhum pod em seu cluster requisitará acesso ao IMDS por outros motivos, como recuperar a região atual da AWS. Você também poderá desabilitar o acesso ao IMDS para pods que não usam redes de

host em um modelo de execução. Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

- Rótulos do Kubernetes – (Opcional) você pode optar por aplicar rótulos do Kubernetes aos nós no grupo de nós gerenciados.
- Taints do Kubernetes: (opcional) você pode optar por aplicar taints do Kubernetes aos nós do seu grupo de nós gerenciados. As opções disponíveis naAs opções disponíveis no menu Effect (Efeito) são **`NoSchedule`** , **`NoExecute`** e **`PreferNoSchedule`** . Para obter mais informações, consulte the section called “Evita que pods sejam agendados em nós específicos”.
- Etiquetas – (Opcional) você pode optar por aplicar etiquetas ao grupo de nós gerenciados pelo Amazon EKS. Essas etiquetas não se propagam para outros recursos no grupo de nós, como instâncias ou grupos do Auto Scaling. Para obter mais informações, consulte the section called “Marcando seus Recursos”.

7. Na página Set compute and scaling configuration (Definir configuração de escalabilidade e computação), preencha os parâmetros adequadamente e escolha Next (Próximo).
   - Tipo de AMI: selecione um tipo de AMI. Se estiver implementando instâncias Arm, certifique-se de analisar as considerações em AMIs Arm do Amazon Linux otimizadas do Amazon EKS antes de implementar.

     Se você especificou um modelo de inicialização na página anterior e especificou nele uma AMI, não poderá selecionar um valor. O valor do modelo é exibido. A AMI especificada no modelo deve atender aos requisitos em Especificando uma AMI.
   - Tipo de capacidade – Selecione um tipo de capacidade. Para obter mais informações sobre a escolha de um tipo de capacidade, consulte the section called “Tipos de capacidade do grupo de nós gerenciados”. Não é possível misturar diferentes tipos de capacidade dentro do mesmo grupo de nós. Se você quiser usar os dois tipos de capacidade, crie grupos de nós separados, cada um com seus próprios tipos de capacidade e instância. Consulte Reserva de GPUs para grupos de nós gerenciados para obter informações sobre provisionamento e o dimensionamento de nós de processamento acelerados por GPU.
   - Tipos de instância: por padrão, um ou mais tipos de instância são especificados. Para remover um tipo de instância padrão, selecione a caixa de seleção X no lado direito do tipo de instância. Escolha o tipo de instância a ser usado no grupo de nós gerenciados. Para obter mais informações, consulte the section called “Tipos de instância do Amazon EC2”.

     O console exibe um conjunto de tipos de instância comumente usados. Se você precisar criar um grupo de nós gerenciados com um tipo de instância que não é exibido, use `eksctl`, a CLI AWS, AWS CloudFormation ou um SDK para criar o grupo de nós. Se você especificou um

modelo de execução na página anterior, não será possível selecionar um valor, porque o tipo de instância deve ser especificado no modelo de execução. O valor do modelo de execução é exibido. Se você selecionou Spot para o tipo de capacidade, recomendamos especificar vários tipos de instância para melhorar a disponibilidade.

- Tamanho do disco: insira o tamanho do disco (em GiB) a ser usado para o volume raiz do nó.

  Se você especificou um modelo de execução na página anterior, não será possível selecionar um valor, porque ele deverá ser especificado no modelo de execução.

- Tamanho desejado – Especifique o número atual de nós que o grupo de nós gerenciados deverá manter na inicialização.

  > **Note**
  >
  > O Amazon EKS não reduz nem aumenta automaticamente a escala na horizontal do grupo de nós. No entanto, você pode configurar o Cluster Autoscaler do Kubernetes para fazer isso por você. Para obter mais informações, consulte Autoscaler do cluster na AWS.

- Tamanho mínimo – Especifique o número mínimo de nós para o qual o grupo de nós gerenciados pode ser reduzido.
- Tamanho máximo – Especifique o número máximo de nós para o qual o grupo de nós gerenciados pode ser expandido.
- Configuração da atualização de grupo de nós – (Opcional) Você pode selecionar o número ou porcentagem de nós a serem atualizados em paralelo. Esses nós estarão indisponíveis durante a atualização. Para o Máximo disponível, selecione uma das seguintes opções e especifique um Valor:
  - Number (Número): selecione e especifique o número de nós em seu grupo de nós que podem ser atualizados em paralelo.
  - Percentage (Porcentagem): selecione e especifique a porcentagem de nós em seu grupo de nós que podem ser atualizados em paralelo. Isso será útil se você tiver um grande número de nós em seu grupo de nós.
- Configuração de reparo automático de nós: (opcional) se você ativar a caixa de seleção Habilitar reparo automático de nós, o Amazon EKS substituirá automaticamente os nós quando ocorrerem problemas. Para obter mais informações, consulte the section called “Integridade de nós”.

8. Na página Specify networking (Especificar a rede), preencha os parâmetros conforme necessário e, em seguida, escolha Next (Próximo).

   - Sub-redes – Escolha as sub-redes nas quais iniciar os nós gerenciados.

   > **Important**
   >
   > Se estiver executando uma aplicação com estado em várias zonas de disponibilidade com suporte de volumes do Amazon EBS e usando o Cluster Autoscaler do Kubernetes, você deverá configurar vários grupos de nós, cada um com escopo de uma única zona de disponibilidade. Além disso, você deverá ativar o recurso `--balance-similar-node-groups`.

   > **Important**
   >
   > - Se você escolher uma sub-rede pública e o cluster tiver somente o endpoint do servidor de API público habilitado, a sub-rede deverá ter o `MapPublicIPOnLaunch` definido como `true` para que as instâncias ingressem com êxito em um cluster. Se a sub-rede tiver sido criada usando `eksctl` ou os modelos do Amazon EKS vended AWS CloudFormation em ou após 26 de março de 2020, essa configuração já estará definida como `true`. Se as sub-redes foram criadas com `eksctl` ou com os modelos AWS CloudFormation antes de 26 de março de 2020, você precisará alterar a configuração manualmente. Para obter mais informações, consulte Modificar o atributo de endereçamento IPv4 público para a sub-rede.
   > - Se você usar um modelo de execução e especificar várias interfaces de rede, o Amazon EC2 não atribuirá automaticamente um endereço `IPv4` público, mesmo que o `MapPublicIpOnLaunch` seja definido como `true`. Para que os nós ingressem no cluster nesse cenário, você deve habilitar o endpoint do servidor de API privado do cluster ou iniciar nós em uma sub-rede privada com acesso de saída à Internet fornecido por meio de um método alternativo, como um Gateway NAT. Para obter mais informações, consulte Endereçamento de IP de instâncias do Amazon EC2, no Guia do usuário do Amazon EC2.

   - Configurar o acesso SSH aos nós (opcional). Habilitar o SSH permite que você se conecte às suas instâncias e reúna informações de diagnóstico quando houver problemas. É altamente

recomendado habilitar o acesso remoto ao criar o grupo de nós. Não será possível habilitar o acesso remoto depois que o grupo estiver criado.

Se você optou por usar um modelo de execução, essa opção não será exibida. Para habilitar o acesso remoto aos nós, especifique um par de chaves no modelo de execução e verifique se a porta adequada está aberta aos nós nos grupos de segurança especificados no modelo de execução. Para obter mais informações, consulte the section called “Usar grupos de segurança personalizados”.

> **Note**
>
> No Windows, esse comando não habilita o SSH. Em vez disso, ele associa seu par de chaves do Amazon EC2 à instância e permite que você faça RDP na instância.

- Para o par de chaves SSH (Opcional), escolha uma chave SSH do Amazon EC2 para usar. Para obter mais informações sobre o Linux, consulte Amazon EC2 key pairs and Linux instances no Guia do usuário do Amazon EC2. Para obter informações sobre o Windows, consulte Amazon EC2 key pairs and Windows instances no Guia do usuário do Amazon EC2. Se você optar por usar um modelo de execução, não será possível selecionar um. Quando uma chave SSH do Amazon EC2 é fornecida para grupos de nós que usam AMIs Bottlerocket, o contêiner administrativo também é habilitado. Para obter mais informações, consulte Contêiner administrador no GitHub.
- Para Allow SSH remote access from (Permitir o acesso remoto de SSH) se você quiser limitar o acesso a instâncias específicas, selecione os grupos de segurança associados a essas instâncias. Se você não selecionar grupos de segurança específicos, o acesso SSH será permitido de qualquer lugar na Internet (`0.0.0.0/0`).

9. Na página Review and create (Revisar e criar), reveja a configuração do grupo de nós gerenciados e escolha Create (Criar).

   Se os nós não conseguirem se juntar ao cluster, consulte the section called “Falha nos nós ao ingressar no cluster” no capítulo Solução de problemas.

10. Observe o status de seus nós e aguarde até que eles atinjam o status `Ready`.

```
kubectl get nodes --watch
```

11. (Somente nós de GPU) Caso escolha um tipo de instância de GPU e a AMI acelerada otimizada para Amazon EKS, você deverá aplicar o plug-in de dispositivo NVIDIA para Kubernetes como

um DaemonSet no cluster. Substitua *vX.X.X* pela versão desejada do NVIDIA/k8s-device-plugin antes de executar o seguinte comando.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Instalar complementos do Kubernetes

Agora que você tem um cluster do Amazon EKS com nós, já poderá iniciar a instalação dos complementos do Kubernetes e implantar as aplicações no cluster. Os seguintes tópicos de documentação ajudam a estender a funcionalidade do seu cluster:

- A entidade principal do IAM que criou o cluster é a única entidade que pode fazer chamadas para o servidor de API do Kubernetes com o `kubectl` ou o Console de gerenciamento da AWS. Se quiser que outras entidades principais do IAM tenham acesso ao cluster, será necessário adicioná-los. Para obter mais informações, consulte the section called “Acesso à API do Kubernetes” e the section called “Permissões obrigatórias”.
- Recomendamos bloquear o acesso dos pods ao IMDS se as seguintes condições a forem verdadeiras:
  - Você planeja atribuir perfis do IAM a todas as suas contas de serviço do Kubernetes para que os pods tenham apenas as permissões mínimas de que precisam.
  - Nenhum pod no cluster exige acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 por outros motivos, como recuperação da região atual da AWS.

  Para obter mais informações, consulte Restringir o acesso ao perfil de instância atribuído ao nó de processamento.
- Configure o Cluster Autoscaler do Kubernetes para ajustar automaticamente o número de nós nos grupos de nós.
- Implante uma aplicação de exemplo no cluster.
- Organize e monitore os recursos do cluster com ferramentas importantes para gerenciar seu cluster.

Atualizar um grupo de nós gerenciados para seu cluster

Quando você inicia uma atualização de grupo de nós gerenciados, o Amazon EKS atualiza automaticamente os nós para você, concluindo as etapas listadas em Entenda cada fase das

atualizações de nós. Se você estiver usando uma AMI otimizada do Amazon EKS, o Amazon EKS aplica automaticamente os patches de segurança mais recentes e as atualizações do sistema operacional aos nós como parte da versão mais recente da AMI.

Existem vários cenários nos quais é útil atualizar a versão ou a configuração do grupo de nós gerenciados do Amazon EKS:

- Você atualizou a versão do Kubernetes para seu cluster do Amazon EKS e quer atualizar os nós para usar a mesma versão do Kubernetes.
- Uma nova versão da AMI está disponível para o grupo de nós gerenciados. Para obter mais informações sobre as versões da AMI, consulte estas seções:
  - the section called “Obter informações de versão”
  - the section called “Bottlerocket”
  - the section called “Obter informações de versão”
- Você deseja ajustar a contagem mínima, máxima ou desejada das instâncias no seu grupo de nós gerenciados.
- Você deseja adicionar ou remover os rótulos do Kubernetes das instâncias no seu grupo de nós gerenciados.
- Você deseja adicionar ou remover as tags da AWS do seu grupo de nós gerenciados.
- Você precisa implantar uma nova versão de um modelo de execução com alterações de configuração, como uma AMI personalizada atualizada.
- Você implantou a versão `1.9.0` ou mais recente do complemento CNI da Amazon VPC, habilitou o complemento para delegação de prefixos e deseja novas instâncias do AWS Nitro System em um grupo de nós para ser compatível com um número significativamente maior de pods. Para obter mais informações, consulte the section called “Aumento dos endereços IP”.
- Você habilitou a delegação de prefixos IP para nós do Windows e deseja que novas instâncias do AWS Nitro System em um grupo de nós sejam compatíveis um número significativamente maior de pods. Para obter mais informações, consulte the section called “Aumento dos endereços IP”.

Se houver uma nova versão da AMI mais recente do que a versão do Kubernetes do grupo de nós gerenciado, você poderá atualizar esta para usar a nova versão da AMI. Da mesma forma, se o cluster estiver executando uma versão mais recente do Kubernetes do que o grupo de nós, você poderá atualizar o grupo de nós para usar a versão mais recente da AMI correspondente à versão do Kubernetes do cluster.

Quando um nó em um grupo de nós gerenciados é encerrado por causa de uma operação de escalabilidade ou atualização, os pods nesse nó são drenados primeiro. Para obter mais informações, consulte the section called “Atualizar detalhes de comportamento”.

Atualizar uma versão do grupo de nós

Você pode atualizar a versão de um grupo de nós com uma das seguintes opções:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”

A versão para a qual você atualiza não pode ser superior à versão do ambiente de gerenciamento.

`eksctl`

Atualizar um grupo de nós gerenciados usando o **`eksctl`**

Atualize um grupo de nós gerenciados para a versão mais recente da AMI da mesma versão do Kubernetes atualmente implantada nos nós com o comando a seguir. Substitua cada *`valor de exemplo`* por seus próprios valores.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code
```

> **Note**
>
> Se você estiver atualizando um grupo de nós implantado com um modelo de execução para uma nova versão do modelo de execução, adicione o `--launch-template-version` *`version-number`* ao comando anterior. O modelo de inicialização deve atender aos requisitos descritos em Personalizar nós gerenciados com modelos de inicialização. Se o modelo de inicialização incluir uma AMI personalizada, a AMI deverá atender aos requisitos em Especificando uma AMI. Quando você atualiza o grupo de nós para uma versão mais recente do modelo de execução, todos os nós são reciclados para corresponder à nova configuração da versão do modelo de execução especificado.

Não é possível atualizar diretamente um grupo de nós implantado sem um modelo de execução para uma nova versão do modelo de execução. Em vez disso, você deve implantar um novo grupo de nós usando o modelo de execução para atualizar o grupo de nós para uma nova versão de modelo de execução.

Você pode atualizar um grupo de nós para a mesma versão que a versão do Kubernetes do ambiente de gerenciamento. Por exemplo, se tiver um cluster executando o Kubernetes `1.33`, você poderá atualizar os nós que estão atualmente executando o Kubernetes `1.32` para a versão `1.33` com o comando a seguir.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code \
  --kubernetes-version=1.33
```

Console de gerenciamento da AWS

Atualizar um grupo de nós gerenciados usando o Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Escolha o cluster que contém o grupo de nós a ser atualizado.
3. Se pelo menos um grupo de nós tiver uma atualização disponível, uma caixa será exibida na parte superior da página notificando você sobre a atualização disponível. Se você selecionar a guia Compute (Computação), verá a opção Update now (Atualizar agora) na coluna AMI release version (Versão de AMI) na tabela Node Groups (Grupos de nós) para o grupo de nós que tem atualizações disponíveis. Para atualizar o grupo de nós, selecione Update now (Atualizar agora).

   Você não verá uma notificação para grupos de nós que foram implantados com uma AMI personalizada. Se os nós forem implantados com uma AMI personalizada, conclua as etapas a seguir para implantar uma nova AMI personalizada atualizada.

   a. Crie uma nova versão da AMI.

   b. Crie uma nova versão do modelo de execução com o novo ID da AMI.

   c. Atualize os nós para a nova versão do modelo de execução.
4. Na caixa de diálogo Update node group version (Atualizar versão do grupo de nós), ative ou desative as seguintes opções:

- Update Node Group version (Atualizar versão do grupo de nós): essa opção não estará disponível se você tiver implantado uma AMI personalizada ou se a AMI otimizada do Amazon EKS estiver atualmente na versão mais recente do cluster.
- Change launch template version (Alterar versão do modelo de inicialização): essa opção não estará disponível se o grupo de nós for implantado sem um modelo de inicialização personalizado. Você só pode atualizar a versão do modelo de execução para um grupo de nós que tenha sido implantado com um modelo de execução personalizado. Selecione a Launch template version (Versão do modelo de inicialização) para a qual você deseja atualizar o grupo de nós. Se o grupo de nós estiver configurado com uma AMI personalizada, a versão selecionada também deverá especificar uma AMI. Quando você atualiza para uma versão mais recente do modelo de execução, todos os nós são reciclados para corresponder à nova configuração da versão do modelo de execução especificada.

5. Em Update strategy (Estratégia de atualização), selecione uma das seguintes opções:
   - Atualização contínua: esta opção respeita os orçamentos de interrupções de pods no cluster. As atualizações falharão se houver um problema de orçamento de interrupções de pods que faça com que o Amazon EKS não consiga drenar corretamente os pods que estão em execução no grupo de nós.
   - Forçar atualização: esta opção não respeita os orçamentos de interrupções de pods. As atualizações ocorrem independentemente de problemas de orçamento de interrupções de pods, forçando a reinicialização dos nós.
6. Selecione Atualizar.

Editar uma configuração do grupo de nós

Você pode modificar algumas configurações de um grupo de nós gerenciados.

1. Abra o console do Amazon EKS.
2. Escolha o cluster que contém o grupo de nós a ser editado.
3. Selecione a guia Compute (Computação).
4. Selecione o grupo de nós a ser editado e escolha Edit (Editar).
5. (Opcional) Na página Edit node group (Editar grupo de nós) faça o seguinte:
   a. Edite a configuração de escalonamento do grupo de nós.
      - Tamanho desejado – Especifique o número atual de nós que o grupo de nós gerenciados deve manter.

- Tamanho mínimo – Especifique o número mínimo de nós para o qual o grupo de nós gerenciados pode ser reduzido.
- Tamanho máximo – Especifique o número máximo de nós para o qual o grupo de nós gerenciados pode ser expandido. Para obter o número máximo de nós permitidos em um grupo de nós, consulte the section called “Cotas de serviço”.

b. (Opcional) Adicione ou remova rótulos do Kubernetes dos nós no grupo de nós. Os rótulos mostrados aqui são apenas aqueles que você aplicou com o Amazon EKS. Outros rótulos que não são exibidos aqui podem existir nos nós.

c. (Opcional) Adicione ou remova taints do Kubernetes dos nós no grupo de nós. As taints adicionadas podem ter o efeito de **`NoSchedule`**, **`NoExecute`** ou **`PreferNoSchedule`**. Para obter mais informações, consulte the section called “Evita que pods sejam agendados em nós específicos”.

d. (Opcional) Adicione ou remova tags do recurso de grupo de nós. Essas etiquetas são aplicadas somente ao grupo de nós do Amazon EKS. Não são propagadas em outros recursos como sub-redes ou instâncias do Amazon EC2 no grupo de nós.

e. (Opcional) Edite a configuração da atualização do grupo de nós. Selecione o número ou a porcentagem.

- Number (Número): selecione e especifique o número de nós em seu grupo de nós que podem ser atualizados em paralelo. Esses nós estarão indisponíveis durante a atualização.
- Percentage (Porcentagem): selecione e especifique a porcentagem de nós em seu grupo de nós que podem ser atualizados em paralelo. Esses nós estarão indisponíveis durante a atualização. Isso será útil se houver muitos nós em seu grupo de nós.

f. Quando terminar de editar, escolha Salvar alterações.

> **Important**
>
> Ao atualizar a configuração do grupo de nós, a modificação de `NodegroupScalingConfig` não respeita os orçamentos de interrupções de pods (PDBs). Ao contrário do processo de atualização do grupo de nós (que drena os nós e respeita os PDBs durante a fase de upgrade), a atualização da configuração de dimensionamento faz com que os nós sejam encerrados imediatamente por meio de uma chamada de redução de escala do Auto Scaling Group (ASG). Isso acontece sem considerar os PDBs, independentemente do tamanho para o qual você está reduzindo a escala. Isso significa que, quando você reduz o `desiredSize`

de um grupo de nós gerenciados do Amazon EKS, os pods são removidos assim que os nós são encerrados, sem honrar nenhum PDB.

Entenda cada fase das atualizações de nós

A estratégia de atualização do nó de processamento gerenciado do Amazon EKS tem quatro fases diferentes descritas nas seções a seguir.

Fase de configuração

A fase de configuração contém estas etapas:

1. Ele cria uma nova versão do modelo de execução do Amazon EC2 para o grupo do Auto Scaling associado ao grupo de nós. A nova versão do modelo de inicialização usa a AMI de destino ou uma versão do modelo de inicialização personalizado para a atualização.
2. Ele atualiza o grupo do Auto Scaling para usar a versão mais recente do modelo de execução.
3. Ele determina a quantidade máxima de nós a serem atualizados em paralelo usando a propriedade `updateConfig` do grupo de nós. O máximo indisponível tem uma cota de 100 nós. O valor padrão é um nó. Para obter mais informações, consulte a propriedade updateConfig na Referência da API do Amazon EKS.

Fase de aumento de escala vertical

Na atualização de nós em um grupo de nós gerenciados, os nós atualizados são iniciados na mesma zona de disponibilidade daqueles que estão sendo atualizados. Para garantir esse posicionamento, usamos o rebalanceamento de zonas de disponibilidade do Amazon EC2. Para obter mais informações, consulte Rebalanceamento de zonas de disponibilidade no Guia do usuário do Amazon EC2 Auto Scaling. Para atender a esse requisito, é possível a inicialização de até duas instâncias por zona de disponibilidade no grupo de nós gerenciados.

A fase de aumento de escala vertical tem estas etapas:

1. Ele aumenta o tamanho máximo do grupo do Auto Scaling e o tamanho desejado pelo que for maior:
   - Até duas vezes o número de zonas de disponibilidade em que o grupo do Auto Scaling está implantado.

- O máximo indisponível de atualização.

  Por exemplo, se o grupo de nós tiver cinco zonas de disponibilidade e `maxUnavailable` como um, o processo de atualização poderá iniciar no máximo dez nós. No entanto, se `maxUnavailable` for 20 (ou qualquer valor superior a 10), o processo iniciaria 20 novos nós.

2. Depois de escalar o grupo do Auto Scaling, ele verifica se os nós que usam a configuração mais recente estão presentes no grupo de nós. Essa etapa só é concluída corretamente quando atende a estes critérios:
   - Pelo menos um novo nó é iniciado em todas as zonas de disponibilidade em que o nó existe.
   - Cada novo nó deve estar no estado `Ready`.
   - Os novos nós devem ter rótulos aplicados ao Amazon EKS.

     São os rótulos do Amazon EKS aplicados aos nós de processamento em um grupo de nós regulares:

     - `eks.amazonaws.com/nodegroup-image=$amiName`
     - `eks.amazonaws.com/nodegroup=$nodeGroupName`

     São os rótulos do Amazon EKS aplicados aos nós de processamento em um modelo de execução personalizado ou em um grupo de nós da AMI:

     - `eks.amazonaws.com/nodegroup-image=$amiName`
     - `eks.amazonaws.com/nodegroup=$nodeGroupName`
     - `eks.amazonaws.com/sourceLaunchTemplateId=$launchTemplateId`
     - `eks.amazonaws.com/sourceLaunchTemplateVersion=$launchTemplateVersion`
3. Ele marca os nós como não agendáveis para evitar o agendamento de novos pods. Também rotula os nós com `node.kubernetes.io/exclude-from-external-load-balancers=true` para remover os nós antigos dos balanceadores de carga antes de encerrá-los.

A seguir, estão os motivos conhecidos que causam um erro `NodeCreationFailure` nesta fase:

Capacidade insuficiente na zona de disponibilidade

Existe a possibilidade de que a zona de disponibilidade não tenha capacidade dos tipos de instância solicitados. É recomendável configurar vários tipos de instância ao criar um grupo de nós gerenciados.

Limites de instâncias do EC2 na sua conta

Pode ser necessário aumentar o número de instâncias do Amazon EC2 que a conta pode executar simultaneamente usando o Service Quotas. Para obter mais informações, consulte Service Quotas do EC2 no Guia do usuário do Amazon Elastic Compute Cloud para instâncias Linux.

Dados de usuário personalizados

Os dados de usuário personalizados às vezes podem interromper o processo de bootstrap. Esse cenário pode fazer com que o `kubelet` não seja iniciado no nó ou com que os nós não recebam os rótulos esperados do Amazon EKS. Para obter mais informações, consulte the section called "Especificar uma AMI".

Quaisquer alterações que prejudiquem a integridade ou a prontidão de um nó

Pressão no disco do nó, pressão na memória e condições semelhantes podem impedir que um nó passe para o estado `Ready`.

Cada nó deve inicializar em até 15 minutos

Se algum nó levar mais de 15 minutos para inicializar e ingressar no cluster, isso fará com que a atualização atinja o tempo limite. Este é o runtime total para inicializar um novo nó, calculado desde quando um novo nó é necessário até quando ele se junta ao cluster. Ao atualizar um grupo de nós gerenciados, o contador de tempo começa assim que o tamanho do grupo do Auto Scaling aumenta.

Fase de atualização

A fase de atualização se comporta de duas maneiras diferentes, dependendo da estratégia de atualização. Há duas estratégias de atualização: padrão e mínima.

Recomendamos a estratégia padrão na maioria dos cenários. Ela cria novos nós antes de encerrar os antigos para que a capacidade disponível seja mantida durante a fase de atualização. A estratégia mínima é útil em cenários em que você está limitado a recursos ou custos, por exemplo, com aceleradores de hardware, como GPUs. Ela encerra os nós antigos antes de criar os novos para que a capacidade total nunca aumente além da quantidade configurada.

A estratégia de atualização padrão tem as seguintes etapas:

1. Ela aumenta a quantidade de nós (contagem desejada) no grupo do Auto Scaling, fazendo com que o grupo de nós crie nós adicionais.

2. Ele seleciona aleatoriamente um nó que precisa ser atualizado, até o máximo indisponível configurado para o grupo de nós.
3. Drena os pods do nó. Se os pods não saírem do nó em até 15 minutos e não houver um sinalizador de força, a fase de atualização falhará com um erro `PodEvictionFailure`. Para tal cenário, você pode aplicar o sinalizador de força com a solicitação `update-nodegroup-version` para excluir os pods.
4. Isola o nó após cada pod ser removido e aguarda 60 segundos. Isso é feito para que o controlador de serviço não envie novas solicitações para este nó e remova esse nó da lista de nós ativos.
5. Ele envia uma solicitação de encerramento para o grupo do Auto Scaling do nó isolado.
6. Ele repete as etapas de atualização anteriores até que não haja nós no grupo de nós implantados com a versão anterior do modelo de execução.

A estratégia de atualização mínima tem as seguintes etapas:

1. Ele isola todos os nós do grupo de nós no início para que o controlador de serviço não envie novas solicitações para esses nós.
2. Ele seleciona aleatoriamente um nó que precisa ser atualizado, até o máximo indisponível configurado para o grupo de nós.
3. Ele drena os pods dos nós selecionados. Se os pods não saírem do nó em até 15 minutos e não houver um sinalizador de força, a fase de atualização falhará com um erro `PodEvictionFailure`. Para tal cenário, você pode aplicar o sinalizador de força com a solicitação `update-nodegroup-version` para excluir os pods.
4. Depois que cada pod é removido e aguarda 60 segundos, ele envia uma solicitação de encerramento para o grupo do Auto Scaling dos nós selecionados. O grupo do Auto Scaling cria novos nós (o mesmo número de nós selecionados) para substituir a capacidade ausente.
5. Ele repete as etapas de atualização anteriores até que não haja nós no grupo de nós implantados com a versão anterior do modelo de execução.

Erros **`PodEvictionFailure`** durante a fase de atualização

A seguir, estão os motivos conhecidos que causam um erro `PodEvictionFailure` nesta fase:

PDB agressivo

Um PDB agressivo está definido no pod, ou há vários PDBs apontando para o mesmo pod.

Implantação que tolera todas as taints

Depois que todos os pods são removidos, espera-se que o nó fique vazio porque o nó sofreu taints nas etapas anteriores. Porém, se a implantação tolerar todos os taints, é mais provável que o nó não esteja vazio, causando falha na remoção dos pods.

Fase de redução de escala vertical

A fase de redução de escala vertical diminui o tamanho máximo do grupo do Auto Scaling e o tamanho desejado para retornar aos valores anteriores ao início da atualização.

Se o fluxo de trabalho Upgrade determinar que o Cluster Autoscaler está aumentando a escala na vertical do grupo de nós durante a fase de redução de escala na vertical do fluxo de trabalho, ele será encerrado imediatamente sem trazer o grupo de nós de volta ao tamanho original.

Personalizar nós gerenciados com modelos de execução

Para obter o máximo de personalização, é possível implantar nós gerenciados com seu próprio modelo de execução, seguindo as etapas descritas nesta página. O uso de um modelo de execução permite, entre outras funcionalidades, fornecer argumentos de inicialização durante a implantação de um nó (por exemplo, argumentos extras do kubelet), atribuir endereços IP aos pods de um bloco CIDR diferente daquele atribuído ao endereço IP do nó, implantar uma AMI ou uma CNI personalizadas com base nos seus requisitos para os nós.

Ao fornecer seu próprio modelo de execução na primeira criação de um grupo de nós gerenciados, você também terá maior flexibilidade posteriormente. Contanto que você implemente um grupo de nós gerenciados com seu próprio modelo de execução, é possível atualizá-lo iterativamente para uma versão diferente do mesmo modelo de execução. Quando você atualiza para o grupo de nós para uma versão mais recente do modelo de execução, todos os nós do grupo são reciclados para corresponder à nova configuração da versão do modelo de execução especificada.

Os grupos de nós gerenciados são sempre implantados com um modelo de execução do grupo do Amazon EC2 Auto Scaling. Quando você não fornece um modelo de execução, a API do Amazon EKS cria um modelo com valores padrão em sua conta automaticamente. No entanto, não recomendamos modificar os modelos de execução gerados automaticamente. Além disso, os grupos de nós existentes que não usarem um modelo de execução personalizado não poderão ser atualizados diretamente. Em vez disso, você deverá criar um novo grupo de nós com um modelo de execução personalizado para fazer isso.

Conceitos básicos do modelo de execução

Você pode criar um modelo de lançamento do Amazon EC2 Auto Scaling com o Console de gerenciamento da AWS, AWS CLI ou um AWS SDK. Para obter mais informações, consulte Criação de um modelo de execução para um grupo do Auto Scaling no Guia do usuário do Amazon EC2 Auto Scaling. Algumas das configurações em um modelo de execução são semelhantes às usadas para a configuração do nó gerenciado. Ao implantar ou atualizar um grupo de nós com um modelo de execução, algumas configurações devem ser especificadas na configuração do grupo de nós ou no modelo de execução. Não especifique uma configuração em ambos os locais. Se uma configuração existir onde não deveria, as operações, como criar ou atualizar um grupo de nós, falharão.

A tabela a seguir lista as configurações proibidas em um modelo de execução. Também são listadas configurações semelhantes, se houver, que são obrigatórias na configuração do grupo de nós gerenciados. As configurações listadas são as configurações que aparecem no console. Eles podem ter nomes semelhantes, mas diferentes, na CLI e no SDK do AWS.

Modelo de execução – Proibido	Configuração do grupo de nós do Amazon EKS
Subnet (Sub-rede) em Network interfaces (Interfaces de rede) (Adicionar interface de rede)	Subnet (Sub-redes) em Node Group network configuration (Configuração de rede do grupo de nós) na página Specify networking (Especificar rede)
IAM instance profile (Perfil de instância do IAM) em Advanced details (Detalhes avançados)	Node IAM Role (Função do IAM do nó) em Node Group configuration (Configuração do grupo de nós) na página Configure Node Group (Configurar o grupo de nós)
Shutdown behavior (Comportamento de desligamento) e Stop - Hibernate behavior (Parar - comportamento de hibernação) em Advanced details (Detalhes avançados). Reter o padrão Não incluir na configuração do modelo de execução para as duas configurações.	Não há equivalente. O Amazon EKS deve controlar o ciclo de vida da instância, não o grupo Auto Scaling.

A tabela a seguir lista as configurações proibidas em uma configuração de grupo de nós gerenciados. Também lista configurações semelhantes, se houver, que serão necessárias em um modelo de execução. As configurações listadas são as configurações que aparecem no console. Eles podem ter nomes semelhantes na CLI e no SDK do AWS.

Configuração do grupo de nós do Amazon EKS - Proibidas	Modelo de execução
(Somente se você tiver especificado uma AMI personalizada em um modelo de execução) Em Tipo de AMI, na Configuração de computação do grupo de nós, na página Definir configuração de computação e escalabilidade, o console exibe Especificado no modelo de execução e o ID da AMI especificada. Se Imagens de aplicações e sistemas operacionais (imagem de máquina da Amazon) não tiver sido especificado no modelo de execução, você poderá selecionar uma AMI na configuração do grupo de nós.	Application and OS Images (Amazon Machine Image) [Imagens de aplicações e sistemas operacionais (imagem de máquina da Amazon)] em Launch template contents (Conteúdo do modelo de execução): especifique um ID se você tiver um dos seguintes requisitos: • Usar uma AMI personalizada. Se você especificar uma AMI que não atenda aos requisitos listados na seção Especificar uma AMI, a implantação do grupo de nós apresentará falhas. • Deseja fornecer dados de usuário para fornecer argumentos para o arquivo `bootstrap.sh` incluído com uma AMI otimizada do Amazon EKS. É possível habilitar para que suas instâncias atribuam um número significativamente maior de endereços IP aos pods, atribuam endereços IP aos pods de um bloco de CIDR diferente do endereço da instância ou implantem um cluster privado sem acesso de saída à internet. Para obter mais informações, consulte os tópicos a seguir. • Atribuir mais endereços IP aos nós do Amazon EKS com prefixos

Configuração do grupo de nós do Amazon EKS - Proibidas	Modelo de execução
	• Implementar pods em sub-redes alternativas com rede personalizada • Implementar clusters privados com acesso limitado à internet • Especificar uma AMI
Tamanho do disco em Node Group compute configuration (Configuração de computação do grupo de nós) a página Set compute and scaling configuration (Definir a configuração da computação e escalabilidade). Exibe o console Especificado no modelo de execução.	Size (Tamanho) em Storage (Volumes) Armazenamento (Volumes) (Add New volume) (Adicionar novo volume). Você deve especificar isso no modelo de execução.
Par de chaves SSH em Node Group configuration (Configuração do grupo de nós) na página Specify Networking (Especificar rede) — O console exibe a chave especificada no modelo de execução ou exibe Not specified in launch template (Não especificado no modelo de execução).	Key pair name (Nome do par de chaves) em Key pair (login) (Par de chaves).
Não é possível especificar grupos de segurança de origem que tenham permissão de acesso remoto ao usar um modelo de execução.	Security groups (Grupos de segurança) em Network settings (Configurações de rede) para a instância ou Security groups (Grupos de segurança) em Network interfaces (Interfaces de rede) (Add network interface (Adicionar interface de rede), mas não os dois. Para obter mais informações, consulte the section called “Usar grupos de segurança personalizados”.

Note

- Se você implantar um grupo de nós usando um modelo de execução, especifique zero ou um tipo de instância em Launch template contents (Iniciar o conteúdo do modelo) em um modelo de execução. Se preferir, você pode especificar de 0 a 20 tipos de instância em Instance types (Tipos de instância) na página Set compute and scaling configuration (Definir configuração de computação e escalabilidade) no console. Ou você pode fazer isso com outras ferramentas que utilizam a API do Amazon EKS. Se você especificar um tipo de instância em um modelo de execução e usar esse modelo para implantar o grupo de nós, não será possível especificar nenhum tipo de instância no console ou usar outras ferramentas que usam a API do Amazon EKS. Se você não especificar um tipo de instância em um modelo de execução, no console ou usando outras ferramentas que usem a API do Amazon EKS, será usado o tipo de instância `t3.medium`. Se o grupo de nós estiver usando o tipo de capacidade spot, recomendamos especificar vários tipos de instância usando o console. Para obter mais informações, consulte the section called "Tipos de capacidade do grupo de nós gerenciados".
- Se algum contêiner implantado no grupo de nós usar o Serviço de Metadados de Instância Versão 2, defina a propriedade Metadata response hop limit (Limite de salto de resposta) como 2 no modelo de execução. Para obter mais informações, consulte Metadados da instância e dados do usuário no Manual do usuário do Amazon EC2.
- Os modelos de execução não são compatíveis com o recurso `InstanceRequirements` que permite a seleção flexível do tipo de instância.

Etiquetar instâncias do Amazon EC2

Você pode usar o parâmetro `TagSpecification` de um modelo de execução para especificar quais etiquetas serão aplicadas às instâncias do Amazon EC2 em seu grupo de nós. A entidade do IAM que chama o método das APIs `CreateNodegroup` ou `UpdateNodegroupVersion` devem ter permissões para `ec2:RunInstances` e `ec2:CreateTags`, e as etiquetas devem ser adicionadas ao modelo de execução.

Usar grupos de segurança personalizados

Você pode usar um modelo de execução para especificar os grupos de segurança do Amazon EC2 a serem aplicados às instâncias do grupo de nós. Isso pode ser no parâmetro dos grupos de segurança do nível de instância ou como parte dos parâmetros de configuração da interface de rede.

Porém, não é possível criar um modelo de execução que especifique o nível da instância e os grupos de segurança da interface de rede. Considere as seguintes condições que se aplicam ao uso de grupos de segurança personalizados com grupos de nós gerenciados:

- Quando o Console de gerenciamento da AWS é usado, o Amazon EKS permite modelos de execução somente com uma única especificação de interface de rede.
- Por padrão, o Amazon EKS aplica o Grupo de segurança de clusterpara as instâncias no grupo de nós, a fim de facilitar a comunicação entre os nós e o ambiente de gerenciamento. Se você especificar grupos de segurança personalizados no modelo de execução usando uma das opções mencionadas anteriormente, o Amazon EKS não adicionará o grupo de segurança do cluster. Então, você deve garantir que as regras de entrada e saída dos grupos de segurança permitam a comunicação com o endpoint do cluster. Se as regras do grupo de segurança estiverem incorretas, os nós de processamento não poderão ingressar no cluster. Para obter mais informações sobre regras do grupo de segurança, consulte the section called “Requisitos para grupos de segurança”.
- Se você precisar de acesso SSH às instâncias no grupo de nós, inclua um grupo de segurança que permita esse acesso.

Dados do usuário do Amazon EC2

O modelo de execução contém uma seção para dados de usuário personalizados. É possível especificar configurações para o grupo de nós desta seção sem criar AMIs personalizadas individuais manualmente. Para obter mais informações sobre as configurações disponíveis para o Bottlerocket, consulte Usar dados de usuário no GitHub.

Você pode fornecer dados de usuário do Amazon EC2 no modelo de execução usando `cloud-init` na inicialização das instâncias. Para obter mais informações, consulte a documentação de cloud-init. Os dados de usuário podem ser usados para executar operações de configuração comuns. Isso inclui as seguintes operações:

- Incluindo usuários ou grupos
- Instalar pacotes

Os dados de usuário do Amazon EC2 em modelo de execução que são usados com grupos de nós gerenciados devem estar no formato arquivo MIME de várias partes para AMIs do Amazon Linux e em formato TOML para AMIs do Bottlerocket. Isso ocorre porque seus dados de usuário são mesclados com os dados de usuário do Amazon EKS necessários para que os nós participem do cluster. Não especifique nenhum comando nos dados de usuário que iniciem ou modifiquem

`kubelet`. Isso é executado como parte dos dados de usuário mesclados pelo Amazon EKS. Certos parâmetros do `kubelet`, como a definição de rótulos em nós, podem ser configurados diretamente por meio da API de grupos de nós gerenciados.

> **Note**
>
> Para obter mais informações sobre personalização avançada do `kubelet`, inclusive para iniciá-la manualmente ou passar parâmetros de configuração personalizados, consulte the section called “Especificar uma AMI”. Seu um ID de AMI personalizada for especificado em um modelo de execução, o Amazon EKS não mesclará os dados de usuário.

Os detalhes a seguir fornecem mais informações sobre a seção de dados de usuário.

Dados do usuário do Amazon Linux 2

Você pode combinar vários blocos de dados de usuário em um único arquivo MIME de várias partes. Por exemplo, você pode combinar um boothook de nuvem que configure o daemon do Docker com um script do shell de dados do usuário que instale um pacote personalizado. Um arquivo em várias partes MIME consiste nos seguintes componentes:

- O tipo de conteúdo e a declaração de limite da parte: – `Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="`
- A declaração da versão MIME: – `MIME-Version: 1.0`
- Um ou mais blocos de dados do usuário, que contêm os seguintes componentes:
  - O limite de abertura, que sinaliza o início de um bloco de dados do usuário: `--==MYBOUNDARY==`
  - A declaração de tipo de conteúdo para o bloco: `Content-Type: text/cloud-config; charset="us-ascii"`. Para obter mais informações sobre os tipos de conteúdo, consulte a documentação do cloud-init.
  - O conteúdo de dados de usuário (por exemplo, uma lista de comandos de shell ou diretivas `cloud-init`).
  - O limite de fechamento, que sinaliza o término do arquivo em várias partes MIME: `--==MYBOUNDARY==--`

Veja a seguir um exemplo de arquivo MIME com várias partes que você pode usar para criar seu próprio.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
echo "Running custom user data script"

--==MYBOUNDARY==--
```

Dados do usuário do Amazon Linux 2023

O Amazon Linux 2023 (AL2023) introduz um novo processo de inicialização do nó `nodeadm` que usa um esquema de configuração YAML. Se estiver usando grupos de nós autogerenciados ou uma AMI com um modelo de inicialização, será necessário fornecer, de forma explícita, metadados de cluster adicionais ao criar um novo grupo de nós. Veja a seguir um exemplo dos parâmetros mínimos necessários, em que `apiServerEndpoint`, `certificateAuthority` e `cidr` do serviço passaram a ser necessários:

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

Normalmente, você definirá essa configuração nos dados do usuário no estado em que se encontra ou incorporada em um documento MIME com várias partes:

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
```

```
kind: NodeConfig spec: [...]

--BOUNDARY--
```

No AL2, os metadados desses parâmetros eram revelados na chamada de API `DescribeCluster` do Amazon EKS. Com o AL2023, esse comportamento foi alterado, uma vez que a chamada de API adicional corre o risco de sofrer controle de utilização durante grandes aumentos de escala vertical para nós. Essa alteração não afetará você se estiver usando grupos de nós gerenciados sem um modelo de inicialização ou se estiver usando o Karpenter. Para obter mais informações sobre `certificateAuthority` e sobre o serviço de `cidr`, consulte `DescribeCluster` na Referência de API do Amazon EKS.

Aqui está um exemplo completo dos dados do usuário do AL2023 que combinam um script de shell para personalizar o nó (como instalar pacotes ou pré-armazenar imagens de contêiner em cache) com a configuração de `nodeadm` necessária. Este exemplo mostra personalizações comuns, incluindo: * Instalação de pacotes adicionais do sistema * Pré-armazenamento em cache de imagens de contêiner para melhorar o tempo de inicialização do pod * Configuração do proxy HTTP * Configuração de sinalizadores `kubelet` para rotulagem de nós

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -o errexit
set -o pipefail
set -o nounset

# Install additional packages
yum install -y htop jq iptables-services

# Pre-cache commonly used container images
nohup docker pull public.ecr.aws/eks-distro/kubernetes/pause:3.2 &

# Configure HTTP proxy if needed
cat > /etc/profile.d/http-proxy.sh << 'EOF'
export HTTP_PROXY="http://proxy.example.com:3128"
export HTTPS_PROXY="http://proxy.example.com:3128"
export NO_PROXY="localhost,127.0.0.1,169.254.169.254,.internal"
```

```
EOF

--BOUNDARY
Content-Type: application/node.eks.aws

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      clusterDNS:
      - 10.100.0.10
    flags:
    - --node-labels=app=my-app,environment=production

--BOUNDARY--
```

Dados do usuário do Bottlerocket

O Bottlerocket estrutura dados de usuário no formato TOML. Informe os dados de usuário a serem mesclados com os dados de usuário fornecidos pelo Amazon EKS. Por exemplo, você pode fornecer outras configurações do `kubelet`.

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

Para obter mais informações sobre as configurações compatíveis, consulte a documentação do Bottlerocket do GitHub. É possível configurar rótulos de nós e taints em seus dados de usuário. Porém, recomendamos configurá-los em seu grupo de nós. O Amazon EKS aplica essas configurações quando você faz isso.

Quando os dados de usuário são mesclados, a formatação não é preservada, mas o conteúdo permanece o mesmo. A configuração que você fornece nos dados de usuário substitui todas as configurações definidas pelo Amazon EKS. Dessa maneira, se você definir

`settings.kubernetes.max-pods` ou `settings.kubernetes.cluster-dns-ip`, esses valores dos dados de usuário são aplicados aos nós.

O Amazon EKS não oferece suporte a todos os TOML válidos. Veja a seguir uma lista de formatos conhecidos não compatíveis:

- Aspas em chaves entre aspas: `'quoted "value"' = "value"`
- Aspas de escape em valores: `str = "I'm a string. \"You can quote me\""`
- Floats mistos e números inteiros: `numbers = [ 0.1, 0.2, 0.5, 1, 2, 5 ]`
- Tipos mistos em matrizes: `contributors = ["foo@example.com", { name = "Baz", email = "baz@example.com" }]`
- Cabeçalhos entre colchetes com chaves entre aspas: `[foo."bar.baz"]`

Dados do usuário do Windows

Os dados do usuário do Windows usam os comandos do PowerShell. Ao criar um grupo de nós gerenciados, os dados de usuário personalizados se combinam com os dados de usuário gerenciados do Amazon EKS. Os comandos do PowerShell vêm em primeiro lugar, seguidos dos comandos de dados de usuário gerenciados, tudo em uma tag `<powershell></powershell>`.

> **Important**
>
> Ao criar grupos de nós do Windows, o Amazon EKS atualiza o `aws-auth ConfigMap` para permitir que os nós baseados em Linux façam parte do cluster. O serviço não configura automaticamente as permissões para AMIs do Windows. Se você estiver usando nós do Windows, precisará gerenciar o acesso por meio da API de entrada de acesso ou atualizando `aws-auth ConfigMap` diretamente. Para obter mais informações, consulte the section called "Habilitar o suporte ao Windows".

> **Note**
>
> Quando nenhum ID de AMI for especificado no modelo de execução, não use o script de boostrap do Amazon EKS para Windows nos dados de usuário para configurar o Amazon EKS.

A seguir, um exemplo de dados do usuário.

```
<powershell>
Write-Host "Running custom user data script"
</powershell>
```

Especificar uma AMI

Se você tiver um dos seguintes requisitos, especifique um ID de AMI na caixa `ImageId` do modelo de execução. Selecione o requisito que você tem para obter informações adicionais.

Fornecer dados de usuário para passar argumentos para o arquivo **`bootstrap.sh`** incluído com uma AMI do Linux/Bottlerocket otimizada para Amazon EKS

Bootstrapping é um termo utilizado para descrever o processo de adicionar comandos que podem ser executados quando uma instância é iniciada. Por exemplo, o bootstrapping permite o uso de argumentos adicionais do kubelet. Você pode passar os argumentos para o script `bootstrap.sh` usando `eksctl` sem especificar um modelo de inicialização. Ou faça isso especificando as informações na seção de dados de usuário de um modelo de execução.

eksctl sem especificar um modelo de inicialização

Crie um arquivo chamado *`my-nodegroup.yaml`* com o seguinte conteúdo. Substitua cada *`valor de exemplo`* por seus próprios valores. Os argumentos `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` são opcionais. No entanto, defini-los permite que o script `bootstrap.sh` evite fazer uma chamada `describeCluster`. Isso é útil em configurações de cluster privado ou clusters em que há aumento e redução de escala na horizontal com frequência. Para obter mais informações sobre o script `bootstrap.sh`, consulte o arquivo bootstrap.sh no GitHub.

- O único argumento necessário é o nome do cluster(*`my-cluster`*).
- Para recuperar um ID de AMI otimizado para `ami-`*`1234567890abcdef0`* , consulte as seguintes seções:
  - Recuperar IDs de AMI do Amazon Linux recomendadas
  - Recuperar IDs de AMI do Bottlerocket recomendadas
  - Recuperar IDs de AMI do Microsoft Windows recomendadas
- Para recuperar o *`certificado de autoridade`* do seu cluster, execute o seguinte comando.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Para recuperar o *api-server-endpoint* do seu cluster, execute o seguinte comando.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- O valor para o `--dns-cluster-ip` é o serviço CIDR com `.10` no final. Para recuperar o *service-cidr* do seu cluster, execute o seguinte comando. Por exemplo, se o valor retornado para for `ipv4 10.100.0.0/16`, então seu valor é *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Este exemplo fornece um argumento adicional do `kubelet` para definir um valor `max-pods` personalizado usando o valor do script `bootstrap.sh` incluído com a AMI otimizada para o Amazon EKS. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Para ajuda com a seleção de *my-max-pods-value*, consulte the section called “O máximo de pods do Amazon EKS recomendado para cada tipo de instância do Amazon EC2”.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    instanceType: m5.large
    privateNetworking: true
    disableIMDSv1: true
    labels: { x86-al2-specified-mng }
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster \
        --b64-cluster-ca certificate-authority \
```

```
        --apiserver-endpoint api-server-endpoint \
        --dns-cluster-ip service-cidr.10 \
        --kubelet-extra-args '--max-pods=my-max-pods-value' \
        --use-max-pods false
```

Para cada opção de arquivo `eksctl config` disponível, consulte Esquema do arquivo config, na documentação do `eksctl`. O utilitário `eksctl` ainda cria um modelo de inicialização para você e preenche seus dados de usuário com dados fornecidos no arquivo `config`.

Crie um grupo de nós com o comando a seguir.

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

Dados do usuário em um modelo de lançamento

Especifique as seguintes informações na seção de dados do usuário do modelo de execução. Substitua cada *valor de exemplo* por seus próprios valores. Os argumentos `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` são opcionais. No entanto, defini-los permite que o script `bootstrap.sh` evite fazer uma chamada `describeCluster`. Isso é útil em configurações de cluster privado ou clusters em que há aumento e redução de escala na horizontal com frequência. Para obter mais informações sobre o script `bootstrap.sh`, consulte o arquivo bootstrap.sh no GitHub.

- O único argumento necessário é o nome do cluster(*my-cluster*).
- Para recuperar o *certificado de autoridade* do seu cluster, execute o seguinte comando.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Para recuperar o *api-server-endpoint* do seu cluster, execute o seguinte comando.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- O valor para o `--dns-cluster-ip` é o serviço CIDR com `.10` no final. Para recuperar o *service-cidr* do seu cluster, execute o seguinte comando. Por exemplo, se o valor retornado para for `ipv4 10.100.0.0/16`, então seu valor é *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Este exemplo fornece um argumento adicional do `kubelet` para definir um valor `max-pods` personalizado usando o valor do script `bootstrap.sh` incluído com a AMI otimizada para o Amazon EKS. Para ajuda com a seleção de *`my-max-pods-value`*, consulte the section called "O máximo de pods do Amazon EKS recomendado para cada tipo de instância do Amazon EC2".

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
/etc/eks/bootstrap.sh my-cluster \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
  --use-max-pods false

--==MYBOUNDARY==--
```

Fornecer dados de usuário para passar argumentos para o arquivo **`Start-EKSBootstrap.ps1`** incluído com uma AMI do Windows otimizada para Amazon EKS

Bootstrapping é um termo utilizado para descrever o processo de adicionar comandos que podem ser executados quando uma instância é iniciada. Você pode passar os argumentos para o script `Start-EKSBootstrap.ps1` usando `eksctl` sem especificar um modelo de inicialização. Ou faça isso especificando as informações na seção de dados de usuário de um modelo de execução.

Se você quiser especificar um ID de AMI do Windows, tenha em mente as seguintes considerações:

- Você deve usar um modelo de inicialização e fornecer os comandos de bootstrap necessários na seção de dados do usuário. Para recuperar o ID do Windows desejado, você pode usar a tabela em Criar nós com AMIs otimizadas do Windows.

- Existem vários limites e condições. Por exemplo, você deve adicionar `eks:kube-proxy-windows` ao mapa de configuração do AWS IAM Authenticator. Para obter mais informações, consulte the section called “Limites e condições ao especificar uma ID de AMI”.

Especifique as seguintes informações na seção de dados do usuário do modelo de execução. Substitua cada *`valor de exemplo`* por seus próprios valores. Os argumentos `-APIServerEndpoint`, `-Base64ClusterCA` e `-DNSClusterIP` são opcionais. No entanto, defini-los permite que o script `Start-EKSBootstrap.ps1` evite fazer uma chamada `describeCluster`.

- O único argumento necessário é o nome do cluster(*`my-cluster`*).
- Para recuperar o *`certificado de autoridade`* do seu cluster, execute o seguinte comando.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- Para recuperar o *`api-server-endpoint`* do seu cluster, execute o seguinte comando.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- O valor para o `--dns-cluster-ip` é o serviço CIDR com `.10` no final. Para recuperar o *`service-cidr`* do seu cluster, execute o seguinte comando. Por exemplo, se o valor retornado para for `ipv4 10.100.0.0/16`, então seu valor é *`10.100.0.10`*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- Para obter argumentos adicionais, consulte the section called “Parâmetros de configuração do script de bootstrap”.

> **Note**
>
> Se você estiver usando um CIDR de serviço personalizado, será necessário especificá-lo usando o parâmetro `-ServiceCIDR`. Do contrário, a resolução de DNS para pods no cluster falhará.

```
<powershell>
```

```
[string]$EKSBootstrapScriptFile = "$env:ProgramFiles\Amazon\EKS\Start-EKSBootstrap.ps1"
& $EKSBootstrapScriptFile -EKSClusterName my-cluster `
  -Base64ClusterCA certificate-authority `
  -APIServerEndpoint api-server-endpoint `
  -DNSClusterIP service-cidr.10
</powershell>
```

Executar uma AMI personalizada devido a requisitos específicos de segurança, conformidade ou política interna

Para obter mais informações, consulte Imagens de máquina da Amazon (AMIs) no Guia do usuário do Amazon EC2. A especificação de compilação da AMI do Amazon EKS contém recursos e scripts de configuração para desenvolver uma AMI do Amazon EKS personalizada baseada no Amazon Linux. Para obter mais informações, consulte Amazon EKS AMI Build Specification (Especificação de compilação da AMI do Amazon EKS) no GitHub. Para criar AMIs personalizadas instaladas com outros sistemas operacionais, consulte Amazon EKS Sample Custom AMIs (AMIs personalizadas de amostra do Amazon EKS) no GitHub.

Não é possível usar referências de parâmetros dinâmicos para IDs de AMI em Modelos de Lançamento usados com grupos de nós gerenciados.

> **Important**
>
> Ao especificar uma AMI, o Amazon EKS não mescla nenhum dado do usuário. Em vez disso, você é responsável por fornecer os comandos do `bootstrap` para que os nós se integrem ao cluster. Se os nós não conseguirem se integrar ao cluster, as ações do Amazon EKS `CreateNodegroup` e do `UpdateNodegroupVersion` também falharão.

Limites e condições ao especificar uma ID de AMI

Estes são os limites e condições envolvidos na especificação de um ID de AMI com grupos de nós gerenciados:

- Você deve criar um novo grupo de nós para alternar entre especificar um ID de AMI em um modelo de execução e não especificar um ID de AMI.
- Você não é notificado no console quando uma versão mais recente da AMI estiver disponível. Para atualizar seu grupo de nós para uma versão mais recente da AMI, é necessário criar uma nova

versão do modelo de execução com um ID de AMI atualizado. Em seguida, é necessário atualizar o grupo de nós com a nova versão do modelo de execução.

- Os campos a seguir não podem ser definidos na API se você especificar um ID de AMI:
  - `amiType`
  - `releaseVersion`
  - `version`
- Qualquer `taints` conjunto na API é aplicado de forma assíncrona se você especificar um ID de AMI. Para aplicar contaminações antes de um nó ingressar no cluster, você deve transmiti-las para `kubelet` os dados do usuário usando a sinalização da linha de comando `--register-with-taints`. Para obter mais informações, consulte kubelet na documentação do Kubernetes.
- Ao especificar um ID de AMI personalizado para grupos de nós gerenciados do Windows, adicione `eks:kube-proxy-windows` ao mapa de configuração do AWS IAM Authenticator. Essa API é necessária para o DNS funcionar bem.
  a. Abra o mapa de configuração do AWS IAM Authenticator para edição.

  ```
  kubectl edit -n kube-system cm aws-auth
  ```

  b. Adicione essa entrada à lista de `groups` abaixo de cada `rolearn` associado aos nós do Windows. Seu mapa de configuração deve ser semelhante a aws-auth-cm-windows.yaml.

  ```
  - eks:kube-proxy-windows
  ```

  c. Salve o arquivo e saia do seu editor de texto.
- Para qualquer AMI que usa um modelo de execução personalizado, o `HttpPutResponseHopLimit` padrão para grupos de nós gerenciados é definido como 2.

Excluir um grupo de nós gerenciados do seu cluster

Este tópico descreve como você pode excluir um grupo de nós gerenciados do Amazon EKS. Quando você exclui um grupo de nós gerenciados, o Amazon EKS primeiro define o tamanho mínimo, máximo e desejado do grupo do Auto Scaling como zero. Isso faz com que seu grupo de nós reduza a escala na vertical.

Antes que cada instância seja encerrada, o Amazon EKS envia um sinal para que a drenagem desse nó seja realizada. Durante o processo de drenagem, o Kubernetes faz o seguinte para cada pod no nó: executa quaisquer ganchos do ciclo de vida `preStop` configurados, envia sinais `SIGTERM`

para os contêineres e, em seguida, aguarda o `terminationGracePeriodSeconds` para um encerramento ordenado. Caso a drenagem do nó não seja concluída após cinco minutos, o Amazon EKS permite que o Auto Scaling dê prosseguimento ao encerramento forçado da instância. Depois que todas as instâncias forem encerradas, o grupo do Auto Scaling será excluído.

> **Important**
>
> Se você excluir um grupo de nós gerenciado que usa um perfil do IAM do nó que não é usado por outro grupo de nós gerenciados no cluster, a função será removida do `ConfigMap aws-auth`. Se algum grupo de nós autogerenciados no cluster estiver usando a mesma função do IAM do nó, os nós autogerenciados serão movidos para o status `NotReady`. Além disso, a operação do cluster também é interrompida. Para adicionar um mapeamento para o perfil que você está usando somente para os grupos de nós autogerenciados, consulte the section called “Criar entradas de acesso”, se a versão da plataforma do seu cluster for pelo menos a versão mínima listada na seção de pré-requisitos de Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso EKS. Se a versão da plataforma for anterior à versão mínima exigida para entradas de acesso, você poderá adicionar a entrada novamente ao `aws-auth ConfigMap`. Para obter mais informações, insira `eksctl create iamidentitymapping --help` no seu terminal.

Você pode excluir um grupo de nós gerenciados com:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”
- the section called “AWS CLI”

`eksctl`

Excluir um grupo de nós gerenciados com **`eksctl`**

Insira o comando a seguir. Substitua `<example value>` por seus próprios valores.

```
eksctl delete nodegroup \
  --cluster <my-cluster> \
  --name <my-mng> \
  --region <region-code>
```

Para obter mais opções, consulte Excluir e drenar grupos de nós na documentação do `eksctl`.

Console de gerenciamento da AWS

Excluir um grupo de nós gerenciados com Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Na página Clusters, escolha o cluster que contém o grupo de nós a ser excluído.
3. Na página do cluster selecionado, escolha a guia Computação.
4. Na seção Node groups (Grupos de nós), escolha o grupo de nós a ser excluído. Escolha Excluir.
5. Na caixa de diálogo de confirmação Excluir grupo de nós, insira o nome do grupo de nós. Escolha Excluir.

AWS CLI

Excluir um grupo de nós gerenciados com AWS CLI

1. Insira o comando a seguir. Substitua `<example value>` por seus próprios valores.

```
aws eks delete-nodegroup \
  --cluster-name <my-cluster> \
  --nodegroup-name <my-mng> \
  --region <region-code>
```

2. Se o parâmetro `cli_pager=` estiver definido na configuração da CLI, use as teclas de seta do teclado para navegar pelo conteúdo da resposta exibida. Pressione a tecla q quando terminar.

   Para obter mais opções, consulte o comando `delete-nodegroup` em AWS CLI Command Reference.

Fazer a manutenção dos nós por conta própria com nós autogerenciados

Um cluster contém um ou mais nós do Amazon EC2 em que os pods estão agendados. Os nós do Amazon EKS são executados na conta da AWS e se conectam ao ambiente de gerenciamento do cluster pelo endpoint do servidor da API do cluster. Você é cobrado por elas com base nos preços do Amazon EC2. Para obter mais informações, consulte Definição de preço do Amazon EC2.

Um cluster pode conter vários grupos de nós. Cada grupo de nós contém um ou mais nós que são implantados em um grupo do Amazon EC2 Auto Scaling. O tipo de instância dos nós dentro do grupo pode variar, por exemplo, ao usar a seleção de tipo de instância baseada em atributos com o Karpenter. Todas as instâncias de um grupo de nós devem usar a perfil do IAM de nós do Amazon EKS.

O Amazon EKS fornece imagens de máquina da Amazon (AMI) especializada que são chamadas de AMIs otimizadas do Amazon EKS. As AMIs estão configuradas para funcionar com o Amazon EKS. Seus componentes incluem `containerd`, `kubelet` e o AWS IAM Authenticator. As AMIs também contêm um script de bootstrap especializado que permite que ele descubra e conecte-se ao ambiente de gerenciamento do cluster automaticamente.

Se você restringir o acesso ao endpoint público do cluster usando blocos CIDR, é recomendável também habilitar o acesso ao endpoint privado. Isso serve para que os nós possam se comunicar com o cluster. Sem o endpoint privado ativado, os blocos CIDR especificados para acesso público devem incluir as origens de saída da VPC. Para obter mais informações, consulte the section called “Acesso ao endpoint do cluster”.

Para adicionar nós autogerenciados ao cluster do Amazon EKS, consulte os tópicos a seguir. Caso execute os nós autogerenciados manualmente, adicione a tag a seguir a cada nó, verificando se `<cluster-name>` corresponde ao cluster. Para obter mais informações, consulte Adicionar e excluir tags em um recurso individual. Se você seguir as etapas nos guias a seguir, a etiqueta necessária será adicionada ao nó para você.

Chave	Valor
`kubernetes.io/cluster/<cluster-name>`	`owned`

Important

As tags no serviço de metadados de instância (IMDS) do Amazon EC2 não são compatíveis com os nós do EKS. Quando as tags de metadados de instância estão habilitadas, não é possível o uso de barras ('/') nos valores das tags. Essa limitação pode causar falhas na inicialização de instâncias, principalmente ao usar ferramentas de gerenciamento de nós, como o Karpenter ou o Cluster Autoscaler, pois esses serviços dependem de tags que contêm barras para um funcionamento adequado.

Para obter mais informações sobre nós, de um ponto de vista geral do Kubernetes, consulte Nodes (Nós) na documentação do Kubernetes.

Tópicos

- Criar nós autogerenciados do Amazon Linux
- Criar nós Bottlerocket autogerenciados do Bottlerocket
- Criar nós autogerenciados do Microsoft Windows
- Criar nós autogerenciados do Ubuntu Linux
- Atualizar nós autogerenciados para seu cluster

Criar nós autogerenciados do Amazon Linux

Este tópico descreve como iniciar grupos do Auto Scaling de nós do Linux que são registrados no cluster do Amazon EKS. Depois que os nós ingressam no cluster, você pode implantar aplicações do Kubernetes neles. Além disso, é possível iniciar nós autogerenciados do Amazon Linux ao usar `eksctl` ou o Console de gerenciamento da AWS. Se você precisar lançar nós em AWS Outposts, consulte the section called “Nós”.

- Um cluster existente do Amazon EKS. Para implantar, consulte the section called “Criar um cluster”. Se você tiver sub-redes na região AWS em que AWS Outposts, AWS Wavelength ou AWS Local Zones estiverem ativados, essas sub-redes não devem ter sido passadas quando você criou o cluster.
- Um perfil do IAM existente para os nós usarem. Para criar uma, consulte the section called “Perfil do IAM do nó”. Se essa função não tiver nenhuma das políticas da VPC CNI, a função separada a seguir será necessária para os pods da VPC CNI.
- (Opcional, mas recomendado) O complemento plug-in CNI da Amazon VPC para Kubernetes configurado com o seu próprio perfil do IAM que tem a política do IAM necessária anexada a ele. Para obter mais informações, consulte the section called “Configuração para IRSA”.
- Familiaridade com as considerações listadas em Escolher um tipo de instância de nó ideal do Amazon EC2. Dependendo do tipo de instância que você escolher, pode ser que haja pré-requisitos adicionais para o seu cluster e VPC.

Você pode iniciar nós do Linux autogerenciados usando uma das seguintes opções:

- the section called “`eksctl`”

- the section called “Console de gerenciamento da AWS”

`eksctl`

Iniciar nós autogerenciados do Linux usando **`eksctl`**

1. Instale a versão `0.215.0` ou posterior da ferramenta de linha de comando da `eksctl` instalada no seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
2. (Opcional) Se a política gerenciada do IAM AmazonEKS_CNI_Policy estiver anexada ao perfil do IAM do nó do Amazon EKS, recomendamos atribuí-la a um perfil do IAM associado à conta de serviço do `aws-node` do Kubernetes. Para obter mais informações, consulte the section called “Configuração para IRSA”.
3. O comando a seguir cria um grupo de nós em um cluster existente. Substitua *`al-nodes`* por um nome para o grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Substitua *`my-cluster`* pelo nome do cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster. Substitua o *`valor de exemplo`* restante por seus próprios valores. Os nós são criados com a mesma versão do Kubernetes que o plano de controle, por padrão.

   Antes de escolher um valor para `--node-type`, revise Escolher um tipo ideal de instância de nó do Amazon EC2.

   Substitua *`my-key`* pelo nome do seu par de chaves ou chave pública do Amazon EC2. Essa chave é usada para SSH em seus nós depois que eles forem iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no Console de gerenciamento da AWS. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2.

   Crie seu grupo de nós com o comando a seguir.

   > **Important**
   >
   > Se você quiser implantar um grupo de nós nas sub-redes AWS Outposts, Wavelength ou Local Zone, há outras considerações a serem feitas:

- As sub-redes não devem ter passado quando você criou o cluster.
- É necessário criar o grupo de nós com um arquivo config que especifique as sub-redes e `volumeType: gp2`. Para obter mais informações, consulte Criar um grupo de nós em um arquivo de configuração e Esquema de arquivo Config na documentação do `eksctl`.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --name al-nodes \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --ssh-access \
  --managed=false \
  --ssh-public-key my-key
```

Para implantar um grupo de nós que:

- Possa atribuir um número significativamente maior de endereços IP as pods do que a configuração padrão, consulte the section called “Aumento dos endereços IP”.
- Possa atribuir endereços `IPv4` aos pods de um bloco CIDR diferente do bloco da instância, consulte the section called “Redes personalizadas”.
- Possa atribuir endereços `IPv6` aos pods e serviços, consulte the section called “IPv6”.
- Não tenha acesso de saída à Internet, consulte the section called “Clusters privados”.

  Para obter uma lista completa de todas as opções e padrões disponíveis, digite o comando a seguir.

  ```
  eksctl create nodegroup --help
  ```

  Se os nós não conseguirem se juntar ao cluster, consulte the section called “Falha nos nós ao ingressar no cluster” no capítulo Solução de problemas.

  Veja abaixo um exemplo de saída. Várias linhas são emitidas enquanto os nós são criados. Uma das últimas linha de saída é o exemplo de linha a seguir.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

4. (Opcional) Implante uma aplicação de exemplo para testar o cluster e os nós do Linux.
5. Recomendamos bloquear o acesso dos pods ao IMDS se as seguintes condições a forem verdadeiras:
   - Você planeja atribuir perfis do IAM a todas as suas contas de serviço do Kubernetes para que os pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum pod no cluster exige acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 por outros motivos, como recuperação da região atual da AWS.

   Para obter mais informações, consulte Restringir o acesso ao perfil de instância atribuído ao nó de processamento.

Console de gerenciamento da AWS

Etapa 1: iniciar nós autogerenciados do Linux usando Console de gerenciamento da AWS

1. Baixe a versão mais recente do modelo do AWS CloudFormation.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2025-11-26/
amazon-eks-nodegroup.yaml
```

2. Aguarde até que o status do cluster seja exibido como `ACTIVE`. Se você executar os nós antes que o cluster esteja ativo, o registro dos nós no cluster falhará, e você terá de executá-los novamente.
3. Abra o console do AWS CloudFormation.
4. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos, padrão).
5. Para Specify template (Especificar modelo), selecione Upload a template file (Fazer upload de um arquivo de modelo) e depois Choose file (Escolher arquivo).
6. Selecione o arquivo `amazon-eks-nodegroup.yaml` que você baixou.
7. Escolha Próximo.
8. Na página Specify stack details (Especificar detalhes da pilha), insira os parâmetros de acordo e escolha Next (Próximo):

- Nome da pilha: escolha um nome para a pilha do AWS CloudFormation. Por exemplo, você pode chamá-lo de *my-cluster-nodes*. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
- ClusterName: insira o nome que você usou ao criar o cluster do Amazon EKS. Esse nome deve ser igual ao nome do cluster, ou seus nós não poderão ingressar no cluster.
- ClusterControlPlaneSecurityGroup: Escolha o valor SecurityGroups da saída do AWS CloudFormation que você gerou ao criar sua VPC.

  As etapas a seguir mostram uma operação para recuperar o grupo aplicável.

  a. Abra o console do Amazon EKS.

  b. Escolha o nome do cluster.

  c. Escolha a guia Redes.

  d. Use o valor de Additional security group (Grupos de segurança adicionais) como referência ao selecionar na lista suspensa ClusterControlPlaneSecurityGroup.
- ApiServerEndpoint: informe o endpoint do servidor de API do cluster do EKS. É possível encontrar essa informação na seção Detalhes do console do cluster do EKS.
- CertificateAuthorityData: informe os dados da Autoridade de Certificação codificados em base64, que também podem ser encontrados na seção Detalhes do console do cluster do EKS.
- ServiceCidr: informe o intervalo CIDR usado para alocar endereços IP aos serviços Kubernetes dentro do cluster. É possível encontrar essa informação na guia Redes do console do cluster do EKS.
- AuthenticationMode: selecione o modo de autenticação em uso no cluster do EKS, consultando a guia Acesso no console do cluster do EKS.
- NodeGroupName: insira um nome para o grupo de nós. Esse nome poderá ser usado superiormente para identificar o grupo de nós do Auto Scaling que for criado para os nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.
- NodeAutoScalingGroupMinSize: digite o número mínimo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeAutoScalingGroupDesiredCapacity: insira o número desejado de nós para o qual dimensionar quando a pilha for criada.

- NodeAutoScalingGroupMaxSize: digite o número máximo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeInstanceType: escolha um tipo de instância para os nós. Para obter mais informações, consulte the section called “Tipos de instância do Amazon EC2”.
- NodeImageIdSSMParam: preenchido previamente com o parâmetro do gerenciador de sistemas do Amazon EC2 de uma AMI do Amazon Linux 2023 otimizada para o Amazon EKS mais recente, que é compatível com uma versão variável do Kubernetes. Para usar uma versão secundária diferente do Kubernetes compatível com o Amazon EKS, substitua *1.XX* por uma versão compatível diferente. Recomendamos especificar a mesma versão do Kubernetes que seu cluster.

  Você também pode substituir *o amazon-linux-2023* por um tipo diferente de AMI. Para obter mais informações, consulte the section called “Obter os IDs mais recentes”.

  > **Note**
  >
  > As AMIs dos nós do Amazon EKS são baseadas no Amazon Linux. Você pode monitorar eventos de segurança ou privacidade para o Amazon Linux 2023 no Centro de segurança do Amazon Linux ou fazendo a assinatura do feed RSS associado. Os eventos de segurança e privacidade incluem uma visão geral do problema, quais pacotes são afetadas e como atualizar suas instâncias para corrigir o problema.

- NodeImageId: (opcional) se estiver usando sua própria AMI personalizada (em vez da AMI otimizada do Amazon EKS), insira um ID de AMI de nó para a região da AWS. Se você especificar um valor aqui, ele substituirá todos os valores no campo NodeImageIdSSMParam.
- NodeVolumeSize: especifique um tamanho de volume raiz para os nós, em GiB.
- NodeVolumeType: especifique um tipo de volume raiz para os nós.
- KeyName: insira o nome de um par de chaves SSH do Amazon EC2; que você pode usar para conectar-se usando SSH em seus nós, depois de iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no Console de gerenciamento da AWS. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2.
- VpcId: insira o ID da VPC que você criou.
- Sub-redes: escolha as sub-redes criadas para a sua VPC. Se você criou sua VPC usando as etapas descritas em Criar uma Amazon VPC para seu cluster do Amazon EKS, especifique apenas as sub-redes privadas dentro da VPC para que seus nós sejam iniciados. É possível

ver quais sub-redes são privadas abrindo cada link de sub-rede na guia Networking (Redes) do cluster.

> **Important**
>
> - Se alguma das sub-redes for pública, ela deverá ter a atribuição automática de atribuição de endereço IP público habilitada. Se a configuração não estiver habilitada para a sub-rede pública, os nós implantados nessa sub-rede pública não receberão um endereço IP público e não poderão se comunicar com o cluster nem com outros produtos da AWS. Se a sub-rede tiver sido implantada antes de 26 de março de 2020 usando um dos modelos de VPC do Amazon EKS AWS CloudFormation ou usando `eksctl`, a atribuição automática de endereço IP público estará desativada para sub-redes públicas. Para obter informações sobre como habilitar a atribuição de endereço IP público para uma sub-rede, consulte Modificar o atributo de endereçamento IPv4 público para a sub-rede. Se o nó for implantado em uma sub-rede privada, ele poderá se comunicar com o cluster e com outros produtos da AWS por um gateway NAT.
> - Se as sub-redes não tiverem acesso à Internet, certifique-se de estar ciente das considerações e etapas adicionais em Implantar clusters privados com acesso limitado à Internet.
> - Se você selecionar as sub-redes AWS Outposts, Wavelength ou Local Zone, as sub-redes não devem ter sido passadas quando você criou o cluster.

9. Selecione as opções desejadas na página Configure stack options (Configurar opções de pilha) e depois escolha Next (Próximo).
10. Marque a caixa de seleção à esqu erda de I acknowledge that AWS CloudFormation might create IAM resources. (Eu reconheço que o CloudFormation pode criar recursos de IAM) e, em seguida, selecione Create stack (Criar pilha).
11. Quando a criação da pilha for concluída, selecione-a no console e escolha Outputs (Saídas). Se você estiver usando os modos de autenticação `EKS API` ou `EKS API and ConfigMap`, esta é a última etapa.
12. Se você estiver usando o modo de autenticação `ConfigMap`, registre o NodeInstanceRole do grupo de nós criado.

Etapa 2: habilite os nós para participar do cluster

> **Note**
>
> As próximas duas etapas são necessárias somente se o modo de autenticação ConfigMap estiver em uso no cluster do EKS. Além disso, caso os nós tenham sido executados em uma VPC privada sem acesso de saída à internet, certifique-se de permitir que os nós se conectem ao cluster usando a VPC.

1. Verifique se você já tem um `aws-auth` `ConfigMap`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se você receber um `aws-auth` `ConfigMap`, atualize-o conforme necessário.

   a. Abra o `ConfigMap` para edição.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Adicione uma nova `mapRoles` entrada conforme necessário. Defina o valor `rolearn` para o valor NodeInstanceRole que você registrou no procedimento anterior.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

   c. Salve o arquivo e saia do seu editor de texto.

3. Se você recebeu um erro informando "`Error from server (NotFound): configmaps "aws-auth" not found`, aplique o estoque`ConfigMap`.

   a. Faça download do mapa de configuração.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

b. No arquivo `aws-auth-cm.yaml`, configure o valor `rolearn` para o valor NodeInstanceRole que você registrou no procedimento anterior. Você pode fazer isso com um editor de texto ou substituindo *`my-node-instance-role`* e executando o seguinte comando:

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
```

c. Aplique a configuração. Esse comando pode demorar alguns minutos para ser concluído.

```
kubectl apply -f aws-auth-cm.yaml
```

4. Observe o status de seus nós e aguarde até que eles atinjam o status `Ready`.

```
kubectl get nodes --watch
```

Insira `Ctrl+C` para retornar a um prompt de shell.

> **Note**
>
> Se você receber qualquer erro de autorização ou de tipo de recurso, consulte the section called “Acesso negado ou não autorizado (`kubectl`)” no tópico de solução de problemas.

Se os nós não conseguirem se juntar ao cluster, consulte the section called “Falha nos nós ao ingressar no cluster” no capítulo Solução de problemas.

5. (Somente nós de GPU) Caso escolha um tipo de instância de GPU e a AMI acelerada otimizada para Amazon EKS, você deverá aplicar o plug-in de dispositivo NVIDIA para Kubernetes como um DaemonSet no cluster. Substitua *`vX.X.X`* pela versão desejada do NVIDIA/k8s-device-plugin antes de executar o seguinte comando.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/deployments/static/nvidia-device-plugin.yml
```

Etapa 3: Ações adicionais

1. (Opcional) Implante uma aplicação de exemplo para testar o cluster e os nós do Linux.

2. (Opcional) Se a política do IAM gerenciada AmazonEKS_CNI_Policy (caso você tenha um cluster IPv4) ou *AmazonEKS_CNI_IPv6_Policy* (que você criou de forma independente, caso tenha um cluster IPv6) estiver anexada ao perfil do IAM dos nós do Amazon EKS, recomendamos atribuí-la a um perfil do IAM que possa ser associado à conta de serviço do Kubernetes aws-node. Para obter mais informações, consulte the section called “Configuração para IRSA”.
3. Recomendamos bloquear o acesso dos pods ao IMDS se as seguintes condições a forem verdadeiras:
   - Você planeja atribuir perfis do IAM a todas as suas contas de serviço do Kubernetes para que os pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum pod no cluster exige acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 por outros motivos, como recuperação da região atual da AWS.

   Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Criar nós Bottlerocket autogerenciados do Bottlerocket

> **Note**
>
> Os grupos de nós gerenciados podem oferecer algumas vantagens para seu caso de uso. Para obter mais informações, consulte the section called “Grupos de nós gerenciados”.

Este tópico descreve como iniciar grupos do Auto Scaling de nós do Bottlerocket que são registrados no cluster do Amazon EKS. O Bottlerocket é um sistema operacional de código aberto baseado em Linux da AWS que você pode usar para executar contêineres em máquinas virtuais ou hosts bare metal. Depois que os nós ingressam no cluster, você pode implantar aplicações do Kubernetes neles. Para obter mais informações sobre o Bottlerocket, consulte Usar uma AMI do Bottlerocket com o Amazon EKS no GitHub e Suporte à AMI personalizada na documentação do eksctl.

Para obter informações sobre atualizações no local, consulte Operador de atualização do Bottlerocket no GitHub.

Important

- Os nós do Amazon EKS são instâncias padrão do Amazon EC2, e você é cobrado por eles com base nos preços normais das instâncias do Amazon EC2. Para obter mais informações, consulte Definição de preço do Amazon EC2.
- Você pode iniciar nós do Bottlerocket em clusters estendidos do Amazon EKS em Outposts da AWS, mas não pode iniciá-los em clusters locais no AWS Outposts. Para obter mais informações, consulte Amazon EKS no AWS Outposts.
- Você pode implantar em instâncias do Amazon EC2 com processadores x86 ou Arm. Porém, não é possível implantar em instâncias que tenham chips Inferentia.
- O Bottlerocket é compatível com o AWS CloudFormation. No entanto, não há um modelo oficial do CloudFormation que possa ser copiado para implantar nós do Bottlerocket para Amazon EKS.
- As imagens do Bottlerocket não vêm com um servidor ou shell SSH. Você pode usar métodos de acesso fora de banda para permitir que o SSH habilite o contêiner do administrador e passar algumas etapas de configuração de bootstrapping com dados de usuário. Para mais informações, consulte essas seções no bottlerocket README.md no GitHub:
  - Exploration (Exploração
  - Admin container (Contêiner Admin
  - Kubernetes settings (Configurações do Kubernetes

Este procedimento exige a versão `eksctl 0.215.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar `eksctl`, consulte Instalação na documentação do `eksctl`. OBSERVAÇÃO: esse procedimento apenas funciona para clusters que foram criados com o `eksctl`.

1. Copie o conteúdo a seguir para o seu dispositivo. Substitua *`my-cluster`* pelo nome do cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome

deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster. Substitua *ng-bottlerocket* por um nome para seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Para implementar em instâncias Arm, substitua *m5.large* por um tipo de instância Arm. Substitua *my-ec2-keypair-name* pelo nome de um par de chaves SSH do Amazon EC2 que você pode usar para se conectar usando SSH em seus nós após o lançamento. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no Console de gerenciamento da AWS. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2. Substitua todos os valores de exemplo restantes por seus próprios valores. Depois de fazer as substituições, execute o comando modificado para criar o arquivo `bottlerocket.yaml`.

Se estiver especificando um tipo de instância Arm do Amazon EC2, analise as considerações em AMIs Amazon Linux Arm otimizadas para Amazon EKS antes da implementação. Para obter instruções sobre como implantar usando uma AMI personalizada, consulte Criar Bottlerocket no GitHub e Suporte à AMI personalizada na documentação do `eksctl`. Para implantar um grupo de nós gerenciados, implante uma AMI personalizada usando um modelo de execução. Para obter mais informações, consulte the section called “Modelos de execução”.

> **Important**
>
> Para implantar um grupo de nós nas sub-redes do AWS Outposts, do AWS Wavelength ou das zonas locais da AWS, não passe as sub-redes do AWS Outposts, do AWS Wavelength ou das zonas locais da AWS ao criar o cluster. É necessário especificar as sub-redes no exemplo a seguir. Para obter mais informações, consulte Create a nodegroup from a config file (Criar um grupo de nós em um arquivo de configuração) e Config file schema (Esquema de arquivo de configuração) na documentação do `eksctl`. Substitua *region-code* pela região da AWS em que seu cluster se encontra.

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
```

```
  version: '1.33'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-bottlerocket
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Bottlerocket
    ami: auto-ssm
    iam:
       attachPolicyARNs:
          - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

2. Implante os nós com o seguinte comando.

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

   Veja um exemplo de saída abaixo.

   Várias linhas são emitidas enquanto os nós são criados. Uma das últimas linha de saída é o exemplo de linha a seguir.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opcional) Crie um volume persistente do Kubernetes em um nó Bottlerocket usando o Plugin de CSI do Amazon EBS. O driver padrão do Amazon EBS depende de ferramentas do sistema de arquivos que não estão incluídas no Bottlerocket. Para obter mais informações sobre como criar uma classe de armazenamento usando o driver, consulte the section called “Amazon EBS”.
4. (Opcional) Por padrão, o `kube-proxy` define o parâmetro do kernel `nf_conntrack_max` com um valor padrão que pode diferir do que o Bottlerocket originalmente define na inicialização. Para manter a configuração padrão do Bottlerocket, edite a configuração do `kube-proxy` com o comando a seguir.

```
kubectl edit -n kube-system daemonset kube-proxy
```

Adicione `--conntrack-max-per-core` e `--conntrack-min` aos argumentos `kube-proxy` que estão no exemplo a seguir. Uma configuração de `0` não implica em nenhuma mudança.

```
      containers:
      - command:
        - kube-proxy
        - --v=2
        - --config=/var/lib/kube-proxy-config/config
        - --conntrack-max-per-core=0
        - --conntrack-min=0
```

5. (Opcional) Implante uma aplicação de exemplo para testar os nós do Bottlerocket.
6. Recomendamos bloquear o acesso dos pods ao IMDS se as seguintes condições a forem verdadeiras:
   - Você planeja atribuir perfis do IAM a todas as suas contas de serviço do Kubernetes para que os pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum pod no cluster exige acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 por outros motivos, como recuperação da região atual da AWS.

   Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Criar nós autogerenciados do Microsoft Windows

Este tópico descreve como iniciar grupos do Auto Scaling de nós Windows que são registrados no seu cluster do Amazon EKS. Depois que os nós ingressam no cluster, você pode implantar aplicações do Kubernetes neles.

> **Important**
>
> - Os nós do Amazon EKS são instâncias padrão do Amazon EC2, e você é cobrado por eles com base nos preços normais das instâncias do Amazon EC2. Para obter mais informações, consulte Definição de preço do Amazon EC2.

- Você pode iniciar nós do Windows em clusters estendidos do Amazon EKS no AWS Outposts, mas não pode iniciá-los em clusters locais no AWS Outposts. Para obter mais informações, consulte Amazon EKS no AWS Outposts.

Habilite o suporte ao Windows para o cluster. Recomendamos revisar considerações importantes antes de iniciar um grupo de nós do Windows. Para obter mais informações, consulte the section called “Habilitar o suporte ao Windows”.

Você pode iniciar nós do autogerenciados do Windows com uma das seguintes opções:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”

`eksctl`

Iniciar nós autogerenciados do Windows usando **`eksctl`**

Este procedimento exige que você instale o `eksctl` e que a versão do `eksctl` seja pelo menos a `0.215.0`. Você pode verificar a versão com o comando a seguir.

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

> **Note**
>
> Este procedimento funciona apenas para clusters que foram criados com o `eksctl`.

1. (Opcional) Se a política do IAM gerenciada AmazonEKS_CNI_Policy (caso você tenha um cluster `IPv4`) ou *`AmazonEKS_CNI_IPv6_Policy`* (que você criou de forma independente, caso tenha um cluster `IPv6`) estiver anexada ao perfil do IAM dos nós do Amazon EKS, recomendamos atribuí-la a um perfil do IAM que possa ser associado à conta de serviço do Kubernetes `aws-node`. Para obter mais informações, consulte the section called “Configuração para IRSA”.
2. Esse procedimento pressupõe que você tem um cluster existente. Se você ainda não tiver um cluster do Amazon EKS e um grupo de nós do Amazon Linux ao qual adicionar um grupo de nós

do Windows, recomendamos seguir the section called “Criar cluster (eksctl)”. Este guia fornece um passo a passo completo sobre como criar um cluster do Amazon EKS com nós do Amazon Linux.

Crie seu grupo de nós com o comando a seguir. Substitua *region-code* pela região da AWS em que seu cluster se encontra. Substitua *my-cluster* pelo nome do cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster. Substitua *ng-windows* por um nome para seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Você pode substituir *2019* por 2022 para usar o Windows Server 2022. Substitua o restante dos valores de exemplo por seus próprios valores.

> **Important**
>
> Para implantar um grupo de nós nas sub-redes AWS Outposts, AWS Wavelength ou AWS Local Zone, não passe as sub-redes AWS Outposts, Wavelength ou Local Zone ao criar o cluster. Crie o grupo de nós com um arquivo de configuração, especificando as sub-redes AWS Outposts, Wavelength ou Local Zone. Para obter mais informações, consulte Criar um grupo de nós em um arquivo de configuração e Esquema de arquivo Config na documentação do `eksctl`.

```
eksctl create nodegroup \
    --region region-code \
    --cluster my-cluster \
    --name ng-windows \
    --node-type t2.large \
    --nodes 3 \
    --nodes-min 1 \
    --nodes-max 4 \
    --managed=false \
    --node-ami-family WindowsServer2019FullContainer
```

> **Note**
>
> - Se os nós não conseguirem juntar-se ao cluster, consulte the section called “Falha nos nós ao ingressar no cluster” no manual Troubleshooting (Solução de problemas).

- Para ver as opções disponíveis para os comandos `eksctl`, insira o comando a seguir.

```
eksctl command -help
```

Veja um exemplo de saída abaixo. Várias linhas são emitidas enquanto os nós são criados. Uma das últimas linha de saída é o exemplo de linha a seguir.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opcional) Implante uma aplicação de exemplo para testar o cluster e os nós do Windows.
4. Recomendamos bloquear o acesso dos pods ao IMDS se as seguintes condições a forem verdadeiras:
   - Você planeja atribuir perfis do IAM a todas as suas contas de serviço do Kubernetes para que os pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum pod no cluster exige acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 por outros motivos, como recuperação da região atual da AWS.

   Para obter mais informações, consulte Restringir o acesso ao perfil de instância atribuído ao nó de processamento.

Console de gerenciamento da AWS

Pré-requisitos

- Um cluster existente do Amazon EKS e um grupo de nós do Linux. Se você não tiver esses recursos, recomendamos que os crie usando um de nossos guias em Conceitos básicos. Estes guias descrevem como criar um cluster do Amazon EKS com nós Linux.
- Uma VPC e um grupo de segurança existentes que atendem aos requisitos de um cluster do Amazon EKS. Para obter mais informações, consulte the section called “Requisitos para sub-rede e VPC” e the section called “Requisitos para grupos de segurança”. Os guias em Conceitos básicos criam uma VPC que atende aos requisitos. Como alternativa, você também pode seguir Criar uma Amazon VPC para seu cluster do Amazon EKS para criar uma manualmente.
- Um cluster existente do Amazon EKS que usa uma VPC e um grupo de segurança que atendam aos requisitos de um cluster do Amazon EKS. Para obter mais informações, consulte the section called “Criar um cluster”. Se você tiver sub-redes na região AWS em que AWS Outposts, AWS

Wavelength ou AWS Local Zones estiverem ativados, essas sub-redes não deverão ter sido transmitidas quando você criou o cluster.

Etapa 1: Inicie nós Windows autogerenciados usando o Console de gerenciamento da AWS

1. Aguarde até que o status do cluster seja exibido como `ACTIVE`. Se você executar os nós antes que o cluster esteja ativo, o registro dos nós no cluster falhará, e você terá de executá-los novamente.
2. Abra o console do AWS CloudFormation.
3. Selecione Criar pilha.
4. Em Specify template, selecione URL do modelo do Amazon S3.
5. Copie a URL a seguir e cole-a no URL do Amazon S3.

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. Escolha Next (Avançar) duas vezes.
7. Na página Quick create stack (Criar pilha rapidamente), insira os parâmetros a seguir de forma adequada:
   - Nome da pilha: escolha um nome para a pilha do AWS CloudFormation. Por exemplo, você pode chamar de `my-cluster-nodes`.
   - ClusterName: insira o nome que você usou ao criar o cluster do Amazon EKS.

     > **Important**
     >
     > Esse nome deve corresponder exatamente ao nome que você usou na Etapa 1: criar seu cluster do Amazon EKS. Caso contrário, seus nós não poderão ser incorporados ao cluster.

   - ClusterControlPlaneSecurityGroup: Escolha o grupo de segurança na saída do AWS CloudFormation que você gerou ao criar sua VPC. As etapas a seguir mostram um método de recuperar o grupo aplicável.
     a. Abra o console do Amazon EKS.
     b. Escolha o nome do cluster.
     c. Escolha a guia Redes.

d. Use o valor de Additional security group (Grupos de segurança adicionais) como referência ao selecionar na lista suspensa ClusterControlPlaneSecurityGroup.

- NodeGroupName: insira um nome para o grupo de nós. Esse nome poderá ser usado superiormente para identificar o grupo de nós do Auto Scaling que for criado para os nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.
- NodeAutoScalingGroupMinSize: digite o número mínimo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeAutoScalingGroupDesiredCapacity: insira o número desejado de nós para o qual dimensionar quando a pilha for criada.
- NodeAutoScalingGroupMaxSize: digite o número máximo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeInstanceType: escolha um tipo de instância para os nós. Para obter mais informações, consulte the section called “Tipos de instância do Amazon EC2”.

> **Note**
>
> Os tipos de instâncias compatíveis com a versão mais recente do plugin Amazon VPC CNI para o Kubernetes estão listados em vpc_ip_resource_limit.go no GitHub. Talvez seja necessário atualizar a versão da CNI para usar os tipos de instâncias compatíveis mais recentes. Para obter mais informações, consulte the section called “CNI da Amazon VPC”.

- NodeImageIdSSMParam: pré-preenchido com o parâmetro do Amazon EC2 Systems Manager do ID da principal AMI do Windows otimizada para Amazon EKS recomendada atualmente. Para usar a versão completa do Windows, substitua *`Core`* por `Full`.
- NodeImageId: (opcional) se estiver usando sua própria AMI personalizada (em vez da AMI otimizada do Amazon EKS), insira um ID de AMI de nó para a região da AWS. Se você especificar um valor para este campo, ele substituirá todos os valores no campo NodeImageIdSSMParam.
- NodeVolumeSize: especifique um tamanho de volume raiz para os nós, em GiB.
- KeyName: insira o nome de um par de chaves SSH do Amazon EC2; que você pode usar para conectar-se usando SSH em seus nós, depois de iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no Console de gerenciamento da AWS. Para

obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2.

> **Note**
>
> Se você não fornecer um par de chaves aqui, a pilha do AWS CloudFormation não será criada.

- BootstrapArguments: especifique argumentos opcionais para passar para o script de bootstrap do nó, como argumentos `kubelet` adicionais usando `-KubeletExtraArgs`.
- DisableIMDSv1: por padrão, cada nó oferece suporte ao serviço de metadados de instância versão 1 (IMDSv1) e IMDSv2. Você pode desabilitar IMDSv1. Para evitar que nós e pods futuros no grupo de nós usem o MDSv1, defina DisableIMDSv1 como true. Para obter mais informações, consulte Configuring the instance metadata service (Configurar o serviço de metadados de instância).
- VpcId: selecione o ID para a VPC que você criou em .
- NodeSecurityGroups: selecione o grupo de segurança que foi criado para o grupo de nós do Linux quando você criou a VPC. Se os nós do Linux tiverem mais de um grupo de segurança anexado a eles, especifique todos eles. Isso serve, por exemplo, quando o grupo de nós do Linux foi criado com `eksctl`.
- Subnets (Sub-redes): escolha as sub-redes que você criou. Se você criou sua VPC usando as etapas em Criar uma Amazon VPC para seu cluster do Amazon EKS, especifique apenas as sub-redes privadas dentro da VPC para que seus nós sejam iniciados.

> **Important**
>
> - Se alguma das sub-redes for pública, ela deverá ter a atribuição automática de atribuição de endereço IP público habilitada. Se a configuração não estiver habilitada para a sub-rede pública, os nós implantados nessa sub-rede pública não receberão um endereço IP público e não poderão se comunicar com o cluster nem com outros produtos da AWS. Se a sub-rede tiver sido implantada antes de 26 de março de 2020 usando um dos modelos de VPC do Amazon EKS AWS CloudFormation ou usando `eksctl`, a atribuição automática de endereço IP público estará desativada para sub-redes públicas. Para obter informações sobre como habilitar a atribuição de endereço IP público para uma sub-rede, consulte Modificar o atributo de endereçamento IPv4

público para a sub-rede. Se o nó for implantado em uma sub-rede privada, ele poderá se comunicar com o cluster e com outros produtos da AWS por um gateway NAT.

- Se as sub-redes não tiverem acesso à Internet, certifique-se de que você está ciente das considerações e etapas adicionais em Implantar clusters privados com acesso limitado à Internet.
- Se você selecionar as sub-redes AWS Outposts, Wavelength ou Local Zone, as sub-redes não deverão ter sido passadas quando você criou o cluster.

8. Confirme que a pilha pode criar recursos do IAM e escolha Create stack (Criar pilha).
9. Quando a criação da pilha for concluída, selecione-a no console e escolha Outputs (Saídas).
10. Registre o valor de NodeInstanceRole para o grupo de nós criado. Você precisará dele ao configurar os nós de Windows do Amazon EKS.

Etapa 2: habilite os nós para participar do cluster

1. Verifique se você já tem um `aws-auth` `ConfigMap`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se você receber um `aws-auth` `ConfigMap`, atualize-o conforme necessário.
   a. Abra o `ConfigMap` para edição.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Adicione novas `mapRoles` entradas conforme necessário. Defina os valores `rolearn` para os valores NodeInstanceRole que você registrou no procedimento anterior.

```
[...]
data:
  mapRoles: |
- rolearn: <ARN of linux instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: <ARN of windows instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
```

```
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
[...]
```

c. Salve o arquivo e saia do seu editor de texto.

3. Se você recebeu um erro informando "`Error from server (NotFound): configmaps "aws-auth" not found`, aplique o estoque`ConfigMap`.

a. Faça download do mapa de configuração.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-auth-cm-windows.yaml
```

b. No arquivo `aws-auth-cm-windows.yaml`, configure os valores `rolearn` para os valores NodeInstanceRole aplicáveis que você registrou nos procedimentos anteriores. Você pode fazer isso com um editor de texto ou substituindo os valores de exemplo e executando o seguinte comando:

```
sed -i.bak -e 's|<ARN of linux instance role (not instance profile)>|my-node-linux-instance-role|' \
    -e 's|<ARN of windows instance role (not instance profile)>|my-node-windows-instance-role|' aws-auth-cm-windows.yaml
```

> **Important**
>
> - Não modifique outras linhas do arquivo.
> - Não use o mesmo perfil do IAM para nós do Windows e Linux.

c. Aplique a configuração. Esse comando pode levar alguns minutos para ser concluído.

```
kubectl apply -f aws-auth-cm-windows.yaml
```

4. Observe o status de seus nós e aguarde até que eles atinjam o status `Ready`.

```
kubectl get nodes --watch
```

Insira `Ctrl+C` para retornar a um prompt de shell.

> **Note**
>
> Se você receber qualquer erro de autorização ou de tipo de recurso, consulte the section called “Acesso negado ou não autorizado (`kubectl`)” no tópico de solução de problemas.

Se os nós não conseguirem se juntar ao cluster, consulte the section called “Falha nos nós ao ingressar no cluster” no capítulo Solução de problemas.

Etapa 3: Ações adicionais

1. (Opcional) Implante uma aplicação de exemplo para testar o cluster e os nós do Windows.
2. (Opcional) Se a política do IAM gerenciada AmazonEKS_CNI_Policy (caso você tenha um cluster `IPv4`) ou *`AmazonEKS_CNI_IPv6_Policy`* (que você criou de forma independente, caso tenha um cluster `IPv6`) estiver anexada ao perfil do IAM dos nós do Amazon EKS, recomendamos atribuí-la a um perfil do IAM que possa ser associado à conta de serviço do Kubernetes `aws-node`. Para obter mais informações, consulte the section called “Configuração para IRSA”.
3. Recomendamos bloquear o acesso dos pods ao IMDS se as seguintes condições a forem verdadeiras:
   - Você planeja atribuir perfis do IAM a todas as suas contas de serviço do Kubernetes para que os pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum pod no cluster exige acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 por outros motivos, como recuperação da região atual da AWS.

   Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Criar nós autogerenciados do Ubuntu Linux

> **Note**
>
> Os grupos de nós gerenciados podem oferecer algumas vantagens para seu caso de uso. Para obter mais informações, consulte the section called “Grupos de nós gerenciados”.

Este tópico descreve como iniciar grupos do Auto Scaling de nós do Ubuntu no Amazon Elastic Kubernetes Service (EKS) ou Ubuntu Pro no Amazon Elastic Kubernetes Service (EKS) que se registram no cluster Amazon EKS. O Ubuntu e o Ubuntu Pro para EKS são baseados no Ubuntu Minimal LTS oficial, incluem o kernel personalizado da AWS, que é desenvolvido em conjunto com a AWS, e foram criados especificamente para o EKS. O Ubuntu Pro adiciona cobertura de segurança adicional ao oferecer suporte a períodos de suporte estendidos do EKS, livepatch do kernel, conformidade com FIPS e a capacidade de executar contêineres Pro ilimitados.

Depois que os nós ingressam no cluster, você pode implantar aplicações em contêiner neles. Para obter mais informações, via documentação do Ubuntu em AWS e o suporte a AMIs personalizadas na documentação `eksctl`.

> **Important**
>
> - Os nós do Amazon EKS são instâncias padrão do Amazon EC2, e você é cobrado por eles com base nos preços normais das instâncias do Amazon EC2. Para obter mais informações, consulte Definição de preço do Amazon EC2.
> - Você pode iniciar nós do Ubuntu em clusters estendidos do Amazon EKS no AWS Outposts, mas não pode iniciá-los em clusters locais no AWS Outposts. Para obter mais informações, consulte Amazon EKS no AWS Outposts.
> - Você pode implantar em instâncias do Amazon EC2 com processadores `x86` ou Arm. No entanto, as instâncias que têm chips Inferentia talvez precisem instalar o SDK do Neuron primeiro.

Este procedimento exige a versão `eksctl 0.215.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar `eksctl`, consulte Instalação na documentação do `eksctl`. OBSERVAÇÃO: esse procedimento apenas funciona para clusters que foram criados com o `eksctl`.

1. Copie o conteúdo a seguir para o seu dispositivo. Substitua o `my-cluster` pelo nome do cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfabético e não pode ter mais de 100 caracteres. Substitua

`ng-ubuntu` por um nome para o seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Para implantar em instâncias do Arm, substitua `m5.large` por um tipo de instância Arm. Substitua `my-ec2-keypair-name` pelo nome de um par de chaves SSH do Amazon EC2; que você pode usar para conectar-se usando SSH em seus nós, depois de iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no Console de gerenciamento da AWS. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2. Substitua todos os valores de exemplo restantes por seus próprios valores. Depois de fazer as substituições, execute o comando modificado para criar o arquivo `ubuntu.yaml`.

> **Important**
>
> Para implantar um grupo de nós nas sub-redes do AWS Outposts, do AWS Wavelength ou das zonas locais da AWS, não passe as sub-redes do AWS Outposts, do AWS Wavelength ou das zonas locais da AWS ao criar o cluster. É necessário especificar as sub-redes no exemplo a seguir. Para obter mais informações, consulte Create a nodegroup from a config file (Criar um grupo de nós em um arquivo de configuração) e Config file schema (Esquema de arquivo de configuração) na documentação do `eksctl`. Substitua *`region-code`* pela região da AWS em que seu cluster se encontra.

```
cat >ubuntu.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.33'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-ubuntu
    instanceType: m5.large
    desiredCapacity: 3
```

```
    amiFamily: Ubuntu2204
    iam:
       attachPolicyARNs:
          - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

Para criar um grupo de nós do Ubuntu Pro, basta alterar o valor de `amiFamily` para `UbuntuPro2204`.

2. Implante os nós com o seguinte comando.

```
eksctl create nodegroup --config-file=ubuntu.yaml
```

Veja um exemplo de saída abaixo.

Várias linhas são emitidas enquanto os nós são criados. Uma das últimas linha de saída é o exemplo de linha a seguir.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opcional) Implante uma aplicação de exemplo para testar os nós do Ubuntu.
4. Recomendamos bloquear o acesso dos pods ao IMDS se as seguintes condições a forem verdadeiras:
   - Você planeja atribuir perfis do IAM a todas as suas contas de serviço do Kubernetes para que os pods tenham apenas as permissões mínimas de que precisam.
   - Nenhum pod no cluster exige acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 por outros motivos, como recuperação da região atual da AWS.

   Para obter mais informações, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.

Atualizar nós autogerenciados para seu cluster

Quando uma nova AMI otimizada para Amazon EKS for lançada, considere substituir os nós no grupo de nós autogerenciados pela nova AMI. Da mesma forma, se você tiver atualizado a versão do Kubernetes para o cluster do Amazon EKS, atualize os nós para usá-los com a mesma versão do Kubernetes.

> **Important**
>
> Este tópico aborda as atualizações do nó para nós autogerenciados. Se você estiver usando grupos de nós gerenciados, consulte the section called “Atualizar”.

Há duas maneiras básicas de atualizar grupos de nós autogerenciados em seus clusters para usar uma nova AMI:

Migrar aplicações para um novo grupo de nós

Crie um novo grupo de nós e migre os pods para esse grupo. A migração para um novo grupo de nós é mais simples do que simplesmente atualizar a ID da AMI em uma pilha existente do AWS CloudFormation. Isso ocorre porque o processo de migração marca o grupo de nós antigo como `NoSchedule` e drena os nós depois que uma nova pilha está pronta para aceitar a workload do pod existente.

Atualizar uma pilha de nós do AWS CloudFormation

Atualize a pilha do AWS CloudFormation para um grupo de nós existente para usar a nova AMI. Esse método não é compatível com grupos de nós que foram criados com o `eksctl`.

Migrar aplicações para um novo grupo de nós

Este tópico descreve como criar um novo grupo de nós, migrar facilmente as aplicações existentes para o novo grupo e remover o grupo de nós antigo do cluster. Você pode migrar para um novo grupo de nós usando o `eksctl` ou o Console de gerenciamento da AWS.

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS e AWS CLI”

`eksctl`

Migre suas aplicações para um novo grupo de nós com **`eksctl`**

Para obter mais informações sobre o uso do eksctl para migração, consulte Unmanaged nodegroups (Grupos de nós não gerenciados) na documentação do `eksctl`.

Este procedimento exige a versão `eksctl` `0.215.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

> **Note**
>
> Este procedimento funciona apenas para clusters e grupos de nós que foram criados com o `eksctl`.

1. Recupere o nome dos grupos de nós existentes, substituindo *`my-cluster`* pelo nome do cluster.

   ```
   eksctl get nodegroups --cluster=my-cluster
   ```

   Veja abaixo um exemplo de saída.

   ```
   CLUSTER      NODEGROUP          CREATED               MIN SIZE      MAX SIZE
    DESIRED CAPACITY     INSTANCE TYPE     IMAGE ID
   default      standard-nodes   2019-05-01T22:26:58Z  1             4             3
                  t3.medium         ami-05a71d034119ffc12
   ```

2. Inicie um novo grupo de nós com `eksctl` usando o comando a seguir. No comando, substitua cada *`valor de exemplo`* por seus próprios valores. O número da versão não pode ser superior à versão do Kubernetes do ambiente de gerenciamento. Além disso, não pode ultrapassar duas versões secundárias anteriores à versão do Kubernetes do ambiente de gerenciamento. Recomendamos usar a mesma versão do ambiente de gerenciamento.

   Recomendamos bloquear o acesso dos pods ao IMDS se as seguintes condições a forem verdadeiras:

- Você planeja atribuir perfis do IAM a todas as suas contas de serviço do Kubernetes para que os pods tenham apenas as permissões mínimas de que precisam.
- Nenhum pod no cluster exige acesso ao serviço de metadados de instância (IMDS) do Amazon EC2 por outros motivos, como recuperação da região atual da AWS.

  Para obter mais informações, consulte Restringir o acesso ao perfil de instância atribuído ao nó de processamento.

  Para bloquear o acesso do pod ao IMDS, adicione a opção `--disable-pod-imds` ao comando a seguir.

  > **Note**
  >
  > Para obter mais sinalizadores disponíveis e suas descrições, consulte https://eksctl.io/.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --version 1.33 \
  --name standard-nodes-new \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --managed=false
```

3. Quando o comando anterior for concluído, verifique se todos os nós atingiram o estado `Ready` com o seguinte comando:

```
kubectl get nodes
```

4. Exclua cada grupo de nós original usando o comando a seguir. No comando, substitua cada *`valor de exemplo`* pelos nomes do seu cluster e grupo de nós:

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes-old
```

Console de gerenciamento da AWS e AWS CLI

Migre suas aplicações para um novo grupo de nós com a CLI Console de gerenciamento da AWS e AWS

1. Inicie um novo grupo de nós seguindo as etapas descritas em Criar nós autogerenciados do Amazon Linux.
2. Quando a criação da pilha for concluída, selecione-a no console e escolha Outputs (Saídas).
3. Registre o valor de NodeInstanceRole para o grupo de nós criado. Você precisará dele para adicionar os novos nós do Amazon EKS ao cluster.

> **Note**
>
> Se você anexou qualquer outra política do IAM à função do IAM do antigo grupo de nós, anexe essas mesmas políticas à função do IAM do novo grupo de nós para manter essa funcionalidade no novo grupo. Isso se aplica a você se você adicionou permissões para o Cluster Autoscaler do Kubernetes, por exemplo.

4. Atualize os grupos de segurança para ambos os grupos de nós, para que eles possam se comunicar um com o outro. Para obter mais informações, consulte the section called “Requisitos para grupos de segurança”.
   a. Registre os IDs do grupo de segurança de ambos os grupos de nós. Isso é mostrado como o valor NodeSecurityGroup nas saídas da pilha do AWS CloudFormation.

      Você pode usar os seguintes comandos da CLI AWS para obter os IDs de grupo de segurança dos nomes de pilha. Nesses comandos, `oldNodes` é o nome da pilha do AWS CloudFormation para a pilha de nós mais antiga e `newNodes` é o nome da pilha para a qual você está migrando. Substitua cada *`valor de exemplo`* por seus próprios valores.

      ```
      oldNodes="old_node_CFN_stack_name"
      newNodes="new_node_CFN_stack_name"

      oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
      --query 'StackResources[?
      ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
      --output text)
      newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
      --query 'StackResources[?
      ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
      --output text)
      ```

   b. Adicione regras de entrada a cada grupo de segurança do nó para que eles aceitem o tráfego uns dos outros.

Os seguintes comandos da CLI AWS adicionam regras de entrada a cada grupo de segurança que permitem todo o tráfego em todos os protocolos do outro grupo de segurança. Essa configuração permite que os pods em cada grupo de nós se comuniquem entre si enquanto estiver migrando a workload para o novo grupo.

```
aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 authorize-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

5. Edite o configmap do `aws-auth` para mapear a nova função da instância do nó no RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Adicione uma nova entrada `mapRoles` para o novo grupo do nós.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile)
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

Substitua o snippet *ARN do perfil da instância (não do perfil da instância)* pelo valor NodeInstanceRole que você registrou em uma etapa anterior. Então, salve e feche o arquivo para aplicar o configmap atualizado.

6. Observe o status dos nós e aguarde até que os novos nós ingressem no cluster e atinjam o status `Ready`.

```
kubectl get nodes --watch
```

7. (Opcional) Se você estiver usando o Cluster Autoscaler do Kubernetes, reduza a implantação para zero (0) réplica para evitar ações de escalabilidade conflitantes.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. Use o comando a seguir para aplicar taint em cada um dos nós que deseja remover com `NoSchedule`. Isso serve para que novos pods não sejam agendados ou reagendados nos nós que você está substituindo. Para obter mais informações, consulte Taints and Tolerations na documentação do Kubernetes.

```
kubectl taint nodes node_name key=value:NoSchedule
```

   Se você estiver atualizando os nós para uma nova versão do Kubernetes, será possível identificar e aplicar taints em todos os nós de determinada versão do Kubernetes (neste caso, `1.31`) com o trecho de código a seguir. O número da versão não pode ser superior à versão do Kubernetes do ambiente de gerenciamento. Também não pode ultrapassar duas versões secundárias anteriores à versão do Kubernetes do ambiente de gerenciamento. Recomendamos usar a mesma versão do ambiente de gerenciamento.

```
K8S_VERSION=1.31
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==
\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Tainting $node"
    kubectl taint nodes $node key=value:NoSchedule
done
```

9. Determine o provedor DNS do cluster.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

   Veja abaixo um exemplo de saída. O cluster está usando o CoreDNS para a resolução de DNS, mas o cluster pode retornar `kube-dns`.

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

10. Se a implantação atual estiver executando menos de duas réplicas, expanda-a para duas réplicas. Substitua *coredns* por `kubedns` se a saída do comando anterior tiver retornado isso.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11. Drene cada um dos nós que você deseja remover do cluster com o seguinte comando:

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

Se você estiver atualizando os nós para uma nova versão do Kubernetes, identifique e drene todos os nós de uma determinada versão do Kubernetes (neste caso, *1.31*) com o trecho de código a seguir.

```
K8S_VERSION=1.31
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==
\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-local-data
done
```

12. Depois que os antigos nós forem esvaziados, revogue as regras de entrada do grupo de segurança que você autorizou anteriormente. Em seguida, exclua a pilha do AWS CloudFormation para encerrar as instâncias.

> **Note**
>
> Se você anexou alguma política IAM adicional à perfil IAM do grupo de nós antigo, como a adição de permissões para o Kubernetes Cluster Autoscaler, desanexe essas políticas adicionais do perfil antes de excluir a pilha do AWS CloudFormation.

a. Revogue as regras de entrada criadas anteriormente para os grupos de segurança do nó. Nesses comandos, `oldNodes` é o nome da pilha do AWS CloudFormation para sua pilha de nós mais antiga e `newNodes` é o nome da pilha para a qual você está migrando.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"
```

```
oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 revoke-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

b. Abra o console do AWS CloudFormation.

c. Selecione a pilha antiga do nó.

d. Escolha Excluir.

e. Na caixa de diálogo de confirmação Delete stack (Excluir pilha), escolha Delete stack (Excluir pilha).

13 Edite o configmap `aws-auth` para remover a antiga função da instância do nó do RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Exclua a entrada `mapRoles` do antigo grupo nós.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-
W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-15-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
```

Salve e feche o arquivo para aplicar o configmap atualizado.

14. (Opcional) Se você estiver usando o Autoscaler do cluster do Kubernetes, reduza a implantação para uma réplica.

> **Note**
>
> Você também deve etiquetar o novo grupo do Auto Scaling corretamente (por exemplo, `k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/my-cluster`) e atualizar o comando da implantação do Cluster Autoscaler de modo a apontar para o grupo do Auto Scaling recém-marcado. Para obter mais informações, consulte Autoscaler do cluster na AWS.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15. (Opcional) Verifique se você está usando a versão mais recente do plugin CNI da Amazon VPC para Kubernetes. Talvez seja necessário atualizar a versão da CNI para usar os tipos de instâncias compatíveis mais recentes. Para obter mais informações, consulte the section called "CNI da Amazon VPC".

16. Se o seu cluster estiver usando o `kube-dns` para resolução de DNS (consulte [migrate-determine-dns-step]), dimensione a implantação do `kube-dns` para uma réplica.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

Atualizar uma pilha de nós do AWS CloudFormation

Este tópico descreve como você pode atualizar uma pilha de nós autogerenciados do AWS CloudFormation existente com uma nova AMI. Você pode usar esse procedimento para atualizar os nós para uma nova versão do Kubernetes após uma atualização do cluster. Senão, é possível atualizar para a AMI otimizada do Amazon EKS mais recente para uma versão existente do Kubernetes.

> **Important**
>
> Este tópico aborda as atualizações do nó para nós autogerenciados. Para obter informações sobre como usar Simplificar o ciclo de vida dos nós com grupos de nós gerenciados, consulte the section called “Atualizar”.

O modelo padrão mais recente do Amazon EKS node AWS CloudFormation está configurado para iniciar uma instância com a nova AMI em seu cluster antes de remover uma antiga, uma de cada vez. Essa configuração garante que você esteja com a contagem desejada do grupo do Auto Scaling de instâncias ativas no cluster durante a implantação da atualização.

> **Note**
>
> Esse método não é compatível com grupos de nós que foram criados com o `eksctl`. Se você criou o cluster ou o grupo de nós com o `eksctl`, consulte the section called “Migração”.

1. Determine o provedor DNS de seu cluster.

   ```
   kubectl get deployments -l k8s-app=kube-dns -n kube-system
   ```

   Veja abaixo um exemplo de saída. O cluster está usando o CoreDNS para a resolução de DNS, mas, em vez disso, o cluster pode retornar `kube-dns`. Sua saída pode parecer diferente dependendo da versão do `kubectl` que você está usando.

   ```
   NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
   coredns   1         1         1            1           31m
   ```

2. Se a implantação atual estiver executando menos de duas réplicas, expanda-a para duas réplicas. Substitua *`coredns`* por `kube-dns` se a saída do comando anterior tiver retornado isso.

   ```
   kubectl scale deployments/coredns --replicas=2 -n kube-system
   ```

3. (Opcional) Se você estiver usando o Cluster Autoscaler do Kubernetes, reduza a implantação para zero (0) réplica para evitar ações de escalabilidade conflitantes.

   ```
   kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
   ```

4. Determine o tipo de instância e a contagem de instâncias desejada do grupo de nós atual. Esses valores são inseridos posteriormente quando você atualiza o modelo do AWS CloudFormation para o grupo.
   a. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
   b. No painel de navegação à esquerda, escolha Launch Configurations (Configurações de execução) e anote o tipo de instância da configuração de execução do nó existente.
   c. No painel de navegação à esquerda, escolha Grupos do Auto Scaling e observe a contagem de instâncias Desejadas para o grupo do Auto Scaling do nó existente.
5. Abra o console do AWS CloudFormation.
6. Selecione a pilha do grupo de nós e escolha Atualizar.
7. Selecione Substituir modelo atual e selecione URL do Amazon S3.
8. Para o URL do Amazon S3, cole o seguinte URL na área de texto para garantir que você esteja usando a versão mais recente do modelo do node AWS CloudFormation. Depois, escolha Próximo:

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

9. Na página Especificar detalhes da pilha, preencha os parâmetros a seguir e escolha Próximo:
   - NodeAutoScalingGroupDesiredCapacity: insira a contagem de instâncias desejada que você registrou em uma etapa anterior. Ou insira o novo número desejado de nós para o qual escalar quando a pilha for criada.
   - NodeAutoScalingGroupMaxSize: digite o número máximo de nós para o qual o grupo Auto Scaling de nós pode ser aumentado. Esse valor deve ser pelo menos um nó a mais do que a capacidade desejada. Assim, você pode realizar uma atualização contínua dos nós sem reduzir a contagem de nós durante a atualização.
   - NodeInstanceType: escolha o tipo de instância registrada em uma etapa anterior. Se preferir, escolha um tipo de instância diferente para os nós. Antes de escolher um tipo de instância diferente, consulte Escolha um tipo ideal de instância de nó do Amazon EC2. Cada tipo de instância do Amazon EC2 suporta um número máximo de interfaces elásticas de rede (interfaces de rede) e cada interface de rede oferece suporte a um número máximo de endereços IP. Como cada nó de processamento e pod recebe seu próprio endereço IP, é importante escolher um tipo de instância que ofereça suporte ao número máximo de pods que você deseja executar em cada nó do Amazon EC2. Para obter uma lista do número de interfaces de rede e endereços IP compatíveis com tipos de instância, consulte Endereços

IP por interface de rede por tipo de instância. Por exemplo, o tipo de instância `m5.large` é compatível com um máximo de trinta endereços IP para o nó de processamento e pods.

> **Note**
>
> Os tipos de instâncias compatíveis com a versão mais recente do plugin CNI da Amazon VPC para Kubernetes estão exibidos em vpc_ip_resource_limit.go no GitHub. Talvez seja necessário atualizar o plug-in CNI da Amazon VPC para Kubernetes para usar os tipos de instâncias compatíveis mais recentes. Para obter mais informações, consulte the section called “CNI da Amazon VPC”.

> **Important**
>
> Alguns tipos de instância podem não estar disponíveis em todas as regiões da AWS.

- NodeImageIdSSMParam: o parâmetro do Amazon EC2 Systems Manager do ID da AMI para o qual você deseja atualizar. O valor a seguir usa a AMI otimizada para o Amazon EKS mais recente para a versão `1.33` do Kubernetes.

```
/aws/service/eks/optimized-ami/1.33/amazon-linux-2/recommended/image_id
```

  É possível substituir *`1.33`* por uma versão da plataforma que seja a mesma. Ou deve ser até uma versão anterior à versão do Kubernetes em execução no ambiente de gerenciamento. Recomendamos que você mantenha seus nós na mesma versão do ambiente de gerenciamento. Você também pode substituir *`o amazon-linux-2`* por um tipo diferente de AMI. Para obter mais informações, consulte the section called “Obter os IDs mais recentes”.

  > **Note**
  >
  > O uso do parâmetro do Amazon EC2 Systems Manager permite atualizar os nós no futuro sem precisar pesquisar e especificar um ID de AMI. Se a pilha do AWS CloudFormation estiver usando esse valor, qualquer atualização da pilha sempre executará a AMI otimizada para Amazon EKS recomendada para a versão específica do Kubernetes. Isso se aplica mesmo que você não altere nenhum valor no template.

- NodeImageId: para usar sua própria AMI personalizada, insira o ID da AMI a ser usada.

> **Important**
>
> Esse valor substitui qualquer valor especificado para NodeImageIdSSMParam. Se você quiser usar o valor de NodeImageIdSSMParam, verifique se o valor de NodeImageId está em branco.

- DisableIMDSv1: por padrão, cada nó oferece suporte ao serviço de metadados de instância versão 1 (IMDSv1) e IMDSv2. No entanto, você pode desabilitar o IMDSv1. Selecione true se você não quiser que nós ou pods agendados no grupo de nós usem IMDSv1. Para obter mais informações, consulte Configurar o serviço de metadados da instância. Você implementou perfis do IAM para contas de serviço e atribuiu as permissões necessárias diretamente a todos os pods que exigem acesso aos serviços da AWS. Dessa forma, nenhum pod no cluster precisa acessar o IMDS por outros motivos, como recuperar a região atual da AWS. Então, você também pode desabilitar o acesso ao IMDSv2 para pods que não usam redes de host. Para obter mais informações, consulte Restringir o acesso ao perfil de instância atribuído ao nó de processamento.

10. (Opcional) Na página Opções, marque os recursos da sua pilha. Escolha Próximo.
11. Na página Review (Revisão), analise suas informações, confirme se a pilha pode criar recursos do IAM e selecione Update stack (Atualizar pilha).

> **Note**
>
> A atualização de cada nó no cluster leva vários minutos. Aguarde até que a atualização de todos os nós seja concluída antes de executar as próximas etapas.

12. Se o seu provedor de DNS do cluster for `kube-dns`, reduza a escala horizontalmente da implantação de `kube-dns` para uma réplica.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13. (Opcional) Se você estiver usando o Cluster Autoscaler do Kubernetes, reduza a implantação para a quantidade de réplicas desejada.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

14. (Opcional) Verifique se você está usando a versão mais recente do plugin CNI da Amazon VPC para Kubernetes. Talvez seja necessário atualizar o plug-in CNI da Amazon VPC para Kubernetes

para usar os tipos de instâncias compatíveis mais recentes. Para obter mais informações, consulte the section called “CNI da Amazon VPC”.

Simplifique o gerenciamento da computação com o AWS Fargate

Este tópico discute o uso do Amazon EKS para executar pods do Kubernetes no AWS Fargate. O Fargate é uma tecnologia que fornece capacidade computacional sob demanda do tamanho certo para contêineres. Com o Fargate, você não tem que provisionar, configurar ou escalar grupos de máquinas virtuais você mesmo para executar contêineres. Você também não precisa escolher tipos de servidor, decidir quando escalar grupos de nós ou otimizar o empacotamento dos clusters.

Você pode controlar quais pods serão iniciados no Fargate e como serão executados com os perfis do Fargate. Os perfis do Fargate são definidos como parte do seu cluster do Amazon EKS. O Amazon EKS integra o Kubernetes ao Fargate com controladores criados pela AWS usando o modelo upstream e extensível fornecido pelo Kubernetes. Esses controladores são executados como parte do ambiente de gerenciamento do Kubernetes gerenciado pelo Amazon EKS e são responsáveis por agendar pods nativos do Kubernetes no Fargate. Os controladores do Fargate incluem um novo agendador que é executado com o agendador do Kubernetes, além de vários controladores de admissão de mutação e de validação. Quando você inicia um pod que atende aos critérios de execução no Fargate, os controladores do Fargate em execução no cluster reconhecem, atualizam e agendam o pod no Fargate.

Este tópico descreve os diferentes componentes dos pods que são executados no Fargate, e faz considerações especiais quanto ao uso do Fargate com o Amazon EKS.

Considerações sobre o AWS Fargate

Veja a seguir algumas considerações sobre o uso do Fargate no Amazon EKS.

- Cada pod executado no Fargate tem seu próprio limite de computação. Um pod não compartilha o kernel subjacente, os recursos de CPU, os recursos de memória nem a interface de rede elástica com outro pod.
- Os balanceadores de carga da rede e os balanceadores de carga da aplicação (ALBs) só podem ser usados com o Fargate com destinos IP. Para obter mais informações, consulte the section called “Criar um balanceador de carga da rede” e the section called “Balanceador de carga da aplicação”.

- Os serviços expostos ao Fargate são executados somente no modo IP do tipo de destino e não no modo IP do nó. A maneira recomendada de verificar a conectividade de um serviço em execução em um nó gerenciado e um serviço em execução no Fargate é se conectar pelo nome de serviço.
- Os pods devem corresponder a um perfil do Fargate no momento em que são agendados para execução no Fargate. Pods que não correspondam a um perfil do Fargate podem ficar retidos como `Pending`. Se houver um perfil do Fargate correspondente, será possível excluir os pods pendentes criados para serem reagendados no Fargate.
- Não há suporte a Deamonsets no Fargate. Se a aplicação exigir um daemon, reconfigure-o para ser executado como um contêiner associado nos seus pods.
- Não há suporte a contêineres privilegiados no Fargate.
- Os pods que são executados no Fargate não podem especificar `HostPort` ou `HostNetwork` no manifesto do pod.
- O limite flexível padrão do `nofile` e do `nproc` é 1.024, e o limite rígido é 65.535 para pods do Fargate.
- No momento, GPUs não estão disponíveis no Fargate.
- Só há suporte a pods em execução no Fargate em sub-redes privadas (com acesso de gateway NAT aos serviços da AWS, mas não uma rota direta para um gateway da Internet), portanto, a VPC do cluster deve ter sub-redes privadas disponíveis. Para clusters sem acesso de saída à Internet, consulte the section called “Clusters privados”.
- Você pode usar a opção Ajustar recursos de pods com o Vertical Pod Autoscaler para definir o tamanho correto inicial da CPU e da memória para os pods do Fargate e, em seguida, usar a opção Escalar implantações de pods com o Horizontal Pod Autoscaler para escalar esses pods. Se quiser que o Vertical Pod Autoscaler reimplante automaticamente os pods no Fargate com combinações maiores de CPU e memória, defina o modo do Vertical Pod Autoscaler como `Auto` ou `Recreate` para garantir a funcionalidade correta. Para obter mais informações, consulte a documentação do Vertical Pod Autoscaler no GitHub.
- A resolução DNS e os nomes de host DNS devem estar habilitados para sua VPC. Para obter mais informações, consulte Visualizar e atualizar o suporte DNS para VPC.
- O Fargate do Amazon EKS adiciona defesa completa para aplicações do Kubernetes isolando cada pod em uma máquina virtual (VM). Essa barreira de VM impede o acesso aos recursos baseados em host usados por outros pods, no caso de um escape de contêiner, um método comum de atacar aplicações em contêiner e obter acesso a recursos fora do contêiner.

  O uso do Amazon EKS não altera suas responsabilidades no modelo de responsabilidade compartilhada. Você deve considerar cuidadosamente a configuração de controles de governança

e segurança do cluster. A maneira mais segura de isolar uma aplicação é sempre executá-la em um cluster separado.

- Os perfis do Fargate são compatíveis com a especificação de sub-redes de blocos CIDR secundários da VPC. Talvez você queira especificar um bloco CIDR secundário. Isso porque existe um número limitado de endereços IP disponíveis em uma sub-rede. Como resultado, existe também um número limitado de pods que podem ser criados no cluster. Usando diferentes sub-redes para os pods, você pode aumentar o número de endereços IP disponíveis. Para obter mais informações, consulte Adicionar blocos CIDR IPv4 a uma VPC.
- O serviço de metadados de instância (IMDS) do Amazon EC2 não está disponível para pods implantados em nós do Fargate. Se você tiver pods implantados no Fargate que precisem de credenciais do IAM, atribua-as aos pods usando perfis do IAM para contas de serviço. Se os pods precisarem de acesso a outras informações disponíveis por meio do IMDS, você deve usar codificação rígida nessas informações na especificação do pod. Isso inclui a zona de disponibilidade ou a região da AWS em que um pod é implantado.
- Não é possível implantar pods do Fargate no AWS Outposts, no AWS Wavelength ou nas zonas locais da AWS.
- O Amazon EKS deve aplicar patches nos pods periodicamente para mantê-los seguros. Tentamos realizar as atualizações de uma forma que reduza o impacto, mas há situações em que os pods precisarão ser excluídos quando não forem removidos com êxito. Há algumas medidas que você pode tomar para minimizar a interrupção. Para obter mais informações, consulte the section called “Eventos de aplicação de patches de sistema operacional”.
- O plugin CNI do Amazon VPC para Amazon EKS é instalado em nós do Fargate. Não é possível usar plug-ins CNI alternativos para clusters do Amazon EKS com nós do Fargate.
- Um pod em execução no Fargate automaticamente monta um sistema de arquivos do Amazon EFS sem precisar das etapas de instalação manual do driver. Você não pode usar o provisionamento dinâmico de volume persistente com nós do Fargate, mas pode usar o provisionamento estático.
- O Amazon EKS não oferece suporte ao Fargate Spot.
- Você não pode montar volumes do Amazon EBS em pods do Fargate.
- Você pode executar o controlador CSI do Amazon EBS em nós do Fargate, mas o DaemonSet do nó CSI do Amazon EBS só pode ser executado em instâncias do Amazon EC2.
- Depois que um trabalho do Kubernetes é marcado como `Completed` ou `Failed`, os pods que o criam normalmente continuam a existir. Esse comportamento permite que você visualize os logs e resultados, mas com o Fargate você incorrerá em custos se não limpar o trabalho posteriormente.

Para excluir automaticamente os pods relacionados após a conclusão do trabalho ou a falha, você pode especificar um período de tempo usando o controlador de vida útil (TTL). O exemplo a seguir mostra a especificação `.spec.ttlSecondsAfterFinished` em seu manifesto de trabalho.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  template:
    spec:
      containers:
      - name: busybox
        image: busybox
        command: ["/bin/sh", "-c", "sleep 10"]
      restartPolicy: Never
  ttlSecondsAfterFinished: 60 # <-- TTL controller
```

Tabela de comparação do Fargate

Critérios	AWS Fargate
Pode ser implantado no AWS Outposts	Não
Pode ser implantado em uma Zona local da AWS	Não
Pode executar contêineres que exijam o Windows	Não
Pode executar contêineres que exijam o Linux	Sim
Pode executar workloads que exijam o chip do Inferentia	Não
Pode executar workloads que exijam uma GPU	Não

Critérios	AWS Fargate
Pode executar workloads que exijam processadores ARM	Não
Pode executar o AWS Bottlerocket	Não
Os pods compartilham um ambiente de runtime do kernel com outros pods	Não; cada pod tem um kernel dedicado
Os pods compartilham recursos de CPU, memória, armazenamento e rede com outros pods.	Não; cada pod tem recursos dedicados e pode ser dimensionado independentemente para maximizar a utilização dos recursos
Os pods podem usar mais hardware e memória do que o solicitado nas especificações do pod	Não; no entanto, o pod pode ser reimplantado usando uma configuração maior de vCPU e memória
Deve implantar e gerenciar instâncias do Amazon EC2	Não
Deve proteger, manter e corrigir o sistema operacional das instâncias do Amazon EC2	Não
Pode fornecer argumentos de bootstrap na implantação de um nó, como argumentos kubelet extras.	Não
Pode atribuir endereços IP a pods de um bloco CIDR diferente do endereço IP atribuído ao nó.	Não
É possível executar o SSH no nó	Não: não há nenhum sistema operacional de host de nó para SSH.
Pode implantar sua própria AMI personalizada em nós	Não
Pode implantar seu próprio CNI personalizado em nós	Não

Critérios	AWS Fargate
Você deve atualizar a AMI do nó por conta própria	Não
Você deve atualizar a versão de Kubernetes do nó por conta própria	Não. Você não gerencia nós.
Pode usar armazenamento do Amazon EBS com pods	Não
Pode usar armazenamento do Amazon EFS com pods	Sim
Pode usar armazenamento do Amazon FSx para Lustre com pods	Não
Pode usar o Network Load Balancer para serviços	Sim, ao usar a opção Criar um balanceador de carga de rede
Pods podem ser executados em uma sub-rede pública	Não
Pode atribuir diferentes grupos de segurança da VPC a pods individuais	Sim
Pode executar o Kubernetes DaemonSets	Não
Compatível com `HostPort` e `HostNetwork` no manifesto do pod	Não
Disponibilidade de regiões do AWS	Algumas regiões compatíveis com o Amazon EKS
Pode executar contêineres em hosts dedicados do Amazon EC2	Não

Critérios	AWS Fargate
Preços	Custo da memória do Fargate e das configurações da CPU. Cada pod tem seu próprio custo. Para obter mais informações, consulte Preços do AWS Fargate.

Comece a usar o AWS Fargate em seu cluster

Este tópico descreve como começar a executar pods no AWS Fargate com o cluster do Amazon EKS.

Se você restringir o acesso ao endpoint público do cluster usando blocos CIDR, é recomendável também habilitar o acesso ao endpoint privado. Dessa forma, os pods do Fargate podem se comunicar com o cluster. Sem o endpoint privado ativado, os blocos CIDR especificados para acesso público devem incluir as origens de saída da VPC. Para obter mais informações, consulte the section called “Acesso ao endpoint do cluster”.

Pré-requisito

Um cluster existente. Se você ainda não tem um cluster do Amazon EKS, consulte Conceitos básicos.

Etapa 1: verificar se os nós existentes podem se comunicar com os pods do Fargate

Se estiver trabalhando com um novo cluster sem nós ou um cluster com apenas grupos de nós gerenciados (consulte the section called “Grupos de nós gerenciados”), pule para the section called “Etapa 2: criar um perfil de execução de pods do Fargate”.

Pressuponha que você esteja trabalhando com um cluster existente que já tenha nós associados a ele. Verifique se os pods nesses nós podem se comunicar livremente com os pods em execução no Fargate. Os pods em execução no Fargate são automaticamente configurados para usar o grupo de segurança do cluster para o cluster ao qual estão associados. Certifique-se de que todos os nós existentes em seu cluster possam enviar e receber tráfego de e para o grupo de segurança do cluster. Os grupos de nós gerenciados são configurados automaticamente para usar também o grupo de segurança do cluster, portanto, não é necessário modificá-los ou verificá-los quanto a essa compatibilidade (consulte the section called “Grupos de nós gerenciados”).

Para grupos de nós existentes que foram criados com `eksctl` ou com os modelos AWS CloudFormation gerenciados pelo Amazon EKS, você pode adicionar manualmente o grupo de segurança do cluster aos nós. Ou então, pode modificar o modelo de lançamento do grupo do Auto Scaling para o grupo de nós para anexar o grupo de segurança do cluster às instâncias. Para obter mais informações, consulte Alterar os grupos de segurança de uma instância no Guia do usuário da Amazon VPC.

Você pode conferir se há um grupo de segurança para o seu cluster no Console de gerenciamento da AWS, na seção Networking (Rede). Ou você pode fazer isso usando o seguinte comando da CLI AWS. Quando usar esse comando, substitua `<my-cluster>` pelo nome de seu cluster.

```
aws eks describe-cluster --name <my-cluster> --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Etapa 2: criar um perfil de execução de pods do Fargate

Quando o cluster cria pods no AWS Fargate, os componentes que são executados na infraestrutura do Fargate devem fazer as chamadas para as APIs da AWS em seu nome. O perfil de execução de pods do Amazon EKS fornece as permissões do IAM para fazer isso. Para criar um perfil de execução de pods do AWS Fargate, consulte the section called “Função do IAM de execução de pod”.

> **Note**
>
> Se você criou o cluster com o `eksctl` usando a opção `--fargate`, ele já tem um perfil de execução de pods que você pode encontrar no console do IAM com o padrão `eksctl-my-cluster-FargatePodExecutionRole-ABCDEFGHIJKL`. Da mesma forma, se você usar o `eksctl` para criar os perfis do Fargate, o `eksctl` criará o perfil de execução de pods, caso ainda não tenha sido criado.

Etapa 3: criar um perfil do Fargate para o cluster

Antes de agendar os pods em execução no Fargate no cluster, é necessário definir um perfil do Fargate que especifique quais pods deverão usá-lo quando forem executados. Para obter mais informações, consulte the section called “Definir perfis”.

> **Note**
>
> Se você criou o cluster com o `eksctl` usando a opção `--fargate`, um perfil do Fargate já foi criado para o cluster com seletores para todos os pods nos namespaces `kube-system` e `default`. Use o procedimento a seguir para criar perfis do Fargate para qualquer outro namespace que você deseja usar com o Fargate.

Você pode criar um perfil do Fargate usando qualquer uma dessas ferramentas:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”

`eksctl`

Este procedimento exige a versão `eksctl` `0.215.0` ou superior. Você pode verificar a versão com o seguinte comando:

```
eksctl version
```

Para obter instruções sobre como instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Para criar um perfil do Fargate com o **`eksctl`**

Crie o perfil do Fargate com o comando `eksctl` a seguir, substituindo cada `<example value>` por seus próprios valores. Você precisa especificar um namespace. Mas, a opção `--labels` não é exigida.

```
eksctl create fargateprofile \
    --cluster <my-cluster> \
    --name <my-fargate-profile> \
    --namespace <my-kubernetes-namespace> \
    --labels <key=value>
```

É possível usar certos curingas para rótulos `<my-kubernetes-namespace>` e `<key=value>` . Para obter mais informações, consulte the section called “Curingas do perfil do Fargate”.

Console de gerenciamento da AWS

Para criar um perfil do Fargate com o Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Escolha o cluster para o qual deve ser criado um perfil do Fargate.
3. Escolha a guia Compute (Computação).
4. Em Fargate profile (Perfil do Fargate), escolha Add Fargate profile (Adicionar perfil do Fargate).
5. Na página Configure Fargate profile (Configurar o perfil do Fargate), faça o seguinte:
   a. Em Name (Nome), insira um nome para o perfil do Fargate. O nome deve ser exclusivo.
   b. Em Perfil de execução de pods, escolha o perfil de execução de pod que deve ser usado com o perfil do Fargate. Somente funções do IAM com o principal de serviço do `eks-fargate-pods.amazonaws.com` são mostradas. Se nenhuma função estiver listada, você deve criar uma. Para obter mais informações, consulte the section called “Função do IAM de execução de pod”.
   c. Modifique as sub-redes selecionadas conforme necessário.

      > **Note**
      >
      > Somente sub-redes privadas são compatíveis com pods em execução no Fargate.

   d. Em Tags (Etiquetas), você tem a opção de etiquetar o perfil do Fargate. Essas tags não são propagadas para outros recursos associados ao perfil, como pods.
   e. Escolha Próximo.
6. Na página Configurar a seleção de pods, faça o seguinte:
   a. Em Namespace, insira um namespace para fazer a correspondência com os pods.
      - É possível usar namespaces específicos para a correspondência, como `kube-system` ou `default`.
      - É possível usar certos curingas (por exemplo, `prod-*`) para fazer a correspondência com vários namespaces (por exemplo, `prod-deployment` e `prod-test`). Para obter mais informações, consulte the section called “Curingas do perfil do Fargate”.
   b. (Opcional) Adicione rótulos do Kubernetes ao seletor. Adicione-os especificamente ao seletor qual os pods no namespace especificado precisam corresponder.

- É possível adicionar o rótulo `infrastructure: fargate` ao seletor para que somente pods no namespace especificado que também tenham o rótulo `infrastructure: fargate` do Kubernetes correspondam ao seletor.
- É possível usar certos curingas (por exemplo, `key?: value?`) para fazer a correspondência com vários namespaces (por exemplo, `keya: valuea` e `keyb: valueb`). Para obter mais informações, consulte the section called “Curingas do perfil do Fargate”.

c. Escolha Próximo.

7. Na página Review and create (Revisar e criar), revise as informações do perfil do Fargate e selecione Create (Criar).

Etapa 4: atualizar o CoreDNS

Por padrão, o CoreDNS é configurado para ser executado na infraestrutura do Amazon EC2 nos clusters do Amazon EKS. Se quiser executar apenas os pods no Fargate no cluster, conclua as etapas a seguir.

> **Note**
>
> Se você criou o cluster com o `eksctl` usando a opção `--fargate`, poderá avançar para the section called “Próximas etapas”.

1. Crie um perfil do Fargate para o CoreDNS com o comando a seguir. Substitua `<my-cluster>` pelo nome do cluster, `<111122223333>` pelo ID da conta, `<AmazonEKSFargatePodExecutionRole>` pelo nome do perfil de execução do pod e `<000000000000000a>`, `<000000000000000b>` e `<000000000000000c>` pelos IDs das sub-redes privadas. Caso não tenha um perfil de execução de pods, você deverá criar um primeiro (consulte the section called “Etapa 2: criar um perfil de execução de pods do Fargate”).

> **Important**
>
> O ARN da função não pode incluir um caminho diferente de `/`.
> Por exemplo, se o nome da sua função for `development/apps/AmazonEKSFargatePodExecutionRole`, você precisará alterá-lo para `AmazonEKSFargatePodExecutionRole` ao especificar o ARN da função. O

formato do ARN da função deve ser `arn:aws:iam::<111122223333>:role/<AmazonEKSFargatePodExecutionRole>`.

```
aws eks create-fargate-profile \
    --fargate-profile-name coredns \
    --cluster-name <my-cluster> \
    --pod-execution-role-arn arn:aws:iam::<111122223333>:role/
<AmazonEKSFargatePodExecutionRole> \
    --selectors namespace=kube-system,labels={k8s-app=kube-dns} \
    --subnets subnet-<000000000000000a> subnet-<000000000000000b> subnet-
<000000000000000c>
```

2. Acione uma reversão da implantação do `coredns`.

```
kubectl rollout restart -n kube-system deployment coredns
```

Próximas etapas

- É possível começar a migrar as aplicações existentes para serem executadas no Fargate com o fluxo de trabalho a seguir.
  a. the section called “Criar um perfil do Fargate” que corresponda ao namespace do Kubernetes da aplicação aos rótulos do Kubernetes.
  b. Exclua e recrie os pods existentes para que eles sejam agendados no Fargate. Modifique o `<namespace>` e `<deployment-type>` para atualizar seus pods específicos.

  ```
  kubectl rollout restart -n <namespace> deployment <deployment-type>
  ```

- Implante o the section called “Balanceador de carga da aplicação” para permitir objetos do Ingress para os pods em execução no Fargate.
- É possível usar o the section called “Vertical Pod Autoscaler” para definir o tamanho inicial correto da CPU e da memória para os pods do Fargate e, em seguida, usar o the section called “Horizontal Pod Autoscaler” para escalá-los. Se quiser que o Vertical Pod Autoscaler reimplante automaticamente os pods no Fargate com combinações maiores de CPU e memória, defina o modo do Vertical Pod Autoscaler como `Auto` ou `Recreate`. Isso é para garantir a funcionalidade correta. Para obter mais informações, consulte a documentação do Vertical Pod Autoscaler no GitHub.

- Você pode configurar o coletor AWS Distro for OpenTelemetry (ADOT) para monitoramento de aplicações seguindo estas instruções.

Definir quais pods usam o AWS Fargate quando iniciado

Antes de agendar pods no Fargate no cluster, você deve definir pelo menos um perfil do Fargate que especifique quais pods usam o Fargate quando iniciado.

Como administrador, você pode usar um perfil do Fargate para declarar quais pods devem ser executados no Fargate. É possível fazer isso por meio dos seletores de perfil. É possível adicionar até cinco seletores a cada perfil. Cada seletor deve conter um namespace. O seletor também pode incluir rótulos. O campo de rótulo consiste em vários pares de chave-valor opcionais. Os pods que correspondem aos seletores são programados no Fargate. Os pods são correspondidos usando um namespace e os rótulos que são especificados no seletor. Se um seletor de namespace for definido sem rótulos, o Amazon EKS tentará agendar todos os pods que são executados nesse namespace no Fargate usando o perfil. Se um pod a ser agendado corresponder a qualquer um dos seletores no perfil do Fargate, então esse pod será agendado no Fargate.

Se um pod corresponder a vários perfis do Fargate, será possível especificar qual perfil um pod usará adicionando o seguinte rótulo do Kubernetes para a especificação do pod: `eks.amazonaws.com/fargate-profile: my-fargate-profile`. O pod deve corresponder a um seletor nesse perfil para ser agendado no Fargate. As regras de afinidade/antiafinidade do Kubernetes não se aplicam e não são necessárias com pods do Fargate para Amazon EKS.

Ao criar um perfil do Fargate, é necessário especificar um perfil de execução do pod. Esse perfil de execução é para os componentes do Amazon EKS que são executados na infraestrutura do Fargate usando o perfil. Ele é adicionado ao Controle de acesso baseado em perfil (RBAC) do Kubernetes do cluster para autorização. Dessa forma, o `kubelet` que está sendo executado na infraestrutura do Fargate pode ser registrado no cluster do Amazon EKS e aparede no cluster como um nó. O perfil de execução do pod também fornece permissões do IAM para a infraestrutura do Fargate para permitir acesso de leitura aos repositórios de imagens do Amazon ECR. Para ter mais informações, consulte the section called “Função do IAM de execução de pod”.

Os perfis do Fargate não podem ser alterados. No entanto, é possível criar um perfil atualizado para substituir um perfil existente e excluir o original.

> **Note**
>
> Qualquer pod que esteja sendo executado usando um perfil do Fargate será interrompido e colocado como pendente quando o perfil for excluído.

Se algum perfil do Fargate em um cluster estiver no status `DELETING`, será necessário aguardar até que a exclusão do perfil do Fargate seja concluída para poder criar qualquer outro perfil nesse cluster.

> **Note**
>
> Atualmente, o Fargate não é compatível com topologySpreadConstraints do Kubernetes.

O Amazon EKS e o Fargate distribuem pods em todas as sub-redes definidas no perfil do Fargate. No entanto, é possível acabar com uma distribuição desigual. Se você precisar ter uma distribuição uniforme, use dois perfis do Fargate. A distribuição uniforme é importante em cenários em que você deseja implantar duas réplicas e não quer nenhum tempo de inatividade. Recomendamos que cada perfil tenha apenas uma sub-rede.

Componentes do perfil do Fargate

Os componentes a seguir estão contidos em um perfil do Fargate.

Função de execução de pod

Quando o cluster cria pods no AWS Fargate, o `kubelet` que está sendo executado na infraestrutura do Fargate deve fazer chamadas para as APIs da AWS em seu nome. Por exemplo, ele precisa fazer chamadas para extrair imagens de contêiner do Amazon ECR. O perfil de execução de pods do Amazon EKS fornece as permissões do IAM para fazer isso.

Ao criar um perfil do Fargate, é necessário especificar um perfil de execução do pod para ser usado com os pods. Esse perfil é adicionado ao Controle de acesso baseado em perfil (RBAC) do Kubernetes do cluster para autorização. Isso permite que o `kubelet` que está sendo executado na infraestrutura do Fargate possa ser registrado no cluster do Amazon EKS e apareça no cluster como um nó. Para ter mais informações, consulte the section called “Função do IAM de execução de pod”.

Sub-redes

Os IDs de sub-redes para iniciar pods que usam esse perfil. No momento, os pods sendo executados no Fargate não recebem endereços IP públicos. Portanto, somente sub-redes privadas (sem rota direta para um gateway da Internet) são aceitas para esse parâmetro.

Seletores

Os seletores a serem correspondidos para que os pods usem esse perfil do Fargate. É possível especificar até cinco seletores em um perfil do Fargate. Os seletores têm os seguintes componentes:

- Namespace: é necessário especificar um namespace para um seletor. O seletor corresponde apenas a pods que são criados nesse namespace. No entanto, é possível criar vários seletores para visar a vários namespaces.
- Rótulos: se preferir, você poderá especificar rótulos do Kubernetes para corresponder ao seletor. O seletor só corresponde pods que têm todos os rótulos especificados no seletor.

Curingas do perfil do Fargate

Além dos caracteres permitidos pelo Kubernetes, você está autorizado a usar `*` e `?` nos critérios de seletor para namespaces, chaves de rótulo e valores de rótulos:

- `*` representa nenhum, um ou vários caracteres. Por exemplo, `prod*` pode representar `prod` e `prod-metrics`.
- `?` representa um único caractere (por exemplo, `value?` pode representar `valuea`). No entanto, não pode representar `value` e `value-a`, porque `?` só pode representar exatamente um caractere.

Esses caracteres curinga podem ser usados em qualquer posição e em combinação (por exemplo, `prod*`, `*dev` e `frontend*?`). Não há suporte para outros curingas e formas de correspondência de padrões, como expressões regulares.

Se houver vários perfis correspondentes para o namespace e os rótulos na especificação do pod, o Fargate selecionará o perfil com base na classificação alfanumérica pelo nome do perfil. Por exemplo, se ambos os perfis A (com o nome `beta-workload`) e B (com o nome `prod-workload`) tiverem seletores correspondentes para os pods a serem iniciados, o Fargate escolherá o perfil A (`beta-workload`) para os pods. Os pods têm rótulos com o perfil A nos pods (por exemplo, `eks.amazonaws.com/fargate-profile=beta-workload`).

Se você quiser migrar os pods existentes do Fargate para novos perfis que usem curingas, há duas maneiras de fazer isso:

- Crie um novo perfil com seletores correspondentes e exclua os perfis antigos. Os pods rotulados com perfis antigos serão reprogramados para novos perfis correspondentes.
- Se você quiser migrar workloads, mas não tiver certeza de quais rótulos do Fargate estão em cada pod do Fargate, será possível usar o método a seguir. Crie um novo perfil com um nome que seja classificado em ordem alfanumérica primeiro entre os perfis no mesmo cluster. Em seguida, recicle os pods do Fargate que precisem ser migrados para novos perfis.

Criar um perfil do Fargate

Esta seção descreve como criar um perfil do Fargate. Também é necessário ter criado uma perfil de execução de pod a ser usado para o perfil do Fargate. Para ter mais informações, consulte the section called “Função do IAM de execução de pod”. Os pods em execução no Fargate são apenas compatíveis com sub-redes privadas com acesso do gateway NAT aos serviços da AWS, mas não uma rota direta para um gateway da internet). Isso ocorre para que a VPC do seu cluster tenha sub-redes privadas disponíveis.

Você pode criar um perfil com o seguinte:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”

`eksctl`

Para criar um perfil do Fargate com o **`eksctl`**

Crie seu perfil do Fargate com o comando do `eksctl` a seguir, substituindo cada valor de exemplo por seus próprios valores. Você precisa especificar um namespace. Mas, a opção `--labels` não é exigida.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

É possível usar certos curingas para rótulos `my-kubernetes-namespace` e `key=value` . Para ter mais informações, consulte the section called “Curingas do perfil do Fargate”.

Console de gerenciamento da AWS

Para criar um perfil do Fargate com o Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Escolha o cluster para o qual deve ser criado um perfil do Fargate.
3. Escolha a guia Compute (Computação).
4. Em Fargate profile (Perfil do Fargate), escolha Add Fargate profile (Adicionar perfil do Fargate).
5. Na página Configure Fargate profile (Configurar o perfil do Fargate), faça o seguinte:
   a. Em Name (Nome), insira um nome exclusivo para o perfil do Fargate, como `my-profile`.
   b. Em Perfil de execução de pods, escolha o perfil de execução de pod que deve ser usado com o perfil do Fargate. Somente funções do IAM com o principal de serviço do `eks-fargate-pods.amazonaws.com` são mostradas. Se nenhuma função estiver listada, você deve criar uma. Para ter mais informações, consulte the section called “Função do IAM de execução de pod”.
   c. Modifique as sub-redes selecionadas conforme necessário.

      > **Note**
      >
      > Somente sub-redes privadas são compatíveis com pods em execução no Fargate.

   d. Em Tags (Etiquetas), você tem a opção de etiquetar o perfil do Fargate. Essas tags não são propagadas para outros recursos associados ao perfil, como pods.
   e. Escolha Próximo.
6. Na página Configurar a seleção de pods, faça o seguinte:
   a. Em Namespace, insira um namespace para fazer a correspondência com os pods.
      - É possível usar namespaces específicos para a correspondência, como `kube-system` ou `default`.
      - É possível usar certos curingas (por exemplo, `prod-*`) para fazer a correspondência com vários namespaces (por exemplo, `prod-deployment` e `prod-test`). Para ter mais informações, consulte the section called “Curingas do perfil do Fargate”.

b. (Opcional) Adicione rótulos do Kubernetes ao seletor. Adicione-os especificamente ao seletor ao qual os pods no namespace especificado precisam corresponder.

- É possível adicionar o rótulo `infrastructure: fargate` ao seletor para que somente pods no namespace especificado que também tenham o rótulo `infrastructure: fargate` do Kubernetes correspondam ao seletor.
- É possível usar certos curingas (por exemplo, `key?: value?`) para fazer a correspondência com vários namespaces (por exemplo, `keya: valuea` e `keyb: valueb`). Para ter mais informações, consulte the section called “Curingas do perfil do Fargate”.

c. Escolha Próximo.

7. Na página Review and create (Revisar e criar), revise as informações do perfil do Fargate e selecione Create (Criar).

Excluir um perfil do Fargate

Este tópico descreve como excluir um perfil do Fargate. Quando você exclui um perfil do Fargate, qualquer pod que foi agendado no Fargate com o perfil é excluído. Se esses pods corresponderem a outro perfil do Fargate, eles serão agendados no Fargate com esse perfil. Se eles não corresponderem mais a nenhum perfil do Fargate, eles não estão agendados para o Fargate e poderão continuar pendentes.

Somente um perfil do Fargate em um cluster pode estar no status `DELETING` de cada vez. Espere que um perfil do Fargate conclua a exclusão antes de excluir qualquer outro perfil desse cluster.

Você pode excluir um perfil com qualquer uma das seguintes ferramentas:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”
- the section called “AWS CLI”

`eksctl`

Excluir um perfil do Fargate com **`eksctl`**

Use o comando a seguir para excluir um perfil de um cluster. Substitua cada *`valor de exemplo`* por seus próprios valores.

```
eksctl delete fargateprofile  --name my-profile --cluster my-cluster
```

Console de gerenciamento da AWS

Excluir um perfil do Fargate com Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, escolha Clusters. Na lista de clusters, escolha o cluster do qual deseja excluir o perfil do Fargate.
3. Escolha a guia Compute (Computação).
4. Escolha o perfil do Fargate a ser excluído e, em seguida, escolha Delete (Excluir).
5. Na página Delete Fargate Profile (Excluir perfil do Fargate), insira o nome do perfil e, em seguida, escolha Delete (Excluir).

AWS CLI

Excluir um perfil Fargate com AWS CLI

Use o comando a seguir para excluir um perfil de um cluster. Substitua cada *valor de exemplo* por seus próprios valores.

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-
cluster
```

Conhecer os detalhes de configuração do pod do Fargate

Esta seção descreve alguns dos detalhes exclusivos da configuração do pod para a execução de pods do Kubernetes no AWS Fargate.

CPU e memória do pod

Com o Kubernetes, você pode definir solicitações, uma quantidade mínima de vCPU e recursos de memória que são alocados em cada contêiner em um pod. Os pods são agendados pelo Kubernetes para garantir que pelo menos os recursos solicitados para cada pod estejam disponíveis no recurso de computação. Para obter mais informações, consulte Managing compute resources for containers na documentação do Kubernetes.

> **Note**
>
> Como o Fargate para Amazon EKS executa apenas um pod por nó, não ocorre a remoção de pods em caso de menos recursos. Todos os pods do Fargate para Amazon EKS são executados com prioridade garantida, então a CPU e a memória solicitadas devem ser iguais ao limite de todos os contêineres. Para obter mais informações, consulte Configurar a qualidade do serviço para pods na documentação do Kubernetes.

Quando os pods são agendados no Fargate, as reservas de vCPU e de memória dentro da especificação do pod determinam a quantidade de CPU e de memória que devem ser provisionadas para o pod.

- A solicitação máxima de qualquer contêiner de inicialização é usada para determinar os requisitos de vCPU e de memória da solicitação de inicialização.
- As solicitações para todos os contêineres de longa execução são adicionadas para determinar os requisitos de vCPU e de memória da solicitação de longa duração.
- O maior dos dois valores anteriores é escolhido para que a solicitação de vCPU e de memória use em seu pod.
- O Fargate adiciona 256 MB a cada reserva de memória do pod para os componentes necessários do Kubernetes (`kubelet`, `kube-proxy` e `containerd`).

O Fargate faz o arredondamento para a configuração de computação seguinte que mais se aproxima da soma das solicitações de vCPU e de memória, a fim de garantir que os pods sempre tenham os recursos necessários para execução.

Se você não especificar uma combinação de vCPU e de memória, será usada a menor combinação (0,25 de vCPU e 0,5 GB de memória).

A tabela a seguir mostra as combinações de vCPU e de memória que estão disponíveis para pods em execução no Fargate.

Valor de vCPU	Valor de memória
0,25 vCPU	0,5 GB, 1 GB, 2 GB
0,5 vCPU	1 GB, 2 GB, 3 GB, 4 GB

Valor de vCPU	Valor de memória
1 vCPU	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB
2 vCPU	Entre 4 GB e 16 GB em incrementos de 1 GB
4 vCPU	Entre 8 GB e 30 GB em incrementos de 1 GB
8 vCPU	Entre 16 GB e 60 GB em incrementos de 4 GB
16 vCPU	Entre 32 GB e 120 GB em incrementos de 8 GB

A memória adicional reservada para os componentes do Kubernetes pode fazer com que uma tarefa do Fargate com mais vCPUs do que o solicitado seja provisionada. Por exemplo, uma solicitação de 1 vCPU e 8 GB de memória terá 256 MB adicionados à solicitação de memória e provisionará uma tarefa do Fargate com 2 vCPUs e 9 GB de memória, já que nenhuma tarefa com 1 vCPU e 9 GB de memória está disponível.

Não há correlação entre o tamanho do pod em execução no Fargate e o tamanho do nó relatado pelo Kubernetes com `kubectl get nodes`. O tamanho do nó relatado geralmente é maior do que a capacidade do pod. Você pode verificar a capacidade do pod com o comando a seguir. Substitua *`default`* pelo namespace do pod e *`pod-name`* pelo nome do pod.

```
kubectl describe pod --namespace default pod-name
```

Veja um exemplo de saída abaixo.

```
[...]
annotations:
    CapacityProvisioned: 0.25vCPU 0.5GB
[...]
```

A anotação `CapacityProvisioned` representa a capacidade do pod imposta e determina o custo do pod em execução no Fargate. Para obter informações sobre preços das configurações de computação, consulte Preço do AWS Fargate.

Armazenamento do Fargate

Um pod em execução no Fargate automaticamente monta um sistema de arquivos do Amazon EFS sem precisar das etapas de instalação manual do driver. Você não pode usar o provisionamento dinâmico de volume persistente com nós do Fargate, mas pode usar o provisionamento estático. Para obter mais informações, consulte Driver da CSI do Amazon EFS no GitHub.

Quando provisionado, cada pod em execução no Fargate recebe um padrão de 20 GiB de armazenamento temporário. Esse tipo de armazenamento é excluído depois que um pod é interrompido. Os novos pods iniciados no Fargate têm a criptografia de volume de armazenamento temporário habilitada por padrão. O armazenamento temporário do pod é criptografado com um algoritmo de criptografia AES-256 usando chaves gerenciadas pelo AWS Fargate.

> **Note**
>
> O armazenamento utilizável padrão para pods do Amazon EKS executados no Fargate é inferior a 20 GiB. Isso ocorre porque uma parte do espaço é usada pelo `kubelet` e outros módulos do Kubernetes que são carregados no pod.

É possível aumentar a quantidade total de armazenamento temporário para até 175 GiB. Para configurar o tamanho com o Kubernetes, especifique as solicitações do recurso `ephemeral-storage` para cada contêiner em um pod. Quando o Kubernetes agendar pods, garanta que a soma das solicitações de recursos para cada pod seja menor que a capacidade da tarefa do Fargate. Para obter mais informações, consulte Gerenciamento de recursos em pods e contêineres na documentação do Kubernetes.

O Amazon EKS Fargate fornece mais armazenamento temporário do que o solicitado para fins de uso do sistema. Por exemplo, uma solicitação de 100 GiB provisionará uma tarefa do Fargate com armazenamento temporário de 115 GiB.

Definir ações para AWS eventos de aplicação de patches do Fargate OS

O Amazon EKS corrige periodicamente o sistema operacional dos nós do AWS Fargate para mantê-los seguros. Como parte do processo de correção, reciclamos os nós para instalar os patches do sistema operacional. Há tentativas de atualizações de uma forma que gera o menor impacto possível nos seus serviços. Porém, se os pods não forem removidos com êxito, haverá momentos em que eles deverão ser excluídos. As ações a seguir podem ser executadas para minimizar possíveis interrupções:

- Defina os orçamentos de interrupção de pods (PDBs) apropriados para controlar o número de pods que se tornam inativos ao mesmo tempo.
- Crie regras do Amazon EventBridge para tratar as remoções com falha antes que os pods sejam excluídos.
- Reinicie manualmente os pods afetados antes da data de remoção publicada na notificação que você recebe.
- Crie uma configuração de notificação em Notificações de Usuários da AWS.

O Amazon EKS trabalha em conjunto com a comunidade do Kubernetes para disponibilizar correções de bugs e patches de segurança da maneira mais rápida possível. Todos os pods do Fargate começam com a versão de patch mais recente do Kubernetes, disponível no Amazon EKS para a versão do Kubernetes do seu cluster. Se você tiver um pod com uma versão de patch mais antiga, o Amazon EKS poderá reciclar para atualizá-lo para a versão mais recente. Isso garante que os pods estejam equipados com as mais recentes atualizações de segurança. Dessa maneira, se houver um problema crítico de Vulnerabilidades e exposições comuns (CVE), você permanecerá em dia para reduzir os riscos de segurança.

Quando o sistema operacional do AWS Fargate for atualizado, o Amazon EKS enviará uma notificação que inclui os recursos afetados e a data das próximas remoções de pod. Se a data da remoção fornecida for inconveniente, você tem a opção de reiniciar manualmente os pods afetados antes da data de remoção publicada na notificação. Todos os pods criados antes do momento em que você recebe a notificação estão sujeitos a remoção. Consulte a Documentação do Kubernetes para obter mais instruções sobre como reiniciar manualmente seus pods.

Para limitar o número de pods inativos ao mesmo tempo quando estes são reciclados, é possível definir orçamentos de interrupção de pods (PDBs). É possível usar PDBs para definir a disponibilidade mínima com base nos requisitos de cada uma das suas aplicações e, ao mesmo tempo, possibilitar que as atualizações ocorram. A disponibilidade mínima do seu PDB deve ser inferior a 100%. Para obter mais informações, consulte Specifying a Disruption Budget for your Application na documentação do Kubernetes.

O Amazon EKS usa a API de remoção para drenar o pod com segurança, respeitando ao mesmo tempo os PDBs definidos para a aplicação. Os pods são removidos por zona de disponibilidade para minimizar os impactos. Se a remoção tiver êxito, o novo pod receberá o patch mais recente e nenhuma ação adicional será necessária.

Quando a remoção de um pod falhar, o Amazon EKS enviará um evento à sua conta com detalhes sobre os pods que falharam. É possível atuar na mensagem antes do horário de encerramento programado. O tempo específico varia dependendo da urgência do patch. No momento certo, o Amazon EKS tentará remover os pods novamente. Porém, desta vez, um novo evento não será enviado se a remoção falhar. Se a remoção falhar novamente, os pods existentes serão excluídos periodicamente para que novos pods possam receber o patch mais recente.

Veja a seguir um exemplo de evento recebido quando a remoção do pod apresenta falhas. Ele inclui detalhes sobre o cluster, o nome do pod, o namespace do pod, o perfil do Fargate e o horário de término agendado.

```
{
    "version": "0",
    "id": "12345678-90ab-cdef-0123-4567890abcde",
    "detail-type": "EKS Fargate Pod Scheduled Termination",
    "source": "aws.eks",
    "account": "111122223333",
    "time": "2021-06-27T12:52:44Z",
    "region": "region-code",
    "resources": [
        "default/my-database-deployment"
    ],
    "detail": {
        "clusterName": "my-cluster",
        "fargateProfileName": "my-fargate-profile",
        "podName": "my-pod-name",
        "podNamespace": "default",
        "evictErrorMessage": "Cannot evict pod as it would violate the pod's disruption
 budget",
        "scheduledTerminationTime": "2021-06-30T12:52:44.832Z[UTC]"
    }
}
```

Além disso, a existência de vários PDBs associados a um pod pode causar um evento de falha de remoção. Esse evento retorna a mensagem de erro a seguir.

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction
 subresource does not support",
```

É possível criar uma ação desejada com base nesse evento. Por exemplo, é possível ajustar o orçamento de interrupção de pods (PDB) para controlar como os pods são removidos. Mais

especificamente, suponha que você comece com um PDB que define a porcentagem-alvo de pods disponíveis. Antes que os pods sejam encerrados à força durante uma atualização, é possível ajustar o PDB para outra porcentagem de pods. Para receber esse evento, você deve criar uma regra do Amazon EventBridge na conta AWS e na região AWS à qual o cluster pertence. A regra deve utilizar o seguinte Padrão personalizado: Para obter mais informações, consulte Criar regras do Amazon EventBridge que reajam a eventos no Guia do usuário do Amazon EventBridge.

```
{
  "source": ["aws.eks"],
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

Um destino apropriado pode ser definido para o evento capturá-lo. Para acessar uma lista completa de destinos disponíveis, consulte Destinos do Amazon EventBridge, no Guia do usuário do Amazon EventBridge. Também é possível criar uma configuração de notificação em Notificações de Usuários da AWS. Ao usar o Console de gerenciamento da AWS para criar a notificação, em Regras de evento, escolha Elastic Kubernetes Service (EKS) para o nome do serviço AWS e EKS Fargate Pod Scheduled Termination para o tipo de evento. Para obter mais informações, consulte Introdução às Notificações de Usuários da AWS no Guia do usuário das Notificações de Usuários da AWS.

Consulte Perguntas frequentes: aviso de remoção do Fargate Pod em AWS re:Post para ver as perguntas mais frequentes sobre remoções de Pods do EKS.

Colete métricas de uso e da aplicação AWS Fargate

Você pode coletar métricas do sistema e métricas de uso do CloudWatch para o AWS Fargate.

Métricas da aplicação

Para aplicações em execução no Amazon EKS e no AWS Fargate, você pode usar o AWS Distro para OpenTelemetry (ADOT). O ADOT permite coletar métricas do sistema e enviá-las para os painéis do CloudWatch Container Insights. Para começar a usar o ADOT para aplicações em execução no Fargate, consulte Using CloudWatch Container Insights with AWS Distro for OpenTelemetry (Usar o CloudWatch Container Insights com Distro para OpenTelemetry) na documentação do ADOT.

Métricas de uso

É possível usar métricas de uso do CloudWatch para fornecer visibilidade sobre o uso de recursos de sua conta. Use essas métricas para visualizar o uso do serviço atual nos gráficos e painéis do CloudWatch.

AWS As métricas de uso do Fargate correspondem às cotas de serviço do AWS. Também é possível configurar alarmes que alertem você quando o uso se aproximar de uma cota de serviço. Para obter mais informações sobre cotas de serviço do Fargate, consulte the section called “Cotas de serviço”.

O AWS Fargate publica as métricas no namespace `AWS/Usage` a seguir.

Métrica	Descrição
`ResourceCount`	O número total dos recursos especificados em execução na sua conta. Os recursos são definidos pelas dimensões associadas à métrica.

As dimensões a seguir são usadas para refinar as métricas de uso publicadas pelo AWS Fargate.

Dimensão	Descrição
`Service`	O nome do serviço da AWS que contém o recurso. Para as métricas de uso do AWS Fargate, o valor para essa dimensão é `Fargate`.
`Type`	O tipo de entidade que está sendo relatada. Atualmente, o único valor válido para as métricas de uso do AWS Fargate é `Resource`.
`Resource`	O tipo de recurso que está em execução. Atualmente, o AWS Fargate retorna informaçõ es sobre o uso do Fargate On-Demand. O valor do recurso para uso sob demanda do Fargate é `OnDemand`.

Dimensão	Descrição
	[OBSERVAÇÃO] ==== O uso sob demanda do Fargate combina pods do Amazon EKS usando o Fargate, tarefas do Amazon ECS usando o tipo de execução do Fargate e tarefas do Amazon ECS usando o provedor de capacidade do `FARGATE`. ====
`Class`	A classe do recurso sob acompanhamento. Atualmente, o AWS Fargate não usa a dimensão de classe.

Criar um alarme do CloudWatch para monitorar métricas de uso dos recursos do Fargate

AWS A Fargate fornece métricas de uso do CloudWatch que correspondem às cotas de serviço do AWS para uso de recursos do Fargate On-Demand. No console do Service Quotas, você pode visualizar seu uso em um gráfico. Também é possível configurar alarmes que alertam você quando o uso se aproxima de uma cota de serviço. Para ter mais informações, consulte the section called “Coletar métricas”.

Use as etapas a seguir para criar um alarme do CloudWatch com base em uma das métricas de uso dos recursos do Fargate.

1. Abra o console Service Quotas em https://console.aws.amazon.com/servicequotas/.
2. No painel de navegação à esquerda, escolha Serviços da AWS.
3. Na lista de serviços AWS, pesquise e selecione AWS Fargate.
4. Na lista Service quotas (Cotas de serviço), escolha a cota de uso do Fargate para a qual você deseja criar um alarme.
5. Na seção “Amazon CloudWatch alarms” (Alarmes do Amazon CloudWatch), escolha Create (Criar).
6. Em Alarm threshold (Limitação do alarme), escolha a porcentagem do valor da cota aplicada que você deseja definir como o valor do alarme.
7. Em Alarm name (Nome do alarme), insira um nome para o alarme e selecione Create (Criar).

Inicie o log no AWS Fargate para o seu cluster

O Amazon EKS no Fargate oferece um roteador de log integrado baseado no Fluent Bit. Isso significa que você não executa explicitamente um contêiner do Fluent Bit como um sidecar, mas a Amazon o executa para você. Tudo o que você precisa fazer é configurar o roteador de log. A configuração acontece por meio de um `ConfigMap` que deve atender aos seguintes critérios:

- Nomeado `aws-logging`
- Criado em um namespace dedicado chamado `aws-observability`
- Não pode exceder 5.300 caracteres.

Depois de criar o `ConfigMap`, o Amazon EKS no Fargate o detecta automaticamente e configura o roteador de log com ele. O Fargate usa uma versão do AWS para Fluent Bit, uma distribuição do Fluent Bit a montante, compatível e gerenciada pela AWS. Para obter mais informações, consulte AWS for Fluent Bit no GitHub.

O roteador de log permite que você use toda a variedade de serviços da AWS para analytics de log e armazenamento. Você pode transmitir logs do Fargate diretamente para o Amazon CloudWatch, Amazon OpenSearch Service. Você também pode transmitir logs para destinos como Amazon S3, Amazon Kinesis Data Streams e ferramentas de parceiros por meio do Amazon Data Firehose.

- Um perfil existente do Fargate que especifica um namespace existente do Kubernetes em que você implanta pods do Fargate. Para obter mais informações, consulte the section called “Etapa 3: criar um perfil do Fargate para o cluster”.
- Um perfil de execução existente de pod do Fargate. Para obter mais informações, consulte the section called “Etapa 2: criar um perfil de execução de pods do Fargate”.

Configuração do roteador de log

> **Important**
>
> Para que os logs sejam publicados com êxito, é necessário haver acesso de rede da VPC em que seu cluster está localizado até o destino do log. Isso diz respeito principalmente aos usuários que personalizam as regras de saída para suas VPCs. Para obter um exemplo de uso do CloudWatch, consulte Usar o CloudWatch Logs com endpoints da VPC de interface no Guia do usuário do Amazon CloudWatch Logs.

Nas etapas a seguir, substitua cada *valor de exemplo* por seus próprios valores.

1. Crie um namespace dedicado do Kubernetes chamado `aws-observability`.
   a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado `aws-observability-namespace.yaml`: O valor para `name` deve ser `aws-observability` e o rótulo `aws-observability: enabled` é obrigatório.

```
kind: Namespace
apiVersion: v1
metadata:
  name: aws-observability
  labels:
    aws-observability: enabled
```

   b. Crie o novo namespace.

```
kubectl apply -f aws-observability-namespace.yaml
```

2. Crie um `ConfigMap` com um valor de dados do `Fluent Conf` para enviar logs de contêiner a um destino. Fluent Conf é o Fluent Bit, uma linguagem de configuração de processador de logs rápida e leve usada para encaminhar logs de contêiner para um destino de sua escolha. Para obter mais informações, consulte Configuration File (Arquivo de configuração) na documentação do Fluent Bit.

> **Important**
>
> As principais seções incluídas em um `Fluent Conf` típico são `Service`, `Input`, `Filter` e `Output`. No entanto, o roteador de log do Fargate só aceita:
>
> - As seções `Filter` e `Output`.
> - Uma seção `Parser`.
>
> Se você fornecer outras seções, elas serão rejeitadas.

   O roteador de log do Fargate gerencia as seções `Input` e `Service`. Ele tem a seção `Input` a seguir, que não pode ser modificada e não é necessária no `ConfigMap`. Porém, ela pode oferecer informações, como o limite do buffer de memória e a tag aplicada aos logs.

```
[INPUT]
    Name tail
```

```
    Buffer_Max_Size 66KB
    DB /var/log/flb_kube.db
    Mem_Buf_Limit 45MB
    Path /var/log/containers/*.log
    Read_From_Head On
    Refresh_Interval 10
    Rotate_Wait 30
    Skip_Long_Lines On
    Tag kube.*
```

Ao criar o `ConfigMap`, leve em consideração as seguintes regras que o Fargate usa para validar os campos:

- `[FILTER]`,`[OUTPUT]`, e `[PARSER]` devem ser especificados sob cada chave correspondente. Por exemplo, o `[FILTER]` deve estar em `filters.conf`. Você pode ter um ou mais `[FILTER]`s em `filters.conf`. As seções `[OUTPUT]` e `[PARSER]` também devem estar sob suas chaves correspondentes. Ao especificar várias seções `[OUTPUT]`, você pode rotear os logs para destinos diferentes ao mesmo tempo.
- O Fargate valida as chaves necessárias para cada seção. `Name` e `match` são necessários para cada `[FILTER]` e `[OUTPUT]`. `Name` e `format` são necessários para cada `[PARSER]`. As chaves não diferenciam maiúsculas de minúsculas.
- Variáveis de ambiente como `${ENV_VAR}` não são permitidas no `ConfigMap`.
- O recuo tem que ser o mesmo para o par de diretivas ou chave-valor dentro de cada `filters.conf`, `output.conf` e `parsers.conf`. Pares chave-valor têm que ser recuados mais do que as diretivas.
- O Fargate valida de acordo com os seguintes filtros compatíveis: `grep`, `parser`, `record_modifier`, `rewrite_tag`, `throttle`, `nest`, `modify` e `kubernetes`.
- O Fargate valida de acordo com os seguintes filtros compatíveis: `es`, `firehose`, `kinesis_firehose`, `cloudwatch`, `cloudwatch_logs` e `kinesis`.
- É necessário fornecer ao menos um plug-in `Output` com suporte no `ConfigMap` para habilitar o registro em log. `Filter` e `Parser` não são necessários para habilitar o registro em log.

  Você também pode executar o Fluent Bit no Amazon EC2 usando a configuração desejada para solucionar quaisquer problemas decorrentes da validação. Crie o `ConfigMap` usando um dos exemplos a seguir.

> **Important**
>
> O registro em log do Fargate no Amazon EKS não oferece suporte para a configuração dinâmica de `ConfigMap`. Quaisquer alterações em um `ConfigMap` são aplicadas somente aos novos pods. As alterações não são aplicadas a pods existentes.

Crie um `ConfigMap` usando o exemplo para o destino de log desejado.

> **Note**
>
> Você também pode usar o Amazon Kinesis Data Streams para seu destino de log. Se você usar o Kinesis Data Streams, verifique se a função de execução do pod recebeu a permissão `kinesis:PutRecords`. Para obter mais informações, consulte as Permissões do Amazon Kinesis Data Streams no Fluent Bit: manual oficial.

Example

CloudWatch

Você tem duas opções de saída ao usar o CloudWatch:

- Um plugin de saída criado em C
- Um plugin de saída criado em Golang

O exemplo a seguir mostra como usar o plug-in `cloudwatch_logs` para enviar logs para o CloudWatch.

a. Salve o conteúdo a seguir em um arquivo denominado `aws-logging-cloudwatch-configmap.yaml`. Substitua *`region-code`* pela região da AWS em que o cluster se encontra. Os parâmetros em `[OUTPUT]` são necessários.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  flb_log_cw: "false"  # Set to true to ship Fluent Bit process logs to
 CloudWatch.
```

```
  filters.conf: |
    [FILTER]
        Name parser
        Match *
        Key_name log
        Parser crio
    [FILTER]
        Name kubernetes
        Match kube.*
        Merge_Log On
        Keep_Log Off
        Buffer_Size 0
        Kube_Meta_Cache_TTL 300s
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
        Match   kube.*
        region region-code
        log_group_name my-logs
        log_stream_prefix from-fluent-bit-
        log_retention_days 60
        auto_create_group true
  parsers.conf: |
    [PARSER]
        Name crio
        Format Regex
        Regex ^(?<time>[^ ]+) (?<stream>stdout|stderr) (?<logtag>P|F) (?
<log>.*)$
        Time_Key    time
        Time_Format %Y-%m-%dT%H:%M:%S.%L%z
```

b. Aplique o manifesto ao cluster.

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

Amazon OpenSearch Service

Se quiser enviar logs para o Amazon OpenSearch Service, é possível usar a saída es, que é um plug-in escrito em C. O exemplo a seguir mostra como usar o plug-in para enviar logs para o OpenSearch.

a. Salve o conteúdo a seguir em um arquivo denominado `aws-logging-opensearch-configmap.yaml`. Substitua cada *`valor de exemplo`* por seus próprios valores.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
      Name  es
      Match *
      Host  search-example-gjxdcilagiprbglqn42jsty66y.region-
code.es.amazonaws.com
      Port  443
      Index example
      Type  example_type
      AWS_Auth On
      AWS_Region region-code
      tls   On
```

b. Aplique o manifesto ao cluster.

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

Firehose

Você tem duas opções de saída ao enviar logs para o Firehose:

- kinesis_firehose: um plug-in de saída escrito em C.
- firehose - Um plug-in de saída escrito em Golang.

O exemplo a seguir mostra como usar o plug-in `kinesis_firehose` para enviar logs para o Firehose.

a. Salve o conteúdo a seguir em um arquivo denominado `aws-logging-firehose-configmap.yaml`. Substitua *region-code* pela região da AWS em que o cluster se encontra.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
```

```
data:
  output.conf: |
    [OUTPUT]
     Name  kinesis_firehose
     Match *
     region region-code
     delivery_stream my-stream-firehose
```

b. Aplique o manifesto ao cluster.

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. Configure as permissões para o perfil de execução do pod do Fargate para enviar logs ao seu destino.

a. Baixe a política do IAM do seu destino para o computador.

Example

CloudWatch

Baixe a política de IAM do CloudWatch no computador. Você também pode visualizar a política no GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/cloudwatchlogs/permissions.json
```

Amazon OpenSearch Service

Baixe a política do IAM OpenSearch para o seu computador. Você também pode visualizar a política no GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-
fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/
permissions.json
```

Verifique se o controle de acesso do OpenSearch Dashboards está configurado corretamente. O `all_access role` no OpenSearch Dashboards precisa que o perfil de execução de pods do Fargate e o perfil do IAM sejam mapeados. O mesmo mapeamento deve ser feito para a função `security_manager`. Você pode adicionar os mapeamentos anteriores selecionando `Menu`, depois `Security`, depois `Roles` e, em seguida, selecionar as respectivas funções. Para obter mais informações, consulte How do I troubleshoot

CloudWatch Logs so that it streams to my Amazon ES domain? (Como solucionar problemas do CloudWatch Logs para que ele transmita para o meu domínio do Amazon ES?).

Firehose

Baixe a política do IAM do Firehose para o seu computador. Você também pode visualizar a política no GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-logging-examples/mainline/examples/fargate/kinesis-firehose/permissions.json
```

b. Crie uma política do IAM com base no arquivo de política que você baixou.

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document
 file://permissions.json
```

c. Anexe a política do IAM à função de execução de pod especificada para o seu perfil do Fargate com o comando a seguir. Substitua *111122223333* pelo ID da sua conta. Substitua *AmazonEKSFargatePodExecutionRole* pelo perfil de execução de pods (para obter mais informações, consulte the section called “Etapa 2: criar um perfil de execução de pods do Fargate”).

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Suporte ao filtro Kubernetes

O filtro Kubernetes Fluent Bit permite adicionar metadados do Kubernetes aos arquivos de log. Para mais informações sobre o filtro, consulte Kubernetes na documentação do Fluent Bit. É possível aplicar um filtro usando o endpoint do servidor da API.

```
filters.conf: |
    [FILTER]
        Name                kubernetes
        Match               kube.*
        Merge_Log           On
        Buffer_Size         0
```

```
        Kube_Meta_Cache_TTL 300s
```

Important

- `Kube_URL`, `Kube_CA_File`, `Kube_Token_Command` e `Kube_Token_File` são parâmetros de configuração de propriedade do serviço e não devem ser especificados. O Amazon EKS Fargate preenche esses valores.
- `Kube_Meta_Cache_TTL` é o tempo em que o Fluent Bit aguarda até se comunicar com o servidor de API para obter os metadados mais recentes. Se `Kube_Meta_Cache_TTL` não for especificado, o Amazon EKS Fargate acrescentará um valor padrão de 30 minutos para diminuir a carga no servidor de API.

Para enviar logs de processo do Fluent Bit para sua conta

Opcionalmente, você pode enviar logs de processo do Fluent Bit para o Amazon CloudWatch usando o `ConfigMap` a seguir. O envio de logs do processo Fluent Bit para o CloudWatch exige custos adicionais de ingestão e armazenamento de logs. Substitua *`region-code`* pela região da AWS em que seu cluster se encontra.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
  labels:
data:
  # Configuration files: server, input, filters and output
  # ======================================================
  flb_log_cw: "true"  # Ships Fluent Bit process logs to CloudWatch.

  output.conf: |
    [OUTPUT]
        Name cloudwatch
        Match kube.*
        region region-code
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
        auto_create_group true
```

Os logs estão no CloudWatch, na mesma região da AWS que o cluster. O nome do grupo de logs é *my-cluster*`-fluent-bit-logs`, e o nome do fluxo de logs do Fluent Bit é `fluent-bit-`*podname*`-`*pod-namespace*.

> **Note**
>
> - Os logs de processo são enviados somente quando o processo do Fluent Bit é iniciado corretamente. Caso haja uma falha ao iniciar o Fluent Bit, os logs do processo serão perdidos. Só é possível enviar logs de processo ao CloudWatch.
> - Para depurar o envio de logs de processo em sua conta, você pode aplicar o `ConfigMap` anterior para obter os logs de processo. A falha de inicialização do Fluent Bit geralmente ocorre porque o `ConfigMap` não foi analisado ou aceito pelo Fluent Bit ao iniciar.

Para interromper os logs do processo de envio do Fluent Bit

O envio de logs do processo Fluent Bit para o CloudWatch exige custos adicionais de ingestão e armazenamento de logs. Para excluir registros do processo em uma configuração `ConfigMap` existente, siga as etapas a seguir.

1. Localize o grupo de logs do CloudWatch criado automaticamente para os logs do processo do Fluent Bit do cluster do Amazon EKS, após habilitar o registro em log do Fargate. Ele segue o formato *my-cluster*`-fluent-bit-logs`.
2. Exclua os fluxos de log existentes do CloudWatch criados para os logs de processo de cada pod no grupo de logs do CloudWatch.
3. Edite `ConfigMap` e configure `flb_log_cw: "false"`.
4. Reinicie todos os pods existentes no cluster.

Testar a aplicação

1. Implante um pod de exemplo.
   a. Salve o seguinte conteúdo no seu computador, em um arquivo chamado `sample-app.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: sample-app
  namespace: same-namespace-as-your-fargate-profile
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - name: http
              containerPort: 80
```

b. Aplique o manifesto ao cluster.

```
kubectl apply -f sample-app.yaml
```

2. Visualize os logs do NGINX usando os destinos que você configurou no `ConfigMap`.

Considerações sobre tamanho

Sugerimos que você planeje até 50 MB de memória para o roteador de log. Se você espera que a aplicação gere logs com um throughput muito alto, planeje até 100 MB.

Solução de problemas

Para confirmar se o recurso de registro em log está habilitado ou desabilitado por algum motivo, como um `ConfigMap` inválido, e por que ele está inválido, verifique os eventos de pods com `kubectl describe pod` *`pod-name`* . A saída pode incluir eventos de pods que esclarecem se o registro em log está habilitado ou não, como a saída de exemplo a seguir.

```
[...]
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
                      Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
[...]
Events:
```

```
  Type     Reason           Age        From
              Message
  ----     ------           ----       ----
              -------
  Warning  LoggingDisabled  <unknown>  fargate-scheduler
               Disabled logging because aws-logging configmap was not found. configmap
"aws-logging" not found
```

Os eventos de pods são temporários, com um intervalo de tempo que depende das configurações. Você também pode visualizar as anotações de um pod usando `kubectl describe pod` *pod-name* . A anotação do pod informa se o recurso de registro em log está habilitado ou desabilitado e o motivo.

Escolher um tipo de instância de nó do Amazon EC2 ideal

O Amazon EC2 fornece uma extensa seleção de tipos de instância para nós de processamento. Cada tipo de instância oferece diferentes capacidades de computação, memória, armazenamento e rede. Cada instância também é agrupada em famílias de acordo com esses recursos. Para obter uma lista, consulte Tipos de instância disponíveis, no Guia do usuário do Amazon EC2. O Amazon EKS lança diversas variações de AMIs do Amazon EC2 para permitir suporte. Para garantir que o tipo de instância selecionado seja compatível com o Amazon EKS, considere estes critérios.

- As AMIs do Amazon EKS no momento não são compatíveis com a família `mac`.
- Arm e AMIs do Amazon EKS não aceleradas não são compatíveis com as famílias `g3`, `g4`, `inf` e `p`.
- AMIs aceleradas do Amazon EKS não têm suporte para as famílias `a`, `c`, `hpc`, `m` e `t`.
- Para instâncias baseadas em Arm, o Amazon Linux 2023 (AL2023) é somente compatível com tipos de instância que usam processadores Graviton2 ou versões mais recentes. O AL2023 não é compatível com instâncias `A1`.

Ao escolher entre tipos de instância que têm suporte pelo Amazon EKS, considere os recursos a seguir de cada tipo.

Número de instâncias em um grupo de nós

> Em geral, um número menor de instâncias maiores é melhor, especialmente se você tiver muitos DaemonSets. Cada instância requer chamadas de API para o servidor de API, portanto, quanto mais instâncias você tiver, maior a carga no servidor de APIs.

Sistema operacional

Revise os tipos de instância compatíveis com o Linux, o Windows e o Bottlerocket. Antes de criar instâncias do Windows, consulte Implantar nós do Windows em clusters do EKS.

Arquitetura de hardware

Você precisa de x86 ou Arm? Antes de implementar instâncias do Arm, analise as AMIs do Amazon EKS otimizadas para Arm Amazon Linux. Você precisa de instâncias criadas no Sistema Nitro (Linux ou Windows) ou que tenham recursos acelerados? Se você precisar de recursos acelerados, só poderá usar o Linux com o Amazon EKS.

Número máximo de pods

Como cada pod é atribuído ao seu próprio endereço IP, o número de endereços IP compatíveis com um tipo de instância é um fator para determinar o número de pods que podem ser executados em uma instância. Para determinar manualmente quantos pods um tipo de instância suporta, consulte the section called “O máximo de pods do Amazon EKS recomendado para cada tipo de instância do Amazon EC2”.

> **Note**
>
> Se estiver usando uma AMI do Amazon Linux 2 otimizada para Amazon EKS da versão `v20220406` ou mais recente, você poderá usar um novo tipo de instância sem fazer atualização para a AMI mais recente. Para essas AMIs, a AMI calcula automaticamente o valor necessário de `max-pods` se ele não estiver listado no arquivo eni-max-pods.txt. Tipos de instância atualmente em versão de demonstração podem não ter suporte pelo Amazon EKS por padrão. Valores para `max-pods` para esses tipos ainda precisam ser adicionados a `eni-max-pods.txt` na nossa AMI.

Os tipos de instância do AWS Nitro System opcionalmente oferecem suporte a bem mais endereços IP do que os tipos de instância que não são do Nitro System. Contudo, nem todos os endereços IP atribuídos a uma instância estão disponíveis para pods. Para atribuir um número significativamente maior de endereços IP às suas instâncias, você deve ter a versão `1.9.0` ou superior do complemento CNI da Amazon VPC instalado em seu cluster e configurado adequadamente. Para obter mais informações, consulte the section called “Aumento dos endereços IP”. Para atribuir o maior número de endereços IP às suas instâncias, a versão `1.10.1` ou superior do complemento CNI da Amazon VPC deverá estar instalada em seu cluster e o cluster deverá ser implantado com a família `IPv6`.

Família IP

Você pode usar qualquer tipo de instância compatível ao usar a família `IPv4` para um cluster, o que permite ao cluster atribuir endereços `IPv4` privados aos pods e serviços. Porém, se você deseja utilizar a família `IPv6` para o seu cluster, deve usar tipos de instância do AWS Nitro System ou tipos de instância de bare metal. Somente o `IPv4` é compatível com instâncias do Windows. O cluster deve executar a versão `1.10.1` ou posterior do complemento CNI da Amazon VPC. Para obter mais informações sobre o uso de `IPv6`, consulte the section called "IPv6".

Versão do complemento Amazon VPC CNI que você está executando

A versão mais recente do plug-in Amazon VPC CNI para Kubernetes é compatível com estes tipos de instância. Talvez seja necessário atualizar a versão do complemento Amazon VPC CNI para poder aproveitar os tipos de instância mais recentes com suporte. Para obter mais informações, consulte the section called "CNI da Amazon VPC". A versão mais recente suporta os recursos mais recentes para serem usados com o Amazon EKS. As versões anteriores não suportam todos os recursos. Você pode visualizar os recursos suportados por diferentes versões no Changelog no GitHub.

Região da AWS em que você está criando seus nós

Nem todos os tipos de instâncias estão disponíveis em todas as regiões da AWS.

Se você estiver usando grupos de segurança para pods

Se você estiver usando grupos de segurança para pods, apenas determinados tipos de instância serão compatíveis. Para obter mais informações, consulte the section called "Grupos de segurança para os pods".

O máximo de pods do Amazon EKS recomendado para cada tipo de instância do Amazon EC2

Como cada pod é atribuído ao seu próprio endereço IP, o número de endereços IP compatíveis com um tipo de instância é um fator para determinar o número de pods que podem ser executados em uma instância. O Amazon EKS fornece um script que você pode baixar e executar para determinar o número máximo de pods do Amazon EKS recomendado a serem executados em cada tipo de instância. O script usa atributos de hardware de cada instância e opções de configuração para determinar o número máximo de pods. Você pode usar o número retornado nessas etapas para

habilitar recursos como atribuição de endereços IP a pods de uma sub-rede diferente da instância e aumentar significativamente o número de endereços IP da sua instância. Se estiver utilizando um grupo de nós gerenciados com vários tipos de instâncias, use um valor que funcione para todos os tipos de instância.

1. Baixe um script que você possa usar para calcular o número máximo de pods para cada tipo de instância.

```
curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/templates/
al2/runtime/max-pods-calculator.sh
```

2. Marque o script como executável no computador.

```
chmod +x max-pods-calculator.sh
```

3. Execute o script, substituindo *m5.large* pelo tipo de instância que você planeja implantar e *1.9.0-eksbuild.1* pela versão do complemento do Amazon VPC CNI. Para determinar a versão do complemento, consulte os procedimentos de atualização em Atribuir IPs a pods com o Amazon VPC CNI.

```
./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
```

   Veja um exemplo de saída abaixo.

```
29
```

   Você pode adicionar as seguintes opções ao script para ver o máximo de pods compatíveis ao usar funcionalidades opcionais.

   - `--cni-custom-networking-enabled`: use essa opção quando quiser atribuir endereços IP de uma sub-rede diferente da sub-rede da sua instância. Para obter mais informações, consulte the section called “Redes personalizadas”. Adicionar essa opção ao script anterior com os mesmos valores do exemplo gera `20`.
   - `--cni-prefix-delegation-enabled`: use esta opção quando quiser atribuir significativamente mais endereços IP a cada interface de rede elástica. Esse recurso requer uma instância do Amazon Linux executada no Sistema Nitro e na versão `1.9.0` ou superior do complemento CNI da Amazon VPC. Para obter mais informações, consulte the section called “Aumento dos endereços IP”. Adicionar essa opção ao script anterior com os mesmos valores do exemplo gera `110`.

Você também pode executar o script com a opção `--help` para ver todas as opções disponíveis.

> **Note**
>
> O script da calculadora max de pods limita o valor de retorno a `110` com base nos limites de escalabilidade do Kubernetes e nas configurações recomendadas. Se o seu tipo de instância tiver mais de 30 vCPUs, esse limite aumentará para `250`, um número baseado em testes internos da equipe de escalabilidade do Amazon EKS. Para obter mais informações, consulte a publicação no blog Plug-in do Amazon VPC CNI aumenta os limites de pods por nó.

Considerações para o Modo Automático do EKS

O Modo Automático do EKS limita o número de pods nos nós inferior:

- À capacidade fixa de 110 pods
- Ao resultado do cálculo do máximo de pods descrito acima.

Crie nós com imagens otimizadas predefinidas

Você pode implantar os nós com Imagens de máquina da Amazon (AMIs) pré-criadas e otimizadas para o Amazon EKS, ou com suas próprias AMIs personalizadas quando usar grupos de nós gerenciados ou nós autogerenciados. Se você estiver executando nós híbridos, consulte the section called “Preparar o sistema operacional”. Para obter informações sobre cada tipo de AMI otimizada para o Amazon EKS, consulte um dos tópicos a seguir. Para instruções sobre como criar a sua própria AMI personalizada, consulte the section called “Desenvolvimento de uma AMI do Amazon Linux personalizada e otimizada para o EKS”.

Com o Modo Automático do Amazon EKS, o EKS gerencia a instância do EC2, incluindo a seleção e a atualização da AMI.

Tópicos

- Guia para os recursos de transição das AMIs do AL2 e aceleradas para AL2
- Criar nós com AMIs do Amazon Linux otimizadas
- Criar nós com AMIs otimizadas do Bottlerocket
- Criar nós com AMIs otimizadas do Ubuntu Linux

- Criar nós com AMIs otimizadas do Windows

Guia para os recursos de transição das AMIs do AL2 e aceleradas para AL2

> **Warning**
>
> A partir de 26 de novembro de 2025, a Amazon EKS não publica mais AMIs do Amazon Linux 2 (AL2) otimizadas para o EKS. As AMIs baseadas no AL2023 e no Bottlerocket para o Amazon EKS estão disponíveis para todas as versões compatíveis com Kubernetes, incluindo a versão 1.33 e as versões posteriores.

A AWS encerrará o suporte para AMIs otimizadas e aceleradas para AL2 do EKS a partir de 26 de novembro de 2025. Embora você possa continuar usando as AMIs do AL2 do EKS após a data de fim do suporte (EOS) (26 de novembro de 2025), o EKS não lançará mais nenhuma nova versão do Kubernetes ou atualizações das AMIs do AL2, incluindo versões menores, patches e correções de bugs após essa data. Recomendamos atualizar para as AMIs do Amazon Linux 2023 (AL2023) ou do Bottlerocket:

- O AL2023 permite uma abordagem segura por padrão com políticas de segurança pré-configuradas, SELinux no modo permissivo, modo somente IMDSv2 ativado por padrão, tempos de inicialização otimizados e gerenciamento aprimorado de pacotes para maior segurança e performance, sendo adequado para infraestrutura que exige personalizações significativas, como acesso direto no nível do sistema operacional ou mudanças extensivas nos nós. Para saber mais informações, consulte Perguntas frequentes do AL2023 ou confira nossas orientações de migração detalhadas em the section called “Atualizar para AL2023”.
- O Bottlerocket possibilita uma maior segurança, tempos de inicialização mais rápidos e superfície de ataque menor para maior eficiência com seu design específico e otimizado para contêineres, sendo adequado para abordagens nativas de contêineres com o mínimo de personalizações de nós. Para saber mais informações, consulte Perguntas frequentes do Bottlerocket ou confira nossas orientações de migração detalhadas em the section called “Bottlerocket”.

Como alternativa, é possível the section called “Desenvolvimento de uma AMI do Amazon Linux personalizada e otimizada para o EKS” até a data de fim do suporte (EOS, na sigla em inglês) em 26

de novembro de 2025. Além disso, você pode criar uma AMI personalizada com uma instância base do Amazon Linux 2 até a data de EOS do Amazon Linux 2, que é 30 de junho de 2026.

Perguntas frequentes sobre migração e suporte

Como faço para migrar do meu AL2 para uma AMI do AL2023?

Recomendamos criar e implementar um plano de migração que inclua testes completos da workload da aplicação e procedimentos de reversão documentados e, em seguida, seguir as instruções passo a passo na Atualização do Amazon Linux 2 para o Amazon Linux 2023 na documentação oficial do EKS.

Posso criar uma AMI do AL2 personalizada após a data de fim do suporte (EOS) do EKS para AMIs do AL2 otimizadas para EKS?

Embora seja recomendado migrar para AMIs otimizadas para EKS com suporte oficial e publicadas para AL2023 ou Bottlerocket, é possível criar AMIs personalizadas otimizadas e aceleradas para AL2 até a data EOS da AMI do AL2 (26 de novembro de 2025). Como alternativa, é possível criar uma AMI personalizada com uma instância base do Amazon Linux 2 até a data de EOS do Amazon Linux 2 (30 de junho de 2026). Para obter instruções passo a passo sobre como criar uma AMI personalizada otimizada para AL2 e acelerada para AL2, consulte Criar uma AMI personalizada do Amazon Linux na documentação oficial do EKS.

A política de suporte à versão do Kubernetes do EKS se aplica às distribuições Amazon Linux?

Não. A data de EOS para AMIs otimizadas para AL2 e aceleradas para AL2 do EKS é independente dos cronogramas de suporte padrão e estendido para as versões do Kubernetes do EKS. Você precisa migrar para AL2023 ou Bottlerocket mesmo se estiver usando o suporte estendido do EKS.

Como a mudança de cgroupv1 para cgroupv2 afeta minha migração?

A comunidade Kubernetes moveu o `cgroupv1` suporte (usado pelo AL2) para o modo de manutenção, o que significa que nenhum novo recurso será adicionado e somente correções de segurança críticas e de bugs importantes serão fornecidas. Para adotar `cgroupv2` no Kubernetes, é necessário garantir a compatibilidade entre o sistema operacional, o kernel, o runtime do contêiner e os componentes do Kubernetes. Isso requer uma distribuição Linux que habilite `cgroupv2` por padrão, como AL2023, Bottlerocket, Red Hat Enterprise Linux (RHEL) 9+, Ubuntu 22.04+ ou Debian 11+. Essas distribuições são fornecidas com versões de kernel ≥ 5.8, que é o requisito mínimo para o suporte a `cgroupv2` no Kubernetes. Para saber mais sobre os grupos, consulte Sobre o cgroup v2.

O que eu faço se eu precisar do Neuron na minha AMI do AL2 personalizada?

Não é possível executar aplicações totalmente baseadas em Neuron de forma nativa em AMIs baseadas no AL2. Para utilizar o AWS Neuron em uma AMI do AL2, é necessário conteinerizar as aplicações usando um contêiner compatível com o Neuron com uma distribuição Linux não AL2 (por exemplo, Ubuntu 22.04, Amazon Linux 2023 etc.) e depois implantar esses contêineres em uma AMI baseada em AL2 que tenha o Neuron Driver (`aws-neuronx-dkms`) instalado.

Eu deveria migrar para uma instância base “bare” do Amazon Linux 2 após 26 de novembro de 2025, que é a data de EOS da AMI do AL2 otimizada para o EKS?

A migração para uma instância base “bare” do Amazon Linux 2 precisa das otimizações específicas, das configurações de runtime do contêiner e das personalizações fornecidas pelas AMIs oficiais do AL2 otimizadas para o EKS e do AL2 aceleradas. Em vez disso, caso você precise continuar utilizando uma solução baseada no AL2, recomendamos criar uma AMI personalizada usando as receitas de AMI do EKS apresentadas em the section called “Desenvolvimento de uma AMI do Amazon Linux personalizada e otimizada para o EKS” ou no repositório Amazon EKS AMI Build Specification. Isso assegura a compatibilidade com as workloads atuais e contempla atualizações do kernel do AL2 até a EOS do Amazon Linux 2, em 30 de junho de 2026.

Ao criar uma AMI do AL2 personalizada usando o repositório destinado à AMI do EKS no GitHub após a data de EOS da AMI do AL2 otimizada para o EKS, que é 26 de novembro de 2025, qual suporte estará disponível para pacotes de repositórios como amzn2-core e amzn2extra-docker?

A receita de AMI do EKS apresentada no repositório Amazon EKS AMI Build Specification obtém pacotes via YUM de softwares padrão do Amazon Linux 2, tais como amzn2-core e amzn2extra-docker. Após a data de EOS da AMI do AL2 otimizada para o EKS em 26 de novembro de 2025, este software continuará a ter suporte até a data de EOS mais abrangente do Amazon Linux 2, que é 30 de junho de 2026. Lembre-se de que o suporte está limitado a atualizações de kernel durante esse período, o que significa que você precisará gerenciar e aplicar manualmente outras atualizações de pacotes, patches de segurança e quaisquer dependências que não sejam do kernel para manter a segurança e a compatibilidade.

Por que aplicações Java que usam versões mais antigas do JDK8 no Amazon EKS com o AL2023 podem sofrer exceções de memória insuficiente (OOM, na sigla em inglês) e reinicializações de pods? Quais são as formas de resolver isso?

Ao serem executadas em nós do Amazon EKS com o AL2023, aplicações Java que dependem de versões do JDK 8 anteriores à versão `jdk8u372` podem causar exceções de OOM e reinicializações

de pods, pois a JVM não é compatível com `cgroupv2`. Este problema surge especificamente da incapacidade da JVM de detectar os limites de memória do contêiner usando `cgroupv2`, que é o padrão no Amazon Linux 2023. Consequentemente, a JVM baseia a alocação do heap na memória total do nó, e não no limite configurado para o pod. Isso ocorre porque o `cgroupv2` altera o local de armazenamento dos dados de limite de memória, fazendo com que versões mais antigas do Java leiam incorretamente a memória disponível e assumam os recursos do nível do nó. Algumas opções possíveis incluem:

- Atualização da versão do JDK: atualizar para a versão `jdk8u372` ou versões posteriores, ou para uma versão mais recente do JDK com suporte total ao `cgroupv2` pode resolver este problema. Para obter uma lista de versões do Java compatíveis que fornecem suporte total ao `cgroupv2`, consulte About cgroup v2.
- Criação de uma AMI personalizada: se você precisar continuar utilizando uma solução baseada no AL2, poderá criar uma AMI personalizada baseada no AL2 até 26 de novembro de 2025, usando the section called “Desenvolvimento de uma AMI do Amazon Linux personalizada e otimizada para o EKS” ou o repositório Amazon EKS AMI Build Specification. Por exemplo, é possível criar uma AMI v1.33 baseada no AL2 até 26 de novembro de 2025. O Amazon EKS fornecerá AMIs baseadas no AL2 até a data de EOS do AL2 no EKS, que é 26 de novembro de 2025. Após 26 de novembro de 2025, que é a data de EOS, você precisará criar sua própria AMI.
- Habilitação do cgroupv1: se você precisar continuar utilizando o `cgroupv1`, poderá habilitar o `cgroupv1` em uma AMI do AL2023 otimizada para o EKS. Para isso, execute `sudo grubby --update-kernel=ALL --args="systemd.unified_cgroup_hierarchy=0"` e reinicie o sistema (por exemplo, a instância do EC2 ou o nó que executa o Amazon Linux 2023). Isso modificará os parâmetros de inicialização do sistema (por exemplo, adicionando o parâmetro de kernel “systemd.unified_cgroup_hierarchy=0” à configuração do GRUB, o que instrui o systemd a usar a hierarquia legada do `cgroupv1`) e habilitará o `cgroupv1`. Lembre-se de que, ao executar este comando da ferramenta grubby, você está reconfigurando o kernel para iniciar com o `cgroupv1` habilitado e o `cgroupv2` desabilitado. Somente uma dessas versões de cgroup é usada para o gerenciamento ativo de recursos em um nó. Este procedimento não equivale a executar `cgroupv2` com suporte de retrocompatibilidade para a API do `cgroupv1`.

> **Warning**
>
> Não recomendamos o uso contínuo do `cgroupv1`. Em vez disso, recomendamos a migração para o `cgroupv2`. A comunidade do Kubernetes colocou o suporte ao `cgroupv1` (utilizado pelo AL2) em modo de manutenção, o que significa que nenhum novo recurso ou atualização

será adicionado, sendo fornecidas apenas correções de segurança críticas e de falhas graves. A remoção total do suporte ao `cgroupv1` é esperada em uma versão futura, embora uma data específica para essa remoção ainda não tenha sido anunciada. Se você enfrentar problemas com o `cgroupv1`, a AWS não poderá fornecer suporte e recomendará que você atualize para o `cgroupv2`.

Compatibilidade e versões

Versões do Kubernetes compatíveis com AMIs do AL2

A versão 1.32 do Kubernetes é a última versão para a qual o Amazon EKS lançará AMIs do Amazon Linux 2 (AL2). Para as versões do Kubernetes com suporte até 1.32, o EKS continuará a lançar AMIs do AL2 (AL2_ARM_64, AL2_x86_64) e AMIs aceleradas para AL2 (AL2_x86_64_GPU) até 26 de novembro de 2025. Após essa data, o EKS deixará de lançar AMIs otimizadas e aceleradas para AL2 para todas as versões do Kubernetes. Observe que data de EOS para AMIs otimizadas para AL2 e aceleradas para AL2 do EKS é independente dos cronogramas de suporte padrão e estendido para as versões do Kubernetes do EKS.

Comparação dos drivers e versões do kernel do Linux compatíveis para AMIs do AL2, AL2023 e Bottlerocket

Componente	AMI do AL2 otimizada para o EKS	AMI do AL2023 otimizada para o EKS	AMI do Bottlerocket otimizada para o EKS
Compatibilidade do sistema operacional base	RHEL7/CentOS 7	Fedora/CentOS 9	N/D
Driver de modo de usuário do CUDA	12.x	12.x e 13.x	12.x e 13.x
Driver da GPU NVIDIA	R570	R580	R570 e R580
Driver do AWS Neuron	2.20 e versões posteriores	2.20 e versões posteriores	2.20 e versões posteriores
Kernel do Linux	5.10	6.1, 6.12	6.1, 6.12

Para obter mais informações sobre a compatibilidade do driver da NVIDIA e do CUDA, consulte a documentação da NVIDIA.

Compatibilidade do AWS Neuron com AMIs do AL2

A partir do AWS Neuron versão 2.20, o runtime do Neuron (`aws-neuronx-runtime-lib`) usado pelas AMIs baseadas em AL do EKS não oferece mais suporte ao Amazon Linux 2 (AL2). O driver do Neuron (`aws-neuronx-dkms`) agora é o único pacote do AWS Neuron que oferece suporte ao Amazon Linux 2. Isso significa que não é possível executar aplicações com tecnologia Neuron nativamente em uma AMI baseada no AL2. Para configurar o Neuron em AMIs do AL2023, consulte o guia de Configuração do AWS Neuron.

Compatibilidade do Kubernetes com AMIs do AL2

A comunidade Kubernetes moveu o suporte ao `cgroupv1` (usado pelo AL2) para o modo de manutenção. Isso significa que nenhum novo recurso será adicionado e somente correções de segurança críticas e de bugs importantes serão fornecidas. Os recursos do Kubernetes que dependem do cgroupv2, como MemoryQoS e isolamento aprimorado de recursos, não estão disponíveis no AL2. Além disso, a versão 1.32 do Kubernetes do Amazon EKS foi a última versão a oferecer suporte às AMIs do AL2. Para manter a compatibilidade com as versões mais recentes do Kubernetes, recomendamos migrar para o AL2023 ou o Bottlerocket, que habilitam o `cgroupv2` por padrão.

Compatibilidade de versões do Linux com AMIs do AL2

O suporte ao Amazon Linux 2 (AL2) será oferecido pela AWS até a data de fim do suporte (EOS) em 30 de junho de 2026. No entanto, à medida que o AL2 envelheceu, o suporte da comunidade Linux em geral a novas aplicações e funcionalidades se tornou mais limitado. As AMIs do AL2 são baseadas no kernel 5.10 do Linux, enquanto o AL2023 usa o kernel 6.1. Ao contrário do AL2023, o AL2 recebe suporte limitado da comunidade Linux em geral. Isso significa que muitos pacotes e ferramentas do Linux upstream precisam ser adaptados para funcionar com a versão mais antiga do kernel do AL2. Alguns recursos modernos e melhorias de segurança do Linux não estão disponíveis devido ao kernel mais antigo e muitos projetos de código aberto oferecem suporte limitado ou não oferecem suporte a versões mais antigas do kernel, como a 5.10.

Pacotes obsoletos não incluídos no AL2023

Alguns dos pacotes mais comuns que não estão incluídos ou que foram alterados no AL2023 incluem:

- Alguns pacotes binários de fontes no Amazon Linux 2 não estão mais disponíveis no Amazon Linux 2023
- Alterações na forma como o Amazon Linux oferece suporte a diferentes versões de pacotes (por exemplo, sistema amazon-linux-extras) no AL2023
- Não há suporte a pacotes extras para Enterprise Linux (EPEL) no AL2023
- Não há suporte a aplicações de 32 bits no AL2023

Para saber mais, consulte Comparação entre o AL2 e o AL2023.

Comparação da validação FIPS entre AL2, AL2023 e Bottlerocket

O Amazon Linux 2 (AL2), Amazon Linux 2023 (AL2023) e o Bottlerocket oferecem suporte à conformidade com os Padrões Federais de Processamento de Informações (FIPS).

- O AL2 é certificado sob o FIPS 140-2 e o AL2023 é certificado sob o FIPS 140-3. Para ativar o modo FIPS no AL2023, instale os pacotes necessários na sua instância do Amazon EC2 e siga as etapas de configuração usando as instruções em Habilitar o modo FIPS no AL2023. Para saber mais, consulte as Perguntas frequentes do AL2023.
- O Bottlerocket fornece variantes específicas para FIPS que restringem os componentes do kernel e do espaço do usuário ao uso de módulos criptográficos que foram submetidos ao Programa de Validação de Módulos Criptográficos do FIPS 140-3.

Log de alterações do driver e versões da AMI do EKS

Para obter uma lista completa de todos os componentes da AMI do EKS e suas versões, consulte as Notas de lançamento da AMI do Amazon EKS no GitHub.

Criar nós com AMIs do Amazon Linux otimizadas

O Amazon Elastic Kubernetes Service (Amazon EKS) fornece imagens de máquina da Amazon (AMIs) especializadas e otimizadas para a execução de nós de processamento do Kubernetes. Estas AMIs do Amazon Linux (AL) otimizadas para o EKS são configuradas previamente com componentes essenciais, como `kubelet`, o AWS IAM Authenticator e o `containerd`, para garantir a integração sem complicações e a segurança em seus clusters. Este guia apresenta detalhes sobre as versões de AMI disponíveis e fornece uma visão geral das opções especializadas para computação acelerada e para arquiteturas baseadas em ARM.

Considerações

- É possível acompanhar os eventos de segurança ou privacidade do Amazon Linux no centro de segurança do Amazon Linux, selecionando a guia da versão desejada. Você também pode assinar o feed RSS aplicável. Os eventos de segurança e privacidade incluem uma visão geral do problema, quais pacotes são afetadas e como atualizar suas instâncias para corrigir o problema.
- Antes de fazer a implantação de uma AMI baseada em ARM ou acelerada, analise as informações contidas em AMIs do Amazon Linux aceleradas e otimizadas para o Amazon EKS e the section called “AMIs do Amazon Linux baseadas em ARM e otimizadas para o Amazon EKS”.
- As instâncias P2 do Amazon EC2 não são compatíveis com o Amazon EKS porque exigem a versão 470 ou anterior do driver `NVIDIA`.
- Todos os grupos de nós gerenciados recém-criados em clusters na versão `1.30` ou mais recente terão como padrão automático o uso do AL2023 como sistema operacional do nó.

AMIs do Amazon Linux aceleradas e otimizadas para o Amazon EKS

As AMIs do Amazon Linux (AL) aceleradas e otimizadas para o Amazon EKS são baseadas nas AMIs padrão do Amazon Linux otimizadas para o EKS. Elas são configuradas para servir como imagens opcionais para nós do Amazon EKS para serem compatíveis com workloads baseadas em GPU, no Inferentia e no Trainium.

Para obter mais informações, consulte the section called “Uso de AMIs do Linux com suporte a GPU para o EKS”.

AMIs do Amazon Linux baseadas em ARM e otimizadas para o Amazon EKS

As instâncias Arm oferecem uma economia significativa para aplicações de expansão e baseadas no Arm, como servidores Web, microsserviços conteinerizados, frotas de armazenamento em cache e datastores distribuídos. Ao adicionar nós do Arm ao cluster, revise as considerações a seguir.

- Se o cluster tiver sido implantado antes de 17 de agosto de 2020, será necessário fazer uma atualização única dos manifestos essenciais do complemento de cluster. Isso serve para que o Kubernetes possa extrair a imagem correta para cada arquitetura de hardware em uso em seu cluster. Para obter mais informações sobre como atualizar complementos do cluster, consulte the section called “Etapa 1: preparar-se para o upgrade”. Se você implantou o cluster depois de 17 de agosto de 2020, então o plug-in CNI do CoreDNS, `kube-proxy` e Amazon VPC para complementos do Kubernetes já tem capacidade para várias arquiteturas.

- As aplicações implantadas em nós do Arm devem ser compiladas para Arm.
- Se tiver DaemonSets implantados em um cluster existente, ou se quiser implantá-los em um novo cluster em que também queira implantar nós do Arm, verifique se o DaemonSet pode ser executado em todas as arquiteturas de hardware do cluster.
- Você pode executar grupos de nós do Arm e grupos de nós x86 no mesmo cluster. Caso o faça, considere implantar imagens de contêiner de várias arquiteturas em um repositório de contêineres, como o Amazon Elastic Container Registry, e depois adicionar seletores de nós aos manifestos para que o Kubernetes saiba em que arquitetura de hardware um pod pode ser implantado. Para obter mais informações, consulte Enviar uma imagem de várias arquiteturas no Guia do usuário do Amazon ECR e a publicação de blog Introducing multi-architecture container images for Amazon ECR.

Mais informações

Para obter mais informações sobre o uso de AMIs do Amazon Linux otimizadas para o Amazon EKS, consulte as seguintes seções:

- Para usar o Amazon Linux com grupos de nós gerenciados, consulte the section called “Grupos de nós gerenciados”.
- Para iniciar nós autogerenciados do Amazon Linux, consulte the section called “Obter os IDs mais recentes”.
- Para obter informações sobre versões, consulte the section called “Obter informações de versão”.
- Para recuperar os IDs mais recentes das AMIs do Amazon Linux otimizadas para o Amazon EKS, consulte the section called “Obter os IDs mais recentes”.
- Para acessar os scripts de código aberto utilizados na criação das AMIs otimizadas para o Amazon EKS, consulte the section called “Desenvolvimento de uma AMI do Amazon Linux personalizada e otimizada para o EKS”.

Atualização do Amazon Linux 2 para o Amazon Linux 2023

> **Warning**
>
> A partir de 26 de novembro de 2025, a Amazon EKS não publica mais AMIs do Amazon Linux 2 (AL2) otimizadas para o EKS. As AMIs baseadas no AL2023 e no Bottlerocket para

o Amazon EKS estão disponíveis para todas as versões compatíveis com Kubernetes, incluindo a versão 1.33 e as versões posteriores.

O AL2023 é um sistema operacional baseado no Linux que foi projetado para fornecer um ambiente seguro, estável e de alta performance para as aplicações em nuvem. Ele corresponde à próxima geração do Amazon Linux da Amazon Web Services e está disponível em todas as versões do Amazon EKS com suporte.

O AL2023 oferece diversas melhorias em relação ao AL2. Para obter uma comparação completa, consulte Comparing AL2 and Amazon Linux 2023 no Guia do usuário do Amazon Linux 2023. Vários pacotes foram adicionados, atualizados e removidos em relação ao AL2. É altamente recomendável testar as aplicações com o AL2023 antes de realizar a atualização. Para obter uma lista de todas as alterações de pacote no AL2023, consulte Package changes in Amazon Linux 2023 nas Notas de lançamento do Amazon Linux 2023.

Além dessas alterações, você deve estar ciente do seguinte:

- O AL2023 introduz um novo processo de inicialização do nó `nodeadm` que usa um esquema de configuração YAML. Se estiver usando grupos de nós autogerenciados ou uma AMI com um modelo de inicialização, será necessário fornecer, de forma explícita, metadados de cluster adicionais ao criar um novo grupo de nós. Veja a seguir um exemplo dos parâmetros mínimos necessários, em que `apiServerEndpoint`, `certificateAuthority` e `cidr` do serviço passaram a ser necessários:

  ```
  ---
  apiVersion: node.eks.aws/v1alpha1
  kind: NodeConfig
  spec:
    cluster:
      name: my-cluster
      apiServerEndpoint: https://example.com
      certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
      cidr: 10.100.0.0/16
  ```

  No AL2, os metadados desses parâmetros eram revelados na chamada de API `DescribeCluster` do Amazon EKS. Com o AL2023, esse comportamento foi alterado, uma vez que a chamada de API adicional corre o risco de sofrer controle de utilização durante grandes aumentos de escala vertical para nós. Essa alteração não afetará você se estiver usando grupos

de nós gerenciados sem um modelo de inicialização ou se estiver usando o Karpenter. Para obter mais informações sobre `certificateAuthority` e sobre o serviço de `cidr`, consulte `DescribeCluster` na Referência de API do Amazon EKS.

- Para o AL2023, o `nodeadm` também altera o formato para aplicar parâmetros ao `kubelet` para cada nó usando `NodeConfigSpec`. No AL2, isso era feito com o parâmetro `--kubelet-extra-args`. Isso é comumente usado para adicionar rótulos e taints aos nós. Um exemplo abaixo mostra a aplicação de `maxPods` e `--node-labels` ao nó.

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: test-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      maxPods: 110
    flags:
      - --node-labels=karpenter.sh/capacity-type=on-demand,karpenter.sh/nodepool=test
```

- A versão `1.16.2` ou as versões posteriores do plug-in CNI da Amazon VPC são necessárias para o AL2023.
- O AL2023 requer `IMDSv2` por padrão. O `IMDSv2` tem diversos benefícios que ajudam a aprimorar a postura de segurança. Ele usa um método de autenticação orientado por sessão que requer a criação de um token secreto em uma solicitação HTTP PUT simples para iniciar a sessão. O token de uma sessão pode ter validade de um segundo a seis horas. Para obter mais informações sobre como fazer a transição do `IMDSv1` para o `IMDSv2`, consulte Transition to using Instance Metadata Service Version 2 e Get the full benefits of IMDSv2 and disable IMDSv1 across your AWS infrastructure. Se desejar usar o `IMDSv1`, você ainda poderá fazê-lo ao substituir manualmente as configurações usando as propriedades de inicialização da opção de metadados da instância.

> **Note**
>
> Para `IMDSv2` com AL2023, a contagem de saltos padrão para os grupos de nós gerenciados pode variar:

- Quando não estiver usando um modelo de lançamento, o padrão será definido como 1. Isso significa que os contêineres não terão acesso às credenciais do nó usando o IMDS. Se você precisar de acesso de contêiner às credenciais do nó, ainda poderá obtê-lo usando um modelo de execução personalizado do Amazon EC2.
- Ao usar uma AMI personalizada em um modelo de execução, o padrão `HttpPutResponseHopLimit` será definido como 2. É possível substituir manualmente o `HttpPutResponseHopLimit` no modelo de execução.

Como alternativa, é possível usar a Identidade de Pods do Amazon EKS para fornecer credenciais em vez de usar o `IMDSv2`.

- O AL2023 apresenta a próxima geração de hierarquia de grupo de controle unificada (`cgroupv2`). A hierarquia `cgroupv2` é usada para implementar um runtime de contêiner e usar `systemd`. Embora o AL2023 ainda inclua códigos que podem fazer o sistema funcionar ao usar `cgroupv1`, esta não é uma configuração recomendada ou com suporte. Essa configuração será completamente removida em uma versão principal futura do Amazon Linux.
- O `eksctl` versão `0.176.0` ou superior é necessário para o `eksctl` oferecer suporte ao AL2023.

Para grupos de nós gerenciados anteriormente existentes, é possível realizar uma atualização no local ou uma atualização azul/verde, dependendo de como você está usando um modelo de inicialização:

- Caso esteja usando uma AMI personalizada com um grupo de nós gerenciados, você poderá realizar uma atualização local ao alterar o ID da AMI no modelo de inicialização. Você deve garantir que as aplicações e quaisquer dados do usuário sejam transferidos para o AL2023 antes de executar essa estratégia de atualização.
- Se você estiver usando grupos de nós gerenciados com o modelo de inicialização padrão ou com um modelo de inicialização personalizado que não especifica o ID da AMI, será necessário fazer upgrade usando uma estratégia azul/verde. Uma atualização azul/verde, normalmente, é mais complexa e envolve a criação de um grupo de nós totalmente novo no qual você especificaria o AL2023 como o tipo de AMI. O novo grupo de nós precisará ser configurado cuidadosamente para garantir que todos os dados personalizados do grupo de nós do AL2 sejam compatíveis com o novo sistema operacional. Depois que o novo grupo de nós tiver sido testado e validado com as aplicações, os pods poderão ser migrados do grupo de nós antigo para o novo grupo de nós. Depois que a migração for concluída, você poderá excluir o grupo de nós antigo.

Se estiver usando o Karpenter e desejar usar o AL2023, você precisará modificar o campo `EC2NodeClass` `amiFamily` com o AL2023. Por padrão, a Oscilação está habilitada no Karpenter. Isso significa que assim que o campo `amiFamily` for alterado, o Karpenter atualizará automaticamente os nós de processamento para a AMI mais recente, quando disponível.

Recuperar informações da versão da AMI do Amazon Linux

O versionamento das AMIs do Amazon Linux otimizadas para o Amazon EKS é feito com base na versão do Kubernetes e na data de lançamento da AMI, no seguinte formato:

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

Cada versão da AMI inclui várias versões do kubelet, do kernel do e do containerd. As AMIs aceleradas também incluem várias versões do driver NVIDIA. Você pode encontrar essas informações de versão no Changelog no GitHub.

Recuperar IDs de AMI do Amazon Linux recomendadas

Ao implantar nós, é possível especificar um ID para uma imagem de máquina da Amazon (AMI) pré-compilada e otimizada para o Amazon EKS. Para recuperar um ID de AMI que se ajuste à configuração desejada, consulte a API AWS Systems Manager Parameter Store. O uso dessa API elimina a necessidade de pesquisar manualmente IDs de AMIs otimizadas para o Amazon EKS. Para obter mais informações, consulte GetParameter. A entidade principal do IAM que você usou deve ter a permissão `ssm:GetParameter` do IAM para recuperar os metadados da AMI otimizada para o Amazon EKS.

Você pode recuperar o ID de imagem da mais recente AMI do Amazon Linux recomendada otimizada para o Amazon EKS com o comando a seguir, que usa o subparâmetro `image_id`. Faça as seguintes modificações no comando, conforme necessário, e execute o comando atualizado:

- Substitua `<kubernetes-version>` por uma versão compatível do Amazon EKS.
- Substitua *`ami-type`* por uma das seguintes opções. Para obter mais informações sobre os tipos de instâncias do Amazon EC2, consulte Tipos de instância do Amazon EC2.
  - Use *`amazon-linux-2023/x86_64/standard`* para instâncias baseadas no Amazon Linux 2023 (AL2023) x86.
  - Use *`amazon-linux-2023/arm64/standard`* para instâncias ARM do AL2023, como as instâncias baseadas no AWS Graviton.

  - Use *amazon-linux-2023/x86_64/nvidia* para as instâncias x86 baseadas do NVIDIA do AL2023 aprovadas mais recentemente.
  - Use *amazon-linux-2023/arm64/nvidia* para as instâncias arm64 do NVIDIA do AL2023 aprovadas mais recentemente.
  - Use *amazon-linux-2023/x86_64/neuron* para as instâncias mais recentes do AL2023 AWS Neuron.
- Substitua <region-code> por uma região da AWS compatível com o Amazon EKS para a qual você deseja o ID da AMI.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/<kubernetes-version>/<ami-
type>/recommended/image_id \
    --region <region-code> --query "Parameter.Value" --output text
```

Aqui está um exemplo de comando após as substituições do espaço reservado terem sido feitas.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.31/amazon-
linux-2023/x86_64/standard/recommended/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

Veja abaixo um exemplo de saída.

```
ami-1234567890abcdef0
```

Desenvolvimento de uma AMI do Amazon Linux personalizada e otimizada para o EKS

> **Warning**
>
> A partir de 26 de novembro de 2025, a Amazon EKS não publica mais AMIs do Amazon Linux 2 (AL2) otimizadas para o EKS. As AMIs baseadas no AL2023 e no Bottlerocket para o Amazon EKS estão disponíveis para todas as versões compatíveis com Kubernetes, incluindo a versão 1.33 e as versões posteriores.

O Amazon EKS fornece scripts de desenvolvimento de código aberto no repositório Amazon EKS AMI Build Specification que você pode usar para visualizar as configurações do `kubelet`, do runtime e do AWS IAM Authenticator for Kubernetes, além de criar sua própria AMI baseada no AL do zero.

Este repositório contém o script de inicialização especializado para AL2 e a ferramenta nodeadm para AL2023, que são executados no momento da inicialização. Esses scripts configuram os dados de certificado da instância, o endpoint do ambiente de gerenciamento, o nome do cluster e muito mais. Os scripts são considerados a fonte de verdade para os desenvolvimentos de AMIs otimizadas para o Amazon EKS, portanto, você pode acompanhar o repositório do GitHub para monitorar as alterações em nossas AMIs.

Ao criar AMIs personalizadas usando AMIs otimizadas para o EKS como base, não é recomendado nem suportado executar uma atualização do sistema operacional (por exemplo, `dnf upgrade`) ou atualizar qualquer um dos pacotes do Kubernetes ou de GPU incluídos nas AMIs otimizadas para o EKS, pois isso pode comprometer a compatibilidade entre os componentes. Se você optar por atualizar o sistema operacional ou os pacotes incluídos nas AMIs otimizadas para o EKS, é recomendável realizar testes completos em um ambiente de desenvolvimento ou de preparação antes da implantação em um ambiente de produção.

Ao criar AMIs personalizadas para instâncias de GPU, recomenda-se desenvolver AMIs personalizadas distintas para cada geração e família de instâncias que você pretende executar. As AMIs aceleradas e otimizadas para o EKS realizam a instalação seletiva de drivers e pacotes durante o runtime, de acordo com a geração e a família do tipo de instância subjacente. Para obter mais informações, consulte os scripts de instalação e de runtime das AMIs do EKS.

Pré-requisitos

- Instalar a AWS CLI
- Instalar o HashiCorp Packer v1.9.4+
- Instalar o GNU Make

Início rápido

Este guia de início rápido apresenta os comandos para criar uma AMI personalizada na conta da AWS. Para saber mais sobre as configurações disponíveis para personalizar sua AMI, consulte as variáveis de modelo na página Amazon Linux 2023.

Pré-requisitos

Instale o plug-in da Amazon necessário. Por exemplo:

```
packer plugins install github.com/hashicorp/amazon
```

Etapa 1. Configurar o ambiente

Clone ou bifurque o repositório oficial de AMIs do Amazon EKS. Por exemplo:

```
git clone https://github.com/awslabs/amazon-eks-ami.git
cd amazon-eks-ami
```

Verifique se o Packer está instalado:

```
packer --version
```

Etapa 2. Criar uma AMI do personalizada

Veja a seguir exemplos de comando de várias AMIs personalizadas.

AMI do NVIDIA AL2 básica:

```
make k8s=1.31 os_distro=al2 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=560 \
  enable_efa=true
```

AMI do NVIDIA AL2023 básica:

```
make k8s=1.31 os_distro=al2023 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=560 \
  enable_efa=true
```

AMI do Neuron AL2023 compatível com STIG:

```
make k8s=1.31 os_distro=al2023 \
```

```
  enable_accelerator=neuron \
  enable_fips=true \
  source_ami_id=ami-0abcd1234efgh5678 \
  kms_key_id=alias/aws-stig
```

Depois de executar esses comandos, o Packer fará o seguinte: * Executará uma instância temporária do Amazon EC2. * Instalará componentes, drivers e configurações do Kubernetes. * Criará a AMI em sua conta da AWS.

A saída esperada deve ser semelhante a essa:

```
==> Wait completed after 8 minutes 42 seconds

==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs: AMIs were created:
us-west-2: ami-0e139a4b1a7a9a3e9

--> amazon-ebs: AMIs were created:
us-west-2: ami-0e139a4b1a7a9a3e9

--> amazon-ebs: AMIs were created:
us-west-2: ami-0e139a4b1a7a9a3e9
```

Etapa 3. Visualizar valores padrão

Para visualizar valores padrão e opções adicionais, execute o seguinte comando:

```
make help
```

Criar nós com AMIs otimizadas do Bottlerocket

O Bottlerocket é uma distribuição de código aberto do Linux patrocinada e apoiada pela AWS. O Bottlerocket foi desenvolvido especificamente para hospedar workloads de contêineres. Com o Bottlerocket, você pode melhorar a disponibilidade de implantações em contêineres e reduzir os custos operacionais automatizando as atualizações em sua infraestrutura de contêineres. O Bottlerocket inclui somente o software essencial para executar contêineres, o que melhora o uso de recursos, reduz as ameaças à segurança e diminui a sobrecarga de gerenciamento. A AMI do Bottlerocket inclui `containerd`, `kubelet` e o AWS IAM Authenticator. Além dos grupos de nós gerenciados e dos nós autogerenciados, o Bottlerocket também é compatível com o Karpenter.

Vantagens

Usar o Bottlerocket com seu cluster do Amazon EKS tem as seguintes vantagens:

- Maior tempo de atividade com menor custo operacional e menor complexidade de gerenciamento: o Bottlerocket tem um volume de recursos menor, tempos de inicialização mais curtos e é menos vulnerável a ameaças de segurança do que outras distribuições do Linux. Um volume menor do Bottlerocket ajuda a reduzir custos usando menos recursos de armazenamento, computação e rede.
- Segurança aprimorada com atualizações automáticas do sistema operacional: as atualizações do Bottlerocket são aplicadas como uma única unidade que pode ser revertida, se necessário. Isso elimina o risco de atualizações corrompidas ou com falha que podem deixar o sistema em um estado inutilizável. Com o Bottlerocket, as atualizações de segurança poderão ser aplicadas automaticamente assim que estiverem disponíveis de forma minimamente disruptiva, e poderão ser revertidas se ocorrerem falhas.
- Suporte Premium: as compilações fornecidas pela AWS do Bottlerocket no Amazon EC2 são cobertas pelos mesmos planos de suporte do AWS Support, que também abrangem serviços da AWS como o Amazon EC2, Amazon EKS e Amazon ECR.

Considerações

Considere o seguinte ao usar o Bottlerocket para seu tipo de AMI:

- O Bottlerocket é compatível com instâncias do Amazon EC2 com processadores `x86_64` e `arm64`.
- O Bottlerocket é compatível com instâncias do Amazon EC2 equipadas com GPUs. Para obter mais informações, consulte the section called “Uso de AMIs do Linux com suporte a GPU para o EKS”.
- As imagens do Bottlerocket não incluem um shell ou um servidor SSH. Você pode usar métodos de acesso fora de banda para permitir o SSH. Essas abordagens habilitam o contêiner do administrador e a aprovação de algumas etapas de configuração de bootstrapping com dados de usuário. Para obter mais informações, consulte as seguintes seções em Bottlerocket OS no GitHub:
  - Exploration (Exploração
  - Admin container (Contêiner Admin
  - Kubernetes settings (Configurações do Kubernetes
- O Bottlerocket usa diferentes tipos de contêineres:

- Por padrão, um contêiner de controle está habilitado. Esse contêiner executa o agente do AWS Systems Manager que você pode usar para executar comandos ou iniciar sessões de shell em instâncias do Bottlerocket para Amazon EC2. Para obter mais informações, consulte Configuração do Session Manager no Guia do Usuário do AWS Systems Manager.
- Se uma chave SSH foi fornecida ao criar o grupo de nós, um contêiner de administrador está habilitado. Recomendamos usar o contêiner administrador somente para casos de desenvolvimento e teste. Não recomendamos usá-lo em ambientes de produção. Para obter mais informações, consulte Contêiner administrador no GitHub.

Mais informações

Para obter mais informações sobre o uso de AMIs do Bottlerocket otimizadas para o Amazon EKS, consulte as seguintes seções:

- Para obter detalhes sobre o Bottlerocket, consulte a documentação do Bottlerocket.
- Para obter recursos de informações da versão, consulte the section called “Obter informações de versão”.
- Para usar o Bottlerocket com grupos de nós gerenciados, consulte the section called “Grupos de nós gerenciados”.
- Para executar os nós autogerenciados do Bottlerocket, consulte the section called “Bottlerocket”.
- Para recuperar os IDs mais recentes das AMIs do Bottlerocket otimizadas para o Amazon EKS, consulte the section called “Obter os IDs mais recentes”.
- Para obter detalhes sobre suporte para conformidade, consulte the section called “Suporte de conformidade”.

Recuperar informações das versões de AMIs do Bottlerocket

Cada versão de AMI do Bottlerocket inclui várias versões do kubelet, do kernel do Bottlerocket e do containerd. As variantes de AMIs aceleradas também incluem várias versões do driver NVIDIA. Você pode encontrar essas informações sobre a versão no tópico OS da documentação do Bottlerocket. Nessa página, navegue até o subtópico de informações da versão aplicável.

A documentação do Bottlerocket pode, às vezes, ficar atrasada em relação às versões disponíveis no GitHub. Você pode encontrar uma lista de alterações das versões mais recentes nas versões no GitHub.

Recuperar IDs da AMI do Bottlerocket recomendadas

Ao implantar nós, é possível especificar um ID para uma imagem de máquina da Amazon (AMI) pré-compilada e otimizada para o Amazon EKS. Para recuperar um ID de AMI que se ajuste à configuração desejada, consulte a API AWS Systems Manager Parameter Store. O uso dessa API elimina a necessidade de pesquisar manualmente IDs de AMIs otimizadas para o Amazon EKS. Para obter mais informações, consulte GetParameter. A entidade principal do IAM que você usou deve ter a permissão `ssm:GetParameter` do IAM para recuperar os metadados da AMI otimizada para o Amazon EKS.

Você pode recuperar o ID da imagem da AMI otimizada mais recente do Bottlerocket para Amazon EKS com o comando a seguir da AWS CLI, que usa o subparâmetro `image_id`. Faça as seguintes modificações no comando, conforme necessário, e execute o comando atualizado:

- Substitua *`kubernetes-version`* por uma versão da plataforma compatível.
- Substitua *`-flavor`* por uma das opções a seguir.
  - Remover *`-flavor`* para variantes sem GPU.
  - Use *`-nvidia`* para variantes habilitadas para GPU.
  - Use *`-fips`* para variantes habilitadas para FIPS.
- Substitua *`architecture`* por uma das opções a seguir.
  - Use *`x86_64`* para instâncias baseadas em x86.
  - Use *`arm64`* para instâncias ARM.
- Substitua *`region-code`* por uma região da AWScompatível com o Amazon EKS para a qual você deseja obter o ID da AMI.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-kubernetes-version-
flavor/architecture/latest/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Aqui está um exemplo de comando após as substituições do espaço reservado terem sido feitas.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.31/x86_64/latest/
image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

Veja abaixo um exemplo de saída.

```
ami-1234567890abcdef0
```

Atender aos requisitos de conformidade com o Bottlerocket

O Bottlerocket está em conformidade com as recomendações definidas por várias organizações:

- Existe um nível de referência do CIS definido para o Bottlerocket. Em uma configuração padrão, a imagem do Bottlerocket tem a maioria dos controles exigidos pelo perfil de configuração CIS Nível 1. É possível implementar os controles necessários para um perfil de configuração CIS de nível 2. Para obter mais informações, consulte Validação da AMI otimizada do Bottlerocket do Amazon EKS em relação ao Benchmark CIS no blog AWS.
- O conjunto de recursos otimizados e a superfície de ataque reduzida significam que as instâncias do Bottlerocket exigem menos configuração para atender aos requisitos do PCI DSS. O CIS Benchmark for Bottlerocket é um excelente recurso para orientação de hardening e apoia seus requisitos de padrões de configuração segura de acordo com o requisito 2.2 do PCI DSS. Também é possível usar o Fluent Bit para atender aos seus requisitos de registro em log de auditoria em nível de sistema operacional, de acordo com o requisito 10.2 do PCI DSS. A AWS publica instâncias novas (com patches) do Bottlerocket periodicamente para ajudar a atender aos requisitos 6.2 do PCI DSS (para v3.2.1) e 6.3.3 (para v4.0).
- O Bottlerocket é um recurso qualificado para HIPAA autorizado para uso com workloads regulamentadas tanto para o Amazon EC2 quanto para o Amazon EKS. Para obter mais informações, consulte a referência de serviços qualificados pelo HIPAA.
- As AMIs do Bottlerocket pré-configuradas estão disponíveis para o uso de módulos de criptografia validados por FIPS 140-3. Inclui os módulos Amazon Linux 2023 Kernel Crypto API Cryptographic e AWS-LC Cryptographic. Para obter mais informações, consulte the section called “AMIs do FIPS do Bottlerocket”.

Prepare seus nós de processamento para o FIPS com as AMIs do FIPS do Bottlerocket

A publicação 140-3 do Federal Information Processing Standard (FIPS) é um padrão dos governos dos EUA e do Canadá que especifica os requisitos de segurança para módulos criptográficos que protegem informações confidenciais. O Bottlerocket facilita o cumprimento do FIPS ao oferecer AMIs com um kernel do FIPS.

Essas AMIs são pré-configuradas para serem usadas com módulos criptográficos validados pelo FIPS 140-3. Inclui os módulos Amazon Linux 2023 Kernel Crypto API Cryptographic e AWS-LC Cryptographic.

O uso das AMIs do FIPS do Bottlerocket torna seus nós de processamento “prontos para o FIPS”, mas não estão automaticamente “em conformidade com o FIPS”. Para obter mais informações, consulte Federal Information Processing Standard (FIPS) 140-3.

Considerações

- Se o cluster usar sub-redes isoladas, o endpoint do FIPS do Amazon ECR poderá não estar acessível. Isso pode fazer com que o bootstrap do nó falhe. Certifique-se de que sua configuração de rede tenha acesso aos endpoints necessários do FIPS. Para obter mais informações, consulte Access a resource through a resource VPC endpoint no Guia do AWS PrivateLink.
- Se o cluster usar uma sub-rede com o PrivateLink, as extrações de imagens falharão porque os endpoints do FIPS do Amazon ECR não estarão disponíveis por meio do PrivateLink.

Criar um grupo de nós gerenciados com uma AMI do FIPS do Bottlerocket

A AMI do FIPS do Bottlerocket vem em duas variantes para ser compatível com as workloads:

- `BOTTLEROCKET_x86_64_FIPS`
- `BOTTLEROCKET_ARM_64_FIPS`

Para criar um grupo de nós gerenciados com uma AMI do FIPS do Bottlerocket, escolha o tipo de AMI aplicável durante o processo de criação. Para obter mais informações, consulte the section called “Criar”.

Para obter mais informações sobre como selecionar variantes habilitadas para FIPS, consulte the section called “Obter os IDs mais recentes”.

Desabilitar o endpoint do FIPS para regiões da AWS não compatíveis

As AMIs do FIPS do Bottlerocket são compatíveis diretamente nos Estados Unidos, incluindo as regiões AWS GovCloud (EUA). Para regiões da AWS em que as AMIs estão disponíveis, mas não são compatíveis diretamente, você ainda poderá usar as AMIs criando um grupo de nós gerenciados com um modelo de execução.

A AMI do FIPS do Bottlerocket depende do endpoint do FIPS do Amazon ECR durante o bootstrap, que geralmente não está disponível fora dos Estados Unidos. Para usar a AMI para o kernel do FIPS nas regiões da AWS que não têm o endpoint do FIPS do Amazon ECR disponível, siga estas etapas para desabilitar o endpoint do FIPS:

1. Crie um novo arquivo de configuração com o conteúdo a seguir ou incorpore o conteúdo ao seu arquivo de configuração existente.

```
[default]
use_fips_endpoint=false
```

1. Codifique o conteúdo do arquivo no formato Base64.
2. No `UserData` do seu modelo de execução, adicione a seguinte string codificada usando o formato TOML:

```
[settings.aws]
config = "<your-base64-encoded-string>"
```

Para outras configurações, consulte Description of settings do Bottlerocket no GitHub.

Veja abaixo um exemplo de `UserData` em um modelo de execução:

```
[settings]
motd = "Hello from eksctl!"
[settings.aws]
config = "W2RlZmF1bHRdCnVzZV9maXBzX2VuZHBvaW50PWZhbHNlCg==" # Base64-encoded string.
[settings.kubernetes]
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
cluster-name = "<cluster-name>"
...<other-settings>
```

Para obter mais informações sobre como criar um modelo de execução com dados de usuário, consulte the section called “Modelos de execução”.

Criar nós com AMIs otimizadas do Ubuntu Linux

A Canonical fez uma parceria com o Amazon EKS para criar AMIs de nós que você pode usar nos clusters.

A Canonical fornece uma imagem de sistema operacional de nó do Kubernetes para fins específicos. A imagem minimizada do Ubuntu é otimizada para o Amazon EKS e inclui o kernel da AWS personalizado que é desenvolvido em conjunto com a AWS. Para obter mais informações, consulte Ubuntu no Amazon Elastic Kubernetes Service (EKS) e the section called “Ubuntu Linux”. Para obter informações sobre suporte, consulte a seção de software de terceiros das Perguntas frequentes sobre suporte premium do AWS.

Criar nós com AMIs otimizadas do Windows

As AMIs do Windows otimizadas para o Amazon EKS são desenvolvidas com base no Windows Server 2019 e 2022. Elas são configuradas para servirem como imagem base para nós do Amazon EKS. As AMIs incluem por padrão os seguintes componentes:

- kubelet
- kube-proxy
- AWS IAM Authenticator para Kubernetes
- csi-proxy
- containerd

> **Note**
>
> É possível rastrear eventos de segurança ou privacidade do Windows Server com o Guia de atualização de segurança da Microsoft.

O Amazon EKS oferece AMIs que são otimizadas para contêineres do Windows nas variantes a seguir:

- AMI do Windows Server 2019 Core otimizada para Amazon EKS
- AMI completa do Windows Server 2019 Core otimizada para Amazon EKS
- AMI do Windows Server 2022 Core otimizada para o Amazon EKS

- AMI Full do Windows Server 2022 otimizada para Amazon EKS

> **Important**
>
> - A AMI Core do Windows Server 20H2 otimizada para Amazon EKS foi descontinuada. Nenhuma nova versão dessa AMI será lançada.
> - Por padrão, para garantir que você tenha as atualizações de segurança mais recentes, o Amazon EKS mantém as AMIs otimizadas do Windows dos últimos quatro meses. Cada nova AMI estará disponível por quatro meses a partir do lançamento inicial. Após esse período, as AMIs mais antigas se tornarão privadas e não estarão mais acessíveis. Incentivamos o uso das AMIs mais recentes para evitar vulnerabilidades de segurança e a perda de acesso às AMIs mais antigas que atingiram o fim da vida útil com suporte. Embora não possamos garantir que poderemos fornecer acesso a AMIs que se tornaram privadas, você pode solicitar acesso ao preencher um tíquete no AWS Support.

Calendário de lançamento

A tabela a seguir lista as datas de lançamento e término do suporte para versões do Windows no Amazon EKS. Se uma data de término estiver em branco, é porque a versão ainda tem suporte.

Versão do Windows	Lançamento do Amazon EKS	Fim do suporte para o Amazon EKS
Windows Server 2022 Core	17/10/2022	
Windows Server 2022 Full	17/10/2022	
Windows Server 20H2 Core	12/8/2021	9/8/2022
Windows Server 2004 Core	19/8/2020	14/12/2021
Windows Server 2019 Core	7/10/2019	
Windows Server 2019 Full	7/10/2019	
Windows Server 1909 Core	7/10/2019	8/12/2020

Parâmetros de configuração do script de bootstrap

Quando você cria um nó do Windows, há um script nesse nó que permite configurar parâmetros diferentes. Dependendo da configuração, esse script pode ser encontrado no nó em um local semelhante a: `C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1`. Você pode especificar valores de parâmetros personalizados definindo-os como argumentos para o script de bootstrap. Por exemplo, você pode atualizar os dados do usuário no modelo de execução. Para obter mais informações, consulte the section called “Dados do usuário do Amazon EC2”.

O script inclui os seguintes parâmetros de linha de comando:

- `-EKSClusterName` – Especifica o nome do cluster do Amazon EKS no qual esse nó de processamento deve ingressar.
- `-KubeletExtraArgs`: especifica argumentos extras para `kubelet` (opcional).
- `-KubeProxyExtraArgs`: especifica argumentos extras para `kube-proxy` (opcional).
- `-APIServerEndpoint` – Especifica o endpoint do servidor de API do cluster do Amazon EKS (opcional). Válido somente quando usado com `-Base64ClusterCA`. Ignorar chamadas `Get-EKSCluster`.
- `-Base64ClusterCA` – Especifica o conteúdo da CA do cluster codificado em base64 (opcional). Válido somente quando usado com `-APIServerEndpoint`. Ignorar chamadas `Get-EKSCluster`.
- `-DNSClusterIP` – Substitui o endereço IP a ser usado para consultas DNS dentro do cluster (opcional). O padrão é `10.100.0.10` ou `172.20.0.10` com base no endereço IP da interface primária.
- `-ServiceCIDR`: substitui o intervalo de endereços IP dos serviços do Kubernetes do qual os serviços do cluster são considerados. O padrão é `172.20.0.0/16` ou `10.100.0.0/16` com base no endereço IP da interface primária.
- `-ExcludedSnatCIDRs`: uma lista de CIDRs `IPv4` a serem excluídos da Source Network Address Translation (SNAT). Isso significa que o IP privado do pod que é endereçável pela VPC não seria convertido no endereço IP do endereço `IPv4` primário da ENI da instância para o tráfego de saída. Por padrão, o CIDR `IPv4` da VPC para o nó do Windows para Amazon EKS é adicionado. Especificar CIDRs para esse parâmetro também exclui adicionalmente os CIDRs especificados. Para obter mais informações, consulte the section called “Tráfego de saída”.

Além dos parâmetros de linha de comando, você também pode especificar alguns parâmetros de variável de ambiente. Ao especificar um parâmetro de linha de comando, ele tem precedência

sobre a respectiva variável de ambiente. A(s) variável(is) de ambiente deve(m) ser definida(s) como escopo de máquina (ou sistema), já que o script de inicialização somente lerá variáveis de escopo de máquina.

O script leva em consideração as seguintes variáveis de ambiente:

- `SERVICE_IPV4_CIDR`— Consulte o parâmetro da linha de comando `ServiceCIDR` para obter a definição.
- `EXCLUDED_SNAT_CIDRS`— Deve ser uma strings separada por vírgula. Consulte o parâmetro da linha de comando `ExcludedSnatCIDRs` para obter a definição.

Compatibilidade com a autenticação gMSA

Os pods do Windows para Amazon EKS permitem diferentes tipos de autenticação group Managed Service Account (gMSA).

- O Amazon EKS é compatível com identidades de domínio do Active Directory para autenticação. Para obter mais informações sobre a autenticação gMSA conectada ao domínio, consulte Windows Authentication on Amazon EKS Windows pods no blog da AWS.
- O Amazon EKS oferece um plug-in que permite que nós do Windows não conectados ao domínio recuperem credenciais de gMSA com uma identidade de usuário portátil. Para obter mais informações sobre a autenticação gMSA não conectada ao domínio, consulte Domainless Windows Authentication for Amazon EKS Windows pods no blog da AWS.

Imagens de contêiner em cache

As AMIs do Windows otimizadas para Amazon EKS têm determinadas imagens de contêiner armazenadas em cache para o runtime `containerd`. Imagens de contêiner são armazenadas em cache ao criar AMIs personalizadas com o uso componentes de compilação gerenciados pela Amazon. Para obter mais informações, consulte the section called “Usar o componente de compilação gerenciado pela Amazon”.

As seguintes imagens de contêiner em cache são para o runtime `containerd`:

- `amazonaws.com/eks/pause-windows`
- `mcr.microsoft.com/windows/nanoserver`
- `mcr.microsoft.com/windows/servercore`

Mais informações

Para obter mais informações sobre o uso de AMIs do Windows otimizadas para Amazon EKS, consulte as seguintes seções:

- Para obter detalhes sobre a execução de workloads em AMIs aceleradas do Amazon Linux otimizadas para o EKS, consulte the section called “Configurar AMIs de GPU do Windows”.
- Para usar o Windows com grupos de nós gerenciados, consulte the section called “Grupos de nós gerenciados”.
- Para iniciar nós autogerenciados do Windows, consulte the section called “Windows”.
- Para obter informações sobre versões, consulte the section called “Obter informações de versão”.
- Para recuperar os IDs mais recentes das AMIs do Windows otimizadas para Amazon EKS, consulte the section called “Obter os IDs mais recentes”.
- Para usar o Amazon EC2 Image Builder para criar AMIs do Windows personalizadas otimizadas para Amazon EKS, consulte the section called “Compilações personalizadas”.
- Para obter as práticas recomendadas, consulte Gerenciamento de AMI do Windows otimizado do Amazon EKS no Guia de práticas recomendadas do EKS.

Criar nós autogerenciados do Windows Server 2022 com o **`eksctl`**

Você pode usar o arquivo `test-windows-2022.yaml` a seguir como referência para criar nós autogerenciados do Windows Server 2022. Substitua cada *`valor de exemplo`* por seus próprios valores.

> **Note**
>
> Você deve usar a versão 0.116.0 ou mais recente do `eksctl` para executar nós autogerenciados do Windows Server 2022.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-2022-cluster
  region: region-code
  version: '1.33'
```

```
nodeGroups:
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2022FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
  - name: linux-ng
    amiFamily: AmazonLinux2
    minSize: 2
    maxSize: 3
```

Os grupos de nós podem ser criados com o uso do seguinte comando:

```
eksctl create cluster -f test-windows-2022.yaml
```

Recuperar informações de versões de AMIs do Windows

Este tópico lista as versões das AMIs do Windows otimizadas para o Amazon EKS e as versões correspondentes do kubelet, containerd e csi-proxy.

Os metadados da AMI otimizada para o Amazon EKS, incluindo o ID da AMI, podem ser recuperados de maneira programática em cada variante. Para obter mais informações, consulte the section called “Obter os IDs mais recentes”.

As versões das AMIs são definidas pela versão do Kubernetes e pela data de lançamento da AMI, no seguinte formato:

```
k8s_major_version.k8s_minor_version-release_date
```

> **Note**
>
> Os grupos de nós gerenciados do Amazon EKS são compatíveis com as versões das AMIs do Windows de novembro de 2022 e posteriores.

Para receber notificações de todas as alterações do arquivo de origem nesta página específica da documentação, você pode se inscrever no seguinte URL com um leitor de RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/nodes/eks-ami-versions-windows.adoc.atom
```

AMI do Windows Server 2022 Core otimizada para o Amazon EKS

As tabelas a seguir listam as versões atuais e anteriores da AMI do Windows Server 2022 Core otimizada para o Amazon EKS.

Example

Versão 1.34 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.34-2025.10.18	1.34.1	2.1.4	1.2.1	
1.34-2025.09.13	1.34.0	2.1.4	1.2.1	

Versão 1.33 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.33-2025.10.18	1.33.5	1.7.28	1.2.1	
1.33-2025.09.13	1.33.4	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.33-2025.08.18	1.33.3	1.7.27	1.2.1	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Versão 1.32 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.32-2025.10.18	1.32.9	1.7.28	1.2.1	
1.32-2025.09.13	1.32.8	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.32-2025.08.18	1.32.7	1.7.27	1.2.1	
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.0	1.7.20	1.1.3	

Versão 1.31 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.31-2025.10.18	1.31.13	1.7.28	1.2.1	
1.31-2025.09.13	1.31.12	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.31-2025.08.18	1.31.11	1.7.27	1.2.1	
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Inclui patches para o CVE-2024-9042 .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Versão 1.30 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.30-2025.10.18	1.30.14	1.7.28	1.2.1	
1.30-2025.09.13	1.30.14	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.30-2025.08.18	1.30.14	1.7.27	1.2.1	
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Inclui patches para o CVE-2024-9042 .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Inclui patches para o CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Atualizado `containerd` para 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Versão 1.29 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2025.10.18	1.29.15	1.7.28	1.2.1	
1.29-2025.09.13	1.29.15	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.29-2025.08.18	1.29.15	1.7.27	1.2.1	
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Inclui patches para o CVE-2024-9042 .
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Atualizado containerd para 1.7.11. Atualizado kubelet para 1.29.3.

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Atualizado `containerd` para `1.6.28`. Reconstrução da CNI e do `csi-proxy` usando o `golang 1.22.1`.
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Corrigido um erro em que a imagem de pausa era excluída incorretamente pelo processo de coleta de resíduos do `kubelet`.

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2024.01.11	1.29.0	1.6.18	1.1.2	Atualização autônoma do Windows KB5034439 excluída nas AMIs principais do Windows Server 2022. O KB se aplica somente às instalações do Windows com uma partição específica do WinRE, que não estão incluídas em nenhuma das nossas AMIs do Windows otimizadas para Amazon EKS.

Versão 1.28 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2025.10.18	1.28.15	1.7.28	1.2.1	
1.28-2025.09.13	1.28.15	1.7.28	1.2.1	Alteração dos logs do plug-

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
				in gMSA para Eventos do Windows
1.28-2025.08.18	1.28.15	1.7.27	1.2.1	
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025.01.15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	Inclui patches para o `CVE-2024-9042` .

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.28.8.

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução da CNI e do csi-proxy usando o golang 1.22.1.
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2024.01.11	1.28.5	1.6.18	1.1.2	Atualização autônoma do Windows KB5034439 excluída nas AMIs principais do Windows Server 2022. O KB se aplica somente às instalações do Windows com uma partição específica do WinRE, que não estão incluídas em nenhuma das nossas AMIs do Windows otimizadas para Amazon EKS.
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

AMI Windows Server 2022 Full otimizada para o Amazon EKS

As tabelas a seguir listam as versões atuais e anteriores da AMI do Windows Server 2022 Full otimizada para o Amazon EKS.

Example

Versão 1.34 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.34-2025.10.18	1.34.1	2.1.4	1.2.1	
1.34-2025.09.13	1.34.0	2.1.4	1.2.1	

Versão 1.33 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.33-2025.10.18	1.33.5	1.7.28	1.2.1	
1.33-2025.09.13	1.33.4	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.33-2025.08.18	1.33.3	1.7.27	1.2.1	
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Versão 1.32 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.32-2025.10.18	1.32.9	1.7.28	1.2.1	
1.32-2025.09.13	1.32.8	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.32-2025.08.18	1.32.7	1.7.27	1.2.1	
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	containrd atualizado para 1.7.27
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.01	1.32.0	1.7.20	1.1.3	

Versão 1.31 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.31-2025.10.18	1.31.13	1.7.28	1.2.1	
1.31-2025.09.13	1.31.12	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.31-2025.08.18	1.31.11	1.7.27	1.2.1	
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Inclui patches para o CVE-2024-9042 .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Versão 1.30 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.30-2025.10.18	1.30.14	1.7.28	1.2.1	
1.30-2025.09.13	1.30.14	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.30-2025.08.18	1.30.14	1.7.27	1.2.1	
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Atualizado `contianerd` para `1.7.20`.
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Inclui patches para o `CVE-2024-9042` .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Inclui patches para o CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Atualizado containerd para 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Versão 1.29 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2025.10.18	1.29.15	1.7.28	1.2.1	
1.29-2025.09.13	1.29.15	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2025.08.18	1.29.15	1.7.27	1.2.1	
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Inclui patches para o `CVE-2024-9042` .
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Atualizado containerd para 1.7.11. Atualizado o kubelet para 1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Reconstrução da CNI e do csi-proxy usando o golang 1.22.1.

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Corrigido um erro em que a imagem de pausa era excluída incorretamente pelo processo de coleta de resíduos do `kubelet`.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Versão 1.28 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2025.10.18	1.28.15	1.7.28	1.2.1	
1.28-2025.09.13	1.28.15	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2025.08.18	1.28.15	1.7.27	1.2.1	
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025.01.15	1.28.15	1.7.14	1.1.3	
1.28-2025.01.01	1.28.15	1.7.14	1.1.3	Inclui patches para o `CVE-2024-9042` .
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Atualizado `containerd` para 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Atualizado `containerd` para 1.6.28. Atualizado `kubelet` para 1.28.8.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Atualizado `containerd` para 1.6.25. Reconstrução da CNI e do `csi-proxy` usando o golang 1.22.1.

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

AMI do Windows Server 2019 Core otimizada para Amazon EKS

As tabelas a seguir listam as versões atuais e anteriores da AMI do Windows Server 2019 Core otimizada para o Amazon EKS.

Example

Versão 1.34 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.34-2025.10.18	1.34.1	2.1.4	1.2.1	
1.34-2025.09.13	1.34.0	2.1.4	1.2.1	

Versão 1.33 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.33-2025.10.18	1.33.5	1.7.28	1.2.1	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.33-2025.09.13	1.33.4	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.33-2025.08.18	1.33.3	1.7.27	1.2.1	
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Versão 1.32 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.32-2025.10.18	1.32.9	1.7.28	1.2.1	
1.32-2025.09.13	1.32.8	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.32-2025.08.18	1.32.7	1.7.27	1.2.1	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.4	1.7.20	1.1.3	

Versão 1.31 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.31-2025.10.18	1.31.13	1.7.28	1.2.1	
1.31-2025.09.13	1.31.12	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.31-2025.08.18	1.31.11	1.7.27	1.2.1	
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Inclui patches para o `CVE-2024-9042` .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Versão 1.30 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.30-2025.10.18	1.30.14	1.7.28	1.2.1	
1.30-2025.09.13	1.30.14	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.30-2025.08.18	1.30.14	1.7.27	1.2.1	
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.30-2025-02-15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Inclui patches para o `CVE-2024-9042` .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Inclui patches para o CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Atualizado containerd para 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Versão 1.29 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2025.10.18	1.29.15	1.7.28	1.2.1	
1.29-2025.09.13	1.29.15	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.29-2025.08.18	1.29.15	1.7.27	1.2.1	
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Inclui patches para o `CVE-2024-9042` .
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Atualizado containerd para 1.7.11. Atualizado kubelet para 1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Reconstrução da CNI e do csi-proxy usando o golang 1.22.1.
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Corrigido um erro em que a imagem de pausa era excluída incorretamente pelo processo de coleta de resíduos do kubelet.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Versão 1.28 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2025.10.18	1.28.15	1.7.28	1.2.1	
1.28-2025.09.13	1.28.15	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.28-2025.08.18	1.28.15	1.7.27	1.2.1	
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2025.06.13	1.28.15	1.7.20	1.2.1	Atualizado `containerd` para `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	Inclui patches para o `CVE-2024-9042` .
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.28.8.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Atualizado containerd para 1.6.25. Reconstrução da CNI e do csi-proxy usando o golang 1.22.1.
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Atualizado containerd para 1.6.18. Foram adicionadas novas variáveis do ambiente de script de bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Corrigido um aviso de segurança em kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

AMI completa do Windows Server 2019 Core otimizada para Amazon EKS

As tabelas a seguir listam as versões atuais e anteriores da AMI do Windows Server 2019 Full otimizada para o Amazon EKS.

Example

Versão 1.34 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.34-2025.10.18	1.34.1	2.1.4	1.2.1	
1.34-2025.09.13	1.34.0	2.1.4	1.2.1	

Versão 1.33 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.33-2025.10.18	1.33.5	1.7.28	1.2.1	
1.33-2025.09.13	1.33.4	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.33-2025.08.18	1.33.3	1.7.27	1.2.1	
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Versão 1.32 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.32-2025.10.18	1.32.9	1.7.28	1.2.1	
1.32-2025.09.13	1.32.8	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.32-2025.08.18	1.32.7	1.7.27	1.2.1	
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.0	1.7.20	1.1.3	

Versão 1.31 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.31-2025.10.18	1.31.13	1.7.28	1.2.1	
1.31-2025.09.13	1.31.12	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.31-2025.08.18	1.31.11	1.7.27	1.2.1	
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Inclui patches para o CVE-2024-9042 .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Versão 1.30 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.30-2025.10.18	1.30.14	1.7.28	1.2.1	
1.30-2025.09.13	1.30.14	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.30-2025.08.18	1.30.14	1.7.27	1.2.1	
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Inclui patches para o CVE-2024-9042 .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Inclui patches para o CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Atualizado `containerd` para 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Versão 1.29 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2025.10.18	1.29.15	1.7.28	1.2.1	
1.29-2025.09.13	1.29.15	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.29-2025.08.18	1.29.15	1.7.27	1.2.1	
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Inclui patches para o CVE-2024-9042 .
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Atualizado containerd para 1.7.11. Atualizado kubelet para 1.29.3.

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Atualizado `containerd` para `1.6.28`. Reconstrução da CNI e do `csi-proxy` usando o `golang 1.22.1`.
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Corrigido um erro em que a imagem de pausa era excluída incorretamente pelo processo de coleta de resíduos do `kubelet`.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Versão 1.28 do Kubernetes

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2025.10.18	1.28.15	1.7.28	1.2.1	
1.28-2025.09.13	1.28.15	1.7.28	1.2.1	Alteração dos logs do plug-in gMSA para Eventos do Windows
1.28-2025.08.18	1.28.15	1.7.27	1.2.1	
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	Atualizado `containerd` para `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Atualizado `containerd` para `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-15	1.28.15	1.7.14	1.1.3	

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	Inclui patches para o CVE-2024-9042 .
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Inclui patches para o CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Atualizado containerd para 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Atualizado containerd para 1.6.28. Atualizado kubelet para 1.28.8.

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Atualizado `containerd` para `1.6.25`. Reconstrução da CNI e do `csi-proxy` usando o `golang 1.22.1`.
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Inclui patches para o CVE-2023-5528 .

AMI version	versão de kubelet	versão de containerd	versão de csi-proxy	Notas da versão
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Atualizado `containerd` para `1.6.18`. Foram adicionadas novas variáveis do ambiente de script de bootstrap (`SERVICE_IPV4_CIDR` e `EXCLUDED_SNAT_CIDRS`).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Corrigido um aviso de segurança em `kubelet`.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Recuperar IDs de AMI do Microsoft Windows recomendadas

Ao implantar nós, é possível especificar um ID para uma imagem de máquina da Amazon (AMI) pré-compilada e otimizada para o Amazon EKS. Para recuperar um ID de AMI que se ajuste à configuração desejada, consulte a API AWS Systems Manager Parameter Store. O uso dessa API elimina a necessidade de pesquisar manualmente IDs de AMIs otimizadas para o Amazon EKS. Para obter mais informações, consulte GetParameter. A entidade principal do IAM que você usou deve ter a permissão `ssm:GetParameter` do IAM para recuperar os metadados da AMI otimizada para o Amazon EKS.

Você pode recuperar o ID de imagem da mais recente AMI do Windows otimizada e recomendada para Amazon EKS com o comando a seguir, que usa o subparâmetro `image_id`. Faça as seguintes modificações no comando, conforme necessário, e execute o comando atualizado:

- Substitua *`release`* por uma das seguintes opções.
  - Use *`2022`* para o Windows Server 2022.
  - Use *`2019`* para o Windows Server 2019.
- Substitua *`installation-option`* por uma das seguintes opções. Para obter mais informações, consulte O que é a opção de instalação do Server Core no Windows Server.
  - Use o *`Core`* para uma instalação mínima com uma superfície de ataque menor.
  - Use *`Full`* para incluir a experiência de desktop do Windows.
- Substitua *`kubernetes-version`* por uma versão da plataforma compatível.
- Substitua *`region-code`* por uma região da AWScompatível com o Amazon EKS para a qual você deseja obter o ID da AMI.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-release-
English-installation-option-EKS_Optimized-kubernetes-version/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Aqui está um exemplo de comando após as substituições do espaço reservado terem sido feitas.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2022-
English-Core-EKS_Optimized-k8s-n-2/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

Veja abaixo um exemplo de saída.

```
ami-1234567890abcdef0
```

Criar uma AMI personalizada do Windows com o Image Builder

Você pode usar o Image Builder do EC2 para criar AMIs personalizadas do Windows otimizadas para o Amazon EKS com uma das seguintes opções:

- Usando uma AMI do Windows otimizada para o Amazon EKS como base

- Usar o componente de compilação gerenciado pela Amazon

Com ambos os métodos, você deve criar sua própria fórmula do Image Builder. Para obter mais informações, consulte Criar uma nova versão de uma fórmula de imagem, no Guia do usuário do Image Builder.

> **Important**
>
> Os seguintes componentes gerenciados pela Amazon para o `eks` incluem patches para o `CVE-2024-5321`.
>
> - `1.28.2` e superior
> - `1.29.2` e superior
> - `1.30.1` e superior
> - Todas as versões para Kubernetes 1.31 e superiores

Usando uma AMI do Windows otimizada para o Amazon EKS como base

Essa opção é a maneira recomendada para criar AMIs personalizadas do Windows. As AMIs do Windows otimizadas para o Amazon EKS que fornecemos são atualizadas com mais frequência do que o componente de compilação gerenciado pela Amazon.

1. Inicie uma fórmula do Image Builder.
   a. Abra o console do EC2 Image Builder em https://console.aws.amazon.com/imagebuilder.
   b. No painel de navegação esquerdo, escolha Image recipes (Fórmulas de imagens).
   c. Escolha Create image recipe (Criar fórmula de imagem).
2. Na seção Detalhes da fórmula, insira um Nome e uma Versão.
3. Especifique o ID da AMI do Windows otimizada para o Amazon EKS na seção Imagem base.
   a. Escolha Inserir ID de AMI personalizada.
   b. Recupere o ID da AMI da versão do sistema operacional do Windows necessária. Para obter mais informações, consulte the section called “Obter os IDs mais recentes”.
   c. Em AMI ID, insira o ID da AMI personalizado. Se o ID da AMI não for encontrada, certifique-se de que a região AWS do ID da AMI corresponda à região AWS mostrada no canto superior direito do console.

4. (Opcional) Para obter as atualizações de segurança mais recentes, adicione o componente `update-windows` na seção Componentes de compilação.
   a. Na lista suspensa à direita da caixa de pesquisa Find components by name (Localizar componentes por nome), escolha Amazon-managed (Gerenciado pela Amazon).
   b. Na caixa Find components by name (Localizar componentes por nome), insira `update-windows`.
   c. Marque a caixa de seleção do resultado da pesquisa de **`update-windows`**. Esse componente inclui os patches mais recentes do Windows para o sistema operacional.
5. Conclua as entradas restantes da fórmula de imagem com as configurações necessárias. Para obter mais informações, consulte Criar uma nova versão de fórmula de imagem (console), no Guia do usuário do Image Builder.
6. Escolha Create recipe (Criar fórmula).
7. Use a nova fórmula de imagem em um pipeline de imagens novo ou existente. Quando o pipeline de imagens for executado com êxito, a AMI personalizada será listada como imagem de saída e estará pronta para uso. Para obter mais informações, consulte Criar um pipeline de imagem usando o assistente do console do EC2 Image Builder.

Usar o componente de compilação gerenciado pela Amazon

Quando o uso de uma AMI do Windows otimizada para o Amazon EKS como base não for viável, você poderá usar o componente de compilação gerenciado pela Amazon. Essa opção pode estar defasada em relação às versões mais recentes compatíveis do Kubernetes.

1. Inicie uma fórmula do Image Builder.
   a. Abra o console do EC2 Image Builder em https://console.aws.amazon.com/imagebuilder.
   b. No painel de navegação esquerdo, escolha Image recipes (Fórmulas de imagens).
   c. Escolha Create image recipe (Criar fórmula de imagem).
2. Na seção Detalhes da fórmula, insira um Nome e uma Versão.
3. Determine qual opção usar para criar sua AMI personalizada na seção Base image (Imagem base):
   - Select managed images (Selecionar imagens gerenciadas): escolha Windows em Image Operating System (OS) (Sistema operacional da imagem). Em seguida, escolha uma das seguintes opções para Origem da imagem.

- Início rápido (gerenciado pela Amazon): na lista suspensa Nome da imagem, escolha uma versão do Windows Server compatível com o Amazon EKS. Para obter mais informações, consulte the section called “Windows”.
- Images owned by me (Imagens que pertencem a mim): para Nome da imagem, escolha o ARN da sua própria imagem com sua própria licença. A imagem que você fornece ainda não pode ter componentes do Amazon EKS instalados.

- Inserir ID da AMI personalizado: para o ID da AMI, insira o ID da AMI com sua própria licença. A imagem que você fornece ainda não pode ter componentes do Amazon EKS instalados.

4. Na seção Componentes de compilação - Windows, faça o seguinte:
   a. Na lista suspensa à direita da caixa de pesquisa Find components by name (Localizar componentes por nome), escolha Amazon-managed (Gerenciado pela Amazon).
   b. Na caixa Find components by name (Localizar componentes por nome), insira `eks`.
   c. Marque a caixa de seleção do resultado da pesquisa de **`eks-optimized-ami-windows`**, mesmo que o resultado retornado não seja a versão desejada.
   d. Na caixa Find components by name (Localizar componentes por nome), insira `update-windows`.
   e. Marque a caixa de seleção do resultado da pesquisa de update-windows. Esse componente inclui os patches mais recentes do Windows para o sistema operacional.
5. Na seção Componentes selecionados, faça o seguinte:
   a. Selecionar Opções de versionamento para **`eks-optimized-ami-windows`** .
   b. Escolha Especificar versão do componente.
   c. No campo Versão do componente, insira *`versão.x`*, substituindo *`versão`* por uma versão compatível do Kubernetes. A inserção de um *`x`* em parte do número da versão indica o uso da versão mais recente do componente que também se alinha com a parte da versão que você define explicitamente. Preste atenção na saída do console, pois ela informará se a versão desejada está disponível como componente gerenciado. Lembre-se de que as versões do Kubernetes mais recentes podem não estar disponíveis para o componente de compilação. Para obter mais informações sobre as versões disponíveis, consulte the section called “Recuperar informações sobre versões de componentes `eks-optimized-ami-windows`”.
6. Conclua as entradas restantes da fórmula de imagem com as configurações necessárias. Para obter mais informações, consulte Criar uma nova versão de fórmula de imagem (console), no Guia do usuário do Image Builder.

7. Escolha Create recipe (Criar fórmula).
8. Use a nova fórmula de imagem em um pipeline de imagens novo ou existente. Quando o pipeline de imagens for executado com êxito, a AMI personalizada será listada como imagem de saída e estará pronta para uso. Para obter mais informações, consulte Criar um pipeline de imagem usando o assistente do console do EC2 Image Builder.

Recuperar informações sobre versões de componentes **`eks-optimized-ami-windows`**

É possível recuperar informações específicas sobre o que está instalado com cada componente. Por exemplo, você pode verificar qual versão do `kubelet` está instalada. Os componentes passam por testes funcionais nas versões dos sistemas operacionais do Windows compatíveis com o Amazon EKS. Para obter mais informações, consulte the section called “Calendário de lançamento”. Quaisquer outras versões do sistema operacional do Windows listadas como não compatíveis ou que tenham chegado ao fim do suporte podem não ser compatíveis com o componente.

1. Abra o console do EC2 Image Builder em https://console.aws.amazon.com/imagebuilder.
2. No painel de navegação à esquerda, escolha Components (Componentes).
3. Na lista suspensa à direita da caixa de pesquisa Find components by name (Localizar componentes por nome), altere Owned by me (Pertencem a mim) para Quick start (Amazon-managed) (Início rápido, gerenciado pela Amazon).
4. Na caixa Localizar componentes por nome, insira `eks`.
5. (Opcional) Se estiver usando uma versão recente, classifique a coluna Version (Versão) em ordem decrescente, escolhendo-a duas vezes.
6. Escolha o link **`eks-optimized-ami-windows`** com a versão desejada.

As informações em Description (Descrição) na página resultante mostra as informações específicas.

Habilitar o reparo automático de nós e investigar os problemas de integridade de nós

A integridade do nó refere-se ao status operacional e à capacidade de um nó executar workloads com eficiência. Um nó íntegro mantém a conectividade esperada, tem recursos suficientes e pode executar pods com sucesso sem interrupções. Para obter informações sobre como obter detalhes sobre os nós, consulte the section called “Visualizar a integridade do nó” e the section called “Obter logs de nós”.

Para ajudar a manter os nós íntegros, o Amazon EKS oferece o agente de monitoramento de nós e o reparo automático de nós.

> **Important**
>
> O agente de monitoramento de nós e o reparo automático de nós estão disponíveis somente no Linux. Estes recursos não estão disponíveis no Windows.

Agente de monitoramento de nós

O agente de monitoramento de nós lê automaticamente os logs dos nós para detectar determinados problemas de integridade. Ele analisa os logs dos nós para detectar falhas e apresenta várias informações de status sobre os nós de processamento. Uma `NodeCondition` dedicada é aplicada nos nós de processamento para cada categoria de problemas detectados, como problemas de armazenamento e rede. As descrições dos problemas de integridade detectados são disponibilizadas no painel de observabilidade. Para obter mais informações, consulte the section called “Problemas de integridade de nós”.

O agente de monitoramento de nós está incluído como um recurso para todos os clusters do Modo Automático do Amazon EKS. Para outros tipos de cluster, você pode adicionar o agente de monitoramento como um complemento do Amazon EKS. Para obter mais informações, consulte the section called “Criação de um complemento”.

Reparo automático de nós

O reparo automático de nós é um recurso adicional que monitora continuamente a integridade dos nós, reagindo automaticamente aos problemas detectados e substituindo os nós quando possível. Isso ajuda na disponibilidade geral do cluster com o mínimo de intervenção manual. Se uma verificação de integridade falhar, o nó será automaticamente isolado para que nenhum novo pod seja programado no nó.

Por si só, o reparo automático de nós pode reagir à condição `Ready` do `kubelet` e a quaisquer objetos de nós que sejam excluídos manualmente. Quando combinado com o agente de monitoramento de nós, o reparo automático de nós pode reagir a mais condições que não seriam detectadas de outra forma. Essas condições adicionais incluem `KernelReady`, `NetworkingReady` e `StorageReady`.

Essa recuperação automatizada de nós resolve automaticamente problemas de nós intermitentes, como falhas na união ao cluster, kubelets que não respondem e aumento de erros no acelerador (dispositivo). A confiabilidade aprimorada ajuda a reduzir o tempo de inatividade da aplicação e a melhorar as operações do cluster. O reparo automático de nós não pode lidar com certos problemas relatados, como `DiskPressure`, `MemoryPressure` e `PIDPressure`. O Amazon EKS espera dez minutos antes de agir nas `NodeConditions` `AcceleratedHardwareReady`, e 30 minutos para todas as outras condições.

Os grupos de nós gerenciados também desabilitarão automaticamente os reparos de nós por motivos de segurança em dois cenários. Quaisquer operações de reparo que estejam previamente em andamento continuarão em ambas as situações.

- Se uma mudança de zona para o cluster tiver sido acionada por meio do Application Recovery Controller (ARC), todas as operações de reparo subsequentes serão interrompidas.
- Se o grupo de nós tiver mais de cinco nós, e mais de 20% dos nós no grupo de nós estiverem em um estado não íntegro, as operações de reparo serão interrompidas.

Você pode habilitar o reparo automático de nós ao criar ou editar um grupo de nós gerenciados.

- Ao usar o console do Amazon EKS, ative a caixa de seleção Habilitar reparo automático de nós para o grupo de nós gerenciados. Para obter mais informações, consulte the section called “Criar”.
- Ao usar a AWS CLI, adicione o parâmetro `--node-repair-config enabled=true` ao comando `eks create nodegroup` ou `eks update-nodegroup-config`.
- Para ver um exemplo do `ClusterConfig` `eksctl` que usa um grupo de nós gerenciados com reparo automático de nós, consulte 44-node-repair.yaml no GitHub.

O Amazon EKS fornece um controle mais detalhado sobre o comportamento de reparo automático de nós por meio dos seguintes itens:

- `maxUnhealthyNodeThresholdCount` e da `maxUnhealthyNodeThresholdPercentage`
  - Com esses campos, você pode definir um limite para o número ou porcentagem de nós com problemas de integridade. Caso esse limite seja ultrapassado, as ações de reparo automático de nós são suspensas. Isso garante maior controle sobre o “raio de impacto” dos reparos automáticos.
  - É possível configurar a contagem absoluta ou a porcentagem, mas nunca as duas simultaneamente.

- `maxParallelNodesRepairedCount` e da `maxParallelNodesRepairedPercentage`
  - Esses campos permitem especificar o número máximo de nós que podem ser reparados simultaneamente ou em paralelo, expresso como um valor absoluto ou uma porcentagem de todos os nós não íntegros. Isso proporciona um controle mais detalhado da velocidade das substituições de nós.
  - Assim como ocorre no limite de nós não íntegros, é possível configurar a contagem absoluta ou a porcentagem, mas nunca as duas simultaneamente.
- `nodeRepairConfigOverrides`
  - Esta é uma estrutura complexa que permite configurar substituições granulares para ações de reparo específicas. Essas substituições controlam a ação de reparo e o tempo de espera antes que um nó seja considerado apto para o processo de reparo.
  - Os campos específicos nesta estrutura são:
    - `nodeMonitoringCondition`: a condição de não integridade relatada pelo agente de monitoramento do nó.
    - `nodeUnhealthyReason`: o motivo pelo qual o agente de monitoramento do nó identificou o nó como não íntegro.
    - `minRepairWaitTimeMins`: o tempo mínimo (em minutos) que a condição de reparo e o motivo da não integridade devem persistir antes que o nó seja elegível para reparo.
    - `repairAction`: a ação que o sistema de reparo deve realizar quando as condições acima forem atendidas.
  - Ao utilizar este campo, é necessário especificar todos os campos na estrutura. Além disso, você pode fornecer uma lista dessas substituições.
  - O `nodeMonitoringCondition` e o `nodeUnhealthyReason` são campos de entrada de texto manual que você define para indicar que deseja divergir do comportamento padrão do sistema.
  - Os campos `minRepairWaitTimeMins` e `repairAction` permitem que você especifique divergências em relação ao comportamento padrão do sistema.

Problemas de integridade de nós

As tabelas a seguir descrevem problemas de integridade de nós que podem ser detectados pelo agente de monitoramento de nós. Há dois tipos de problemas:

- Condição: um problema terminal que justifica uma ação de remediação, como a substituição ou reinicialização de uma instância. Quando o reparo automático estiver habilitado, o Amazon EKS

executará uma ação de reparo, seja uma substituição ou uma reinicialização do nó. Para obter mais informações, consulte the section called “Condições de nós”.

- Evento: um problema temporário ou uma configuração de nós abaixo do ideal. Nenhuma ação de reparo automático ocorrerá. Para obter mais informações, consulte the section called “Eventos de nós”.

Problemas de integridade de nós com AcceleratedHardware

Para os problemas listados na tabela abaixo com gravidade do tipo “Condição”, a condição de monitoramento utilizada é `AcceleratedHardwareReady`.

Se o reparo automático estiver habilitado, as ações de reparo listadas serão iniciadas dez minutos após a detecção do problema. Para obter mais informações sobre erros do XID, consulte Xid Errors na Documentação de implantação e gerenciamento da GPU NVIDIA. Para obter mais informações sobre as mensagens individuais do XID, consulte Understanding Xid Messages na Documentação de implantação e gerenciamento da GPU NVIDIA.

Nome	Gravidade	Descrição
DCGMDiagnosticFailure	Condição	Um caso de teste do conjunto de testes de diagnóstico ativo do DCGM falhou.
DCGMError	Condição	Houve perda de conexão com o processo host do DCGM ou não foi possível estabelecê-la.
DCGMFieldError[Code]	Event	O DCGM detectou degradaçã o da GPU por meio de um identificador de campo.
DCGMHealthCode[Code]	Event	Uma verificação de integrida de do DCGM falhou de forma não fatal.
DCGMHealthCode[Code]	Condição	Uma verificação de integrida de do DCGM falhou de forma fatal.

Nome	Gravidade	Descrição
NeuronDMAError	Condição	Um mecanismo de DMA encontrou um erro irrecuperável.
NeuronHBMUncorrectableError	Condição	Um HBM encontrou um erro incorrigível e produziu resultados incorretos.
NeuronNCUncorrectableError	Condição	Foi detectado um erro de memória incorrigível do Neuron Core.
NeuronSRAMUncorrectableError	Condição	Uma SRAM no chip encontrou um erro de paridade e produziu resultados incorretos.
NvidiaDeviceCountMismatch	Event	O número de GPUs visíveis por meio da biblioteca NVML é inconsistente com o número de dispositivos da NVIDIA no sistema de arquivos.
NvidiaDoubleBitError	Condição	Um erro de dois bits foi produzido pelo driver da GPU.
NvidiaNCCLError	Event	Ocorreu uma falha de segmentação na NVIDIA Collective Communications Library (`libnccl`).
NvidiaNVLinkError	Condição	Erros do NVLink foram relatados pelo driver da GPU.
NvidiaPCIeError	Event	As retransmissões de PCIe foram acionadas para a recuperação de erros de transmissão.

Nome	Gravidade	Descrição
NvidiaPageRetirement	Event	O driver da GPU marcou uma página de memória para retirada. Isso poderá ocorrer se houver um único erro de bit duplo ou se dois erros de bit único forem encontrados no mesmo endereço.
NvidiaPowerError	Event	O uso de energia pelas GPUs ultrapassou os limites permitidos.
NvidiaThermalError	Event	O status térmico das GPUs ultrapassou os limites permitidos.
NvidiaXID[Code]Error	Condição	Ocorreu um erro crítico relacionado com a GPU.
NvidiaXID[Code]Warning	Event	Ocorreu um erro não crítico relacionado com a GPU.

Problemas de integridade de nós com ContainerRuntime

Para os problemas listados na tabela abaixo com gravidade do tipo “Condição”, a condição de monitoramento utilizada é `ContainerRuntimeReady`.

Nome	Gravidade	Descrição
ContainerRuntimeFailed	Event	O runtime do contêiner falhou ao criar um contêiner , provavelmente relaciona do a quaisquer problemas relatados, caso tenham ocorrido repetidamente.

Nome	Gravidade	Descrição
DeprecatedContainerdConfiguration	Event	Uma imagem de contêiner usando a versão do manifesto de imagem obsoleta (versão 2 e esquema 1) foi extraída recentemente para o nó por meio do `containerd` .
KubeletFailed	Event	O kubelet entrou em um estado de falha.
LivenessProbeFailures	Event	Uma falha na sonda de liveness foi detectada, possivelmente indicando problemas no código da aplicação ou valores de tempo limite insuficientes, caso tenham ocorrido repetidam ente.
PodStuckTerminating	Condição	Um pod está ou estava preso no encerramento por um tempo excessivo, o que pode ser causado por erros de CRI impedindo a progressão do estado do pod.
ReadinessProbeFailures	Event	Uma falha na sonda de readiness foi detectada, possivelmente indicando problemas no código da aplicação ou valores de tempo limite insuficientes, caso tenham ocorrido repetidam ente.

Nome	Gravidade	Descrição
[Name]RepeatedRestart	Event	Uma unidade do systemd está reiniciando com frequência.
ServiceFailedToStart	Event	Uma unidade do systemd falhou ao iniciar.

Problemas de integridade de nós do kernel

Para os problemas listados na tabela abaixo com gravidade do tipo “Condição”, a condição de monitoramento utilizada é `KernelReady`.

Nome	Gravidade	Descrição
AppBlocked	Event	A tarefa foi bloqueada da programação por um longo período, o que geralmente ocorre devido a um bloqueio na entrada ou na saída.
AppCrash	Event	Uma aplicação no nó falhou.
ApproachingKernelPidMax	Event	A quantidade de processos está chegando ao limite máximo de PIDs permitido pela atual configuração `kernel.pid_max` do sistema. Se esse limite for atingido, não será possível iniciar novos processos.
ApproachingMaxOpenFiles	Event	O número de arquivos abertos está se aproximando do número máximo de arquivos abertos possíveis, de acordo com as configurações atuais

Nome	Gravidade	Descrição
		do kernel, após o qual a abertura de novos arquivos falhará.
ConntrackExceededKernel	Event	O rastreamento de conexões excedeu o máximo para o kernel e novas conexões não puderam ser estabelecidas, o que pode resultar em perda de pacotes.
ExcessiveZombieProcesses	Event	Processos que não podem ser totalmente recuperados estão se acumulando em grandes números, o que indica problemas na aplicação e pode levar a atingir os limites de processos do sistema.
ForkFailedOutOfPIDs	Condição	Uma chamada fork ou exec falhou porque o sistema ficou sem IDs de processo ou memória, o que pode ser causado por processos zumbis ou esgotamento da memória física.
KernelBug	Event	Um bug do kernel foi detectado e relatado pelo próprio Kernel Linux, embora às vezes isso possa ser causado por nós com alto uso de CPU ou memória, levando ao atraso no processamento de eventos.

Nome	Gravidade	Descrição
LargeEnvironment	Event	O número de variáveis de ambiente para este processo é maior do que o esperado. Isso pode ser causado pelo excesso de serviços com a configuração `enableServiceLinks` definida como “verdadeira”, resultando em possíveis problemas de performance.
RapidCron	Event	Um cron job está sendo executado com uma frequência maior que a cada cinco minutos neste nó, o que poderá afetar a performance se o trabalho consumir recursos significativos.
SoftLockup	Event	A CPU ficou paralisada por um determinado período de tempo.

Problemas de integridade de nós de rede

Para os problemas listados na tabela abaixo com gravidade do tipo “Condição”, a condição de monitoramento utilizada é `NetworkingReady`.

Nome	Gravidade	Descrição
BandwidthInExceeded	Event	Os pacotes foram enfileirados ou descartados porque a largura de banda agregada

Nome	Gravidade	Descrição
		de entrada excedeu o máximo para a instância.
BandwidthOutExceeded	Event	Os pacotes foram enfileira dos ou descartados porque a largura de banda agregada de saída excedeu o máximo para a instância.
ConntrackExceeded	Event	O rastreamento de conexões excedeu o máximo para a instância e novas conexões não puderam ser estabelec idas, o que pode resultar em perda de pacotes.
IPAMDInconsistentState	Event	O arquivo de ponto de verificação do IPAMD, no disco, está inconsistente com os endereços IP registrados no runtime do contêiner.
IPAMDNoIPs	Event	O IPAMD está sem endereços IP disponíveis.
IPAMDNotReady	Condição	O IPAMD não consegue se conectar ao servidor da API.
IPAMDNotRunning	Condição	Não foi possível localizar o processo da CNI do Amazon VPC em execução no sistema.
IPAMDRepeatedlyRestart	Event	Ocorreram várias reinicial izações no serviço do IPAMD.

Nome	Gravidade	Descrição
InterfaceNotRunning	Condição	Esta interface parece não estar funcionando ou há problemas de rede.
InterfaceNotUp	Condição	Esta interface parece não estar ativa ou há problemas de rede.
KubeProxyNotReady	Event	O kube-proxy falhou ao observar ou listar recursos.
LinkLocalExceeded	Event	Os pacotes foram descartad os porque o PPS do tráfego para os serviços de proxy local excedeu o máximo para a interface da rede.
MACAddressPolicyMisconfigur ed	Event	O valor definido para `MACAddressPolicy` na configuração de link do systemd-networkd está incorreto.
MissingDefaultRoutes	Event	Há regras de rota padrão ausentes.
MissingIPRoutes	Event	Existem rotas ausentes para os endereços IP dos pods.
MissingIPRules	Event	Existem regras ausentes para os endereços IP dos pods.
MissingLoopbackInterface	Condição	A interface de loopback está ausente nesta instância, causando falha nos serviços que dependem da conectivi dade local.

Nome	Gravidade	Descrição
NetworkSysctl	Event	As configurações de `sysctl` da rede deste nó podem estar configuradas incorretamente.
PPSExceeded	Event	Os pacotes foram enfileirados ou descartados porque o PPS bidirecional excedeu o máximo para a instância.
PortConflict	Event	Se um pod usar hostPort, ele pode gravar regras de `iptables` que substituem as portas já vinculadas do host. Isso pode bloquear a comunicação entre o servidor de API e o `kubelet`.
UnexpectedRejectRule	Event	Uma regra inesperada de `REJECT` ou `DROP` foi encontrada no `iptables`, o que pode estar bloqueando o tráfego esperado.

Problemas de integridade do nó de armazenamento

Para os problemas listados na tabela abaixo com gravidade do tipo “Condição”, a condição de monitoramento utilizada é `StorageReady`.

Nome	Gravidade	Descrição
EBSInstanceIOPSExceeded	Event	O limite máximo de IOPS da instância foi excedido.
EBSInstanceThroughputExceeded	Event	O limite máximo de throughput da instância foi excedido.

Nome	Gravidade	Descrição
EBSVolumeIOPSExceeded	Event	O limite máximo de IOPS de um volume do EBS específico foi excedido.
EBSVolumeThroughputExceeded	Event	O limite máximo de throughput de um volume do Amazon EBS específico foi excedido.
EtcHostsMountFailed	Event	A montagem do arquivo `/etc/hosts` gerado pelo kubelet falhou devido à remontagem de `/var/lib/kubelet/pods` via userdata durante a operação `kubelet-container` .
IODelays	Event	Atraso de entrada ou saída detectado em um processo, possivelmente indicando provisionamento insuficiente de entrada-saída, se excessivo.
KubeletDiskUsageSlow	Event	O `kubelet` está relatando lentidão no uso do disco ao tentar acessar o sistema de arquivos. Isso pode indicar que as operações de entrada e de saída do disco são insuficientes ou que há problemas no sistema de arquivos.

Nome	Gravidade	Descrição
XFSSmallAverageClusterSize	Event	O tamanho médio do cluster XFS é pequeno, indicando fragmentação excessiva do espaço livre. Isso pode impedir a criação de arquivos, mesmo que haja inodes ou espaço livre disponíveis.

Visualizar o status de integridade dos nós

Este tópico explica as ferramentas e os métodos disponíveis para monitorar o status de integridade dos nós nos clusters do Amazon EKS. As informações abrangem condições, eventos e casos de detecção de nós que ajudam a identificar e diagnosticar problemas no nível do nó. Use os comandos e padrões descritos aqui para inspecionar os recursos de integridade do nó, interpretar as condições de status e analisar os eventos do nó para solucionar problemas operacionais.

Você pode obter algumas informações sobre a integridade dos nós com comandos do Kubernetes para todos os nós. E se você usar o agente de monitoramento de nós por meio do Modo Automático do Amazon EKS ou do complemento gerenciado do Amazon EKS, você obterá uma variedade maior de sinais de nós para ajudar na solução de problemas. As descrições dos problemas de integridade detectados pelo agente de monitoramento de nós também são disponibilizadas no painel de observabilidade. Para ter mais informações, consulte the section called “Integridade de nós”.

Condições de nós

As condições do nó representam problemas terminais que exigem ações de correção, como a substituição ou reinicialização da instância.

Para obter as condições de todos os nós:

```
kubectl get nodes -o 'custom-
columns=NAME:.metadata.name,CONDITIONS:.status.conditions[*].type,STATUS:.status.conditions[*].
```

Para obter as condições detalhadas de um nó específico:

```
kubectl describe node node-name
```

Exemplo de saída de condição de um nó íntegro:

```
  - lastHeartbeatTime: "2024-11-21T19:07:40Z"
    lastTransitionTime: "2024-11-08T03:57:40Z"
    message: Monitoring for the Networking system is active
    reason: NetworkingIsReady
    status: "True"
    type: NetworkingReady
```

Exemplo de condição de um nó não íntegro com um problema de rede:

```
  - lastHeartbeatTime: "2024-11-21T19:12:29Z"
    lastTransitionTime: "2024-11-08T17:04:17Z"
    message: IPAM-D has failed to connect to API Server which could be an issue with
      IPTable rules or any other network configuration.
    reason: IPAMDNotReady
    status: "False"
    type: NetworkingReady
```

Eventos de nós

Os eventos de nós indicam problemas temporários ou configurações abaixo do ideal.

Para obter todos os eventos relatados pelo agente de monitoramento de nós

Quando o agente de monitoramento de nós estiver disponível, você poderá executar o comando a seguir.

```
kubectl get events --field-selector=reportingComponent=eks-node-monitoring-agent
```

Exemplo de resultado:

```
LAST SEEN   TYPE      REASON       OBJECT
 MESSAGE
4s          Warning   SoftLockup   node/ip-192-168-71-251.us-west-2.compute.internal
 CPU stuck for 23s
```

Para obter os eventos de todos os nós

```
kubectl get events --field-selector involvedObject.kind=Node
```

Para obter os eventos para um nó específico

```
kubectl get events --field-selector involvedObject.kind=Node,involvedObject.name=node-name
```

Para observar os eventos em tempo real

```
kubectl get events -w --field-selector involvedObject.kind=Node
```

Exemplo de saída de evento:

```
LAST SEEN   TYPE     REASON           OBJECT         MESSAGE
2m          Warning  MemoryPressure   Node/node-1    Node experiencing memory pressure
5m          Normal   NodeReady        Node/node-1    Node became ready
```

Comandos comuns de solução de problemas

```
# Get comprehensive node status
kubectl get node node-name -o yaml

# Watch node status changes
kubectl get nodes -w

# Get node metrics
kubectl top node
```

Recuperar logs de nós de um nó gerenciado usando kubectl e S3

Saiba como recuperar logs de nós de um nó gerenciado do Amazon EKS que tem o agente de monitoramento de nós.

Pré-requisitos

Certifique-se de ter o seguinte:

- Um cluster existente do Amazon EKS com o agente de monitoramento de nós. Para obter mais informações, consulte the section called “Integridade de nós”.

- A ferramenta de linha de comandos da `kubectl` instalada e configurada para se comunicar com o cluster.
- A AWS CLI instalada e conectada com permissões suficientes para criar buckets e objetos do S3.
- Uma versão recente do Python 3 instalada
- O AWS SDK para Python 3, Boto 3, instalado.

Etapa 1: criar um bucket de destino do S3 (opcional)

Caso ainda não tenha um bucket do S3 para armazenar os logs, crie um. Use o comando da AWS CLI a seguir. O bucket usa como padrão a lista de controle de acesso `private`. Substitua *`bucket-name`* pelo nome exclusivo do bucket escolhido.

```
aws s3api create-bucket --bucket <bucket-name>
```

Etapa 2: criar um URL do S3 pré-assinado para HTTP PUT

O Amazon EKS retorna os logs dos nós fazendo uma operação HTTP PUT em um URL especificado por você. Neste tutorial, vamos gerar um URL de HTTP PUT do S3 pré-assinado.

Os logs serão retornados como um tarball gzip, com a extensão `.tar.gz`.

> **Note**
>
> Você deve usar a API ou um SDK da AWS para criar o URL de upload do S3 pré-assinado para que o EKS faça o upload do arquivo de logs. Você não pode criar um URL de upload do S3 pré-assinado usando a AWS CLI.

1. Determine em que parte do bucket você deseja armazenar os logs. Por exemplo, você pode usar *`2024-11-12/logs1.tar.gz`* como a chave.
2. Salve o código Python abaixo no arquivo *`presign-upload.py`*. Substitua *`<bucket-name>`* e *`<key>`*. A chave deve terminar com `.tar.gz`.

```
import boto3; print(boto3.client('s3').generate_presigned_url(
   ClientMethod='put_object',
   Params={'Bucket': '[.replaceable]`<bucket-name>`', 'Key':
 '[.replaceable]`<key>`'},
```

```
    ExpiresIn=[.replaceable]`1000`
))
```

3. Execute o script com

```
python presign-upload.py
```

4. Observe a saída do URL. Use esse valor na próxima etapa como `http-put-destination`.

Para obter mais informações, consulte Generate a presigned URL to upload a file na documentação do AWS Boto3 SDK para Python.

Etapa 3: criar um recurso NodeDiagnostic

Identifique o nome do nós do qual deseja coletar os logs.

Crie um manifesto `NodeDiagnostic` que use o nome do nó como nome do recurso e forneça um destino de URL de HTTP PUT.

```
apiVersion: eks.amazonaws.com/v1alpha1
kind: NodeDiagnostic
metadata:
    name: <node-name>
spec:
    logCapture:
        destination: http-put-destination
```

Aplique o manifesto ao cluster.

```
kubectl apply -f nodediagnostic.yaml
```

Você pode verificar o status da coleta descrevendo o recurso `NodeDiagnostic`:

- Um status de `Success` ou `SuccessWithErrors` indica que a tarefa foi concluída e os logs foram enviados para o destino fornecido (`SuccessWithErrors` indica que alguns logs podem estar ausentes)
- Se o status for Falha, confirme se o URL de upload está formado de forma correta e não expirou.

```
kubectl describe nodediagnostics.eks.amazonaws.com/<node-name>
```

Etapa 4: fazer download dos logs do S3

Aguarde aproximadamente um minuto antes de tentar fazer download dos logs. Em seguida, use a CLI do S3 para fazer download dos logs.

```
# Once NodeDiagnostic shows Success status, download the logs
aws s3 cp s3://<bucket-name>/key ./<path-to-node-logs>.tar.gz
```

Etapa 5: limpar o recurso NodeDiagnostic

- Os recursos `NodeDiagnostic` não são excluídos automaticamente. Você deve limpá-los por conta própria depois de obter os artefatos de logs.

```
# Delete the NodeDiagnostic resource
kubectl delete nodediagnostics.eks.amazonaws.com/<node-name>
```

Visão geral do Amazon EKS Hybrid Nodes

Com o Amazon EKS Hybrid Nodes, você pode usar a infraestrutura on-premises e de borda como nós nos clusters do Amazon EKS. A AWS gerencia o ambiente de gerenciamento do Kubernetes hospedado na AWS do cluster do Amazon EKS, e você gerencia os nós híbridos que são executados nos ambientes on-premises ou de borda. Isso unifica o gerenciamento do Kubernetes nos ambientes e transfere o gerenciamento do ambiente de gerenciamento do Kubernetes para a AWS das aplicações on-premises e de borda.

O Amazon EKS Hybrid Nodes funciona com qualquer hardware on-premises ou máquina virtual, trazendo a eficiência, a escalabilidade e a disponibilidade do Amazon EKS para onde quer que as aplicações precisem ser executadas. Você pode usar uma ampla variedade de recursos do Amazon EKS com o Amazon EKS Hybrid Nodes, incluindo complementos do Amazon EKS, a Identidade de Pods do Amazon EKS, entradas de acesso ao cluster, insights do cluster e suporte estendido à versão do Kubernetes. O Amazon EKS Hybrid Nodes se integra de forma nativa com os serviços da AWS, incluindo o AWS Systems Manager, o AWS IAM Roles Anywhere, o Amazon Managed Service for Prometheus e o Amazon CloudWatch para monitoramento centralizado, registro em log e gerenciamento de identidades.

Com o Amazon EKS Hybrid Nodes, não há compromissos iniciais nem taxas mínimas, e você é cobrado por hora pelos recursos de vCPU dos nós híbridos quando eles estão anexados aos clusters do Amazon EKS. Para obter informações sobre preços, consulte Preços do Amazon EKS.

Atributos

O EKS Hybrid Nodes apresenta os seguintes recursos de alto nível:

- Ambiente de gerenciamento do Kubernetes gerenciado: o AWS gerencia o ambiente de gerenciamento do Kubernetes hospedado no AWS do cluster do EKS, e você gerencia os nós híbridos que são executados nos seus ambientes on-premises ou de borda. Isso unifica o gerenciamento do Kubernetes nos ambientes e transfere o gerenciamento do ambiente de gerenciamento do Kubernetes para a AWS das aplicações on-premises e de borda. Ao mover o ambiente de gerenciamento do Kubernetes para a AWS, você pode conservar a capacidade on-premises para suas aplicações e confiar que o ambiente de gerenciamento do Kubernetes escalará com suas workloads.
- Experiência de EKS consistente: a maioria dos recursos do EKS é compatível com o EKS Hybrid Nodes para proporcionar uma experiência de EKS consistente em ambientes on-premises e na nuvem, incluindo complementos do EKS, Identidade de Pods do EKS, entradas de acesso ao cluster, insights do cluster, suporte estendido a versões do Kubernetes e muito mais. Consulte the section called “Configurar complementos” para obter mais informações sobre os complementos do EKS compatíveis com o EKS Hybrid Nodes.
- Observabilidade e gerenciamento de identidades centralizados: o EKS Hybrid Nodes se integra de forma nativa com os serviços da AWS, incluindo o AWS Systems Manager, o AWS IAM Roles Anywhere, o Amazon Managed Service for Prometheus e o Amazon CloudWatch para monitoramento centralizado, registro em log e gerenciamento de identidades.
- Burst-to-cloud ou adição de capacidade on-premises: um único cluster do EKS pode ser usado para executar nós híbridos e nós em regiões da AWS, zonas Locais da AWS ou AWS Outposts para expandir para a nuvem em caso de necessidade (burst-to-cloud) ou adicionar capacidade on-premises aos seus clusters do EKS. Consulte Considerações sobre clusters de modo misto para obter mais informações.
- Infraestrutura flexível: o EKS Hybrid Nodes segue uma abordagem de traga sua própria infraestrutura e é agnóstico em relação à infraestrutura utilizada para nós híbridos. É possível executar nós híbridos em máquinas físicas ou virtuais e nas arquiteturas x86 e ARM, o que torna possível a migração de workloads on-premises executadas em nós híbridos entre diferentes tipos de infraestrutura.

- Rede flexível: com o EKS Hybrid Nodes, a comunicação entre o ambiente de gerenciamento do EKS e os nós híbridos é roteada por meio da VPC e das sub-redes que você passa durante a criação do cluster, o que se baseia no mecanismo existente no EKS para a conexão de rede entre o ambiente de gerenciamento e o nó. Isso é flexível quanto ao seu método de preferência de conectar as redes on-premises a uma VPC na AWS. Há várias opções documentadas disponíveis, incluindo o AWS VPN Site-to-Site, o AWS Direct Connect e sua própria solução de VPN, e você pode escolher o método que melhor se adapta ao seu caso de uso.

Limites

- Você pode usar até 15 CIDRs para redes remotas de nós e 15 CIDRs para redes remotas de pods por cluster.

Considerações

- O EKS Hybrid Nodes pode ser usado com clusters existentes ou novos do EKS.
- O EKS Hybrid Nodes está disponível em todas as regiões da AWS, exceto nas regiões AWS GovCloud (EUA) e nas regiões da AWS na China.
- O EKS Hybrid Nodes deve ter uma conexão confiável entre o ambiente on-premises e a AWS. O EKS Hybrid Nodes não é adequado para ambientes desconectados, interrompidos, intermitentes ou limitados (DDIL). Se você estiver executando em um ambiente DDIL, considere o Amazon EKS Anywhere. Consulte as práticas recomendadas do EKS Hybrid Nodes para obter informações sobre como os nós híbridos se comportam durante cenários de desconexão de rede.
- Não há suporte à execução do EKS Hybrid Nodes em infraestrutura de nuvem, incluindo as regiões da AWS, as zonas locais da AWS, o AWS Outposts ou outras nuvens. Você será cobrado pela taxa de nós híbridos se executar nós híbridos em instâncias do Amazon EC2.
- A cobrança dos nós híbridos começa quando os nós se unem ao cluster do EKS e termina quando os nós são removidos do cluster. Certifique-se de remover os nós híbridos do cluster do EKS caso não os esteja usando.

Recursos adicionais

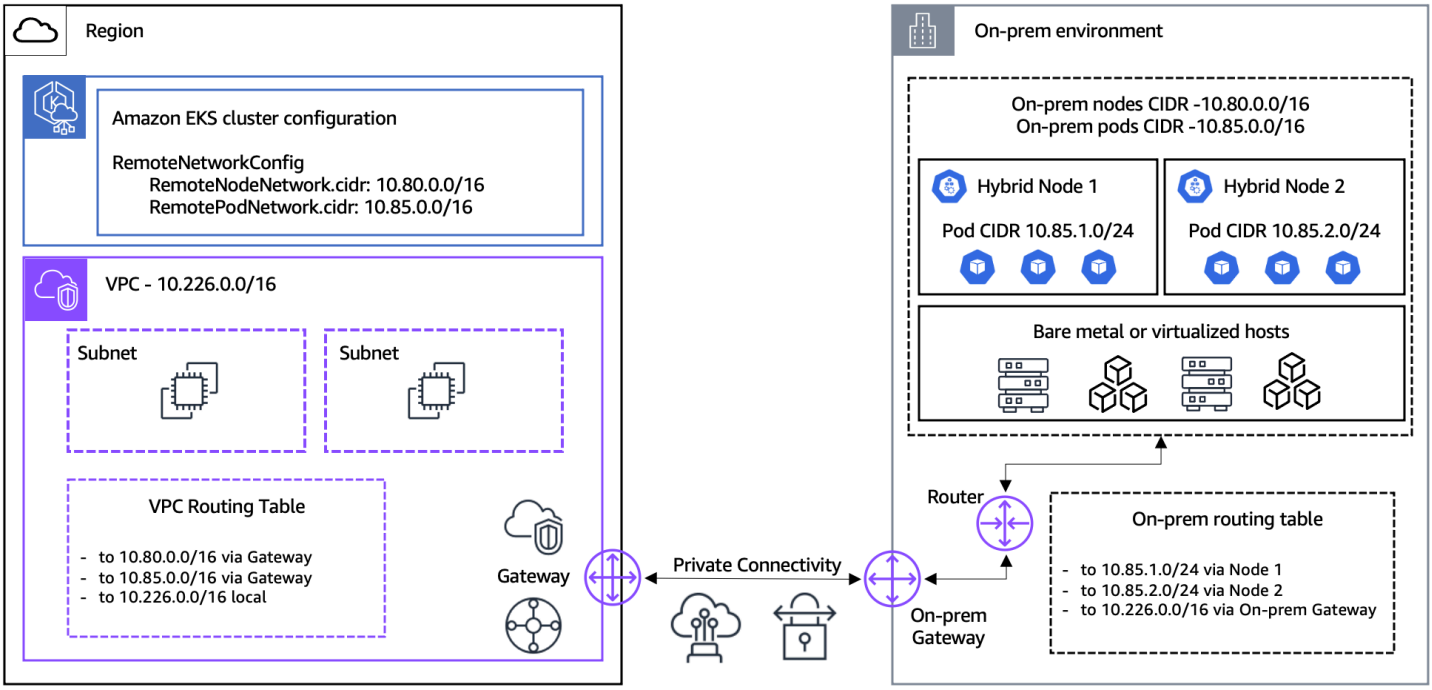
- Workshop sobre o EKS Hybrid Nodes: instruções passo a passo para implantar o EKS Hybrid Nodes em um ambiente de demonstração.

- AWS re:Invent: EKS Hybrid Nodes: sessão do AWS re:Invent apresentando o lançamento do EKS Hybrid Nodes com um cliente, mostrando como ele está usando os EKS Hybrid Nodes em seu ambiente.
- AWS re:Post: Cluster networking for EKS Hybrid Nodes: artigo explicando vários métodos para configurar redes para o EKS Hybrid Nodes.
- Blog da AWS: Run GenAI inference across environments with EKS Hybrid Nodes: postagem no blog mostrando como executar a inferência de IA generativa em ambientes com o EKS Hybrid Nodes.

Configuração de pré-requisitos para nós híbridos

Para usar o Amazon EKS Hybrid Nodes, você deve ter uma conectividade privada do ambiente on-premises de/para a AWS, servidores bare metal ou máquinas virtuais com um sistema operacional compatível e ativações híbridas do AWS IAM Roles Anywhere ou do AWS Systems Manager (SSM) configuradas. Você é responsável por gerenciar esses pré-requisitos durante todo o ciclo de vida dos nós híbridos.

- Conectividade de rede híbrida do ambiente on-premises de/para a AWS
- Infraestrutura na forma de máquinas físicas ou virtuais
- Sistema operacional compatível com nós híbridos
- Provedor de credenciais do IAM on-premises configurado



Conectividade de rede híbrida

A comunicação entre o ambiente de gerenciamento do Amazon EKS e os nós híbridos é roteada pela VPC e pelas sub-redes que você passa durante a criação do cluster, que se baseia no mecanismo existente no Amazon EKS do ambiente de gerenciamento para a rede de nós. Há várias opções documentadas disponíveis para você conectar o ambiente on-premises à VPC, incluindo o AWS VPN Site-to-Site, o AWS Direct Connect ou sua própria conexão da VPN. Consulte os guias de usuário do AWS VPN Site-to-Site e do AWS Direct Connect para obter mais informações sobre como usar essas soluções para a conexão de rede híbrida.

Para uma experiência ideal, recomendamos uma conectividade de rede confiável de pelo menos 100 Mbps e uma latência máxima de ida e vinda de 200 ms para a conexão dos nós híbridos com a região da AWS. Esta é uma orientação geral que engloba a maioria dos casos de uso, mas não é um requisito rigoroso. Os requisitos de largura de banda e latência podem variar dependendo do número de nós híbridos e das características da workload, como tamanho da imagem da aplicação, elasticidade da aplicação, configurações de monitoramento e registro em log e dependências da aplicação no acesso aos dados armazenados em outros serviços da AWS. Recomendamos que você teste com suas próprias aplicações e seus próprios ambientes antes da implantação na produção para validar se a configuração de rede atende aos requisitos das workloads.

Configuração de rede on-premises

Você deve habilitar o acesso à rede de entrada no ambiente de gerenciamento do Amazon EKS para o ambiente on-premises a fim de permitir que o ambiente de gerenciamento do Amazon EKS se comunique com o `kubelet` em execução em nós híbridos e, opcionalmente, com webhooks em execução nos nós híbridos. Além disso, você deve habilitar o acesso à rede de saída para que os nós híbridos e os componentes executados neles se comuniquem com o ambiente de gerenciamento do Amazon EKS. Você pode configurar essa comunicação para permanecer totalmente privada no AWS Direct Connect, no AWS VPN Site-to-Site ou em sua própria conexão da VPN.

Os intervalos de Encaminhamento Entre Domínios Sem Classificação (CIDR) que você usa para as redes on-premises de nós e pods devem usar intervalos de endereços IPv4 RFC-1918. O roteador on-premises deve ser configurado com rotas para os nós on-premises e, opcionalmente, pods. Consulte the section called “Configuração de rede on-premises” para obter mais informações sobre os requisitos de rede on-premises, incluindo a lista completa de portas e protocolos necessários que devem ser habilitados em seu firewall e ambiente on-premises.

Configuração do cluster do EKS

Para minimizar a latência, recomendamos criar o cluster do Amazon EKS na região da AWS mais próxima do ambiente on-premises ou de borda. Você passa os CIDRs de nós e pods on-premises durante a criação do cluster do Amazon EKS por meio de dois campos de API: `RemoteNodeNetwork` e `RemotePodNetwork`. Talvez seja necessário discutir com a equipe de rede on-premises para identificar os CIDRs de nós e pods on-premises. O CIDR de nós é alocado da rede on-premises, e o CIDR de pods é alocado da interface de rede de contêineres (CNI) que você usa caso esteja utilizando uma rede de sobreposição para a CNI. O Cilium e o Calico usam redes de sobreposição por padrão.

Os CIDRs de nós e pods on-premises que você configura nos campos `RemoteNodeNetwork` e `RemotePodNetwork` são usados para configurar o ambiente de gerenciamento do Amazon EKS para rotear o tráfego por meio da VPC para o `kubelet` e os pods em execução nos nós híbridos. Os CIDRs de nós e pods on-premises não podem se sobrepor entre si, com o CIDR da VPC que você passa durante a criação do cluster ou com a configuração do IPv4 do serviço para o cluster do Amazon EKS. Além disso, os Pod CIDRs devem ser exclusivos para cada cluster EKS para que seu roteador on-premises possa rotear o tráfego.

Recomendamos usar o acesso de endpoint público ou privado para o endpoint do servidor da API do Kubernetes do Amazon EKS. Se você escolher “Público e Privado”, o endpoint do servidor da

API do Kubernetes do Amazon EKS sempre resolverá para os IPs públicos dos nós híbridos em execução fora da VPC, o que pode impedir que os nós híbridos se juntem ao cluster. Quando você usa o acesso ao endpoint público, o endpoint do servidor da API do Kubernetes é resolvido para IPs públicos, e a comunicação dos nós híbridos com o ambiente de gerenciamento do Amazon EKS será roteada pela internet. Quando você escolhe o acesso privado ao endpoint, o endpoint do servidor da API do Kubernetes é resolvido para IPs privados, e a comunicação dos nós híbridos com o ambiente de gerenciamento do Amazon EKS será roteada pelo link de conectividade privada, na maioria dos casos o AWS Direct Connect ou o AWS Site-to-Site VPN.

Configuração de VPC

Você deve configurar a VPC que você passa durante a criação do cluster do Amazon EKS com rotas em sua tabela de roteamento para o nó on-premises e, opcionalmente, redes de pods como o gateway privado virtual (VGW) ou o gateway de trânsito (TGW) como destino. Um exemplo é mostrado abaixo. Substitua `REMOTE_NODE_CIDR` e `REMOTE_POD_CIDR` pelos valores da rede on-premises.

Destino	Alvo	Descrição
10,226.0.0/16	local	O tráfego local para as rotas da VPC dentro da VPC
REMOTE_NODE_CIDR	tgw-abcdef123456	CIDR de nós on-premises, rotear tráfego para o TGW
REMODE_POD_CIDR	tgw-abcdef123456	CIDR de pods on-premises, rotear tráfego para o TGW

Configuração do security group

Ao criar um cluster, o Amazon EKS cria um grupo de segurança com o nome `eks-cluster-sg-<cluster-name>-<uniqueID>`. Você não pode alterar as regras de entrada desse grupo de segurança de clusters, mas pode restringir as regras de saída. Você deve adicionar um grupo de segurança adicional ao cluster para permitir que o kubelet e, opcionalmente, os webhooks em execução nos nós híbridos façam contato com o ambiente de gerenciamento do Amazon EKS. As regras de entrada necessárias para esse grupo de segurança adicional são mostradas abaixo. Substitua `REMOTE_NODE_CIDR` e `REMOTE_POD_CIDR` pelos valores da rede on-premises.

Name	ID da regra do grupo de segurança	Versão de IP	Tipo	Protocolo	Intervalo de portas	Origem
Entrada de nós on-premises	sgr-abcdef123456	IPv4	HTTPS	TCP	443	REMOTE_NODE_CIDR
Entrada de pods on-premises	sgr-abcdef654321	IPv4	HTTPS	TCP	443	REMOTE_POD_CIDR

Infraestrutura

Você deve ter servidores bare metal ou máquinas virtuais disponíveis para uso como nós híbridos. Os nós híbridos são independentes da infraestrutura subjacente e são compatíveis com as arquiteturas x86 e ARM. O Amazon EKS Hybrid Nodes segue uma abordagem de “traga sua própria infraestrutura”, em que você é responsável por provisionar e gerenciar os servidores bare metal ou as máquinas virtuais que você usa para nós híbridos. Embora não haja um requisito mínimo rigoroso de recursos, recomendamos usar hosts com pelo menos 1 vCPU e 1 GiB de RAM para nós híbridos.

Sistema operacional

O Bottlerocket, o Amazon Linux 2023 (AL2023), o Ubuntu e o RHEL são validados continuamente para uso como o sistema operacional de nós para nós híbridos. O Bottlerocket tem suporte da AWS somente em ambientes VMware vSphere. O AL2023 não é coberto pelos planos do AWS Support quando executado fora do Amazon EC2. O AL2023 só pode ser usado em ambientes virtualizados on-premises. Consulte o Guia do usuário do Amazon Linux 2023 para obter mais informações. A AWS oferece suporte à integração de nós híbridos com os sistemas operacionais Ubuntu e RHEL, mas não oferece suporte para os sistemas operacionais em si.

Você é responsável pelo provisionamento e gerenciamento do sistema operacional. Quando você estiver testando nós híbridos pela primeira vez, é mais fácil executar a CLI (`nodeadm`) do Amazon EKS Hybrid Nodes em um host já provisionado. Para implantações de produção, recomendamos incluir `nodeadm` nas imagens golden do sistema operacional configurado para ser executado

como um serviço systemd para unir automaticamente os hosts aos clusters do Amazon EKS na inicialização do host.

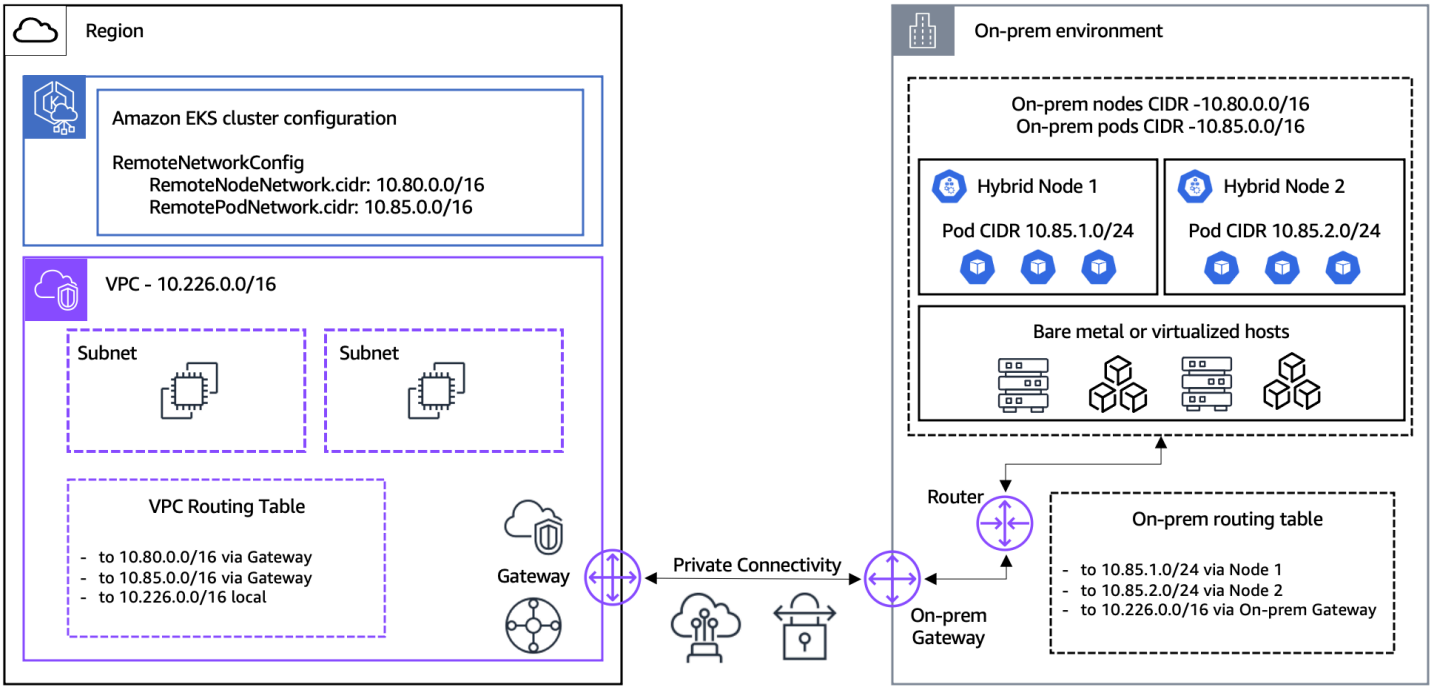
Provedor de credenciais do IAM on-premises

O Amazon EKS Hybrid Nodes usa credenciais temporárias do IAM provisionadas por ativações híbridas do AWS SSM ou pelo AWS IAM Roles Anywhere para se autenticar com o cluster do Amazon EKS. Você deve usar as ativações híbridas do AWS SSM ou o AWS IAM Roles Anywhere com a CLI do Amazon EKS Hybrid Nodes (`nodeadm`). Recomendamos usar as ativações híbridas do AWS SSM caso não tenha uma infraestrutura de chave pública (PKI) existente com uma Certificate Authority (CA) e certificados para seus ambientes on-premises. Se você já tem uma PKI e certificados on-premises, use o AWS IAM Roles Anywhere.

Semelhante ao the section called “Perfil do IAM do nó” para nós em execução no Amazon EC2, você criará um perfil do IAM de nós híbridos com as permissões necessárias para unir nós híbridos aos clusters do Amazon EKS. Caso esteja usando o AWS IAM Roles Anywhere, configure uma política de confiança que permita que o AWS IAM Roles Anywhere assuma o perfil do IAM de nós híbridos, e configure seu usuário do AWS IAM Roles Anywhere com o perfil do IAM de nós híbridos como um perfil assumível. Caso esteja usando o AWS SSM, configure uma política de confiança que permita que o AWS SSM assuma o perfil do IAM de nós híbridos e crie a ativação híbrida com o perfil do IAM de nós híbridos. Confira the section called “Preparar credenciais” para saber como criar o perfil do IAM de nós híbridos com as permissões necessárias.

Preparar a rede para nós híbridos

Este tópico fornece uma visão geral da configuração de rede que você deve configurar antes de criar o cluster do Amazon EKS e anexar nós híbridos. Este guia pressupõe que você atendeu aos pré-requisitos para conectividade de rede híbrida usando a AWS Site-to-Site VPN, o AWS Direct Connect ou sua própria solução de VPN.



Configuração de rede on-premises

Requisitos mínimos de rede

Para uma experiência ideal, recomendamos uma conectividade de rede confiável de pelo menos 100 Mbps e uma latência máxima de ida e vinda de 200 ms para a conexão dos nós híbridos com a região da AWS. Esta é uma orientação geral que engloba a maioria dos casos de uso, mas não é um requisito rigoroso. Os requisitos de largura de banda e latência podem variar dependendo do número de nós híbridos e das características da workload, como tamanho da imagem da aplicação, elasticidade da aplicação, configurações de monitoramento e registro em log e dependências da aplicação no acesso aos dados armazenados em outros serviços da AWS. Recomendamos que você teste com suas próprias aplicações e seus próprios ambientes antes da implantação na produção para validar se a configuração de rede atende aos requisitos das workloads.

CIDRs de nós e pods on-premises

Identifique os CIDRs de nós e pods que você usará para os nós híbridos e as workloads executadas neles. O CIDR de nós é alocado da rede on-premises e o CIDR de pods é alocado da interface de rede de contêineres (CNI), caso esteja usando uma rede de sobreposição para a CNI. Você passa os CIDRs de nós on-premises e, opcionalmente, CIDRs de pods como entradas ao criar o cluster do EKS com os campos `RemoteNodeNetwork` e `RemotePodNetwork`. Seus CIDRs de nós on-premises devem ser roteáveis na rede on-premises. Consulte a seção a seguir para obter informações sobre a roteabilidade do CIDR de pods on-premises.

Os blocos CIDR de nós e pods on-premises devem atender aos seguintes requisitos:

1. Estar dentro de um dos seguintes intervalos de IPv4 RFC-1918: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.
2. Não se sobrepor entre si, com o CIDR da VPC do cluster do EKS ou com o CIDR de IPv4 do serviço do Kubernetes.

Roteamento de rede de pods on-premises

Ao usar o EKS Hybrid Nodes, geralmente recomendamos que você torne seus CIDRs de pods on-premises roteáveis em sua rede on-premises para permitir a comunicação e a funcionalidade completas do cluster entre os ambientes na nuvem e on-premises.

Redes de pods roteáveis

Se você conseguir tornar a rede de pods roteável na rede on-premises, siga as orientações abaixo.

1. Configure o campo `RemotePodNetwork` para o cluster do EKS com o CIDR do pod on-premises, as tabelas de rotas da VPC com o CIDR do pod on-premises e o grupo de segurança do cluster do EKS com o CIDR do pod on-premises.
2. Você pode usar várias técnicas para tornar seu pod CIDR on-premises roteável na rede on-premises, incluindo o Protocolo de Gateway da Borda (BGP), rotas estáticas ou outras soluções de roteamento personalizadas. O BGP é a solução recomendada, pois é mais escalável e fácil de gerenciar do que soluções alternativas, que demandam uma configuração de rota personalizada ou manual. A AWS dá suporte aos recursos do BGP do Cilium e do Calico para anunciar CIDRs do pod. Consulte the section called “Configurar CNI” e the section called “CIDRs de pods remotos roteáveis” para obter mais informações.
3. Os webhooks podem ser executados em nós híbridos, pois o ambiente de gerenciamento do EKS é capaz de se comunicar com os endereços IP de pods atribuídos aos webhooks.
4. As workloads executadas em nós de nuvem podem se comunicar diretamente com workloads executadas em nós híbridos no mesmo cluster do EKS.
5. Outros serviços da AWS, como o AWS Application Load Balancers e o Amazon Managed Service for Prometheus, são capazes de se comunicar com workloads executadas em nós híbridos para equilibrar o tráfego da rede e extrair métricas de pods.

Redes de pods não roteáveis

Se você não conseguir tornar suas redes de pods roteáveis na rede on-premises, siga as orientações abaixo.

1. Os webhooks não podem ser executados em nós híbridos porque eles exigem uma conectividade do ambiente de gerenciamento do EKS com os endereços IP de pods atribuídos aos webhooks. Nesse caso, recomendamos que você execute webhooks em nós de nuvem no mesmo cluster do EKS que seus nós híbridos. Consulte the section called “Configurar webhooks” para obter mais informações.
2. As workloads executadas em nós de nuvem não conseguem se comunicar diretamente com as workloads executadas em nós híbridos ao usar a CNI da VPC para nós de nuvem e o Cilium ou o Calico para nós híbridos.
3. Use a Distribuição de Tráfego de Serviços para manter o tráfego local na zona de onde ele é originado. Para obter mais informações sobre Distribuição de Tráfego de Serviços, consulte the section called “Configurar a Distribuição de Tráfego de Serviços”.
4. Configure sua CNI para usar mascaramento de saída ou conversão de endereços de rede (NAT) para o tráfego de pods quando ela sair de seus hosts on-premises. Esta opção está habilitada por padrão no Cilium. O Calico requer que `natOutgoing` seja definido como `true`.
5. Outros serviços da AWS, como o AWS Application Load Balancers e o Amazon Managed Service for Prometheus, não são capazes de se comunicar com workloads executadas em nós híbridos.

Acesso necessário durante a instalação e atualização de nós híbridos

Você deve ter acesso aos domínios a seguir durante o processo de instalação em que instala as dependências dos nós híbridos nos hosts. Esse processo pode ser feito uma vez quando você está criando as imagens do sistema operacional ou em cada host no runtime. Isso inclui a instalação inicial e quando você atualiza a versão do Kubernetes dos nós híbridos.

Alguns pacotes são instalados usando o gerenciador de pacotes padrão do sistema operacional. Para AL2023 e RHEL, o `yum` comando é usado para instalar `containerd` , `ca-certificates` , `iptables` e `amazon-ssm-agent` . Para o Ubuntu, `apt` é usado para instalar `containerd` , `ca-certificates` e `iptables` , e `snap` é usado para instalar `amazon-ssm-agent` .

Componente	URL	Protocolo	Porta
Artefatos dos nós do EKS (S3)	https://hybrid-assets.eks.amazonaws.com	HTTPS	443
Endpoints de serviços da VPC	https://eks.*region*.amazonaws.com	HTTPS	443
Endpoints de serviços do ECR	https://api.ecr.*region*.amazonaws.com	HTTPS	443
Endpoints do ECR do EKS	Consulte the section called “Visualizar registros de imagens da Amazon” dos endpoints regionais.	HTTPS	443
Endpoint binário do SSM [1]	https://amazon-ssm-*region*.s3.*region*.amazonaws.com	HTTPS	443
Endpoint do serviço do SSM [1]	https://ssm.*region*.amazonaws.com	HTTPS	443
Endpoint binário do IAM Anywhere [2]	https://rolesanywhere.amazonaws.com	HTTPS	443
Endpoint do serviço do IAM Anywhere [2]	https://rolesanywhere.*region*.amazonaws.com	HTTPS	443
Endpoints do gerenciador de pacotes do sistema operacional	Os endpoints do repositório de pacotes são específicos do sistema operacion	HTTPS	443

Componente	URL	Protocolo	Porta
	al e podem variar de acordo com a região geográfica.		

> **Note**
>
> [1] O acesso aos endpoints do AWS SSM só será necessário se você estiver usando ativações híbridas do AWS SSM para o provedor de credenciais do IAM on-premises.
> [2] O acesso aos endpoints do AWS IAM só será necessário se você estiver usando o AWS IAM Roles Anywhere para seu provedor de credenciais do IAM on-premises.

Acesso necessário para operações contínuas do cluster

O acesso à rede a seguir para o firewall on-premises é necessário para operações contínuas do cluster.

> **Important**
>
> Dependendo da sua escolha de CNI, você precisa configurar regras adicionais de acesso à rede para as portas da CNI. Consulte a documentação do Cilium e a documentação do Calico para obter detalhes.

Tipo	Protocolo	Direção	Porta	Origem	Destino	Usage
HTTPS	TCP	Saída	443	CIDRs remotos de nós	IPs do cluster de EKS [1]	Kubelet para servidor de API do Kubernetes
HTTPS	TCP	Saída	443	CIDR(s) remoto(s) de pods	IPs do cluster de EKS [1]	Pod para servidor

Tipo	Protocolo	Direção	Porta	Origem	Destino	Usage
						de API do Kubernetes
HTTPS	TCP	Saída	443	CIDRs remotos de nós	Endpoint do serviço do SSM	Atualização de credenciais de ativações híbridas do SSM e heartbeats do SSM a cada cinco minutos
HTTPS	TCP	Saída	443	CIDRs remotos de nós	Endpoint do serviço do IAM Anywhere	Atualização de credenciais do IAM Roles Anywhere
HTTPS	TCP	Saída	443	CIDR(s) remoto(s) de pods	Endpoint regional do STS	Pod para endpoint do STS, necessário somente para IRSA

Tipo	Protocolo	Direção	Porta	Origem	Destino	Usage
HTTPS	TCP	Saída	443	CIDRs remotos de nós	Endpoint do serviço de autenticação do Amazon EKS	Nó para endpoint de autenticação do Amazon EKS, necessário somente para a Identidade de Pods do Amazon EKS
HTTPS	TCP	Entrada	10250	IPs do cluster de EKS [1]	CIDR(s) remoto(s) de nós	Servidor de API do Kubernetes para kubelet
HTTPS	TCP	Entrada	Portas de webhook	IPs do cluster de EKS [1]	CIDR(s) remoto(s) de pods	Servidor de API do Kubernetes para webhooks

Tipo	Protocolo	Direção	Porta	Origem	Destino	Usage
HTTPS	TCP e UDP	Entrada e saída	53	CIDR(s) remoto(s) de pods	CIDR(s) remoto(s) de pods	Pod para CoreDNS. Caso execute pelo menos uma réplica do CoreDNS na nuvem, você deverá permitir o tráfego de DNS para a VPC em que o CoreDNS está sendo executado.
Definido pelo usuário	Definido pelo usuário	Entrada e saída	Portas da aplicação	CIDR(s) remoto(s) de pods	CIDR(s) remoto(s) de pods	Pod para Pod

> **Note**
>
> [1] Os IPs do cluster do EKS. Consulte a seção a seguir sobre as interfaces de rede elástica do Amazon EKS.

Interfaces de rede do Amazon EKS

O Amazon EKS anexa interfaces de rede às sub-redes na VPC que você passa durante a criação do cluster para permitir a comunicação entre o ambiente de gerenciamento do EKS e a VPC. As

interfaces de rede que o Amazon EKS cria podem ser encontradas após a criação do cluster no console do Amazon EC2 ou com a AWS CLI. As interfaces de rede originais são excluídas, e novas interfaces de rede são criadas quando as alterações são aplicadas no cluster do EKS, como atualizações de versão do Kubernetes. Você pode restringir o intervalo de IPs das interfaces de rede do Amazon EKS usando tamanhos de sub-rede restritos para as sub-redes que você passa durante a criação do cluster, o que facilita a configuração do firewall on-premises para permitir conectividade de entrada e saída com esse conjunto conhecido e restrito de IPs. Para controlar quais interfaces de rede de sub-redes são criadas, você poderá limitar o número de sub-redes especificadas ao criar um cluster ou atualizar as sub-redes depois de criar o cluster.

As interfaces de rede provisionadas pelo Amazon EKS têm uma descrição do formato `Amazon EKS your-cluster-name`. Veja o exemplo abaixo para um comando da AWS CLI que você pode usar para encontrar os endereços IP das interfaces de rede que o Amazon EKS provisiona. Substitua `VPC_ID` pelo ID da VPC que você passa durante a criação do cluster.

```
aws ec2 describe-network-interfaces \
--query 'NetworkInterfaces[?(VpcId == VPC_ID && contains(Description,Amazon
 EKS))].PrivateIpAddress'
```

Configuração de sub-redes e da AWS VPC

Os requisitos existentes de VPC e sub-redes para o Amazon EKS se aplicam a clusters com nós híbridos. Além disso, o CIDR da VPC não pode se sobrepor aos CIDRs de nós e pods on-premises. Você deve configurar rotas na tabela de roteamento da VPC para seu nó on-premises e, opcionalmente, para os CIDRs de pods. Essas rotas devem ser configuradas para rotear o tráfego para o gateway que você está usando para sua conectividade de rede híbrida, que geralmente é um gateway privado virtual (VGW) ou gateway de trânsito (TGW). Caso esteja usando o TGW ou VGW para conectar a VPC ao ambiente on-premises, você deverá criar um anexo do TGW ou VGW para a VPC. Sua VPC deve ter suporte de nomes de host de DNS e resolução de DNS.

As etapas a seguir usam a AWS CLI. Você também pode criar esses recursos no Console de gerenciamento da AWS ou com outras interfaces, como o AWS CloudFormation, AWS CDK ou Terraform.

Etapa 1: criar uma VPC

1. Execute o comando a seguir para criar uma VPC. Substitua `VPC_CIDR` por um intervalo de CIDRs de `IPv4` RFC-1918 (privado) ou não RFC-1918 (público) (por exemplo, `10.0.0.0/16`).

Observação: a resolução de DNS, que é um requisito do EKS, está habilitada para a VPC por padrão.

```
aws ec2 create-vpc --cidr-block VPC_CIDR
```

2. Habilite nomes de host DNS para a VPC. Observe que a resolução de DNS está habilitada para a VPC por padrão. Substitua `VPC_ID` pelo ID da VPC que você criou na etapa anterior.

```
aws ec2 modify-vpc-attribute --vpc-id VPC_ID --enable-dns-hostnames
```

Etapa 2: criar sub-redes

Crie pelo menos duas sub-redes. O Amazon EKS usa essas sub-redes para as interfaces de rede do cluster. Para obter mais informações, consulte os requisitos e considerações sobre sub-redes.

1. Você pode encontrar as zonas de disponibilidade de uma região da AWS com o comando a seguir. Substitua `us-west-2` pela sua região.

```
aws ec2 describe-availability-zones \
     --query 'AvailabilityZones[?(RegionName == us-west-2)].ZoneName'
```

2. Crie uma sub-rede. Substitua `VPC_ID` pelo ID da VPC. Substitua `SUBNET_CIDR` pelo bloco CIDR da sub-rede (por exemplo, 10.0.1.0/24). Substitua `AZ` pela zona de disponibilidade em que a sub-rede será criada (por exemplo, us-west-2a). As sub-redes que você criar devem estar em ao menos duas diferentes zonas de disponibilidade.

```
aws ec2 create-subnet \
    --vpc-id VPC_ID \
    --cidr-block SUBNET_CIDR \
    --availability-zone AZ
```

(Opcional) Etapa 3: anexar a VPC ao gateway de trânsito (TGW) da Amazon VPC ou ao gateway privado virtual (VGW) do AWS Direct Connect.

Se você estiver usando um TGW ou VGW, anexe sua VPC ao TGW ou VGW. Para obter mais informações, consulte os anexos da Amazon VPC em Gateways de trânsito da Amazon VPC ou as associações de gateway privado virtual do AWS Direct Connect.

Transit Gateway

Execute o comando a seguir para anexar um gateway de trânsito. Substitua `VPC_ID` pelo ID da VPC. Substitua `SUBNET_ID1` e `SUBNET_ID2` pelos IDs das sub-redes que você criou na etapa anterior. Substitua `TGW_ID` pelo ID do TGW.

```
aws ec2 create-transit-gateway-vpc-attachment \
    --vpc-id VPC_ID \
    --subnet-ids SUBNET_ID1 SUBNET_ID2 \
    --transit-gateway-id TGW_ID
```

Gateway privado virtual

Execute o comando a seguir para anexar um gateway de trânsito. Substitua `VPN_ID` pelo ID do VGW. Substitua `VPC_ID` pelo ID da VPC.

```
aws ec2 attach-vpn-gateway \
    --vpn-gateway-id VPN_ID \
    --vpc-id VPC_ID
```

(Opcional) Etapa 4: Criar tabela de rotas

Você pode modificar a tabela de rotas principal para a VPC ou criar uma tabela de rotas personalizada. As etapas a seguir criam uma tabela de rotas personalizada com as rotas para os CIDRs de nós e pods on-premises. Para obter mais informações, consulte Tabelas de rotas de sub-redes. Substitua `VPC_ID` pelo ID da VPC.

```
aws ec2 create-route-table --vpc-id VPC_ID
```

Etapa 5: criar rotas para nós e pods on-premises

Crie rotas na tabela de rotas para cada um dos nós remotos on-premises. Você pode modificar a tabela de rotas principal da VPC ou usar a tabela de rotas personalizada que você criou na etapa anterior.

Os exemplos abaixo mostram como criar rotas para os CIDRs de nós e pods on-premises. Nos exemplos, um gateway de trânsito (TGW) é usado para conectar a VPC ao ambiente on-premises. Se você tiver vários CIDRs de nós e pods on-premises, repita as etapas para cada CIDR.

- Se você estiver usando um gateway da internet ou um gateway privado virtual (VGW), substitua `--transit-gateway-id` por `--gateway-id`.
- Substitua `RT_ID` pelo ID da tabela de rotas que você criou na etapa anterior.
- Substitua `REMOTE_NODE_CIDR` pelo intervalo de CIDRs que você usará para os nós híbridos.
- Substitua `REMOTE_POD_CIDR` pelo intervalo de CIDRs que você usará para os pods executados em nós híbridos. O intervalo de CIDRs de pods corresponde à configuração da interface de rede de contêineres (CNI), que geralmente usa uma rede de sobreposição on-premises. Para obter mais informações, consulte the section called “Configurar CNI”.
- Substitua `TGW_ID` pelo ID do TGW.

Rede remota de nós

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_NODE_CIDR \
    --transit-gateway-id TGW_ID
```

Rede remota de pods

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_POD_CIDR \
    --transit-gateway-id TGW_ID
```

(Opcional) Etapa 6: associar sub-redes à tabela de rotas

Caso tenha criado uma tabela de rotas personalizada na etapa anterior, associe cada uma das sub-redes criadas na etapa anterior à tabela de rotas personalizada. Caso esteja modificando a tabela de rotas principal da VPC, as sub-redes serão automaticamente associadas à tabela de rotas principal da VPC, e você poderá pular esta etapa.

Execute o comando a seguir para cada uma das sub-redes que você criou nas etapas anteriores. Substitua `RT_ID` pela tabela de rotas que você criou na etapa anterior. Substitua `SUBNET_ID` pelo ID de uma sub-rede.

```
aws ec2 associate-route-table --route-table-id RT_ID --subnet-id SUBNET_ID
```

Configuração do grupo de segurança do cluster

O acesso a seguir para o grupo de segurança do cluster do EKS é necessário para operações contínuas do cluster. O Amazon EKS cria automaticamente as regras de grupo de segurança de entrada necessárias para nós híbridos quando você cria ou atualiza seu cluster com redes remotas de nós e pods configuradas. Como os grupos de segurança permitem todo o tráfego de saída por padrão, o Amazon EKS não modifica automaticamente as regras de saída do grupo de segurança do cluster para nós híbridos. Se quiser personalizar o grupo de segurança do cluster, você pode limitar o tráfego às regras na tabela a seguir.

Tipo	Protocolo	Direção	Porta	Origem	Destino	Usage
HTTPS	TCP	Entrada	443	CIDRs remotos de nós	N/D	Kubelet para o servidor de API do Kubernetes
HTTPS	TCP	Entrada	443	CIDRs remotos de pods	N/D	Pods que exigem acesso ao servidor de API do K8s quando

Tipo	Protocolo	Direção	Porta	Origem	Destino	Usage
						a CNI não está usando NAT para o tráfego de pods.
HTTPS	TCP	Saída	10250	N/D	CIDRs remotos de nós	Servidor de API do Kubernetes para Kubelet
HTTPS	TCP	Saída	Portas de webhook	N/D	CIDRs remotos de pods	Servidor de API do Kubernetes para webhook (se estiver executando webhooks em nós híbridos)

Important

Limites das regras de grupos de segurança: os grupos de segurança do Amazon EC2 têm no máximo 60 regras de entrada por padrão. As regras de entrada do grupo de segurança podem não se aplicar se o grupo de segurança do cluster se aproximar desse limite. Nesse caso, pode ser necessário adicionar manualmente as regras de entrada ausentes. Responsabilidade de limpeza do CIDR: se você remover redes remotas de nós ou pods dos clusters do EKS, o EKS não removerá automaticamente as regras de grupo de segurança

correspondentes. Você é responsável por remover manualmente redes remotas de nós ou pods não utilizadas das regras do seu grupo de segurança.

Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte the section called “Requisitos para grupos de segurança”.

(Opcional) Configuração manual de grupos de segurança

Se precisar criar grupos de segurança adicionais ou modificar as regras criadas automaticamente, é possível usar os comandos a seguir como referência. Por padrão, o comando abaixo cria um grupo de segurança que permite todo o acesso externo. Você pode restringir o acesso externo para incluir somente as regras acima. Caso esteja pensando em limitar regras de saída, recomendamos testar completamente todas as aplicações e a conectividade de pods antes de aplicar as regras modificadas a um cluster do produção.

- No primeiro comando, substitua `SG_NAME` por um nome para o grupo de segurança
- No primeiro comando, substitua `VPC_ID` pelo ID da VPC que você criou na etapa anterior
- No segundo comando, substitua `SG_ID` pelo ID do grupo de segurança que você criou no primeiro comando
- No segundo comando, substitua `REMOTE_NODE_CIDR` e `REMOTE_POD_CIDR` pelos valores dos nós híbridos e da rede on-premises.

```
aws ec2 create-security-group \
    --group-name SG_NAME \
    --description "security group for hybrid nodes" \
    --vpc-id VPC_ID
```

```
aws ec2 authorize-security-group-ingress \
    --group-id SG_ID \
    --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort": 443,
 "IpRanges": [{"CidrIp": "REMOTE_NODE_CIDR"}, {"CidrIp": "REMOTE_POD_CIDR"}]}]'
```

Preparar o sistema operacional para nós híbridos

O Bottlerocket, o Amazon Linux 2023 (AL2023), o Ubuntu e o RHEL são validados continuamente para uso como o sistema operacional de nós para nós híbridos. O Bottlerocket tem suporte da AWS

somente em ambientes VMware vSphere. O AL2023 não é coberto pelos planos do AWS Support quando executado fora do Amazon EC2. O AL2023 só pode ser usado em ambientes virtualizados on-premises. Consulte o Guia do usuário do Amazon Linux 2023 para obter mais informações. A AWS oferece suporte à integração de nós híbridos com os sistemas operacionais Ubuntu e RHEL, mas não oferece suporte para os sistemas operacionais em si.

Você é responsável pelo provisionamento e gerenciamento do sistema operacional. Quando você estiver testando nós híbridos pela primeira vez, é mais fácil executar a CLI (`nodeadm`) do Amazon EKS Hybrid Nodes em um host já provisionado. Para implantações de produção, recomenda-se incluir o `nodeadm` nas imagens do sistema operacional configurado para ser executado como um serviço systemd para unir automaticamente os hosts aos clusters do Amazon EKS na inicialização do host. Se você estiver usando o Bottlerocket como seu sistema operacional de nó no vSphere, não precisará usar `nodeadm`, pois o Bottlerocket já contém as dependências necessárias para nós híbridos e se conectará automaticamente ao cluster configurado na inicialização do host.

Compatibilidade da versão

A tabela abaixo representa as versões do sistema operacional que são compatíveis e validadas para uso como sistema operacional de nó para nós híbridos. Se você estiver usando outras variantes ou versões do sistema operacional que não estão incluídas nesta tabela, então a compatibilidade dos nós híbridos com a variante ou a versão do sistema operacional não é coberta pelo AWS Support. Os nós híbridos são independentes da infraestrutura subjacente e são compatíveis com as arquiteturas x86 e ARM.

Sistema operacional	Versões
Amazon Linux	Amazon Linux 2023 (AL2023)
Bottlerocket	Variantes do VMware v1.37.0 e posteriores que executam o Kubernetes v1.28 e superior
Ubuntu	Ubuntu 20.04, Ubuntu 22.04 e Ubuntu 24.04
Red Hat Enterprise Linux	RHEL 8, RHEL 9

Considerações sobre sistemas operacionais

Geral

- A CLI (`nodeadm`) do Amazon EKS Hybrid Nodes pode ser usada para simplificar a instalação e a configuração dos componentes e dependências dos nós híbridos. Você pode executar o processo `nodeadm install` durante os pipelines de criação de imagens do sistema operacional ou no runtime em cada host on-premises. Para obter mais informações sobre os componentes que o `nodeadm` instala, consulte the section called “Nós híbridos nodeadm”.
- Se você estiver usando um proxy no ambiente on-premises para acessar a internet, é necessária uma configuração adicional do sistema operacional para que os processos de instalação e atualização configurem o gerenciador de pacotes para usar o proxy. Para obter instruções, consulte the section called “Configurar proxy”.

Bottlerocket

- As etapas e ferramentas para conectar um nó do Bottlerocket são diferentes das etapas de outros sistemas operacionais e são abordadas separadamente em the section called “Conectar nós híbridos com o Bottlerocket”, em vez das etapas em the section called “Conectar nós híbridos”.
- As etapas do Bottlerocket não usam a ferramenta de nós híbridos da CLI, `nodeadm`.
- Somente as variantes VMware do Bottlerocket versão v1.37.0 e superior são compatíveis com os nós híbridos do EKS. As variantes de VMware do Bottlerocket estão disponíveis para as versões v1.28 e superiores do Kubernetes. Outras variantes do Bottlerocket não são aceitas como sistema operacional de nós híbridos. OBSERVAÇÃO: as variantes de VMware do Bottlerocket estão disponíveis apenas para a arquitetura x86_64.

Containerd

- O Containerd é o runtime de contêiner padrão do Kubernetes e é uma dependência para nós híbridos, bem como para todos os tipos de computação de nós do Amazon EKS. A CLI do Amazon EKS Hybrid Nodes (`nodeadm`) tenta instalar o containerd durante o processo `nodeadm install`. Você pode configurar a instalação do containerd no runtime de `nodeadm install` com a opção da linha de comando `--containerd-source`. As opções válidas são: `none`, `distro` e `docker`. Caso esteja usando o RHEL, o `distro` não será uma opção válida e você pode configurar o `nodeadm` para instalar o build do containerd dos repositórios do Docker, ou pode instalar manualmente o containerd. Ao usar o AL2023 ou o Ubuntu, o `nodeadm` instala por padrão

o containerd da distribuição do sistema operacional. Se você não quiser que o nodeadm instale o containerd, use a opção `--containerd-source none`.

Ubuntu

- Caso esteja usando o Ubuntu 24.04, talvez seja necessário atualizar a versão do containerd ou alterar a configuração do AppArmor para adotar uma correção que permita que os pods terminem corretamente. Consulte Ubuntu #2065423. É necessária uma reinicialização para aplicar as alterações no usuário do AppArmor. A versão mais recente do Ubuntu 24.04 tem uma versão atualizada do containerd em seu gerenciador de pacotes com a correção (containerd versão 1.7.19+).

Arm

- Se você estiver usando hardware ARM, um processador compatível com ARMv8.2 com a extensão de criptografia (ARMv8.2+crypto) é necessário para executar a versão 1.31 e mais recente do complemento kube-proxy do EKS. Todos os sistemas Raspberry Pi anteriores ao Raspberry Pi 5, bem como os processadores baseados em Cortex-A72, não atendem a esse requisito. Como solução alternativa, você pode continuar a usar a versão 1.30 do complemento kube-proxy do EKS até o fim do suporte estendido em julho de 2026, consulte Calendário de lançamentos do Kubernetes, ou pode usar uma imagem do kube-proxy personalizada do upstream.
- A mensagem de erro a seguir no log do kube-proxy indica esta incompatibilidade:

```
Fatal glibc error: This version of Amazon Linux requires a newer ARM64 processor
 compliant with at least ARM architecture 8.2-a with Cryptographic extensions. On EC2
 this is Graviton 2 or later.
```

Criar imagens do sistema operacional

O Amazon EKS fornece modelos de exemplo do Packer que você pode usar para criar imagens do sistema operacional que incluem o `nodeadm` e o configuram para execução na inicialização do host. Esse processo é recomendado para evitar extrair as dependências dos nós híbridos individualmente em cada host e para automatizar o processo de bootstrap dos nós híbridos. Você pode usar os modelos de exemplo do Packer com uma imagem ISO do Ubuntu 22.04, Ubuntu 24.04, RHEL 8 ou RHEL 9 e pode gerar imagens com estes formatos: OVA, Qcow2 ou raw.

Pré-requisitos

Antes de usar os modelos de exemplo do Packer, você deve ter o seguinte instalado na máquina de onde está executando o Packer:

- Packer versão 1.11.0 ou posterior. Para obter instruções sobre como instalar o Packer, consulte Install Packer na documentação do Packer.
- Se estiver criando OVAs, plug-in do VMware vSphere 1.4.0 ou superior
- Se estiver criando imagens `Qcow2` ou raw, plug-in do QEMU versão 1.x

Definir variáveis de ambiente

Antes de executar o build do Packer, defina as variáveis de ambiente a seguir na máquina de onde está executando o Packer.

Geral

As variáveis de ambiente a seguir devem ser definidas para criar imagens com todos os sistemas operacionais e formatos de saída.

Variável de ambiente	Tipo	Descrição
PKR_SSH_PASSWORD	String	O Packer usa as variáveis `ssh_username` e `ssh_password` para usar SSH na máquina criada durante o provisionamento. Isso precisa corresponder às senhas usadas para criar o usuário inicial nos arquivos de kickstart ou user-data do respectivo sistema operacional. O padrão é definido como "builder" ou "ubuntu", dependendo do sistema operacional. Ao definir a senha, certifique-se de

Variável de ambiente	Tipo	Descrição
		alterá-la no arquivo `ks.cfg` ou `user-data` correspondente para que haja correspondência.
ISO_URL	String	URL da ISO a ser usada. Pode ser um link da Web para download de um servidor ou um caminho absoluto para um arquivo local.
ISO_CHECKSUM	String	Soma de verificação associada para a ISO fornecida.
CREDENTIAL_PROVIDER	String	Provedor de credenciais para nós híbridos. Os valores válidos são `ssm` (padrão) para ativações híbridas do SSM e o `iam` para IAM Roles Anywhere
K8S_VERSION	String	Versão do Kubernetes para nós híbridos (por exemplo, `1.31`). Para saber sobre as versões compatíveis com o Kubernetes, consulte Versões compatíveis com o Amazon EKS.
NODEADM_ARCH	String	Arquitetura para `nodeadm install`: Selecione `amd` ou `arm`.

RHEL

Se você estiver usando o RHEL, as variáveis de ambiente a seguir devem ser definidas.

Variável de ambiente	Tipo	Descrição
RH_USERNAME	String	Nome de usuário do gerenciador de assinaturas do RHEL
RH_PASSWORD	String	Senha do gerenciador de assinaturas RHEL
RHEL_VERSION	String	Versão ISO do RHEL sendo usada. Os valores válidos são 8 ou 9.

Ubuntu

Não são necessárias variáveis de ambiente específicas do Ubuntu.

vSphere

Se você estiver criando um OVA para VMware vSphere, as variáveis de ambiente a seguir deverão ser definidas.

Variável de ambiente	Tipo	Descrição
VSPHERE_SERVER	String	Endereço do servidor vSphere
VSPHERE_USER	String	Nome de usuário do vSphere
VSPHERE_PASSWORD	String	Senha do vSphere
VSPHERE_DATACENTER	String	Nome do data center do vSphere
VSPHERE_CLUSTER	String	Nome do cluster do vSphere
VSPHERE_DATASTORE	String	Nome do datastore do vSphere
VSPHERE_NETWORK	String	Nome da rede do vSphere

Variável de ambiente	Tipo	Descrição
VSPHERE_OUTPUT_FOLDER	String	Pasta de saída do vSphere para os modelos

QEMU

Variável de ambiente	Tipo	Descrição
PACKER_OUTPUT_FORMAT	String	Formato de saída para o construtor do QEMU. Os valores válidos são `qcow2` e `raw`.

Validar modelo

Antes de executar o build, valide o modelo com o comando a seguir após definir as variáveis de ambiente. Substitua `template.pkr.hcl` caso esteja usando um nome diferente para o modelo.

```
packer validate template.pkr.hcl
```

Criar imagens

Crie as imagens com os comandos a seguir e use o sinalizador `-only` para especificar o destino e o sistema operacional para as imagens. Substitua `template.pkr.hcl` caso esteja usando um nome diferente para o modelo.

OVAs para vSphere

> **Note**
>
> Caso esteja usando o RHEL com o vSphere, você precisará converter os arquivos kickstart em uma imagem OEMDRV e passá-la como uma ISO para inicializar. Para obter mais informações, consulte Packer Readme no Repositório do EKS Hybrid Nodes GitHub.

OVA para Ubuntu

```
packer build -only=general-build.vsphere-iso.ubuntu22 template.pkr.hcl
```

OVA para Ubuntu

```
packer build -only=general-build.vsphere-iso.ubuntu24 template.pkr.hcl
```

OVA para RHEL

```
packer build -only=general-build.vsphere-iso.rhel8 template.pkr.hcl
```

OVA para RHEL

```
packer build -only=general-build.vsphere-iso.rhel9 template.pkr.hcl
```

QEMU

> **Note**
>
> Se você estiver criando uma imagem para uma CPU host específica que não corresponde ao host construtor, consulte a documentação do QEMU para obter o nome que corresponda à CPU host e use o sinalizador `-cpu` com o nome da CPU host ao executar os comandos a seguir.

Ubuntu 22.04 Qcow2/Raw

```
packer build -only=general-build.qemu.ubuntu22 template.pkr.hcl
```

Ubuntu 24.04 Qcow2/Raw

```
packer build -only=general-build.qemu.ubuntu24 template.pkr.hcl
```

RHEL 8 Qcow2/Raw

```
packer build -only=general-build.qemu.rhel8 template.pkr.hcl
```

RHEL 9 Qcow2/Raw

```
packer build -only=general-build.qemu.rhel9 template.pkr.hcl
```

Passar a configuração do nodeadm por meio dos dados do usuário

Você pode passar a configuração para `nodeadm` nos dados do usuário por meio da inicialização da nuvem para configurar e conectar automaticamente os nós híbridos ao cluster do EKS na inicialização do host. Veja abaixo um exemplo de como fazer isso ao usar o VMware vSphere como a infraestrutura para seus nós híbridos.

1. Instale a CLI `govc` seguindo as instruções no readme do govc no GitHub.
2. Depois de executar o comando build do Packer na seção anterior e provisionar o modelo, você pode clonar o modelo para criar vários nós diferentes usando o seguinte: Você deve clonar o modelo de cada nova VM que você está criando que será usada para nós híbridos. Substitua as variáveis no comando abaixo pelos valores do seu ambiente. `VM_NAME` no comando abaixo é usado como seu `NODE_NAME` quando você injeta os nomes das VMs por meio do arquivo `metadata.yaml`.

   ```
   govc vm.clone -vm "/PATH/TO/TEMPLATE" -ds="YOUR_DATASTORE" \
       -on=false -template=false -folder=/FOLDER/TO/SAVE/VM "VM_NAME"
   ```

3. Depois de clonar o modelo para cada uma das suas novas VMs, crie um `userdata.yaml` e `metadata.yaml` para as VMs. Suas VMs podem compartilhar o mesmo `userdata.yaml` e `metadata.yaml` e você os preencherá por VM nas etapas abaixo. A configuração do `nodeadm` é criada e definida na seção `write_files` do `userdata.yaml`. O exemplo abaixo usa as ativações híbridas do AWS SSM como o provedor de credenciais on-premises para nós híbridos. Para obter mais informações sobre a configuração do `nodeadm`, consulte the section called “Nós híbridos nodeadm”.

   userdata.yaml:

   ```
   #cloud-config
   users:
     - name: # username for login. Use 'builder' for RHEL or 'ubuntu' for Ubuntu.
       passwd: # password to login. Default is 'builder' for RHEL.
       groups: [adm, cdrom, dip, plugdev, lxd, sudo]
       lock-passwd: false
       sudo: ALL=(ALL) NOPASSWD:ALL
       shell: /bin/bash
   ```

```
write_files:
  - path: /usr/local/bin/nodeConfig.yaml
    permissions: '0644'
    content: |
      apiVersion: node.eks.aws/v1alpha1
      kind: NodeConfig
      spec:
          cluster:
              name: # Cluster Name
              region: # AWS region
          hybrid:
              ssm:
                  activationCode: # Your ssm activation code
                  activationId: # Your ssm activation id

runcmd:
  - /usr/local/bin/nodeadm init -c file:///usr/local/bin/nodeConfig.yaml >> /var/log/
nodeadm-init.log 2>&1
```

metadata.yaml:

Crie um `metadata.yaml` para o ambiente. Mantenha o formato da variável `"$NODE_NAME"` no arquivo, pois ela será preenchida com valores em uma etapa subsequente.

```
instance-id: "$NODE_NAME"
local-hostname: "$NODE_NAME"
network:
  version: 2
  ethernets:
    nics:
      match:
        name: ens*
      dhcp4: yes
```

4. Adicione os arquivos `userdata.yaml` e `metadata.yaml` como strings `gzip+base64` com os comandos a seguir. Os comandos a seguir devem ser executados para cada uma das VMs que você está criando. Substitua `VM_NAME` pelo nome da VM que você está atualizando.

```
export NODE_NAME="VM_NAME"
export USER_DATA=$(gzip -c9 <userdata.yaml | base64)
```

```
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.userdata="${USER_DATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.userdata.encoding=gzip+base64

envsubst '$NODE_NAME' < metadata.yaml > metadata.yaml.tmp
export METADATA=$(gzip -c9 <metadata.yaml.tmp | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.metadata="${METADATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.metadata.encoding=gzip+base64
```

5. Ative as novas VMs, que devem se conectar automaticamente ao cluster do EKS que você configurou.

```
govc vm.power -on "${NODE_NAME}"
```

Preparar as credenciais para nós híbridos

O Amazon EKS Hybrid Nodes usa credenciais temporárias do IAM provisionadas por ativações híbridas do AWS SSM ou pelo AWS IAM Roles Anywhere para se autenticar com o cluster do Amazon EKS. Você deve usar as ativações híbridas do AWS SSM ou o AWS IAM Roles Anywhere com a CLI do Amazon EKS Hybrid Nodes (`nodeadm`). Você não deve usar as ativações híbridas do AWS SSM e o AWS IAM Roles Anywhere juntos. Recomendamos usar as ativações híbridas do AWS SSM caso não tenha uma infraestrutura de chave pública (PKI) existente com uma Certificate Authority (CA) e certificados para seus ambientes on-premises. Se você já tem uma PKI e certificados on-premises, use o AWS IAM Roles Anywhere.

Perfil do IAM para nós híbridos

Antes de poder conectar nós híbridos ao cluster do Amazon EKS, você deve criar um perfil do IAM que será usado com as ativações híbridas do AWS SSM ou o AWS IAM Roles Anywhere para as credenciais dos nós híbridos. Após a criação do cluster, você usará esse perfil com uma entrada de acesso do Amazon EKS ou uma entrada do ConfigMap `aws-auth` para mapear o perfil do IAM para o controle de acesso por perfil (RBAC) do Kubernetes. Para obter mais informações sobre como associar o perfil do IAM de nós híbridos ao RBAC do Kubernetes, consulte the section called “Preparar acesso ao cluster”.

O perfil do IAM de nós híbridos deve ter as permissões a seguir.

- Permissões para o `nodeadm` usar a ação `eks:DescribeCluster` a fim de obter informações sobre o cluster ao qual você deseja conectar os nós híbridos. Se você não habilitar a ação `eks:DescribeCluster`, será necessário informar o endpoint da API do Kubernetes, o pacote da CA do cluster e o intervalo CIDR IPv4 do serviço na configuração do nó que você transmite para o comando `nodeadm init`.
- Permissões para o `nodeadm` usar a ação `eks:ListAccessEntries` a fim de listar as entradas de acesso no cluster ao qual você deseja conectar os nós híbridos. Se você não habilitar a ação `eks:ListAccessEntries`, será necessário fornecer o sinalizador `--skip cluster-access-validation` ao executar o comando `nodeadm init`.
- Permissões para que o kubelet use imagens de contêiner do Amazon Elastic Container Registry (Amazon ECR), conforme definido pela política AmazonEC2ContainerRegistryPullOnly.
- Se estiver usando o AWS SSM, permissões para o `nodeadm init` usar ativações híbridas do AWS SSM, conforme definido na política aws-managed-policy/latest/reference/AmazonSSMManagedInstanceCore.html.
- Caso esteja usando o AWS SSM, as permissões para usar a ação `ssm:DeregisterManagedInstance` e a ação `ssm:DescribeInstanceInformation` para `nodeadm uninstall` cancelar o registro de instâncias.
- (Opcional) Permissões para que o Amazon EKS Pod Identity Agent use a ação `eks-auth:AssumeRoleForPodIdentity` para recuperar credenciais para pods.

Configurar ativações híbridas do AWS SSM

Antes de configurar as ativações híbridas do AWS SSM, você deve ter um perfil do IAM de nós híbridos criado e configurado. Para obter mais informações, consulte the section called “Criar o perfil do IAM de nós híbridos”. Siga as instruções em Criar uma ativação híbrida para registrar nós no Systems Manager no Guia do usuário do AWS Systems Manager para criar uma ativação híbrida do AWS SSM para os nós híbridos. O ID e o código de ativação que você recebe são usados com `nodeadm` quando você registra os hosts como nós híbridos com o cluster do Amazon EKS. Você pode voltar a esta etapa posteriormente depois de criar e preparar os clusters do Amazon EKS para nós híbridos.

> **Important**
>
> O Systems Manager retorna imediatamente o ID e o código de ativação para o console ou a janela de comando, dependendo de como você criou a ativação. Copie essas informações e

armazene-as em um local seguro. Se você sair do console ou fechar a janela de comando, poderá perder essas informações. Se perdê-las, você deverá criar outra ativação.

Por padrão, as ativações híbridas do AWS SSM ficam ativas por 24 horas. Como alternativa, você pode especificar uma `--expiration-date` ao criar a ativação híbrida no formato de carimbo de data e hora, como `2024-08-01T00:00:00`. Quando você usa o AWS SSM como o provedor de credenciais, o nome do nó para os nós híbridos não é configurável e é gerado automaticamente pelo AWS SSM. Você pode visualizar e gerenciar as instâncias gerenciadas pelo AWS SSM no console do AWS Systems Manager em Fleet Manager. Você pode registrar até mil nós padrão ativados para ambiente híbrido por conta por região da AWS sem custo adicional. No entanto, para registrar mais de 1.000 nós híbridos, você precisa ativar o nível de instâncias avançadas. Há uma cobrança para o uso do nível de instâncias avançadas que não está incluída nos preços do Amazon EKS Hybrid Nodes. Para obter mais informações, consulte Preços do AWS Systems Manager.

Veja o exemplo abaixo para saber como criar uma ativação híbrida do AWS SSM com o perfil do IAM de nós híbridos. Quando você usa ativações híbridas do AWS SSM para as credenciais de nós híbridos, os nomes dos nós híbridos terão o formato `mi-012345678abcdefgh` e as credenciais temporárias provisionadas pelo AWS SSM serão válidas por uma hora. Você não pode alterar o nome do nó ou a duração da credencial ao usar o AWS SSM como seu provedor de credenciais. As credenciais temporárias são rotacionadas automaticamente pelo AWS SSM, e a rotação não afeta o status dos nós ou das aplicações.

Recomendamos usar uma ativação híbrida do AWS SSM por cluster do EKS para definir o escopo da permissão `ssm:DeregisterManagedInstance` do AWS SSM do perfil do IAM de nós híbridos para somente poder cancelar o registro de instâncias associadas à ativação híbrida do AWS SSM. No exemplo desta página, é usada uma tag com o ARN do cluster do EKS, que pode ser usada para mapear a ativação híbrida do AWS SSM para o cluster do EKS. Como alternativa, você pode usar a tag e o método de preferência para definir o escopo das permissões do AWS SSM com base nos requisitos e limites de permissões. A opção `REGISTRATION_LIMIT` no comando abaixo é um número inteiro usado para limitar o número de máquinas que podem usar a ativação híbrida do AWS SSM (por exemplo, `10`)

```
aws ssm create-activation \
     --region AWS_REGION \
     --default-instance-name eks-hybrid-nodes \
     --description "Activation for EKS hybrid nodes" \
     --iam-role AmazonEKSHybridNodesRole \
```

```
    --tags Key=EKSClusterARN,Value=arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/
CLUSTER_NAME \
    --registration-limit REGISTRATION_LIMIT
```

Consulte as instruções em Criar uma ativação híbrida para registrar nós no Systems Manager para obter mais informações sobre as configurações disponíveis para ativações híbridas do AWS SSM.

Configurar o AWS IAM Roles Anywhere

Siga as instruções em Getting started with IAM Roles Anywhere no Guia do usuário do IAM Roles Anywhere para configurar a âncora de confiança e o usuário que você usará para as credenciais temporárias do IAM para o perfil do IAM de nós híbridos. Ao criar seu usuário, você poderá criá-lo sem adicionar nenhum perfil. Você pode criar o usuário, retornar a essas etapas para criar o perfil do IAM de nós híbridos e, em seguida, adicionar o perfil ao seu usuário após a criação. Como alternativa, você pode usar as etapas do AWS CloudFormation mais adiante nesta página para concluir a configuração do IAM Roles Anywhere para nós híbridos.

Ao adicionar o perfil do IAM de nós híbridos ao seu usuário, selecione Aceitar nome da sessão do perfil personalizado no painel do nome da sessão Perfil personalizado na parte inferior da página Editar usuário no console do AWS IAM Roles Anywhere. Isso corresponde ao campo acceptRoleSessionName da API `CreateProfile`. Isso permite que você forneça um nome de nó personalizado para os nós híbridos na configuração que você passa para `nodeadm` durante o processo de bootstrap. É necessário passar um nome de nó personalizado durante o processo de `nodeadm init`. Você pode atualizar seu usuário para aceitar um nome de sessão de usuário personalizado depois de criá-lo.

Você pode configurar a duração da validade da credencial com o AWS IAM Roles Anywhere por meio do campo durationSeconds do seu perfil do AWS IAM Roles Anywhere. A duração padrão é de uma hora, com um máximo de 12 horas. A configuração de `MaxSessionDuration` no perfil do IAM de nós híbridos deve ser maior do que a configuração de `durationSeconds` no seu usuário do AWS IAM Roles Anywhere. Para obter mais informações sobre `MaxSessionDuration`, consulte a documentação da API UpdateRole.

Os certificados e as chaves por máquina que você gera da autoridade de certificação (CA) devem ser colocados no diretório `/etc/iam/pki` em cada nó híbrido com os nomes de arquivo `server.pem` para o certificado e `server.key` para a chave.

Criar o perfil do IAM de nós híbridos

Para executar as etapas desta seção, a entidade principal do IAM que usa o Console da AWS ou a AWS CLI deve ter as permissões a seguir.

- `iam:CreatePolicy`
- `iam:CreateRole`
- `iam:AttachRolePolicy`
- Se estiver usando o AWS IAM Roles Anywhere
  - `rolesanywhere:CreateTrustAnchor`
  - `rolesanywhere:CreateProfile`
  - `iam:PassRole`

AWS CloudFormation

Instale e configure a AWS CLI, caso ainda não o tenha feito. Consulte Installing or updating to the last version of the AWS CLI.

Etapas para as ativações híbridas do AWS SSM

A pilha do CloudFormation cria o perfil do IAM de nós híbridos com as permissões descritas acima. O modelo do CloudFormation não cria a ativação híbrida do AWS SSM.

1. Faça download do modelo do CloudFormation do AWS SSM para nós híbridos:

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ssm-cfn.yaml'
```

2. Crie um `cfn-ssm-parameters.json` com as seguintes opções:
   a. Substitua `ROLE_NAME` pelo nome do perfil do IAM de nós híbridos. Por padrão, o modelo do CloudFormation usa o `AmazonEKSHybridNodesRole` como nome do perfil que ele cria, caso você não especifique um nome.
   b. Substitua `TAG_KEY` pela chave de tag de recurso do AWS SSM que você usou ao criar a ativação híbrida do AWS SSM. A combinação da chave e do valor da tag é usada na condição de `ssm:DeregisterManagedInstance` para permitir que somente o perfil do IAM de nós híbridos cancele o registro das instâncias gerenciadas pelo AWS SSM associadas à ativação híbrida do AWS SSM. No modelo do CloudFormation, `TAG_KEY` é definido como `EKSClusterARN`.

c. Substitua `TAG_VALUE` pelo valor da tag de recurso do AWS SSM que você usou ao criar a ativação híbrida do AWS SSM. A combinação da chave e do valor da tag é usada na condição de `ssm:DeregisterManagedInstance` para permitir que somente o perfil do IAM de nós híbridos cancele o registro das instâncias gerenciadas pelo AWS SSM associadas à ativação híbrida do AWS SSM. Caso esteja usando o padrão `TAG_KEY` de `EKSClusterARN`, passe o ARN do cluster do EKS como `TAG_VALUE`. Os ARNs dos clusters do EKS têm o formato `arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`.

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "SSMDeregisterConditionTagKey": "TAG_KEY",
    "SSMDeregisterConditionTagValue": "TAG_VALUE"
  }
}
```

3. Implante a pilha do CloudFormation. Substitua `STACK_NAME` pelo nome da pilha do CloudFormation.

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --template-file hybrid-ssm-cfn.yaml \
    --parameter-overrides file://cfn-ssm-parameters.json \
    --capabilities CAPABILITY_NAMED_IAM
```

Etapas do AWS IAM Roles Anywhere

A pilha do CloudFormation cria a âncora de confiança do AWS IAM Roles Anywhere com a autoridade de certificação (CA) que você configura, cria o usuário do AWS IAM Roles Anywhere e cria o perfil do IAM de nós híbridos com as permissões descritas anteriormente.

1. Para configurar uma autoridade de certificação (CA)
   a. Para usar um recurso do AWS Private CA, abra o console do AWS Private Certificate Authority. Siga as instruções no Guia do usuário do AWS Private CA.
   b. Para usar uma CA externa, siga as instruções fornecidas pela CA. Você fornece o corpo do certificado em uma etapa posterior.
   c. Os certificados emitidos por CAs públicas não podem ser usados como âncoras de confiança.
2. Fazer download do modelo do CloudFormation do AWS IAM Roles Anywhere para nós híbridos

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ira-cfn.yaml'
```

3. Crie um `cfn-iamra-parameters.json` com as seguintes opções:

   a. Substitua `ROLE_NAME` pelo nome do perfil do IAM de nós híbridos. Por padrão, o modelo do CloudFormation usa o `AmazonEKSHybridNodesRole` como nome do perfil que ele cria, caso você não especifique um nome.

   b. Substitua `CERT_ATTRIBUTE` pelo atributo de certificado por máquina que identifica exclusivamente o host. O atributo de certificado que você usa deve corresponder ao nodeName que você usa para a configuração de `nodeadm` ao conectar nós híbridos ao cluster. Para obter mais informações, consulte a the section called “Nós híbridos nodeadm”. Por padrão, o modelo do CloudFormation usa `${aws:PrincipalTag/x509Subject/CN}` como `CERT_ATTRIBUTE`, o que corresponde ao campo CN dos certificados por máquina. Como alternativa, você pode passar `$(aws:PrincipalTag/x509SAN/Name/CN}` como `CERT_ATTRIBUTE`.

   c. Substitua `CA_CERT_BODY` pelo corpo do certificado da CA sem quebras de linha. `CA_CERT_BODY` deve estar no formato Privacy Enhanced Mail (PEM). Se você tiver um certificado CA no formato PEM, remova as quebras de linha e as linhas BEGIN CERTIFICATE e END CERTIFICATE antes de colocar o corpo do certificado CA no arquivo `cfn-iamra-parameters.json`.

   ```
   {
     "Parameters": {
       "RoleName": "ROLE_NAME",
       "CertAttributeTrustPolicy": "CERT_ATTRIBUTE",
       "CABundleCert": "CA_CERT_BODY"
     }
   }
   ```

4. Implemente o modelo do CloudFormation Substitua `STACK_NAME` pelo nome da pilha do CloudFormation.

   ```
   aws cloudformation deploy \
       --stack-name STACK_NAME \
       --template-file hybrid-ira-cfn.yaml \
       --parameter-overrides file://cfn-iamra-parameters.json
       --capabilities CAPABILITY_NAMED_IAM
   ```

AWS CLI

Instale e configure a AWS CLI, caso ainda não o tenha feito. Consulte Installing or updating to the last version of the AWS CLI.

Criar política de descrição de cluster do EKS

1. Crie um arquivo denominado `eks-describe-cluster-policy.json` com o conteúdo a seguir:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Crie a política com o seguinte comando:

```
aws iam create-policy \
    --policy-name EKSDescribeClusterPolicy \
    --policy-document file://eks-describe-cluster-policy.json
```

Etapas para as ativações híbridas do AWS SSM

1. Crie um arquivo denominado `eks-hybrid-ssm-policy.json` com os conteúdos a seguir. A política concede permissão para duas ações: `ssm:DescribeInstanceInformation` e `ssm:DeregisterManagedInstance`. A política restringe a permissão `ssm:DeregisterManagedInstance` às instâncias gerenciadas pelo AWS SSM associadas à ativação híbrida do AWS SSM com base na tag de recurso que você especifica na política de confiança.

   a. Substitua `AWS_REGION` pela região da AWS da ativação híbrida do AWS SSM.

   b. Substitua `AWS_ACCOUNT_ID` pelo ID de sua conta da AWS.

c. Substitua `TAG_KEY` pela chave de tag de recurso do AWS SSM que você usou ao criar a ativação híbrida do AWS SSM. A combinação da chave e do valor da tag é usada na condição de `ssm:DeregisterManagedInstance` para permitir que somente o perfil do IAM de nós híbridos cancele o registro das instâncias gerenciadas pelo AWS SSM associadas à ativação híbrida do AWS SSM. No modelo do CloudFormation, `TAG_KEY` é definido como `EKSClusterARN`.

d. Substitua `TAG_VALUE` pelo valor da tag de recurso do AWS SSM que você usou ao criar a ativação híbrida do AWS SSM. A combinação da chave e do valor da tag é usada na condição de `ssm:DeregisterManagedInstance` para permitir que somente o perfil do IAM de nós híbridos cancele o registro das instâncias gerenciadas pelo AWS SSM associadas à ativação híbrida do AWS SSM. Caso esteja usando o padrão `TAG_KEY` de `EKSClusterARN`, passe o ARN do cluster do EKS como `TAG_VALUE`. Os ARNs dos clusters do EKS têm o formato `arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeInstanceInformation",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DeregisterManagedInstance",
            "Resource": "arn:aws:ssm:us-east-1:123456789012:managed-instance/*",
            "Condition": {
                "StringEquals": {
                    "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
                }
            }
        }
    ]
}
```

2. Crie a política com o comando a seguir.

```
aws iam create-policy \
    --policy-name EKSHybridSSMPolicy \
    --policy-document file://eks-hybrid-ssm-policy.json
```

3. Crie um arquivo chamado `eks-hybrid-ssm-trust.json`. Substitua `AWS_REGION` pela região da AWS da ativação híbrida do AWS SSM e `AWS_ACCOUNT_ID` pelo ID da conta da AWS.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"ssm.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws:ssm:us-east-1:123456789012:*"
            }
         }
      }
   ]
}
```

4. Crie o perfil com o comando a seguir.

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-ssm-trust.json
```

5. Anexe as políticas `EKSDescribeClusterPolicy` e `EKSHybridSSMPolicy` que você criou nas etapas anteriores. Substitua `AWS_ACCOUNT_ID` pelo ID de sua conta da AWS.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws:iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws:iam::AWS_ACCOUNT_ID:policy/EKSHybridSSMPolicy
```

6. Anexe as políticas `AmazonEC2ContainerRegistryPullOnly` e `AmazonSSMManagedInstanceCore` gerenciadas pela AWS.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
```

Etapas do AWS IAM Roles Anywhere

Para usar o AWS IAM Roles Anywhere, você deve configurar a âncora de confiança do AWS IAM Roles Anywhere antes de criar o perfil do IAM de nós híbridos. Para obter instruções, consulte the section called “Configurar o AWS IAM Roles Anywhere”.

1. Crie um arquivo chamado `eks-hybrid-iamra-trust.json`. Substitua `TRUST_ANCHOR ARN` pelo ARN da âncora de confiança criada nas etapas de the section called “Configurar o AWS IAM Roles Anywhere”. A condição nessa política de confiança restringe a capacidade do AWS IAM Roles Anywhere de assumir o perfil do IAM de nós híbridos para trocar as credenciais temporárias do IAM somente quando o nome da sessão do perfil corresponder ao CN no certificado x509 instalado nos nós híbridos. Como alternativa, você pode usar outros atributos de certificado para identificar o nó de forma exclusiva. O atributo do certificado que você usa na política de confiança deve corresponder ao `nodeName` que você definiu na configuração de `nodeadm`. Para obter mais informações, consulte a the section called “Nós híbridos nodeadm”.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
```

```
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}",
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        }
    ]
}
```

2. Crie o perfil com o comando a seguir.

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-iamra-trust.json
```

3. Anexe a política `EKSDescribeClusterPolicy` que você criou nas etapas anteriores. Substitua `AWS_ACCOUNT_ID` pelo ID de sua conta da AWS.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
```

```
    --policy-arn arn:aws:iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

4. Anexar a política `AmazonEC2ContainerRegistryPullOnly` gerenciada pela AWS

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Console de gerenciamento da AWS

Criar política de descrição de cluster do EKS

1. Abra o console do Amazon IAM.
2. No painel de navegação à esquerda, escolha Políticas.
3. Na página Políticas, escolha Criar política.
4. Na página Especificar permissões, no painel Selecionar um serviço, escolha EKS.
   a. Filtre as ações de DescribeCluster e selecione a ação de leitura DescribeCluster.
   b. Escolha Próximo.
5. Na página Analisar e criar
   a. Insira um Nome de política para a política, como `EKSDescribeClusterPolicy`.
   b. Escolha Criar política.

Etapas para as ativações híbridas do AWS SSM

1. Abra o console do Amazon IAM.
2. No painel de navegação à esquerda, escolha Políticas.
3. Na página Políticas, escolha Criar política.
4. Na página Especificar permissões, no Editor de políticas na parte superior direita da navegação, escolha JSON. Cole o trecho a seguir. Substitua `AWS_REGION` pela região da AWS da ativação híbrida do AWS SSM e substitua `AWS_ACCOUNT_ID` pelo ID da conta da AWS. Substitua `TAG_KEY` e `TAG_VALUE` pela chave de tag de recurso do AWS SSM que você usou ao criar a ativação híbrida do AWS SSM.

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeInstanceInformation",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DeregisterManagedInstance",
            "Resource": "arn:aws:ssm:us-east-1:123456789012:managed-instance/*",
            "Condition": {
                "StringEquals": {
                    "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
                }
            }
        }
    ]
}
```

a. Escolha Próximo.

5. Na página Analisar e criar.
   a. Insira um nome de Política para a política, como `EKSHybridSSMPolicy`.
   b. Escolha Create Policy.
6. No painel de navegação à esquerda, selecione Perfis.
7. Na página Perfis, selecione Criar perfil.
8. Na página Selecionar entidade confiável, faça o seguinte:
   a. Na Seção do tipo de entidade confiável, escolha Política de confiança personalizada. Cole o indicado abaixo no editor de políticas de confiança personalizadas. Substitua `AWS_REGION` pela região da AWS da ativação híbrida do AWS SSM e `AWS_ACCOUNT_ID` pelo ID da conta da AWS.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"ssm.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
```

```
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws:ssm:us-east-1:123456789012:*"
            }
         }
      }
   ]
}
```

b. Escolha Next (Próximo).

9. Na página Add permissions (Adicionar permissões), faça o seguinte:
   a. Na caixa Filtrar políticas, insira `EKSDescribeClusterPolicy` ou o nome da política que você criou acima. Marque a caixa de seleção à esquerda do nome da política nos resultados da pesquisa.
   b. Na caixa Filtrar políticas, insira `EKSHybridSSMPolicy` ou o nome da política que você criou acima. Marque a caixa de seleção à esquerda do nome da política nos resultados da pesquisa.
   c. Na caixa Filtrar políticas insira `AmazonEC2ContainerRegistryPullOnly`. Marque a caixa de seleção à esquerda de `AmazonEC2ContainerRegistryPullOnly` nos resultados da pesquisa.
   d. Na caixa Filtrar políticas insira `AmazonSSMManagedInstanceCore`. Marque a caixa de seleção à esquerda de `AmazonSSMManagedInstanceCore` nos resultados da pesquisa.
   e. Escolha Próximo.
10. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:
   a. Em Nome do perfil, insira um nome exclusivo para o perfil, como `AmazonEKSHybridNodesRole`.
   b. Em Description (Descrição), substitua o texto atual por um texto descritivo como `Amazon EKS - Hybrid Nodes role`.
   c. Selecione Criar perfil.

Etapas do AWS IAM Roles Anywhere

Para usar o AWS IAM Roles Anywhere, você deve configurar a âncora de confiança do AWS IAM Roles Anywhere antes de criar o perfil do IAM de nós híbridos. Para obter instruções, consulte the section called “Configurar o AWS IAM Roles Anywhere”.

1. Abra o console do Amazon IAM.
2. No painel de navegação à esquerda, selecione Perfis.
3. Na página Perfis, selecione Criar perfil.
4. Na página Selecionar entidade confiável, faça o seguinte:
   a. Na Seção do tipo de entidade confiável, escolha Política de confiança personalizada. Cole o indicado abaixo no editor de políticas de confiança personalizadas. Substitua `TRUST_ANCHOR ARN` pelo ARN da âncora de confiança criada nas etapas de the section called “Configurar o AWS IAM Roles Anywhere”. A condição nessa política de confiança restringe a capacidade do AWS IAM Roles Anywhere de assumir o perfil do IAM de nós híbridos para trocar as credenciais temporárias do IAM somente quando o nome da sessão do perfil corresponder ao CN no certificado x509 instalado nos nós híbridos. Como alternativa, você pode usar outros atributos de certificado para identificar o nó de forma exclusiva. O atributo de certificado que você usa na política de confiança deve corresponder ao nodeName definido na configuração de nodeadm. Para obter mais informações, consulte a the section called “Nós híbridos nodeadm”.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        },
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}",
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        }
    ]
}
```

b. Escolha Next (Próximo).

5. Na página Add permissions (Adicionar permissões), faça o seguinte:

   a. Na caixa Filtrar políticas, insira `EKSDescribeClusterPolicy` ou o nome da política que você criou acima. Marque a caixa de seleção à esquerda do nome da política nos resultados da pesquisa.

   b. Na caixa Filtrar políticas insira `AmazonEC2ContainerRegistryPullOnly`. Marque a caixa de seleção à esquerda de `AmazonEC2ContainerRegistryPullOnly` nos resultados da pesquisa.

   c. Escolha Próximo.

6. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:

   a. Em Nome do perfil, insira um nome exclusivo para o perfil, como `AmazonEKSHybridNodesRole`.

   b. Em Description (Descrição), substitua o texto atual por um texto descritivo como `Amazon EKS - Hybrid Nodes role`.

   c. Selecione Criar perfil.

Criar um cluster do Amazon EKS com nós híbridos

Este tópico fornece uma visão geral das opções disponíveis e descreve o que deve ser considerado ao criar um cluster do Amazon EKS habilitado para nós híbridos. O EKS Hybrid Nodes tem o mesmo suporte a versão do Kubernetes que os clusters do Amazon EKS com nós na nuvem, incluindo suporte padrão e estendido.

Se você não planeja usar o EKS Hybrid Nodes, consulte a documentação principal de criação de clusters do Amazon EKS em the section called “Criar um cluster”.

Pré-requisitos

- O the section called “Pré-requisitos” foi concluído. Antes de criar o cluster habilitado para nós híbridos, você deve identificar o nó on-premises e, opcionalmente, os CIDRs de pods, criar sua VPC e sub-redes de acordo com os requisitos do EKS e os requisitos de nós híbridos, e configurar o grupo de segurança com regras de entrada para os CIDRs on-premises e, opcionalmente, de pods. Para obter mais informações sobre esses pré-requisitos, consulte the section called “Preparar a rede”.
- A versão mais recente da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo. Para verificar sua versão atual, use `aws --version`. Os gerenciadores de pacotes, como yum, apt-get ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar ou atualizar a versão mais recente da AWS CLI e Como definir as configurações da AWS CLI no Guia do usuário da AWS Command Line Interface.
- Uma entidade principal do IAM com permissões para criar perfis do IAM e anexar políticas, além de criar e descrever clusters do EKS

Considerações

- O cluster dove usar a `API` ou `API_AND_CONFIG_MAP` para o modo de autenticação do cluster.
- O cluster dove usar a família de endereços IPv4.
- O cluster dove usar a conectividade de endpoint de cluster público ou privado. O cluster não pode usar a conectividade de endpoint de cluster "pública e privada" porque o endpoint do servidor de API do Amazon EKS Kubernetes resolverá para os IPs públicos dos nós híbridos executados fora da VPC.
- A autenticação OIDC é compatível com clusters do EKS habilitados para nós híbridos.

- Você pode adicionar, alterar ou remover a configuração de nós híbridos do cluster existente. Para obter mais informações, consulte the section called “Cluster existente”.

Etapa 1: Criar perfil IAM de cluster

Se já tiver um perfil do IAM do cluster ou se for criar o cluster com o `eksctl`ou o AWS CloudFormation, você poderá ignorar esta etapa. Por padrão, o `eksctl` e o modelo do AWS CloudFormation criam o perfil do IAM do cluster para você.

1. Execute o comando a seguir para criar um arquivo JSON de política de confiança do IAM:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Crie o perfil do IAM do cluster do Amazon EKS. Se necessário, prefixe eks-cluster-role-trust-policy.json com o caminho no computador no qual você gravou o arquivo na etapa anterior. O comando associa a política de confiança criada na etapa anterior à função. Para criar um perfil do IAM, a entidade principal do IAM que estiver criando o perfil deverá ser atribuída à seguinte ação `iam:CreateRole` (permissão):

```
aws iam create-role \
    --role-name myAmazonEKSClusterRole \
    --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

3. Você pode atribuir a política gerenciada do Amazon EKS ou criar sua própria política personalizada. Para obter as permissões mínimas que você deve usar em sua política personalizada, consulte the section called “Perfil do IAM do nó”. Anexe a política gerenciada do Amazon EKS denominada `AmazonEKSClusterPolicy` à função. Para anexar uma política do IAM a uma entidade principal do IAM a entidade principal do IAM que está anexando a política

deve receber uma das seguintes ações do IAM (permissões): `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
    --role-name myAmazonEKSClusterRole
```

Etapa 2: criar um cluster habilitado para nós híbridos

Você pode criar um cluster usando:

- eksctl
- AWS CloudFormation
- AWS CLI
- Console de gerenciamento da AWS

Criar cluster habilitado para nós híbridos: eksctl

Você precisa instalar a versão mais recente da ferramenta da linha do comando `eksctl`. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

1. Crie `cluster-config.yaml` para definir um cluster IPv4 do Amazon EKS habilitado para nós híbridos. Faça as seguintes substituições no `cluster-config.yaml`: Para obter uma lista completa das configurações, consulte a documentação do eksctl.
   a. Substitua `CLUSTER_NAME` por um nome de cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
   b. Substitua `AWS_REGION` pela região da AWS na qual você deseja criar o cluster.
   c. Substitua `K8S_VERSION` por qualquer versão compatível do Amazon EKS.
   d. Substitua `CREDS_PROVIDER` por `ssm` ou `ira` com base no provedor de credenciais que você configurou nas etapas para the section called “Preparar credenciais”.
   e. Substitua `CA_BUNDLE_CERT` se o provedor de credenciais estiver definido como `ira`, que usa o AWS IAM Roles Anywhere como provedor de credenciais. O CA_BUNDLE_CERT é o corpo do certificado da autoridade de certificação (CA) e depende da sua escolha de CA. O certificado deve estar no formato Privacy Enhanced Mail (PEM).

f. Substitua `GATEWAY_ID` pelo ID do seu gateway privado virtual ou gateway de trânsito a ser anexado à VPC.

g. Substitua `REMOTE_NODE_CIDRS` pelo CIDR de nós on-premises dos nós híbridos.

h. Substitua `REMOTE_POD_CIDRS` pelo CIDR de pods on-premises para workloads executadas em nós híbridos, ou remova a linha da configuração se não estiver executando webhooks em nós híbridos. Você deverá configurar `REMOTE_POD_CIDRS` se a CNI não usar a conversão de endereços de rede (NAT) ou o mascaramento para endereços IP de pods quando o tráfego do pod sair dos hosts on-premises. Você deverá configurar `REMOTE_POD_CIDRS` se estiver executando webhooks em nós híbridos. Consulte the section called “Configurar webhooks” para obter mais informações.

i. Os blocos CIDR de nós e pods on-premises devem atender aos seguintes requisitos:

   i. Estar dentro de um dos intervalos IPv4 RFC-1918: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.

   ii. Não se sobrepor entre si, ao `VPC CIDR` do cluster ou ao CIDR IPv4 do serviço do Kubernetes

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: CLUSTER_NAME
  region: AWS_REGION
  version: "K8S_VERSION"

remoteNetworkConfig:
  iam:
    provider: CREDS_PROVIDER # default SSM, can also be set to IRA
    # caBundleCert: CA_BUNDLE_CERT
  vpcGatewayID: GATEWAY_ID
  remoteNodeNetworks:
  - cidrs: ["REMOTE_NODE_CIDRS"]
  remotePodNetworks:
  - cidrs: ["REMOTE_POD_CIDRS"]
```

2. Execute o seguinte comando:

```
eksctl create cluster -f cluster-config.yaml
```

O provisionamento de cluster leva alguns minutos. Durante a criação do cluster, várias linhas de saída são exibidas. A última linha do resultado é semelhante ao exemplo de linha a seguir.

```
[#]  EKS cluster "CLUSTER_NAME" in "REGION" region is ready
```

3. Continue com the section called “Etapa 3: atualizar o kubeconfig”.

Criar um cluster habilitado para nós híbridos: AWS CloudFormation

A pilha do CloudFormation cria o perfil do IAM do cluster do EKS e um cluster do EKS com o `RemoteNodeNetwork` e `RemotePodNetwork` que você especifica. Modifique o modelo do CloudFormation se precisar personalizar as configurações do cluster do EKS que não estejam expostas no modelo do CloudFormation.

1. Baixe o modelo do CloudFormation.

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-eks-cfn.yaml'
```

2. Crie um `cfn-eks-parameters.json` e especifique a configuração para cada valor.
   a. `CLUSTER_NAME`: o nome do cluster do EKS a ser criado
   b. `CLUSTER_ROLE_NAME`: o nome do perfil do IAM do cluster do EKS a ser criado. O padrão no modelo é "EKSClusterRole".
   c. `SUBNET1_ID`: o ID da primeira sub-rede que você criou nas etapas de pré-requisitos.
   d. `SUBNET2_ID`: o ID da segunda sub-rede que você criou nas etapas de pré-requisitos.
   e. `SG_ID`: o ID do grupo de segurança que você criou nas etapas de pré-requisitos.
   f. `REMOTE_NODE_CIDRS`: o CIDR de nós on-premises para os nós híbridos.
   g. `REMOTE_POD_CIDRS`: o CIDR de pods on-premises para workloads executadas em nós híbridos. Você deverá configurar `REMOTE_POD_CIDRS` se a CNI não usar a conversão de endereços de rede (NAT) ou o mascaramento para endereços IP de pods quando o tráfego do pod sair dos hosts on-premises. Você deverá configurar `REMOTE_POD_CIDRS` se estiver executando webhooks em nós híbridos. Consulte the section called “Configurar webhooks” para obter mais informações.
   h. Os blocos CIDR de nós e pods on-premises devem atender aos seguintes requisitos:
      i. Estar dentro de um dos intervalos IPv4 RFC-1918: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.

ii. Não se sobrepor entre si, ao VPC CIDR do seu cluster ou ao CIDR IPv4 do seu serviço do Kubernetes.

i. CLUSTER_AUTH: o modo de autenticação do cluster para seu cluster. Os valores válidos são API e API_AND_CONFIG_MAP. O padrão no modelo é API_AND_CONFIG_MAP.

j. CLUSTER_ENDPOINT: a conectividade do endpoint do cluster para seu cluster. Os valores válidos são "Público" e "Privado". O padrão no modelo é Privado, o que significa que você só poderá se conectar ao endpoint da API do Kubernetes de dentro da VPC.

k. K8S_VERSION: a versão do Kubernetes para usar para o cluster. Consulte as versões compatíveis com o Amazon EKS.

```
{
  "Parameters": {
    "ClusterName": "CLUSTER_NAME",
    "ClusterRoleName": "CLUSTER_ROLE_NAME",
    "SubnetId1": "SUBNET1_ID",
    "SubnetId2": "SUBNET2_ID",
    "SecurityGroupId" "SG_ID",
    "RemoteNodeCIDR": "REMOTE_NODE_CIDRS",
    "RemotePodCIDR": "REMOTE_POD_CIDRS",
    "ClusterAuthMode": "CLUSTER_AUTH",
    "ClusterEndpointConnectivity": "CLUSTER_ENDPOINT",
    "K8sVersion": "K8S_VERSION"
  }
 }
```

3. Implante a pilha do CloudFormation. Substitua STACK_NAME pelo nome da pilha do CloudFormation e AWS_REGION pela região da AWS em que o cluster será criado.

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --template-file hybrid-eks-cfn.yaml \
    --parameter-overrides file://cfn-eks-parameters.json \
    --capabilities CAPABILITY_NAMED_IAM
```

O provisionamento de cluster leva alguns minutos. Você pode verificar o status da pilha com o comando a seguir. Substitua STACK_NAME pelo nome da pilha do CloudFormation e AWS_REGION pela região da AWS em que o cluster será criado.

```
aws cloudformation describe-stacks \
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --query 'Stacks[].StackStatus'
```

4. Continue com the section called “Etapa 3: atualizar o kubeconfig”.

Criar o cluster habilitado para nós híbridos: AWS CLI

1. Execute o comando apresentado a seguir para criar um cluster do EKS habilitado para nós híbridos. Antes de executar o comando, substitua as informações a seguir por suas próprias configurações. Para obter uma lista completa de configurações, consulte a documentação de the section called “Criar um cluster”.
   a. `CLUSTER_NAME`: o nome do cluster do EKS a ser criado
   b. `AWS_REGION`: região da AWS onde o cluster será criado.
   c. `K8S_VERSION`: a versão do Kubernetes para usar para o cluster. Consulte as versões compatíveis com o Amazon EKS.
   d. `ROLE_ARN`: o perfil do cluster do Amazon EKS que você configurou para seu cluster. Para obter mais informações, consulte perfil do IAM do cluster do Amazon EKS.
   e. `SUBNET1_ID`: o ID da primeira sub-rede que você criou nas etapas de pré-requisitos.
   f. `SUBNET2_ID`: o ID da segunda sub-rede que você criou nas etapas de pré-requisitos.
   g. `SG_ID`: o ID do grupo de segurança que você criou nas etapas de pré-requisitos.
   h. Você pode usar a `API` e `API_AND_CONFIG_MAP` para o modo de autenticação de acesso ao cluster. No comando abaixo, o modo de autenticação de acesso ao cluster está definido como `API_AND_CONFIG_MAP`.
   i. Você pode usar os parâmetros `endpointPublicAccess` e `endpointPrivateAccess` para habilitar ou desabilitar o acesso público e privado ao endpoint do servidor de API do Kubernetes do cluster. No comando abaixo, `endpointPublicAccess` está definido como false e `endpointPrivateAccess` está definido como true.
   j. `REMOTE_NODE_CIDRS`: o CIDR de nós on-premises para seus nós híbridos.
   k. `REMOTE_POD_CIDRS` (opcional): o CIDR de pods on-premises para workloads executadas em nós híbridos.
   l. Os blocos CIDR de nós e pods on-premises devem atender aos seguintes requisitos:

i. Estar dentro de um dos intervalos IPv4 RFC-1918: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.

ii. Não se sobrepor entre si, ao `VPC CIDR` do cluster do Amazon EKS ou ao CIDR IPv4 do serviço do Kubernetes.

```
aws eks create-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --kubernetes-version K8S_VERSION \
    --role-arn ROLE_ARN \
    --resources-vpc-config
 subnetIds=SUBNET1_ID,SUBNET2_ID,securityGroupIds=SG_ID,endpointPrivateAccess=true,endpo
 \
    --access-config authenticationMode=API_AND_CONFIG_MAP \
    --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. Leva alguns minutos para provisionar o cluster. Você pode consultar o status do cluster com o comando a seguir. Substitua `CLUSTER_NAME` pelo nome do cluster que você está criando e `AWS_REGION` pela região da AWS onde o cluster está sendo criado. Não prossiga para a próxima etapa até que a saída recebida esteja `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. Continue com the section called “Etapa 3: atualizar o kubeconfig”.

Criar o cluster habilitado para nós híbridos: Console de gerenciamento da AWS

1. Abra o console do Amazon EKS em Console do Amazon EKS.
2. Escolha Add cluster (Adicionar cluster) e, em seguida, Create (Criar).
3. Na página Configure cluster (Configurar cluster), preencha os seguintes campos:
   a. Name (Nome): um nome exclusivo para o cluster. O nome pode conter apenas caracteres alfanuméricos (diferencia maiúsculas de minúsculas), hifens e sublinhados. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.

b. Perfil do IAM do cluster: escolha o perfil do IAM do cluster do Amazon EKS que você criou para permitir que o ambiente de gerenciamento do Kubernetes gerencie os recursos da AWS em seu nome.

c. Kubernetes version (Versão do Kubernetes) – a versão do Kubernetes a ser usada para o cluster. Recomendamos que você selecione a versão mais recente, a menos que precise de uma versão anterior.

d. Política de atualização: escolha Estendida ou Padrão.

   i. Estendida: esta opção é compatível com a versão do Kubernetes por 26 meses após a data de lançamento. O período de suporte estendido tem um custo adicional por hora que começa após o término do período de suporte padrão. Quando o suporte estendido terminar, seu cluster será atualizado automaticamente para a próxima versão.

   ii. Padrão: esta opção é compatível com a versão do Kubernetes por 14 meses após a data de lançamento. Não há custo adicional. Quando o suporte padrão terminar, o cluster será atualizado automaticamente para a próxima versão.

e. Acesso ao cluster: escolha permitir ou não permitir o acesso do administrador do cluster e selecione um modo de autenticação. Os modos de autenticação a seguir são compatíveis com clusters habilitados para nós híbridos.

   i. API do EKS: o cluster obterá entidades principais do IAM autenticadas somente das APIs de entrada de acesso do EKS.

   ii. ConfigMap e API do EKS: o cluster obterá entidades principais do IAM autenticadas tanto das APIs de entrada de acesso do EKS quanto do ConfigMap `aws-auth`.

f. Secrets encryption (Criptografia de segredos): (Opcional) Escolha habilitar a criptografia de segredos do Kubernetes usando uma chave do KMS. Também é possível isso depois de criar o cluster. Antes de habilitar esse recurso, familiarize-se com as informações em the section called “Habilitar a criptografia de segredos”.

g. Mudança de zona do ARC: se habilitado, o EKS registrará o cluster com a mudança de zona do ARC para possibilitar o uso da mudança de zona para redirecionar o tráfego da aplicação para fora de uma AZ.

h. Tags (Etiquetas) – (opcional) adicione etiquetas ao cluster. Para obter mais informações, consulte the section called “Marcando seus Recursos”.

i. Após terminar com essa página, escolha Próximo.

4. Na página Specify networking (Especificar redes), selecione valores para os seguintes campos:

   a. VPC: escolha uma VPC existente que atenda aos requisitos do Amazon EKS Hybrid Nodes e the section called “Requisitos para sub-rede e VPC”. Antes de escolher uma VPC,

recomendamos que você esteja familiarizado com todos os requisitos e considerações apresentados em Visualizar os requisitos de rede do Amazon EKS para VPC, sub-redes e nós híbridos. Não será possível alterar quais VPCs você deseja utilizar depois de criar o cluster. Se nenhuma VPC estiver listada, você precisará criar uma primeiro. Para obter mais informações, consulte the section called “Crie uma VPC” e os requisitos de rede do Amazon EKS Hybrid Nodes.

b. Subnets (Sub-redes): por padrão, as sub-redes disponíveis na VPC especificada no campo anterior são pré-selecionadas. É necessário selecionar pelo menos duas.

c. Security groups (Grupos de segurança) (Opcional): especifique um ou mais grupos de segurança que você deseja que o Amazon EKS associe às interfaces de rede que ele cria. Pelo menos um dos grupos de segurança que você especificar deve ter regras de entrada para o nó on-premises e, opcionalmente, para os CIDRs de pods. Consulte os requisitos de rede do Amazon EKS Hybrid Nodes para obter mais informações. Independentemente de você escolher algum grupo de segurança, o Amazon EKS cria um grupo de segurança que permite comunicação entre seu cluster e sua VPC. O Amazon EKS associa esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para obter mais informações sobre o grupo de segurança do cluster que o Amazon EKS cria, consulte the section called “Requisitos para grupos de segurança”. Você pode modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS.

d. Escolha a família de endereços IP do cluster: você deve escolher IPv4 para clusters habilitados para nós híbridos.

e. (Opcional) Escolha Configurar intervalo de endereços IP do Kubernetes e especifique um Intervalo de serviços IPv4.

f. Escolha Configurar redes remotas para habilitar nós híbridos e especifique seus CIDRs de nós e pods on-premises para nós híbridos.

g. Você deverá configurar o CIDR remoto de pods se a CNI não usar a conversão de endereços de rede (NAT) ou o mascaramento para endereços IP de pods quando o tráfego de pods sair dos hosts on-premises. Você deverá configurar o CIDR remoto de pods se estiver executando webhooks em nós híbridos.

h. Os blocos CIDR de nós e pods on-premises devem atender aos seguintes requisitos:

   i. Estar dentro de um dos intervalos IPv4 RFC-1918: `10.0.0.0/8`, `172.16.0.0/12` ou `192.168.0.0/16`.

   ii. Não se sobrepor entre si, ao `VPC CIDR` do cluster ou ao CIDR IPv4 do serviço do Kubernetes

i. Para Cluster endpoint access (Acesso ao endpoint do cluster), selecione uma opção. Depois de criar o cluster, você poderá alterar essa opção. Para clusters habilitados para nós híbridos, você deve escolher entre Público ou Privado. Antes de selecionar uma opção não padrão, familiarize-se com as opções e suas implicações. Para obter mais informações, consulte the section called “Acesso ao endpoint do cluster”.

j. Após terminar com essa página, escolha Próximo.

5. (Opcional) Na página Configurar observabilidade, escolha quais opções de métricas e registros em log do ambiente de gerenciamento deseja ativar. Por padrão, o cada tipo de log está desativado.

   a. Para obter mais informações sobre as opções de métricas do Prometheus, consulte the section called “Métricas do Prometheus”.

   b. Para obter mais informações sobre as opções de registro em log do ambiente de gerenciamento do EKS, consulte the section called “Logs do ambiente de gerenciamento”.

   c. Após terminar com essa página, escolha Próximo.

6. Na página Selecionar complementos, escolha os complementos que você deseja adicionar ao cluster.

   a. Você pode escolher quantos complementos do Amazon EKS e do AWS Marketplace desejar. Os complementos do Amazon EKS que não são compatíveis com nós híbridos são marcados com "Não compatível com nós híbridos", e os complementos têm uma regra de antiafinidade para evitar que sejam executados em nós híbridos. Consulte Configurar complementos para nós híbridos para obter mais informações. Se os complementos do AWS Marketplace que você deseja instalar não estiverem listados, você poderá pesquisar os complementos disponíveis no AWS Marketplace inserindo um texto na caixa de pesquisa. Você também poderá pesquisar por category (categoria), vendor (fornecedor) ou pricing model (modelo de preços) e depois escolher os complementos nos resultados da pesquisa.

   b. Alguns complementos, como CoreDNS e kube-proxy, são instalados por padrão. Se você desabilitar qualquer um dos complementos padrão, isso poderá afetar sua capacidade de executar aplicações do Kubernetes.

   c. Após terminar com esta página, escolha `Next`.

7. Na página Configurar opções de complementos selecionados, selecione a versão que deseja instalar.

   a. Você sempre pode atualizar para uma versão posterior após a criação do cluster. Você pode atualizar a configuração de cada complemento após a criação do cluster. Para obter mais informações sobre a configuração de complementos, consulte the section called “Atualização

de um complemento". Para as versões de complementos que são compatíveis com nós híbridos, consulte the section called "Configurar complementos".

b. Após terminar com essa página, escolha Próximo.

8. Na página Review and create (Revisar e criar), revise as informações que você inseriu ou selecionou nas páginas anteriores. Se precisar fazer alterações, escolha Edit (Editar). Quando estiver satisfeito, escolha Criar. O campo Status mostra o campo CREATING (Criando) enquanto o cluster é provisionado. O provisionamento de cluster leva alguns minutos.
9. Continue com the section called "Etapa 3: atualizar o kubeconfig".

Etapa 3: atualizar o kubeconfig

Se você criou seu cluster usando `eksctl`, pode ignorar esta etapa. Isso ocorre porque `eksctl` já concluiu essa etapa para você. Habilite o `kubectl` para se comunicar com o cluster adicionando um novo contexto ao arquivo de configuração do `kubectl`. Para obter mais informações sobre como criar e atualizar o arquivo, consulte the section called "Acessar meu cluster com o kubectl".

```
aws eks update-kubeconfig --name CLUSTER_NAME --region AWS_REGION
```

Veja um exemplo de saída abaixo.

```
Added new context arn:aws:eks:AWS_REGION:111122223333:cluster/CLUSTER_NAME to /home/
username/.kube/config
```

Confirme a comunicação com o cluster, executando o seguinte comando:

```
kubectl get svc
```

Veja abaixo um exemplo de saída.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

Etapa 4: configuração do cluster

Como próxima etapa, consulte the section called "Preparar acesso ao cluster" para habilitar o acesso para que os nós híbridos se juntem ao cluster.

Habilitar nós híbridos em um cluster existente do Amazon EKS ou modificar a configuração

Este tópico fornece uma visão geral das opções disponíveis e descreve o que deve ser considerado ao adicionar, alterar ou remover a configuração de nós híbridos de um cluster do Amazon EKS.

Para permitir que um cluster do Amazon EKS use nós híbridos, adicione os intervalos de CIDR de endereços IP do seu nó on-premises e, opcionalmente, a rede de pods na configuração `RemoteNetworkConfig`. O EKS usa essa lista de CIDRs para habilitar a conectividade entre o cluster e suas redes on-premises. Para obter uma lista completa de opções ao atualizar sua configuração de cluster, consulte UpdateClusterConfig na Referência de APIs do Amazon EKS.

Você pode realizar qualquer uma das seguintes ações na configuração de rede do EKS Hybrid Nodes em um cluster:

- Adicione a configuração de rede remota para habilitar o EKS Hybrid Nodes em um cluster existente.
- Adicione, altere ou remova as redes de nós remotos ou as redes de pods remotos em um cluster existente.
- Remova todos os intervalos de CIDR da rede de nós remotos para desabilitar o EKS Hybrid Nodes em um cluster existente.

Pré-requisitos

- Antes de habilitar o cluster do Amazon EKS para nós híbridos, certifique-se de que seu ambiente atenda aos requisitos descritos em the section called “Pré-requisitos” e detalhados em the section called “Preparar a rede” the section called “Preparar o sistema operacional” e the section called “Preparar credenciais”.
- O cluster deve usar a família de endereços IPv4.
- O cluster deve usar a `API` ou `API_AND_CONFIG_MAP` para o modo de autenticação do cluster. O processo para modificar o modo de autenticação do cluster está descrito em the section called “Modo de autenticação”.
- Recomendamos usar o acesso de endpoint público ou privado para o endpoint do servidor da API do Kubernetes do Amazon EKS, mas não ambos. Se você escolher “Público e Privado”, o endpoint do servidor da API do Kubernetes do Amazon EKS sempre resolverá para os IPs públicos dos nós híbridos em execução fora da VPC, o que pode impedir que os nós híbridos se juntem

ao cluster. O processo para modificar o acesso à rede ao seu cluster está descrito em the section called “Acesso ao endpoint do cluster”.

- A versão mais recente da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo. Para verificar sua versão atual, use `aws --version`. Os gerenciadores de pacotes, como yum, apt-get ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar ou atualizar a versão mais recente da AWS CLI e Como definir as configurações da AWS CLI no Guia do usuário da AWS Command Line Interface.
- Uma entidade principal do IAM com permissão para chamar UpdateClusterConfig no cluster do Amazon EKS.
- Atualize os complementos para as versões que são compatíveis com nós híbridos. Para as versões de complementos que são compatíveis com nós híbridos, consulte the section called “Configurar complementos”.
- Se você estiver executando complementos que não sejam compatíveis com nós híbridos, certifique-se de que o complemento DaemonSet ou Deployment tenha a regra de afinidade a seguir para evitar a implantação em nós híbridos. Adicione a regra de afinidade a seguir caso não esteja presente.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Considerações

O objeto JSON `remoteNetworkConfig` tem o seguinte comportamento durante uma atualização:

- Qualquer parte existente da configuração que você não especificar permanecerá inalterada. Se você não especificar nenhuma das `remoteNodeNetworks` ou `remotePodNetworks`, essa parte permanecerá a mesma.
- Caso esteja modificando as listas de CIDRs das `remoteNodeNetworks` ou `remotePodNetworks`, você deverá especificar a lista completa de CIDRs que deseja

em sua configuração final. Quando você especifica uma alteração na lista de CIDRs nas `remoteNodeNetworks` ou `remotePodNetworks`, o EKS substitui a lista original durante a atualização.

- Os blocos CIDR de nós e pods on-premises devem atender aos seguintes requisitos:
  1. Estar dentro de um dos seguintes intervalos de IPv4 RFC-1918: 10.0.0.0/8, 172.16.0.0/12 ou 192.168.0.0/16.
  2. Não se sobrepor entre si, todos os CIDRs da VPC do cluster do Amazon EKS ou com o CIDR de IPv4 do serviço do Kubernetes.

Habilitar nós híbridos em um cluster existente

Você pode habilitar o EKS Hybrid Nodes em um cluster existente usando:

- AWS CloudFormation
- AWS CLI
- Console de gerenciamento da AWS

Habilitar o EKS Hybrid Nodes em um cluster existente: AWS CloudFormation

1. Para habilitar o EKS Hybrid Nodes em seu cluster, adicione a `RemoteNodeNetwork` e (opcional) a `RemotePodNetwork` ao seu modelo do CloudFormation, e atualize a pilha. Observe que a `RemoteNodeNetwork` é uma lista com no máximo um item `Cidrs`. e o `Cidrs` é uma lista de vários intervalos de CIDRs de IP.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [RemoteNodeCIDR]
  RemotePodNetworks:
    - Cidrs: [RemotePodCIDR]
```

2. Avance para the section called “Preparar acesso ao cluster”.

Habilitar o EKS Hybrid Nodes em um cluster existente: AWS CLI

1. Execute o comando a seguir para habilitar a `RemoteNetworkConfig` para o EKS Hybrid Nodes para seu cluster do EKS. Antes de executar o comando, substitua as informações a seguir

por suas próprias configurações. Para obter uma lista completa das configurações, consulte UpdateClusterConfig na Referência de APIs do Amazon EKS.

a. `CLUSTER_NAME`: nome do cluster do EKS a ser atualizado.

b. `AWS_REGION`: região da AWS em que o cluster do EKS está em execução.

c. `REMOTE_NODE_CIDRS`: o CIDR de nós on-premises para seus nós híbridos.

d. `REMOTE_POD_CIDRS` (opcional): o CIDR de pods on-premises para workloads executadas em nós híbridos.

```
aws eks update-cluster-config \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. Leva alguns minutos para atualizar o cluster. Você pode consultar o status do cluster com o comando a seguir. Substitua `CLUSTER_NAME` pelo nome do cluster que você está modificando e `AWS_REGION` pela região da AWS em que o cluster está sendo executado. Não prossiga para a próxima etapa até que a saída recebida esteja `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. Avance para the section called “Preparar acesso ao cluster”.

Habilitar o EKS Hybrid Nodes em um cluster existente: Console de gerenciamento da AWS

1. Abra o console do Amazon EKS em Console do Amazon EKS.
2. Selecione o nome do cluster para exibir as informações dele.
3. Escolha a guia Redes e depois Gerenciar.
4. No menu suspenso, escolha Redes remotas.
5. Escolha Configurar redes remotas para habilitar nós híbridos e especifique seus CIDRs de nós e pods on-premises para nós híbridos.
6. Escolha Save changes (Salvar alterações) para terminar. Aguarde até que o status do cluster retorne para Active.
7. Avance para the section called “Preparar acesso ao cluster”.

Atualizar a configuração de nós híbridos em um cluster existente

É possível modificar `remoteNetworkConfig` em um cluster híbrido existente usando qualquer uma destas opções:

- AWS CloudFormation
- AWS CLI
- Console de gerenciamento da AWS

Atualizar a configuração híbrida em um cluster existente: AWS CloudFormation

1. Atualize seu modelo do CloudFormation com os novos valores de CIDR da rede.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [NEW_REMOTE_NODE_CIDRS]
  RemotePodNetworks:
    - Cidrs: [NEW_REMOTE_POD_CIDRS]
```

> **Note**
>
> Ao atualizar as listas de CIDRs de `RemoteNodeNetworks` ou `RemotePodNetworks`, inclua todos os CIDRs (novos e existentes). O EKS substitui toda a lista durante as atualizações. A omissão desses campos da solicitação de atualização mantém as configurações existentes.

2. Atualize sua pilha do CloudFormation com o modelo modificado e aguarde a conclusão da atualização da pilha.

Atualizar a configuração híbrida em um cluster existente: AWS CLI

1. Para modificar os CIDRs da rede remota, execute o comando a seguir. Substitua os valores por suas próprias configurações:

```
aws eks update-cluster-config
--name CLUSTER_NAME
--region AWS_REGION
```

```
--remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["NEW_REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":
["NEW_REMOTE_POD_CIDRS"]}]}'
```

> **Note**
>
> Ao atualizar as listas de CIDRs de `remoteNodeNetworks` ou `remotePodNetworks`, inclua todos os CIDRs (novos e existentes). O EKS substitui toda a lista durante as atualizações. A omissão desses campos da solicitação de atualização mantém as configurações existentes.

2. Aguarde até que o status do cluster retorne para ACTIVE antes de continuar.

Atualizar a configuração híbrida em um cluster existente: Console de gerenciamento da AWS

1. Abra o console do Amazon EKS em Console do Amazon EKS.
2. Selecione o nome do cluster para exibir as informações dele.
3. Escolha a guia Redes e depois Gerenciar.
4. No menu suspenso, escolha Redes remotas.
5. Atualize os CIDRs em `Remote node networks` e `Remote pod networks - Optional`, conforme necessário.
6. Escolha Salvar alterações e aguarde até que o status do cluster retorne para Active.

Desabilitar nós híbridos em um cluster existente

Você pode desabilitar o EKS Hybrid Nodes em um cluster existente usando:

- AWS CloudFormation
- AWS CLI
- Console de gerenciamento da AWS

Desabilitar o EKS Hybrid Nodes em um cluster existente: AWS CloudFormation

1. Para desabilitar o EKS Hybrid Nodes em seu cluster, defina `RemoteNodeNetworks` e `RemotePodNetworks` para esvaziar matrizes em seu modelo do CloudFormation e atualize a pilha.

```
RemoteNetworkConfig:
  RemoteNodeNetworks: []
  RemotePodNetworks: []
```

Desabilitar o EKS Hybrid Nodes em um cluster existente: AWS CLI

1. Execute o comando a seguir para remover a `RemoteNetworkConfig` do cluster do EKS. Antes de executar o comando, substitua as informações a seguir por suas próprias configurações. Para obter uma lista completa das configurações, consulte UpdateClusterConfig na Referência de APIs do Amazon EKS.

   a. `CLUSTER_NAME`: nome do cluster do EKS a ser atualizado.

   b. `AWS_REGION`: região da AWS em que o cluster do EKS está em execução.

   ```
   aws eks update-cluster-config \
       --name CLUSTER_NAME \
       --region AWS_REGION \
       --remote-network-config '{"remoteNodeNetworks":[],"remotePodNetworks":[]}'
   ```

2. Leva alguns minutos para atualizar o cluster. Você pode consultar o status do cluster com o comando a seguir. Substitua `CLUSTER_NAME` pelo nome do cluster que você está modificando e `AWS_REGION` pela região da AWS em que o cluster está sendo executado. Não prossiga para a próxima etapa até que a saída recebida esteja `ACTIVE`.

   ```
   aws eks describe-cluster \
       --name CLUSTER_NAME \
       --region AWS_REGION \
       --query "cluster.status"
   ```

Desabilitar o EKS Hybrid Nodes em um cluster existente: Console de gerenciamento da AWS

1. Abra o console do Amazon EKS em Console do Amazon EKS.
2. Selecione o nome do cluster para exibir as informações dele.
3. Escolha a guia Redes e depois Gerenciar.
4. No menu suspenso, escolha Redes remotas.

5. Escolha Configurar redes remotas para habilitar nós híbridos e remova todos os CIDRs em `Remote node networks` e `Remote pod networks - Optional`.
6. Escolha Save changes (Salvar alterações) para terminar. Aguarde até que o status do cluster retorne para Active.

Preparar o acesso ao cluster para nós híbridos

Antes de conectar nós híbridos ao cluster do Amazon EKS, você deve habilitar o perfil do IAM de nós híbridos com permissões do Kubernetes para ingressar no cluster. Para obter informações sobre como criar um perfil do IAM de nós híbridos, consulte the section called “Preparar credenciais”.
O Amazon EKS é compatível com duas maneiras de associar as entidades principais do IAM ao controle de acesso por perfil (RBAC) do Kubernetes, entradas de acesso do Amazon EKS e o ConfigMap `aws-auth`. Para obter mais informações sobre o gerenciamento de acesso do Amazon EKS, consulte the section called “Acesso à API do Kubernetes”.

Use os procedimentos abaixo para associar o perfil do IAM de nós híbridos às permissões do Kubernetes. Para usar as entradas de acesso do Amazon EKS, o cluster deve ter sido criado com os modos de autenticação da `API` ou `API_AND_CONFIG_MAP`. Para usar o ConfigMap `aws-auth`, o cluster deve ter sido criado com o modo de autenticação `API_AND_CONFIG_MAP`. O modo de autenticação somente `CONFIG_MAP` não é compatível com clusters do Amazon EKS habilitados para nós híbridos.

Usar entradas de acesso do Amazon EKS para o perfil do IAM de nós híbridos

Há um tipo de entrada de acesso do Amazon EKS para nós híbridos denominado HYBRID_LINUX que pode ser usado com um perfil do IAM. Com esse tipo de entrada de acesso, o nome de usuário é automaticamente definido como system:node:{{SessionName}}. Para obter mais informações sobre como criar entradas de acesso, consulte the section called “Criar entradas de acesso”.

AWS CLI

1. Você deve ter a versão mais recente da AWS CLI instalada e configurada no dispositivo. Para verificar sua versão atual, use `aws --version`. Os gerenciadores de pacotes, como yum, apt-get ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Instalar e Configuração rápida com aws configure no Guia do usuário da AWS Command Line Interface.

2. Crie a entrada de acesso com o comando a seguir. Substitua CLUSTER_NAME pelo nome do cluster e HYBRID_NODES_ROLE_ARN pelo ARN do perfil que você criou nas etapas para the section called “Preparar credenciais”.

```
aws eks create-access-entry --cluster-name CLUSTER_NAME \
    --principal-arn HYBRID_NODES_ROLE_ARN \
    --type HYBRID_LINUX
```

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS em Console do Amazon EKS.
2. Escolha o nome do cluster habilitado para nós híbridos.
3. Escolha a guia Acesso.
4. Selecione Criar entrada de acesso.
5. Em Entidade principal do IAM, selecione o perfil do IAM de nós híbridos que você criou nas etapas para the section called “Preparar credenciais”.
6. Para Tipo, selecione Hybrid Linux.
7. (Opcional) Para Tags, atribua rótulos à entrada de acesso. Por exemplo, para facilitar a localização de todos os recursos com a mesma tag.
8. Selecione Ignorar para analisar e criar. Você não pode adicionar políticas à entrada de acesso do Hybrid Linux nem alterar o escopo de acesso.
9. Revise a configuração da entrada de acesso. Se algo parecer errado, escolha Anterior para voltar às etapas anteriores e corrigir o erro. Se a configuração estiver correta, escolha Criar.

Usar o ConfigMap aws-auth para o perfil do IAM de nós híbridos

Nas etapas a seguir, você criará ou atualizará o ConfigMap `aws-auth` com o ARN do perfil do IAM de nós híbridos que você criou nas etapas para the section called “Preparar credenciais”.

1. Verifique se você tem um ConfigMap `aws-auth` existente para o cluster. Observe que, se você estiver usando um arquivo `kubeconfig` específico, use o sinalizador `--kubeconfig`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se você receber um ConfigMap `aws-auth`, atualize-o conforme necessário.

a. Abra o ConfigMap para edição.

```
kubectl edit -n kube-system configmap/aws-auth
```

b. Adicione uma nova `mapRoles` entrada conforme necessário. Substitua `HYBRID_NODES_ROLE_ARN` pelo ARN do perfil do IAM de nós híbridos. Observe que `{{SessionName}}` é o formato de modelo correto para salvar no ConfigMap. Não o substitua por outros valores.

```
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
```

c. Salve o arquivo e saia do seu editor de texto.

3. Se não houver um ConfigMap `aws-auth` existente para o cluster, crie-o com o comando a seguir. Substitua `HYBRID_NODES_ROLE_ARN` pelo ARN do perfil do IAM de nós híbridos. Observe que `{{SessionName}}` é o formato de modelo correto para salvar no ConfigMap. Não o substitua por outros valores.

```
kubectl apply -f=/dev/stdin <<-EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
EOF
```

Execução de workloads on-premises em nós híbridos

Em um cluster do EKS com nós híbridos habilitados, você pode executar aplicações on-premises e de borda em sua própria infraestrutura com os mesmos clusters, recursos e ferramentas do Amazon EKS que você usa na Nuvem AWS.

As seções a seguir contêm instruções passo a passo para usar nós híbridos.

Tópicos

- Conectar nós híbridos
- Conectar nós híbridos com o Bottlerocket
- Atualizar os nós híbridos do cluster
- Atualizações de segurança de patches para nós híbridos
- Remoção de nós híbridos

Conectar nós híbridos

> **Note**
>
> As etapas a seguir se aplicam aos nós híbridos que executam sistemas operacionais compatíveis, exceto o Bottlerocket. Para obter as etapas para conectar um nó híbrido que executa o Bottlerocket, consulte the section called “Conectar nós híbridos com o Bottlerocket”.

Este tópico descreve como conectar nós híbridos a um cluster do Amazon EKS. Depois que os nós híbridos se unirem ao cluster, eles aparecerão com o status Não pronto no console do Amazon EKS e em ferramentas compatíveis com o Kubernetes, como o kubectl. Depois de concluir as etapas desta página, prossiga para the section called “Configurar CNI” para preparar os nós híbridos para executar aplicações.

Pré-requisitos

Antes de conectar nós híbridos ao cluster do Amazon EKS, certifique-se de ter concluído as etapas de pré-requisitos.

- Você tem conectividade de rede do ambiente on-premises para a região da AWS que hospeda o cluster do Amazon EKS. Consulte the section called “Preparar a rede” para obter mais informações.
- Você tem um sistema operacional compatível para nós híbridos instalado nos hosts on-premises. Consulte the section called “Preparar o sistema operacional” para obter mais informações.
- Você criou o perfil do IAM de nós híbridos e configurou o provedor de credenciais on-premises (ativações híbridas do AWS Systems Manager ou o AWS IAM Roles Anywhere). Consulte the section called “Preparar credenciais” para obter mais informações.
- Você criou o cluster do Amazon EKS habilitado para nós híbridos. Consulte the section called “Criação de cluster” para obter mais informações.
- Você associou o perfil do IAM de nós híbridos às permissões de controle de acesso por perfil (RBAC) do Kubernetes. Consulte the section called “Preparar acesso ao cluster” para obter mais informações.

Etapa 1: instalar a CLI de nós híbridos (**`nodeadm`**) em cada host on-premises

Caso esteja incluindo a CLI (`nodeadm`) do Amazon EKS Hybrid Nodes nas imagens pré-criadas do sistema operacional, você pode pular esta etapa. Para obter mais informações sobre a versão de nós híbridos de `nodeadm`, consulte the section called “Nós híbridos nodeadm”.

A versão de nós híbridos do `nodeadm` está hospedada no Amazon S3 com o Amazon CloudFront na frente. Para instalar `nodeadm` em cada host on-premises, você pode executar o comando a seguir nos hosts on-premises.

Para hosts x86_64:

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/
nodeadm'
```

Para hosts ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/
nodeadm'
```

Adicione uma permissão de arquivo executável ao binário baixado em cada host.

```
chmod +x nodeadm
```

Etapa 2: instalar as dependências dos nós híbridos com **`nodeadm`**

Caso esteja instalando as dependências de nós híbridos em imagens pré-criadas do sistema operacional, você pode pular esta etapa. O comando `nodeadm install` pode ser usado para instalar todas as dependências necessárias para nós híbridos. As dependências dos nós híbridos incluem containerd, kubelet, kubectl e componentes do AWS SSM ou AWS IAM Roles Anywhere. Consulte the section called “Nós híbridos nodeadm” para obter mais informações sobre os componentes e os locais dos arquivos instalados por `nodeadm install`. Consulte the section called “Preparar a rede” para conferir os nós híbridos e obter mais informações sobre os domínios que devem ser permitidos no firewall on-premises para o processo de `nodeadm install`.

Execute o comando abaixo para instalar as dependências dos nós híbridos no host on-premises. O comando abaixo deve ser executado com um usuário que tenha acesso sudo/root no host.

> **Important**
>
> A CLI de nós híbridos (`nodeadm`) deve ser executada com um usuário que tenha acesso sudo/root no host.

- Substitua `K8S_VERSION` pela versão secundária do Kubernetes do cluster do Amazon EKS, por exemplo, `1.31`. Consulte as versões compatíveis com o Amazon EKS para obter uma lista das versões do Kubernetes compatíveis.
- Substitua `CREDS_PROVIDER` pelo provedor de credenciais on-premises que você está usando. Os valores válidos são `ssm` para o AWS SSM e `iam-ra` para o AWS IAM Roles Anywhere.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER
```

Etapa 3: conectar nós híbridos ao cluster

Antes de conectar os nós híbridos ao cluster, certifique-se de ter permitido o acesso necessário no firewall on-premises e no grupo de segurança do cluster para a comunicação do ambiente de gerenciamento do Amazon EKS de/para nós híbridos. A maioria dos problemas nesta etapa está relacionada à configuração do firewall, à configuração do grupo de segurança ou à configuração do perfil do IAM de nós híbridos.

> **Important**
>
> A CLI de nós híbridos (`nodeadm`) deve ser executada com um usuário que tenha acesso sudo/root no host.

1. Crie um arquivo `nodeConfig.yaml` em cada host com os valores da implantação. Para obter uma descrição completa das configurações disponíveis, consulte the section called "Nós híbridos nodeadm". Se o perfil do IAM de nós híbridos não tiver permissão para a ação `eks:DescribeCluster`, você deverá passar o endpoint da API Kubernetes, o pacote de CA do cluster e o CIDR IPv4 do serviço do Kubernetes na seção de clusters do `nodeConfig.yaml`.
   a. Use o exemplo de `nodeConfig.yaml` abaixo se estiver usando ativações híbridas do AWS SSM para o provedor de credenciais on-premises.
      i. Substitua o `CLUSTER_NAME` pelo nome do cluster.
      ii. Substitua `AWS_REGION` pela região da AWS que está hospedando o cluster. Por exemplo, `.us-west-2`
      iii. Substitua `ACTIVATION_CODE` pelo código de ativação que você recebeu ao criar a ativação híbrida do AWS SSM. Consulte the section called "Preparar credenciais" para obter mais informações.
      iv. Substitua `ACTIVATION_ID` pelo ID de ativação que você recebeu ao criar a ativação híbrida do AWS SSM. Você pode recuperar essas informações no console do AWS Systems Manager ou com o comando `aws ssm describe-activations` da AWS CLI.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    ssm:
      activationCode: ACTIVATION_CODE
      activationId: ACTIVATION_ID
```

   b. Use o exemplo de `nodeConfig.yaml` abaixo se estiver usando o AWS IAM Roles Anywhere para o provedor de credenciais on-premises.
      i. Substitua o `CLUSTER_NAME` pelo nome do cluster.

ii. Substitua `AWS_REGION` pela região da AWS que está hospedando o cluster. Por exemplo, `.us-west-2`

iii. Substitua `NODE_NAME` pelo nome do nó. O nome do nó deve corresponder ao CN do certificado no host caso tenha configurado a política de confiança do perfil do IAM de nós híbridos com a condição de recurso `"sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"`. O `nodeName` que você usar não deve ter mais de 64 caracteres.

iv. Substitua `TRUST_ANCHOR_ARN` pelo ARN da âncora de confiança que você configurou nas etapas de Preparar credenciais para nós híbridos.

v. Substitua `PROFILE_ARN` pelo ARN da âncora de confiança que você configurou nas etapas de the section called “Preparar credenciais”.

vi. Substitua `ROLE_ARN` pelo ARN do perfil do IAM de nós híbridos.

vii. Substitua `CERTIFICATE_PATH` pelo caminho no disco para o certificado do nó. Caso não seja especificado, o padrão será `/etc/iam/pki/server.pem`.

viii. Substitua `KEY_PATH` pelo caminho no disco para a chave privada do certificado. Caso não seja especificado, o padrão será `/etc/iam/pki/server.key`.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    iamRolesAnywhere:
      nodeName: NODE_NAME
      trustAnchorArn: TRUST_ANCHOR_ARN
      profileArn: PROFILE_ARN
      roleArn: ROLE_ARN
      certificatePath: CERTIFICATE_PATH
      privateKeyPath: KEY_PATH
```

2. Execute o comando `nodeadm init` com o `nodeConfig.yaml` para conectar os nós híbridos ao cluster do Amazon EKS.

```
nodeadm init -c file://nodeConfig.yaml
```

Se o comando acima for concluído com sucesso, o nó híbrido terá se unido ao cluster do Amazon EKS. Você pode verificar isso no console do Amazon EKS navegando até a guia Computação do cluster (certifique-se de que a entidade principal do IAM tenha permissões para visualizar) ou com `kubectl get nodes`.

> **Important**
>
> Seus nós terão o status `Not Ready`, o que é esperado e se deve à falta de uma CNI em execução nos nós híbridos. Se os nós não tiverem se unido ao cluster, consulte the section called “Solução de problemas”.

Etapa 4: configurar uma CNI para nós híbridos

Para preparar os nós híbridos para executar aplicações, continue com as etapas em the section called “Configurar CNI”.

Conectar nós híbridos com o Bottlerocket

Este tópico descreve como conectar nós híbridos que executam o Bottlerocket a um cluster do Amazon EKS. O Bottlerocket é uma distribuição de código aberto do Linux patrocinada e apoiada pela AWS. O Bottlerocket foi desenvolvido especificamente para hospedar workloads de contêineres. Com o Bottlerocket, você pode melhorar a disponibilidade de implantações em contêineres e reduzir os custos operacionais automatizando as atualizações em sua infraestrutura de contêineres. O Bottlerocket inclui somente o software essencial para executar contêineres, o que melhora o uso de recursos, reduz as ameaças à segurança e diminui a sobrecarga de gerenciamento.

Somente as variantes VMware do Bottlerocket versão v1.37.0 e superior são compatíveis com os nós híbridos do EKS. As variantes de VMware do Bottlerocket estão disponíveis para as versões v1.28 e superiores do Kubernetes. As imagens do sistema operacional para essas variantes incluem o kubelet, containerd, aws-iam-authenticator e outros pré-requisitos de software para nós híbridos do EKS. É possível configurar esses componentes usando um arquivo de configurações do Bottlerocket que inclui dados do usuário codificados em base64 para os contêineres de bootstrap e administração do Bottlerocket. Configurar essas opções permite que o Bottlerocket use seu provedor de credenciais de nós híbridos para autenticar nós híbridos em seu cluster. Depois que os nós híbridos se unirem ao cluster, eles aparecerão com o status `Not Ready` no console do Amazon EKS e em ferramentas compatíveis com o Kubernetes, como o `kubectl`. Depois de concluir as etapas desta página, prossiga para the section called “Configurar CNI” para preparar os nós híbridos para executar aplicações.

Pré-requisitos

Antes de conectar nós híbridos ao cluster do Amazon EKS, certifique-se de ter concluído as etapas de pré-requisitos.

- Você tem conectividade de rede do ambiente on-premises para a região da AWS que hospeda o cluster do Amazon EKS. Consulte the section called “Preparar a rede” para obter mais informações.
- Você criou o perfil do IAM de nós híbridos e configurou o provedor de credenciais on-premises (ativações híbridas do AWS Systems Manager ou o AWS IAM Roles Anywhere). Consulte the section called “Preparar credenciais” para obter mais informações.
- Você criou o cluster do Amazon EKS habilitado para nós híbridos. Consulte the section called “Criação de cluster” para obter mais informações.
- Você associou o perfil do IAM de nós híbridos às permissões de controle de acesso por perfil (RBAC) do Kubernetes. Consulte the section called “Preparar acesso ao cluster” para obter mais informações.

Etapa 1: criar o arquivo TOML de configurações do Bottlerocket

Para configurar o Bottlerocket para nós híbridos, você precisa criar um arquivo `settings.toml` com a configuração necessária. O conteúdo do arquivo TOML será diferente com base no provedor de credenciais utilizado (SSM ou IAM Roles Anywhere). Esse arquivo será passado como dados do usuário ao provisionar a instância do Bottlerocket.

> **Note**
>
> Os arquivos no formato TOML fornecidos abaixo representam somente as configurações mínimas necessárias para inicializar uma máquina com sistema operacional Bottlerocket na VMWare como um nó em um cluster do EKS. O Bottlerocket oferece uma ampla variedade de configurações para atender a diversos casos de uso. Portanto, para explorar outras opções de configuração além da inicialização de nós híbridos, consulte a documentação do Bottlerocket. Nessa documentação, você encontrará a lista completa de todas as configurações documentadas da versão do Bottlerocket que você está usando (por exemplo, estas são as configurações disponíveis para o Bottlerocket 1.51.x).

SSM

Se você estiver usando o AWS Systems Manager como seu provedor de credenciais, crie um arquivo `settings.toml` com o seguinte conteúdo:

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"
cloud-provider = ""
server-tls-bootstrap = true

[settings.network]
hostname = "<hostname>"

[settings.aws]
region = "<region>"

[settings.kubernetes.credential-providers.ecr-credential-provider]
enabled = true
cache-duration = "12h"
image-patterns = [
    "*.dkr.ecr.*.amazonaws.com",
    "*.dkr.ecr.*.amazonaws.com.cn",
    "*.dkr.ecr.*.amazonaws.eu",
    "*.dkr.ecr-fips.*.amazonaws.com",
    "*.dkr.ecr-fips.*.amazonaws.eu",
    "public.ecr.aws"
]

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
"eks.amazonaws.com/hybrid-credential-provider" = "ssm"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"
```

```
[settings.host-containers.control]
enabled = true
```

Substitua os espaços reservados pelos seguintes valores:

- `<cluster-name>`: o nome do cluster do Amazon EKS.
- `<api-server-endpoint>`: o endpoint do servidor de API do seu cluster.
- `<cluster-certificate-authority>`: o pacote de CA codificado em base64 do seu cluster.
- `<region>`: a região da AWS que hospeda seu cluster, por exemplo, "us-east-1".
- `<hostname>`: o nome do host da instância do Bottlerocket que também será configurado como o nome do nó. Esse pode ser qualquer valor exclusivo escolhido por você, mas deve seguir as convenções de nomenclatura de objetos do Kubernetes. Além disso, o nome do host utilizado não pode ter mais de 64 caracteres. OBSERVAÇÃO: quando o provedor SSM for usado, esse nome de host e nome de nó serão substituídos pelo ID da instância gerenciada (por exemplo, ID `mi-*`) depois que a instância for registrada no SSM.
- `<base64-encoded-admin-container-userdata>`: o conteúdo codificado em base64 da configuração do contêiner administrativo do Bottlerocket. A ativação do contêiner administrativo permite que você se conecte à sua instância do Bottlerocket com SSH para exploração e depuração do sistema. Embora essa não seja uma configuração obrigatória, recomendamos ativá-la para facilitar a solução de problemas. Consulte a documentação do contêiner administrativo do Bottlerocket para obter mais informações sobre a autenticação com o contêiner administrativo. O contêiner administrativo recebe a entrada do usuário e da chave SSH no formato JSON, por exemplo,

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>`: o conteúdo codificado em base64 da configuração do contêiner de bootstrap do Bottlerocket. Consulte a documentação do contêiner de bootstrap do Bottlerocket para obter mais informações sobre sua configuração. O contêiner

de bootstrap é responsável por registrar a instância como uma instância gerenciada do AWS SSM e adicioná-la como um nó Kubernetes em seu cluster do Amazon EKS. Os dados do usuário passados para o contêiner de bootstrap assumem a forma de uma invocação de comando que aceita como entrada o código de ativação híbrida do SSM e o ID que você criou anteriormente:

```
eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-code=<activation-
code> --region=<region>
```

IAM Roles Anywhere

Se você estiver usando o AWS IAM Roles Anywhere como seu provedor de credenciais, crie um arquivo `settings.toml` com o seguinte conteúdo:

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"
cloud-provider = ""
server-tls-bootstrap = true

[settings.network]
hostname = "<hostname>"

[settings.aws]
region = "<region>"
config = "<base64-encoded-aws-config-file>"

[settings.kubernetes.credential-providers.ecr-credential-provider]
enabled = true
cache-duration = "12h"
image-patterns = [
    "*.dkr.ecr.*.amazonaws.com",
    "*.dkr.ecr.*.amazonaws.com.cn",
    "*.dkr.ecr.*.amazonaws.eu",
    "*.dkr.ecr-fips.*.amazonaws.com",
    "*.dkr.ecr-fips.*.amazonaws.eu",
    "public.ecr.aws"
]
```

```
[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
"eks.amazonaws.com/hybrid-credential-provider" = "iam-ra"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"
```

Substitua os espaços reservados pelos seguintes valores:

- `<cluster-name>`: o nome do cluster do Amazon EKS.
- `<api-server-endpoint>`: o endpoint do servidor de API do seu cluster.
- `<cluster-certificate-authority>`: o pacote de CA codificado em base64 do seu cluster.
- `<region>`: a região da AWS que hospeda seu cluster, por exemplo, "us-east-1"
- `<hostname>`: o nome do host da instância do Bottlerocket que também será configurado como o nome do nó. Esse pode ser qualquer valor exclusivo escolhido por você, mas deve seguir as convenções de nomenclatura de objetos do Kubernetes. Além disso, o nome do host utilizado não pode ter mais de 64 caracteres. OBSERVAÇÃO: quando um provedor IAM-RA é usado, o nome do nó deve corresponder ao CN do certificado no host caso tenha configurado a política de confiança do perfil do IAM do Hybrid Nodes com a condição de recurso `"sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"`.
- `<base64-encoded-aws-config-file>`: o conteúdo codificado em base64 do seu arquivo de configuração da AWS. O conteúdo do arquivo deve ser o seguinte:

```
[default]
credential_process = aws_signing_helper credential-process --certificate /root/.aws/
node.crt --private-key /root/.aws/node.key --profile-arn <profile-arn> --role-arn
 <role-arn> --trust-anchor-arn <trust-anchor-arn> --role-session-name <role-session-
name>
```

- `<base64-encoded-admin-container-userdata>`: o conteúdo codificado em base64 da configuração do contêiner administrativo do Bottlerocket. A ativação do contêiner administrativo permite que você se conecte à sua instância do Bottlerocket com SSH para exploração e

depuração do sistema. Embora essa não seja uma configuração obrigatória, recomendamos ativá-la para facilitar a solução de problemas. Consulte a documentação do contêiner administrativo do Bottlerocket para obter mais informações sobre a autenticação com o contêiner administrativo. O contêiner administrativo recebe a entrada do usuário e da chave SSH no formato JSON, por exemplo,

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>`: o conteúdo codificado em base64 da configuração do contêiner de bootstrap do Bottlerocket. Consulte a documentação do contêiner de bootstrap do Bottlerocket para obter mais informações sobre sua configuração. O contêiner de bootstrap é responsável por criar os arquivos de certificado de host e chave privada do certificado do IAM Roles Anywhere na instância. Eles serão consumidos pelo `aws_signing_helper` para obter as credenciais temporárias necessárias para autenticação com seu cluster do Amazon EKS. Os dados do usuário passados para o contêiner de bootstrap assumem a forma de uma invocação de comando que aceita como entrada o conteúdo do arquivo de certificado e chave privada que você criou anteriormente:

```
eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key>
```

Etapa 2: provisionar a VM vSphere do Bottlerocket com dados do usuário

Depois de construir o arquivo TOML, passe-o como dados do usuário durante a criação da VM vSphere. Lembre-se de que os dados do usuário devem ser configurados antes que a VM seja ligada pela primeira vez. Dessa forma, você precisará fornecê-los ao criar a instância ou, se desejar criar a VM com antecedência, a VM deverá estar no estado poweredOff até que você configure os dados do usuário para ela. Por exemplo, se estiver usando a CLI do `govc`:

Criar uma VM pela primeira vez

```
govc vm.create \
  -on=true \
  -c=2 \
  -m=4096 \
  -net.adapter=<network-adapter> \
  -net=<network-name> \
  -e guestinfo.userdata.encoding="base64" \
  -e guestinfo.userdata="$(base64 -w0 settings.toml)" \
  -template=<template-name> \
  <vm-name>
```

Atualizar dados do usuário para uma VM existente

```
govc vm.create \
    -on=false \
    -c=2 \
    -m=4096 \
    -net.adapter=<network-adapter> \
    -net=<network-name> \
    -template=<template-name> \
    <vm-name>

govc vm.change
    -vm <vm-name> \
    -e guestinfo.userdata="$(base64 -w0 settings.toml)" \
    -e guestinfo.userdata.encoding="base64"

govc vm.power -on <vm-name>
```

Nas seções acima, a opção `-e guestinfo.userdata.encoding="base64"` especifica que os dados do usuário são codificados em base64. A opção `-e guestinfo.userdata` passa o conteúdo codificado em base64 do arquivo `settings.toml` como dados do usuário para a instância do Bottlerocket. Substitua os espaços reservados por seus valores específicos, como o modelo OVA do Bottlerocket e detalhes da rede.

Etapa 3: verificar a conexão do nó híbrido

Depois que a instância do Bottlerocket for iniciada, ela tentará se juntar ao seu cluster do Amazon EKS. Você pode verificar a conexão no console do Amazon EKS navegando até a guia Computação do seu cluster ou executando o seguinte comando:

```
kubectl get nodes
```

> **Important**
>
> Seus nós terão o status `Not Ready`, o que é esperado e se deve à falta de uma CNI em execução nos nós híbridos. Se os nós não tiverem se unido ao cluster, consulte the section called “Solução de problemas”.

Etapa 4: configurar uma CNI para nós híbridos

Para preparar os nós híbridos para executar aplicações, continue com as etapas em the section called “Configurar CNI”.

Atualizar os nós híbridos do cluster

A orientação para atualizar nós híbridos é semelhante aos nós autogerenciados do Amazon EKS que são executados no Amazon EC2. Recomendamos que você crie nós híbridos na versão de destino do Kubernetes, migre gradualmente as aplicações existentes para os nós híbridos na nova versão do Kubernetes e remova os nós híbridos na versão antiga do Kubernetes do cluster. Certifique-se de revisar as Práticas recomendadas do Amazon EKS para obter atualizações antes de iniciar uma atualização. O Amazon EKS Hybrid Nodes tem o mesmo suporte de versão do Kubernetes para clusters do Amazon EKS com nós na nuvem, incluindo suporte padrão e estendido.

O Amazon EKS Hybrid Nodes segue a mesma política de diferença de versão para nós que o Kubernetes upstream. O Amazon EKS Hybrid Nodes não pode estar em uma versão mais recente do que o ambiente de gerenciamento do Amazon EKS, e os nós híbridos podem estar até três versões secundárias do Kubernetes mais antigas do que a versão secundária do ambiente de gerenciamento do Amazon EKS.

Se você não tiver capacidade disponível para criar nós híbridos na versão de destino do Kubernetes para uma estratégia de atualização de migração por substituição, você também poderá usar a CLI (`nodeadm`) do Amazon EKS Hybrid Nodes para atualizar a versão do Kubernetes dos nós híbridos no local.

> **Important**
>
> Se você estiver atualizando os nós híbridos localmente com `nodeadm`, haverá um tempo de inatividade do nó durante o processo em que a versão mais antiga dos componentes do

Kubernetes é desligada e os novos componentes da versão do Kubernetes são instalados e iniciados.

Pré-requisitos

Antes de atualizar, certifique-se de ter atendido aos pré-requisitos a seguir.

- A versão de destino do Kubernetes para a atualização dos nós híbridos deve ser igual ou inferior à versão do ambiente de gerenciamento do Amazon EKS.
- Se você estiver seguindo uma estratégia de atualização de migração por substituição, os novos nós híbridos que você está instalando na versão de destino do Kubernetes deverão atender aos requisitos de the section called “Pré-requisitos”. Isso inclui ter endereços IP no CIDR da rede remota de nós que você passou durante a criação do cluster do Amazon EKS.
- Tanto para a migração por substituição quanto para as atualizações locais, os nós híbridos devem ter acesso aos domínios necessários para extrair as novas versões das dependências dos nós híbridos.
- Você deve ter o kubectl instalado na instância ou máquina local que você está usando para interagir com o endpoint da API do Kubernetes do Amazon EKS.
- A versão da CNI deve ser compatível com a versão do Kubernetes para a qual você está atualizando. Caso contrário, atualize a versão da CNI antes de atualizar os nós híbridos. Consulte the section called “Configurar CNI” para obter mais informações.

Atualizações de migração por substituição (azul-verde)

As atualizações de migração por substituição referem-se ao processo de criar nós híbridos em novos hosts com a versão de destino do Kubernetes, migrar tranquilamente as aplicações existentes para os novos nós híbridos na versão de destino do Kubernetes e remover os nós híbridos da versão antiga do Kubernetes do cluster. Essa estratégia também é chamada de migração azul-verde.

1. Conecte os novos hosts como nós híbridos seguindo as etapas de the section called “Conectar nós híbridos”. Ao executar o comando `nodeadm install`, use a versão de destino do Kubernetes.
2. Habilite a comunicação entre os novos nós híbridos na versão de destino do Kubernetes e os nós híbridos na versão antiga do Kubernetes. Essa configuração permite que os pods se comuniquem entre si enquanto você está migrando a workload para os nós híbridos na versão de destino do Kubernetes.

3. Confirme se os nós híbridos na versão de destino do Kubernetes se uniram com êxito ao cluster e se têm o status Pronto.
4. Use o comando a seguir para marcar cada um dos nós que deseja remover como não programável. Isso serve para que novos pods não sejam programados ou reprogramados nos nós que você está substituindo. Para obter mais informações, consulte isolar kubectl na documentação do Kubernetes. Substitua `NODE_NAME` pelo nome dos nós híbridos na versão antiga do Kubernetes.

```
kubectl cordon NODE_NAME
```

   Você pode identificar e isolar todos os nós de uma determinada versão do Kubernetes (neste caso, `1.28`) com o trecho de código a seguir.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Cordoning $node"
    kubectl cordon $node
done
```

5. Se a implantação atual estiver executando menos de duas réplicas de CoreDNS nos nós híbridos, aumente a escala horizontalmente da implantação para pelo menos duas réplicas. Recomendamos executar pelo menos duas réplicas do CoreDNS em nós híbridos para resiliência durante as operações normais.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

6. Drene cada um dos nós híbridos na versão antiga do Kubernetes que você deseja remover do cluster com o comando a seguir. Para obter mais informações sobre como drenar nós, consulte Safely Drain a Node na documentação do Kubernetes. Substitua `NODE_NAME` pelo nome dos nós híbridos na versão antiga do Kubernetes.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

   Você pode identificar e drenar todos os nós de determinada versão do Kubernetes (neste caso, `1.28`) com o trecho de código a seguir.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-emptydir-data
done
```

7. Você pode usar `nodeadm` para interromper e remover os artefatos dos nós híbridos do host. Você deve executar `nodeadm` com um usuário que tenha privilégios root/sudo. Por padrão, `nodeadm uninstall` não prosseguirá se houver pods restantes no nó. Para obter mais informações, consulte the section called “Nós híbridos nodeadm”.

```
nodeadm uninstall
```

8. Com os artefatos dos nós híbridos interrompidos e desinstalados, remova o recurso do nó do cluster.

```
kubectl delete node node-name
```

   Você pode identificar e excluir todos os nós de uma determinada versão do Kubernetes (neste caso, `1.28`) com o trecho de código a seguir.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Deleting $node"
    kubectl delete node $node
done
```

9. Dependendo da sua escolha de CNI, poderá haver artefatos restantes nos nós híbridos após a execução das etapas acima. Consulte the section called “Configurar CNI” para obter mais informações.

Atualizações no local

O processo de atualização no local refere-se ao uso de `nodeadm upgrade` para atualizar a versão do Kubernetes para nós híbridos sem usar novos hosts físicos ou virtuais e uma estratégia de migração por substituição. O processo de `nodeadm upgrade` desliga os componentes antigos existentes do Kubernetes em execução no nó híbrido, desinstala os componentes antigos do Kubernetes existentes, instala os novos componentes do Kubernetes de destino e inicia os novos componentes do Kubernetes de destino. É altamente recomendável atualizar um nó por vez para minimizar o impacto nas aplicações em execução nos nós híbridos. A duração desse processo depende da largura de banda e da latência da rede.

1. Use o comando a seguir para marcar o nó que você está atualizando como não programável. Isso serve para que novos pods não sejam programados ou reprogramados nos nós que você está atualizando. Para obter mais informações, consulte isolar kubectl na documentação do Kubernetes. Substitua `NODE_NAME` pelo nome do nó híbrido que você está atualizando

   ```
   kubectl cordon NODE_NAME
   ```

2. Drene o nó que você está atualizando com o comando a seguir. Para obter mais informações sobre como drenar nós, consulte Safely Drain a Node na documentação do Kubernetes. Substitua `NODE_NAME` pelo nome do nó híbrido que você está atualizando.

   ```
   kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
   ```

3. Execute `nodeadm upgrade` no nó híbrido que você está atualizando. Você deve executar `nodeadm` com um usuário que tenha privilégios root/sudo. O nome do nó é preservado durante a atualização para os provedores de credenciais do AWS SSM e AWS IAM Roles Anywhere. Você não pode alterar os provedores de credenciais durante o processo de atualização. Consulte the section called “Nós híbridos nodeadm” para obter os valores de configuração de `nodeConfig.yaml`. Substitua `K8S_VERSION` pela versão do Kubernetes de destino para a qual você está atualizando.

   ```
   nodeadm upgrade K8S_VERSION -c file://nodeConfig.yaml
   ```

4. Para permitir que os pods sejam agendados no nó após a atualização, digite o seguinte. Substitua `NODE_NAME` pelo nome do nó.

   ```
   kubectl uncordon NODE_NAME
   ```

5. Observe o status dos nós híbridos e aguarde até que os nós sejam desligados e reiniciados na nova versão do Kubernetes com o status Pronto.

```
kubectl get nodes -o wide -w
```

Atualizações de segurança de patches para nós híbridos

Este tópico descreve o procedimento para executar patches in-loco de atualizações de segurança para pacotes e dependências específicos em execução nos nós híbridos. Como prática recomendada, sugerimos que você atualize regularmente seus nós híbridos para receber CVEs e patches de segurança.

Para ver as etapas de atualização da versão do Kubernetes, consulte the section called “Atualizar os nós híbridos”.

Um exemplo de software que pode precisar de patches de segurança é `containerd`.

`Containerd`

`containerd` é o runtime padrão do contêiner Kubernetes e a dependência principal dos EKS Hybrid Nodes, usado para gerenciar o ciclo de vida do contêiner, incluindo a extração de imagens e o gerenciamento da execução do contêiner. Em um nó híbrido, você pode instalar o `containerd` por meio da CLI do nodeadm ou manualmente. Dependendo do sistema operacional do seu nó, o `nodeadm` instalará o `containerd` por meio do pacote distribuído pelo sistema operacional ou do pacote Docker.

Quando um CVE em `containerd` é publicado, você tem as opções a seguir de atualização para a versão corrigida do `containerd` em seus nós híbridos.

Etapa 1: verificar se o patch foi publicado nos gerenciadores de pacotes

Você pode verificar se o patch `containerd` CVE foi publicado em cada gerenciador de pacotes do sistema operacional respectivo consultando os boletins de segurança correspondentes:

- Amazon Linux 2023
- RHEL
- Ubuntu 20.04
- Ubuntu 22.04
- Ubuntu 24.04

Se você usa o repositório Docker como fonte do `containerd`, pode verificar os anúncios de segurança do Docker para identificar a disponibilidade da versão corrigida no repositório Docker.

Etapa 2: escolher o método para instalar o patch

Existem três métodos para corrigir e instalar atualizações de segurança in-loco nos nós. O método que você pode usar depende de se o patch está ou não disponível no sistema operacional do gerenciador de pacotes:

1. Instale os patches com `nodeadm upgrade` que estão publicados nos gerenciadores de pacotes. Consulte a Etapa 2 a.
2. Instale os patches diretamente com os gerenciadores de pacotes. Consulte a Etapa 2 b.
3. Instale patches personalizados que não estão publicados nos gerenciadores de pacotes. Observe que há considerações especiais sobre patches personalizados para `containerd`, Etapa 2 c.

Etapa 2 a: aplicar patches com **`nodeadm upgrade`**

Depois de confirmar que o patch `containerd` da CVE foi publicado nos repositórios do sistema operacional ou do Docker (Apt ou RPM), você pode usar o comando `nodeadm upgrade` a fim de atualizar para a versão mais recente do `containerd`. Como não se trata de uma atualização da versão do Kubernetes, você deve passar sua versão atual do Kubernetes para o comando de upgrade do `nodeadm`.

```
nodeadm upgrade K8S_VERSION --config-source file:///root/nodeConfig.yaml
```

Etapa 2 b: aplicar patches com gerenciadores de pacotes do sistema operacional

Alternativamente, você também pode atualizar por meio do respectivo gerenciador de pacotes e usá-lo para atualizar o pacote `containerd` como se segue.

Amazon Linux 2023

```
sudo yum update -y
sudo yum install -y containerd
```

RHEL

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/rhel/docker-
ce.repo
```

```
sudo yum update -y
sudo yum install -y containerd
```

Ubuntu

```
sudo mkdir -p /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/
docker.asc
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
 https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update -y
sudo apt install -y --only-upgrade containerd.io
```

Etapa 2 c: patch **`Containerd`** da CVE não publicado nos gerenciadores de pacotes

Se a versão corrigida de `containerd` estiver disponível apenas por outros meios e não no gerenciador de pacotes, por exemplo, nas versões do GitHub, você poderá instalar o `containerd` pelo site oficial do GitHub.

1. Se a máquina já fizer parte do cluster como um nó híbrido, será necessário executar o comando `nodeadm uninstall`.
2. Instale os arquivos binários oficiais do `containerd` . Você pode usar as etapas oficiais de instalação no GitHub.
3. Execute o comando `nodeadm install` com o argumento `--containerd-source` definido como `none`, o que ignorará a instalação do `containerd` por meio do `nodeadm`. Você pode usar o valor de `none` na origem do `containerd` para qualquer sistema operacional que o nó estiver executando.

   ```
   nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --containerd-source
    none
   ```

Remoção de nós híbridos

Este tópico descreve como excluir nós híbridos do cluster do Amazon EKS. Você deve excluir os nós híbridos com as ferramentas de sua escolha compatíveis com o Kubernetes, como o kubectl. As

cobranças por nós híbridos param quando o objeto do nó é removido do cluster do Amazon EKS. Para obter mais informações sobre preços de nós híbridos, consulte Preços do Amazon EKS.

> **Important**
>
> Remover os nós interrompe as workloads em execução no nó. Antes de excluir os nós híbridos, recomendamos primeiro drenar o nó para mover os pods para outro nó ativo. Para obter mais informações sobre como drenar nós, consulte Safely Drain a Node na documentação do Kubernetes.

Execute as etapas do kubectl abaixo na instância ou máquina local que você usa para interagir com o endpoint da API do Kubernetes do cluster do Amazon EKS. Se você estiver usando um arquivo `kubeconfig` específico, use o sinalizador `--kubeconfig`.

Etapa 1: listar os nós

```
kubectl get nodes
```

Etapa 2: drenar o nó

Consulte kubectl drain na documentação do Kubernetes para obter mais informações sobre o comando `kubectl drain`.

```
kubectl drain --ignore-daemonsets <node-name>
```

Etapa 3: interromper e desinstalar os artefatos dos nós híbridos

Você pode usar a CLI (`nodeadm`) do Amazon EKS Hybrid Nodes para interromper e remover os artefatos dos nós híbridos do host. Você deve executar `nodeadm` com um usuário que tenha privilégios root/sudo. Por padrão, `nodeadm uninstall` não prosseguirá se houver pods restantes no nó. Se você estiver usando o AWS Systems Manager (SSM) como o provedor de credenciais, o comando `nodeadm uninstall` cancelará o registro do host como uma instância gerenciada do AWS SSM. Para obter mais informações, consulte the section called “Nós híbridos nodeadm”.

```
nodeadm uninstall
```

Etapa 4: excluir o nó do cluster

Com os artefatos dos nós híbridos interrompidos e desinstalados, remova o recurso do nó do cluster.

```
kubectl delete node <node-name>
```

Etapa 5: verificar se há artefatos restantes

Dependendo da sua escolha de CNI, poderá haver artefatos restantes nos nós híbridos após a execução das etapas acima. Consulte the section called “Configurar CNI” para obter mais informações.

Configurar redes de aplicações, complementos e webhooks para nós híbridos

Depois de criar um cluster EKS para nós híbridos, configure recursos adicionais para redes de aplicações (CNI, BGP, Ingress, balanceamento de carga, políticas de rede), complementos, webhooks e configurações de proxy. Para obter a lista completa dos complementos do EKS e da comunidade que são compatíveis com nós híbridos, consulte the section called “Configurar complementos”.

Insights de clusters do EKS O EKS inclui verificações de insights para configurações incorretas na configuração dos seus nós híbridos que podem prejudicar a funcionalidade do cluster ou das workloads. Para obter mais informações sobre insights de clusters, consulte the section called “Insights de cluster”.

A seguir, são listados os recursos e complementos comuns que você pode usar com nós híbridos:

- Interface de rede de contêineres (CNI): a AWS oferece suporte para o Cilium como a CNI para nós híbridos. Para obter mais informações, consulte the section called “Configurar CNI”. Observe que a CNI da AWS VPC não pode ser usada com nós híbridos.
- CoreDNS e **`kube-proxy`**: o CoreDNS e o `kube-proxy` são instalados automaticamente quando nós híbridos se unem ao cluster do Amazon EKS. Esses complementos podem ser gerenciados como complementos do EKS após a criação do cluster.
- Ingress e balanceamento de carga: você pode usar o AWS Load Balancer Controller e o Application Load Balancer (ALB) ou Network Load Balancer (NLB) com o tipo de destino `ip` para workloads em execução em nós híbridos. A AWS oferece suporte para os atributos integrados de balanceamento de carga do Ingress, Gateway e Kubernetes Service do Cilium para workloads executadas em nós híbridos. Para obter mais informações, consulte the section called “Configurar o Ingress” e the section called “Configurar os serviços LoadBalancer”.
- Métricas: você pode usar os extratores sem atendente do Amazon Managed Service for Prometheus (AMP), o AWS Distro for Open Telemetry (ADOT) e o atendente de observabilidade

do Amazon CloudWatch com nós híbridos. Para usar extratores sem atendente do AMP para métricas de pods em nós híbridos, os pods devem estar acessíveis na VPC que você usa para o cluster do EKS.

- Logs: é possível habilitar o log do ambiente de gerenciamento do EKS para clusters habilitados para nós híbridos. Você pode usar o complemento do EKS para ADOT e o complemento do EKS do atendente de observabilidade do Amazon CloudWatch para o log de nós híbridos e pods.
- Identidades do pod e IRSA: você pode usar Identidades de Pods do EKS e perfis do IAM para contas de serviço (IRSA) com aplicações executadas em nós híbridos para permitir acesso granular aos pods executados em nós híbridos com outros serviços da AWS.
- Webhooks: se estiver executando webhooks, consulte the section called “Configurar webhooks” para considerações e etapas para, opcionalmente, executar webhooks em nós de nuvem, caso não consiga tornar suas redes de pod on-premises roteáveis.
- Proxy: caso esteja usando um servidor proxy no ambiente on-premises para tráfego que sai do data center ou do ambiente de borda, é possível configurar os nós e o cluster para usar o servidor proxy. Para obter mais informações, consulte the section called “Configurar proxy”.

Tópicos

- Configurar a CNI para nós híbridos
- Configuração de complementos para nós híbridos
- Configurar webhooks para nós híbridos
- Configurar proxy para nós híbridos
- Configurar o BGP do Cilium para nós híbridos
- Configurar o Ingress do Kubernetes para nós híbridos
- Configurar serviços do tipo LoadBalancer para nós híbridos
- Configurar políticas de rede do Kubernetes para nós híbridos

Configurar a CNI para nós híbridos

O Cilium é interface de rede de contêineres (CNI) compatível com a AWS para o Amazon EKS Hybrid Nodes. Você deve instalar uma CNI para que os nós híbridos fiquem prontos para atender às workloads. Os nós híbridos aparecem com o status `Not Ready` até que uma CNI esteja em execução. Você pode gerenciar essa CNI com ferramentas de sua escolha, como o Helm. As instruções nesta página abrangem o gerenciamento do ciclo de vida do Cilium (instalação, atualização, exclusão). Consulte the section called “Visão geral do Ingress do Cilium e do Gateway

do Cilium", the section called "Tipo de serviço LoadBalancer", e the section called "Configurar políticas de rede" para saber como configurar o Cilium para Ingress, balanceamento de carga e políticas de rede.

O Cilium não é compatível com a AWS quando executados em nós na Nuvem AWS. A CNI da Amazon VPC não é compatível com nós híbridos e está configurada com antiafinidade para o rótulo `eks.amazonaws.com/compute-type: hybrid`.

A documentação anterior do Calico nesta página foi movida para o Repositório de exemplos híbridos do EKS.

Compatibilidade da versão

As versões `v1.17.x` e `v1.18.x` do Cilium são compatíveis com o EKS Hybrid Nodes para todas as versões do Kubernetes com suporte por parte do Amazon EKS.

> **Note**
>
> Requisito de kernel do Cilium v1.18.3: devido ao requisito de kernel (kernel do Linux igual ou superior à versão 5.10), o Cilium v1.18.3 não é compatível com:

- Ubuntu 20.04
- Red Hat Enterprise Linux (RHEL) 8

Para obter os requisitos do sistema, consulte Requisitos do sistema Cilium.

Consulte o suporte a versões do Kubernetes para obter uma lista das versões do Kubernetes compatíveis com o Amazon EKS. O EKS Hybrid Nodes tem o mesmo suporte a versão do Kubernetes que os clusters do Amazon EKS com nós na nuvem.

Recursos com suporte

A AWS mantém compilações do Cilium para o EKS Hybrid Nodes baseadas no projeto Cilium de código aberto. Para receber suporte da AWS para o Cilium, você deve usar as compilações do Cilium fornecidas pela AWS e as versões compatíveis com o Cilium.

A AWS oferece suporte técnico para a configuração padrão dos recursos do Cilium para uso com o EKS Hybrid Nodes. Se você planeja usar o recurso fora do escopo do suporte da AWS,

recomendamos que obtenha suporte comercial para o Cilium ou tenha a experiência interna para solucionar problemas e contribuir com correções para o projeto do Cilium.

Atributo do Cilium	Com suporte da AWS
Conformidade da rede do Kubernetes	Sim
Conectividade do cluster principal	Sim
Família IP	IPv4
Gerenciamento de ciclo de vida	Helm
Modo de rede	Encapsulamento VXLAN
Gerenciamento de endereços IP (IPAM)	Escopo do cluster de IPAM do Cilium
Política de rede	Política de rede do Kubernetes
Protocolo de Gateway da Borda (BGP)	Ambiente de gerenciamento BGP do Cilium
Ingress do Kubernetes	Ingress do Cilium, Gateway do Cilium
Alocação de IP do Service LoadBalancer	Balanceador de Carga de IPAM do Cilium
Anúncio de Endereço IP do Service LoadBalancer	Ambiente de gerenciamento BGP do Cilium
substituição do kube-proxy	Sim

Considerações sobre o Cilium

- Repositório Helm: a AWS hospeda o chart do Helm do Cilium no Amazon Elastic Container Registry Public (Amazon ECR Public) no Amazon EKS Cilium/Cilium. As versões disponíveis são:
  - Cilium v1.17.9: `oci://public.ecr.aws/eks/cilium/cilium:1.17.9-0`

- Cilium v1.18.3: `oci://public.ecr.aws/eks/cilium/cilium:1.18.3-0`

  Os comandos neste tópico usam esse repositório. Observe que certos comandos `helm repo` não são válidos para repositórios do Helm no Amazon ECR Public, então você não pode se referir a esse repositório a partir de um nome de repositório local do Helm. Em vez disso, use o URI completo na maioria dos comandos.

- Por padrão, o Cilium é configurado para ser executado no modo sobreposto e túnel com VXLAN como método de encapsulamento. Esse modo tem o menor número de requisitos para a rede física subjacente.
- Por padrão, o Cilium mascara o endereço IP de origem de todo o tráfego de pods que sai do cluster para o endereço IP do nó. Se você desabilitar o mascaramento, os CIDRs do pod devem ser roteáveis na sua rede on-premises.
- Se você estiver executando webhooks em nós híbridos, os CIDRs do pod deverão ser roteáveis em sua rede on-premises. Se os CIDRs de pod não forem roteáveis na rede on-premises, é recomendável executar webhooks em nós de nuvem no mesmo cluster. Consulte the section called “Configurar webhooks” e the section called “Preparar a rede” para obter mais informações.
- A AWS recomenda o uso da funcionalidade BGP integrada do Cilium para tornar os CIDRs do pod roteáveis na sua rede on-premises. Para obter mais informações sobre como configurar o BGP do Cilium com nós híbridos, consulte the section called “Configurar o BGP”.
- O Gerenciamento de endereços IP (IPAM) padrão no Cilium é chamado de Escopo do cluster, em que o operador do Cilium aloca endereços IP para cada nó com base nos CIDRs de pod configurados pelo usuário.

Instalar o Cilium em nós híbridos

Procedimento

1. Crie um novo arquivo YAML denominado `cilium-values.yaml`. O exemplo a seguir configura o Cilium para ser executado somente em nós híbridos, definindo a afinidade para o rótulo `eks.amazonaws.com/compute-type: hybrid` para o atendente e operador do Cilium.
   - Configure `clusterPoolIpv4PodCIDRList` com os mesmos CIDRs de pods que configurou as redes remotas de pods do seu cluster do EKS. Por exemplo, `10.100.0.0/24`. O operador do Cilium aloca fatias de endereço IP por meio do espaço de IP configurado `clusterPoolIpv4PodCIDRList`. O CIDR do seu pod não deve se sobrepor ao CIDR do nó on-premises, ao CIDR da VPC ou ao CIDR do serviço do Kubernetes.

- Configure `clusterPoolIpv4MaskSize` com base nos pods necessários por nó. Por exemplo, 25 para um tamanho de segmento /25 de 128 pods por nó.
- Não altere `clusterPoolIpv4PodCIDRList` ou `clusterPoolIpv4MaskSize` depois de implantar o Cilium em seu cluster, consulte Expandir o pool de clusters para obter mais informações.
- Se você estiver executando o Cilium no modo de substituição do kube-proxy, defina `kubeProxyReplacement: "true"` em seus valores de Helm e certifique-se de não ter uma implantação existente do kube-proxy em execução nos mesmos nós que o Cilium.
- O exemplo abaixo desabilita o proxy Envoy Layer 7 (L7) que o Cilium usa para políticas de rede L7 e Ingress. Para obter mais informações, consulte the section called “Configurar políticas de rede” e the section called “Visão geral do Ingress do Cilium e do Gateway do Cilium”.
- O exemplo abaixo configura `loadBalancer.serviceTopology: true` para que a Distribuição de tráfego de serviços funcione corretamente se você a configurar para seus serviços. Para obter mais informações, consulte the section called “Configurar a Distribuição de Tráfego de Serviços”.
- Para obter uma lista completa dos valores do Helm para o Cilium, consulte a referência do Helm na documentação do Cilium.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: In
          values:
          - hybrid
ipam:
  mode: cluster-pool
  operator:
    clusterPoolIPv4MaskSize: 25
    clusterPoolIPv4PodCIDRList:
    - POD_CIDR
loadBalancer:
  serviceTopology: true
operator:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
```

```
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
              - hybrid
  unmanagedPodWatcher:
    restart: false
loadBalancer:
  serviceTopology: true
envoy:
  enabled: false
kubeProxyReplacement: "false"
```

2. Instale o Cilium no cluster.
   - Substitua `CILIUM_VERSION` por uma versão do Cilium (por exemplo, `1.17.9-0` ou `1.18.3-0`). É recomendável usar a versão de patch mais recente para a versão secundária do Cilium.
   - Certifique-se de que seus nós atendam aos requisitos de kernel da versão escolhida. O Cilium v1.18.3 necessita do kernel do Linux igual ou superior à versão 5.10.
   - Caso esteja usando um arquivo kubeconfig específico, use o sinalizador `--kubeconfig` com o comando de instalação do Helm.

```
helm install cilium oci://public.ecr.aws/eks/cilium/cilium \
    --version CILIUM_VERSION \
    --namespace kube-system \
    --values cilium-values.yaml
```

3. Confirme que a instalação do Cilium teve êxito com os comandos a seguir. Você deve ver a implantação `cilium-operator` e o `cilium-agent` em execução em cada um dos seus nós híbridos. Além disso, os nós híbridos agora devem ter o status `Ready`. Para obter informações sobre como configurar o BGP do Cilium para anunciar seus CIDRs de pod em sua rede on-premises, acesse the section called “Configurar o BGP”.

```
kubectl get pods -n kube-system
```

```
NAME                              READY   STATUS    RESTARTS   AGE
cilium-jjjn8                      1/1     Running   0          11m
cilium-operator-d4f4d7fcb-sc5xn   1/1     Running   0          11m
```

```
kubectl get nodes
```

```
NAME                   STATUS   ROLES    AGE   VERSION
mi-04a2cf999b7112233   Ready    <none>   19m   v1.31.0-eks-a737599
```

Atualizar o Cilium em nós híbridos

Antes de atualizar a implantação do Cilium, analise cuidadosamente a documentação de atualização do Cilium e as notas de atualização para entender as mudanças na versão de destino do Cilium.

1. Certifique-se de ter instalado a CLI do `helm` no ambiente de linha de comandos. Consulte a documentação do Helm para obter instruções da instalação.
2. Execute a verificação de pré-lançamento da atualização do Cilium. Substitua `CILIUM_VERSION` pela sua versão de destino do Cilium. Recomendamos executar a versão de patch mais recente para a versão secundária do Cilium. Você pode encontrar a versão de patch mais recente para uma determinada versão secundária do Cilium na seção Stable Releases da documentação do Cilium.

```
helm install cilium-preflight oci://public.ecr.aws/eks/cilium/cilium --version
 CILIUM_VERSION \
  --namespace=kube-system \
  --set preflight.enabled=true \
  --set agent=false \
  --set operator.enabled=false
```

3. Depois de aplicar o `cilium-preflight.yaml`, certifique-se de que o número de pods `READY` seja o mesmo número de pods do Cilium em execução.

```
kubectl get ds -n kube-system | sed -n '1p;/cilium/p'
```

```
NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
cilium                    2         2         2       2            2           <none>
          1h20m
cilium-pre-flight-check   2         2         2       2            2           <none>
          7m15s
```

4. Quando o número de pods READY for igual, certifique-se de que a implantação de pré-lançamento do Cilium também esteja marcada como READY 1/1. Caso mostre READY 0/1, consulte a seção Validação de CNP e resolva problemas com a implantação antes de continuar com a atualização.

```
kubectl get deployment -n kube-system cilium-pre-flight-check -w
```

```
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
cilium-pre-flight-check   1/1     1            0           12s
```

5. Excluir o pré-lançamento

```
helm uninstall cilium-preflight --namespace kube-system
```

6. Antes de executar o comando `helm upgrade`, preserve os valores da implantação em um `existing-cilium-values.yaml` ou use as opções da linha de comando `--set` para as configurações quando estiver executando o comando de upgrade. A operação de atualização substitui o ConfigMap do Cilium, portanto, é fundamental que os valores de configuração sejam passados durante a atualização.

```
helm get values cilium --namespace kube-system -o yaml > existing-cilium-values.yaml
```

7. Durante as operações normais do cluster, todos os componentes do Cilium devem executar a mesma versão. As etapas a seguir descrevem como atualizar todos os componentes de uma versão estável para uma versão estável posterior. Ao atualizar de uma versão secundária para outra versão secundária, é recomendável atualizar primeiro para a versão de patch mais recente da versão secundária existente do Cilium. Para minimizar a interrupção, a opção `upgradeCompatibility` deve ser definida para a versão inicial do Cilium que foi instalada no cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium --version CILIUM_VERSION \
  --namespace kube-system \
  --set upgradeCompatibility=1.X \
  -f existing-cilium-values.yaml
```

8. (Opcional) Se você precisar reverter a atualização devido a problemas, execute os comandos a seguir.

```
helm history cilium --namespace kube-system
```

```
helm rollback cilium [REVISION] --namespace kube-system
```

Excluir o Cilium dos nós híbridos

1. Execute o comando a seguir para desinstalar todos os componentes do Cilium do cluster. Observe que a desinstalação da CNI pode afetar a integridade dos nós e pods e não deve ser realizada em clusters de produção.

   ```
   helm uninstall cilium --namespace kube-system
   ```

   As interfaces e rotas configuradas pelo Cilium não são removidas por padrão quando a CNI é removida do cluster. Consulte o problema do GitHub para obter mais informações.
2. Para limpar os arquivos e recursos de configuração em disco, caso esteja usando os diretórios de configuração padrão, você poderá remover os arquivos conforme mostrado pelo script `cni-uninstall.sh` no repositório do Cilium no GitHub.
3. Para remover as definições de recursos personalizados (CRDs) do Cilium do cluster, você pode executar os comandos a seguir.

   ```
   kubectl get crds -oname | grep "cilium" | xargs kubectl delete
   ```

Configuração de complementos para nós híbridos

Esta página descreve as considerações para a execução de complementos da AWS e complementos da comunidade no Amazon EKS Hybrid Nodes. Para saber mais sobre os complementos do Amazon EKS e os processos de criação, atualização e remoção de complementos do cluster, consulte the section called “Complementos do Amazon EKS”. Salvo indicação em contrário nesta página, os processos para criar, atualizar e remover complementos do Amazon EKS são os mesmos para clusters do Amazon EKS com nós híbridos e para clusters do Amazon EKS com nós em execução na Nuvem AWS. Somente os complementos incluídos nesta página foram validados quanto à compatibilidade com o Amazon EKS Hybrid Nodes.

Os complementos da AWS a seguir são compatíveis com o Amazon EKS Hybrid Nodes.

complemento da AWS	Versões de complementos compatíveis
kube-proxy	v1.25.14-eksbuild.2 e superior

complemento da AWS	Versões de complementos compatíveis
CoreDNS	v1.9.3-eksbuild.7 e superior
AWS Distro para OpenTelemetry (ADOT)	v0.102.1-eksbuild.2 e superior
Agente de observabilidade do CloudWatch	v2.2.1-eksbuild.1 e superior
Atendente de Identidade de Pods do EKS	• v1.3.3-eksbuild.1 e superior, exceto para o Bottlerocket • v1.3.7-eksbuild.2 e superior para o Bottlerocket
Agente de monitoramento de nós	v1.2.0-eksbuild.1 e superior
Controlador de snapshots da CSI	v8.1.0-eksbuild.1 e superior
Conector de CA Privado da AWS para Kubernetes	v1.6.0-eksbuild.1 e superior

Os complementos da comunidade a seguir são compatíveis com o Amazon EKS Hybrid Nodes. Para saber mais sobre os complementos da comunidade, consulte the section called “Complementos da comunidade”.

Complemento da comunidade	Versões de complementos compatíveis
Servidor de métricas do Kubernetes	v0.7.2-eksbuild.1 e superior
cert-manager	v1.17.2-eksbuild.1 e superior
Exportador de nós do Prometheus	v1.9.1-eksbuild.2 e superior
kube-state-metrics	v2.15.0-eksbuild.4 e superior
DNS externo	v0.19.0-eksbuild.1 e superior

Além dos complementos do Amazon EKS nas tabelas acima, o Amazon Managed Service for Prometheus Collector e o AWS Load Balancer Controller para entrada de aplicações (HTTP) e balanceamento de carga (TCP e UDP) são compatíveis com nós híbridos.

Alguns complementos da AWS e complementos da comunidade não são compatíveis com o Amazon EKS Hybrid Nodes. As versões mais recentes desses complementos têm uma regra de antiafinidade para o rótulo padrão `eks.amazonaws.com/compute-type: hybrid` aplicado aos nós híbridos. Isso impede que eles sejam executados em nós híbridos quando implantados nos clusters. Caso tenha clusters com nós híbridos e nós em execução na Nuvem AWS, você poderá implantar esses complementos no cluster para nós executados na Nuvem AWS. A CNI da Amazon VPC não é compatível com nós híbridos, e o Cilium e o Calico são compatíveis como as interfaces de rede de contêineres (CNIs) do Amazon EKS Hybrid Nodes. Consulte the section called “Configurar CNI” para obter mais informações.

Complementos da AWS

As próximas seções descrevem as diferenças entre a execução de complementos compatíveis da AWS em nós híbridos, em comparação com os outros tipos de computação do Amazon EKS.

kube-proxy e CoreDNS

Por padrão, o EKS instala o kube-proxy e o CoreDNS como complementos autogerenciados quando você cria um cluster do EKS com a API da AWS e AWS SDKs, inclusive da AWS CLI. Você pode substituir esses complementos pelos complementos do Amazon EKS após a criação do cluster. Consulte a documentação do EKS para obter detalhes sobre the section called “kube-proxy” e the section called “CoreDNS”. Se você estiver executando um cluster com nós híbridos e não híbridos na AWS Nuvem, a AWS recomenda ter pelo menos uma réplica do CoreDNS nos nós híbridos e pelo menos uma réplica do CoreDNS nos nós na Nuvem AWS. Consulte the section called “Configurar réplicas do CoreDNS” para saber sobre as etapas de configuração.

Agente de observabilidade do CloudWatch

O operador do Agente de observabilidade do CloudWatch usa webhooks. Caso você esteja executando o operador em nós híbridos, o CIDR do pod on-premises deverá ser roteável na rede on-premises e você deverá configurar o cluster do EKS com a rede remota de pods. Para obter mais informações, consulte Configurar webhooks para nós híbridos.

As métricas no nível do nó não estão disponíveis para nós híbridos porque o CloudWatch Container Insights depende da disponibilidade do Serviço de metadados de instância (IMDS) das métricas

no nível do nó. Métricas no nível de cluster, workload, pod e contêiner estão disponíveis para nós híbridos.

Depois de instalar o complemento seguindo as etapas descritas em Instalar o atendente do CloudWatch com a observabilidade do Amazon CloudWatch, o manifesto do complemento deve ser atualizado antes que o atendente possa ser executado com êxito em nós híbridos. Edite o recurso `amazoncloudwatchagents` no cluster para adicionar a variável `RUN_WITH_IRSA` de ambiente, conforme mostrado abaixo.

```
kubectl edit amazoncloudwatchagents -n amazon-cloudwatch cloudwatch-agent
```

```
apiVersion: v1
items:
- apiVersion: cloudwatch.aws.amazon.com/v1alpha1
  kind: AmazonCloudWatchAgent
  metadata:
    ...
    name: cloudwatch-agent
    namespace: amazon-cloudwatch
    ...
  spec:
    ...
    env:
    - name: RUN_WITH_IRSA # <-- Add this
      value: "True" # <-- Add this
    - name: K8S_NODE_NAME
      valueFrom:
        fieldRef:
          fieldPath: spec.nodeName
          ...
```

Coletor gerenciado do Amazon Managed Prometheus para nós híbridos

Um coletor gerenciado do Amazon Managed Service for Prometheus (AMP) consiste em um extrator que descobre e coleta métricas dos recursos em um cluster do Amazon EKS. O AMP gerencia o extrator para você, eliminando a necessidade de gerenciar instâncias, atendentes ou extratores por conta própria.

Você pode usar coletores gerenciados pelo AMP sem nenhuma configuração adicional específica para nós híbridos. No entanto, os endpoints de métricas das aplicações nos nós híbridos devem ser

acessíveis na VPC, incluindo rotas da VPC para CIDRs de rede remota de pods e as portas abertas no firewall on-premises. Além disso, o cluster deve ter acesso ao endpoint privado do cluster.

Siga as etapas em Usar um coletor gerenciado pela AWS no Guia do usuário do Amazon Managed Service for Prometheus.

AWS Distro para OpenTelemetry (ADOT)

Você pode usar o complemento do AWS Distro for OpenTelemetry (ADOT) para coletar métricas, logs e dados de rastreamento das aplicações executadas em nós híbridos. O ADOT usa webhooks de admissão para alterar e validar as solicitações do Collector Custom Resource. Caso esteja executando o operador ADOT em nós híbridos, o CIDR de pods on-premises deverá ser roteável na rede on-premises e você deverá configurar o cluster do EKS com a rede remota de pods. Para obter mais informações, consulte Configurar webhooks para nós híbridos.

Siga as etapas em Getting Started with AWS Distro for OpenTelemetry using EKS Add-Ons na documentação do AWS Distro para OpenTelemetry.

AWS Load Balancer Controller

Você pode usar o AWS Load Balancer Controller e o Application Load Balancer (ALB) ou Network Load Balancer (NLB) com o tipo de destino `ip` para workloads em nós híbridos. Os destinos de IP usados com o ALB ou o NLB devem ser roteáveis pela AWS. O AWS Load Balancer Controller também usawebhooks. Caso você execute o operador do AWS Load Balancer Controller em nós híbridos, o CIDR de pods on-premises deverá ser roteável na rede on-premises e você deverá configurar o cluster do EKS com a rede remota de pods. Para obter mais informações, consulte Configurar webhooks para nós híbridos.

Para instalar o AWS Load Balancer Controller, siga as etapas em the section called “Application Load Balancer da AWS” ou the section called “AWS Network Load Balancer”.

Para entrar com o ALB, você deve especificar as anotações abaixo. Consulte the section called “Balanceador de carga da aplicação” para obter mais informações.

```
alb.ingress.kubernetes.io/target-type: ip
```

Para o balanceamento de carga com NLB, você deve especificar as anotações abaixo. Consulte the section called “Balanceamento de carga da rede” para obter mais informações.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
```

```
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

Atendente de Identidade de Pods do EKS

> **Note**
>
> Para implantar com êxito o complemento do atendente da Identidade de Pods do EKS em nós híbridos que executam o Bottlerocket, certifique-se de que a versão do Bottlerocket seja pelo menos a v1.39.0. O atendente da Identidade de Pods não é compatível com versões anteriores do Bottlerocket em ambientes de nós híbridos.

O DaemonSet do atendente original da Identidade de Pods do Amazon EKS depende da disponibilidade do IMDS do EC2 no nó para obter as credenciais da AWS necessárias. Como o IMDS não está disponível em nós híbridos, a partir da versão 1.3.3-eksbuild.1, o complemento do atendente da Identidade de Pods implanta opcionalmente um DaemonSet que monta as credenciais necessárias. Os nós híbridos que executam o Bottlerocket exigem um método diferente para montar as credenciais e, a partir da versão 1.3.7-eksbuild.2, o complemento do atendente da Identidade de Pods implanta opcionalmente um DaemonSet voltado especificamente para nós híbridos do Bottlerocket. As seções a seguir descrevem o processo para habilitar os DaemonSets opcionais.

Ubuntu/RHEL/AL2023

1. Para usar o atendente da Identidade de Pods em nós híbridos do Ubuntu/RHEL/AL2023, defina `enableCredentialsFile: true` na seção híbrida da configuração de `nodeadm`, conforme mostrado abaixo:

   ```
   apiVersion: node.eks.aws/v1alpha1
   kind: NodeConfig
   spec:
       hybrid:
           enableCredentialsFile: true # <-- Add this
   ```

   Isso vai configurar `nodeadm` para criar um arquivo de credenciais a ser configurado no nó abaixo de `/eks-hybrid/.aws/credentials`, que será usado pelos pods `eks-pod-identity-agent`. Esse arquivo de credenciais conterá credenciais temporárias da AWS que serão atualizadas periodicamente.

2. Depois de atualizar a configuração de `nodeadm` em cada nó, execute o comando `nodeadm init` a seguir com o `nodeConfig.yaml` para unir os nós híbridos ao cluster do Amazon EKS. Se os nós já se uniram ao cluster anteriormente, execute o comando `nodeadm init` novamente.

```
nodeadm init -c file://nodeConfig.yaml
```

3. Instale o `eks-pod-identity-agent` com o suporte para nós híbridos habilitado usando a AWS CLI ou o Console de gerenciamento da AWS.

   a. AWS CLI: na máquina que você está usando para administrar o cluster, execute o comando a seguir para instalar `eks-pod-identity-agent` com o suporte para nós híbridos habilitado. Substitua o `my-cluster` pelo nome do cluster.

```
aws eks create-addon \
    --cluster-name my-cluster \
    --addon-name eks-pod-identity-agent \
    --configuration-values '{"daemonsets":{"hybrid":{"create": true}}}'
```

   b. Console de gerenciamento da AWS: se você estiver instalando o complemento do atendente da Identidade de Pods no Console da AWS, adicione o seguinte à configuração opcional para implantar o DaemonSet que tem como destino os nós híbridos.

```
{"daemonsets":{"hybrid":{"create": true}}}
```

Bottlerocket

1. Para usar o atendente da Identidade de Pods nos nós híbridos do Bottlerocket, adicione o sinalizador `--enable-credentials-file=true` ao comando usado para os dados do usuário do contêiner de bootstrap do Bottlerocket, conforme descrito em the section called “Conectar nós híbridos com o Bottlerocket”.

   a. Se você estiver usando o provedor de credenciais do SSM, seu comando deverá ser semelhante a este:

```
eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-
code=<activation-code> --region=<region> --enable-credentials-file=true
```

   b. Se você estiver usando o provedor de credenciais do IAM Roles Anywhere, seu comando deverá ser semelhante a este:

```
eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key> --enable-
credentials-file=true
```

Isso vai configurar o script de bootstrap para criar um arquivo de credenciais no nó em `/var/eks-hybrid/.aws/credentials`, que será usado pelos pods do `eks-pod-identity-agent`. Esse arquivo de credenciais conterá credenciais temporárias da AWS que serão atualizadas periodicamente.

2. Instale o `eks-pod-identity-agent` com o suporte para nós híbridos do Bottlerocket habilitado usando a AWS CLI ou o Console de gerenciamento da AWS.
   a. AWS CLI: na máquina que você está usando para administrar o cluster, execute o comando a seguir para instalar o `eks-pod-identity-agent` com o suporte para nós híbridos do Bottlerocket habilitado. Substitua o `my-cluster` pelo nome do cluster.

```
aws eks create-addon \
    --cluster-name my-cluster \
    --addon-name eks-pod-identity-agent \
    --configuration-values '{"daemonsets":{"hybrid-bottlerocket":{"create":
 true}}}'
```

   b. Console de gerenciamento da AWS: se você estiver instalando o complemento do atendente da Identidade de Pods no Console da AWS, adicione o seguinte à configuração opcional para implantar o DaemonSet que tem como destino os nós híbridos do Bottlerocket.

```
{"daemonsets":{"hybrid-bottlerocket":{"create": true}}}
```

Controlador de snapshots da CSI

A partir da versão `v8.1.0-eksbuild.2`, o complemento do controlador de snapshots da CSI aplica uma regra flexível de antiafinidade para nós híbridos, preferindo que o controlador `deployment` seja executado no EC2 na mesma região da AWS do ambiente de gerenciamento do Amazon EKS. Colocar a `deployment` na mesma região da AWS do ambiente de gerenciamento do Amazon EKS melhora a latência.

Complementos da comunidade

As próximas seções descrevem as diferenças entre a execução de complementos compatíveis da comunidade em nós híbridos, em comparação com os outros tipos de computação do Amazon EKS.

Servidor de métricas do Kubernetes

O ambiente de gerenciamento precisa alcançar o IP do pod do Metrics Server (ou IP do nó se o HostNetwork estiver habilitado). Portanto, a menos que execute o Metrics Server no modo hostNetwork, você deverá configurar uma rede de pod remota ao criar o cluster do Amazon EKS e deverá tornar os endereços IP do pod roteáveis. A implementação do Protocolo de Gateway da Borda (BGP) com o CNI é uma forma comum de tornar os endereços IP do pod roteáveis.

cert-manager

O `cert-manager` usa webhooks. Caso execute o `cert-manager` em nós híbridos, o CIDR de pods on-premises deverá ser roteável na rede on-premises e você deverá configurar o cluster do EKS com a rede remota de pods. Para obter mais informações, consulte Configurar webhooks para nós híbridos.

Configurar webhooks para nós híbridos

Esta página detalha os aspectos da execução de webhooks com nós híbridos. Os webhooks são usados em aplicações Kubernetes e projetos de código aberto, como o AWS Load Balancer Controller e o CloudWatch Observability Agent, para executar recursos de mutação e validação no runtime.

Redes de pods roteáveis

Se for possível tornar o CIDR de pods on-premises roteável em sua rede on-premises, você poderá executar webhooks em nós híbridos. Você pode usar várias técnicas para tornar seu pod CIDR on-premises roteável na rede on-premises, incluindo o Protocolo de Gateway da Borda (BGP), rotas estáticas ou outras soluções de roteamento personalizadas. O BGP é a solução recomendada, pois é mais escalável e fácil de gerenciar do que soluções alternativas, que demandam uma configuração de rota personalizada ou manual. A AWS dá suporte aos recursos do BGP do Cilium e do Calico para anunciar CIDRs do pod. Consulte the section called “Configurar CNI” e the section called “CIDRs de pods remotos roteáveis” para obter mais informações.

Redes de pods não roteáveis

Se você não puder tornar seu CIDR de pods on-premises roteável na rede on-premises e precisar executar webhooks, recomendamos executar todos os webhooks em nós da nuvem no mesmo cluster do EKS que os nós híbridos.

Considerações sobre clusters de modo misto

Os clusters de modo misto são definidos como clusters do EKS que têm nós híbridos e nós em execução na Nuvem AWS. Ao executar um cluster de modo misto, considere as seguintes recomendações:

- Execute a VPC CNI em nós na Nuvem AWS e o Cilium ou o Calico em nós híbridos. O Cilium e o Calico não são compatíveis com a AWS quando executados em nós na Nuvem AWS.
- Configurar os webhooks para execução em nós na Nuvem AWS. Confira the section called “Configurar webhooks para complementos” para saber como configurar os webhooks da AWS e os complementos da comunidade.
- Se as aplicações precisarem de pods executados em nós na Nuvem AWS para se comunicar diretamente com os pods executados em nós híbridos (“comunicação leste-oeste”), e você estiver usando a CNI da VPC nos nós na Nuvem AWS e o Cilium ou o Calico em nós híbridos, então o CIDR de pods on-premises deverá ser roteável na rede on-premises.
- Execute pelo menos uma réplica do CoreDNS em nós na Nuvem AWS e uma réplica do CoreDNS em nós híbridos.
- Configure a Distribuição de Tráfego de Serviços para manter o tráfego de serviços local na zona de origem. Para obter mais informações sobre Distribuição de Tráfego de Serviços, consulte the section called “Configurar a Distribuição de Tráfego de Serviços”.
- Se você estiver usando o AWS Application Load Balancers (ALB) ou o Network Load Balancers (NLB) para tráfego de workloads executado em nós híbridos, os destinos de IP usados com o ALB ou o NLB deverão ser roteáveis da AWS.
- O complemento Metrics Server requer conectividade do ambiente de gerenciamento do EKS com o endereço IP do pod do Metrics Server. Se você estiver executando o complemento Metrics Server em nós híbridos, o CIDR do pod on-premises deverá ser roteável em sua rede on-premises.
- Para coletar métricas de nós híbridos usando coletores gerenciados do Amazon Managed Service for Prometheus (AMP), o CIDR do pod on-premises deverá ser roteável na rede local. Outra opção é usar o coletor gerenciado do AMP para recursos e métricas do ambiente de gerenciamento do EKS executados na Nuvem AWS, e o complemento do AWS Distro para OpenTelemetry (ADOT) para coletar métricas de nós híbridos.

Configurar clusters de modo misto

Para visualizar os webhooks mutantes e validados em execução no cluster, você pode ver o tipo de recurso Extensões no painel Recursos do console do EKS do cluster ou usar os comandos a seguir.

O EKS também informa as métricas do webhook no painel de observabilidade do cluster. Consulte the section called “Painel de observabilidade” para obter mais informações.

```
kubectl get mutatingwebhookconfigurations
```

```
kubectl get validatingwebhookconfigurations
```

Configurar a Distribuição de Tráfego de Serviços

Ao executar clusters de modo misto, recomendamos que você use a Distribuição de Tráfego de Serviços para manter o tráfego de serviços local na zona de onde ele é originado. A Distribuição de Tráfego de Serviços (disponível para as versões 1.31 e posteriores do Kubernetes no EKS) é a solução recomendada em relação ao Roteamento com Reconhecimento de Topologia porque é a mais previsível. Na Distribuição de Tráfego de Serviços, os endpoints íntegros na zona receberão todo o tráfego dela. No Roteamento com Reconhecimento de Topologia, cada serviço deve atender a várias condições na zona para aplicar o roteamento personalizado. Do contrário, ele vai rotear o tráfego uniformemente para todos os endpoints.

Se você estiver usando o Cilium como seu CNI, você deve executar o CNI com o `enable-service-topology` definido como `true` para habilitar a Distribuição de tráfego de serviço. Você pode passar essa configuração com o sinalizador de instalação do Helm `--set loadBalancer.serviceTopology=true` ou atualizar uma instalação existente com o comando `cilium config set enable-service-topology true` da CLI do Cilium. O agente do Cilium em execução em cada nó deve ser reiniciado após a atualização da configuração de uma instalação existente.

Um exemplo de como configurar a Distribuição de Tráfego de Serviços para o serviço CoreDNS é mostrado na seção a seguir, e recomendamos que você habilite o mesmo para todos os serviços no cluster para evitar tráfego não intencional entre ambientes.

Configurar réplicas do CoreDNS

Se você estiver executando um cluster em modo misto com nós híbridos e nós na Nuvem AWS, é recomendável ter pelo menos uma réplica do CoreDNS nos nós híbridos e pelo menos uma réplica do CoreDNS nos nós na Nuvem AWS. Para evitar problemas de latência e rede em uma configuração de cluster em modo misto, é possível configurar o serviço CoreDNS para preferir a réplica do CoreDNS mais próxima com Distribuição de tráfego de serviço.

A Distribuição de tráfego de serviço (disponível para as versões 1.31 e posteriores do Kubernetes no EKS) é a solução recomendada em relação ao Roteamento com reconhecimento de topologia porque é a mais previsível. Na Distribuição de tráfego de serviço, os endpoints saudáveis na zona receberão todo o tráfego dessa zona. No roteamento com reconhecimento de topologia, cada serviço deve atender a várias condições nessa zona para aplicar o roteamento personalizado, caso contrário, ele roteia o tráfego uniformemente para todos os endpoints. As etapas a seguir configuram a Distribuição de tráfego de serviço.

Se você estiver usando o Cilium como seu CNI, você deve executar o CNI com o `enable-service-topology` definido como `true` para habilitar a Distribuição de tráfego de serviço. Você pode passar essa configuração com o sinalizador de instalação do Helm `--set loadBalancer.serviceTopology=true` ou atualizar uma instalação existente com o comando `cilium config set enable-service-topology true` da CLI do Cilium. O agente do Cilium em execução em cada nó deve ser reiniciado após a atualização da configuração de uma instalação existente.

1. Adicione um rótulo de zona de topologia para cada um dos seus nós híbridos, por exemplo, `topology.kubernetes.io/zone: onprem`. Ou você pode definir o rótulo na fase `nodeadm init` especificando o rótulo na configuração de `nodeadm`, consulte the section called “Configuração de nó para personalizar o kubelet (opcional)”. Observe que os nós executados na Nuvem AWS recebem automaticamente um rótulo de zona de topologia aplicado a eles, correspondente à zona de disponibilidade (AZ) do nó.

   ```
   kubectl label node hybrid-node-name topology.kubernetes.io/zone=zone
   ```

2. Adicione `podAntiAffinity` à implantação do CoreDNS com a chave da zona de topologia. Ou você pode configurar a implantação do CoreDNS durante a instalação com os complementos do EKS.

   ```
   kubectl edit deployment coredns -n kube-system
   ```

   ```
   spec:
     template:
       spec:
         affinity:
           ...
            podAntiAffinity:
              preferredDuringSchedulingIgnoredDuringExecution:
   ```

```
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: k8s-app
                  operator: In
                  values:
                  - kube-dns
              topologyKey: kubernetes.io/hostname
            weight: 100
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: k8s-app
                  operator: In
                  values:
                  - kube-dns
              topologyKey: topology.kubernetes.io/zone
            weight: 50
      ...
```

3. Adicione a configuração `trafficDistribution: PreferClose` à configuração do serviço `kube-dns` para habilitar a Distribuição de Tráfego de Serviços.

```
kubectl patch svc kube-dns -n kube-system --type=merge -p '{
  "spec": {
    "trafficDistribution": "PreferClose"
  }
}'
```

4. É possível confirmar se a Distribuição de tráfego de serviço está habilitada visualizando as fatias de endpoint do serviço `kube-dns`. Suas fatias de endpoint devem mostrar o `hints` para seus rótulos de zona de topologia, o que confirma que a Distribuição de tráfego de serviço está habilitada. Se você não consegue ver o `hints` para cada endereço de endpoint, então a Distribuição de tráfego de serviço não está habilitada.

```
kubectl get endpointslice -A | grep "kube-dns"
```

```
kubectl get endpointslice [.replaceable]`kube-dns-<id>`  -n kube-system -o yaml
```

```
addressType: IPv4
apiVersion: discovery.k8s.io/v1
```

```
endpoints:
- addresses:
  - <your-hybrid-node-pod-ip>
  hints:
    forZones:
    - name: onprem
  nodeName: <your-hybrid-node-name>
  zone: onprem
- addresses:
  - <your-cloud-node-pod-ip>
  hints:
    forZones:
    - name: us-west-2a
  nodeName: <your-cloud-node-name>
  zone: us-west-2a
```

Configurar webhooks para complementos

Os complementos a seguir usam webhooks e são compatíveis para uso com nós híbridos.

- AWS Load Balancer Controller
- Agente de observabilidade do CloudWatch
- AWS Distro para OpenTelemetry (ADOT)
- `cert-manager`

Consulte as seções a seguir para configurar os webhooks usados por esses complementos para executar em nós na Nuvem AWS.

AWS Load Balancer Controller

Para usar o AWS Load Balancer Controller em uma configuração de cluster em modo misto, é necessário executar o controlador em nós na Nuvem AWS. Para fazer isso, adicione as opções a seguir à sua configuração de valores do Helm ou especifique os valores usando a configuração de add-on do EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
```

```
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Agente de observabilidade do CloudWatch

O complemento do Agente de observabilidade do CloudWatch tem um operador do Kubernetes que usa webhooks. Para executar o operador em nós na Nuvem AWS em uma configuração de cluster de modo misto, edite a configuração do operador do Agente de observabilidade do CloudWatch. Não é possível configurar a afinidade do operador durante a instalação com os complementos do Helm e EKS (consulte o problema nº 2431 do mapa de contêineres).

```
kubectl edit -n amazon-cloudwatch deployment amazon-cloudwatch-observability-
controller-manager
```

```
spec:
  ...
  template:
    ...
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: eks.amazonaws.com/compute-type
                operator: NotIn
                values:
                - hybrid
```

AWS Distro para OpenTelemetry (ADOT)

O complemento AWS Distro for OpenTelemetry (ADOT) tem um operador do Kubernetes que usa webhooks. Para executar o operador em nós na Nuvem AWS em uma configuração de cluster de modo misto, adicione as informações abaixo à sua configuração de valores do Helm ou especifique os valores usando a configuração do complemento do EKS.

```
affinity:
  nodeAffinity:
```

```
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Se o CIDR do pod não for roteável na rede on-premises, o coletor do ADOT deverá ser executado em nós híbridos para extrair as métricas de seus nós híbridos e das workloads executadas neles. Para fazer isso, edite a Definição de recurso personalizado (CRD).

```
kubectl -n opentelemetry-operator-system edit opentelemetrycollectors.opentelemetry.io
 adot-col-prom-metrics
```

```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
            - hybrid
```

Você pode configurar o coletor ADOT para extrair métricas apenas dos nós híbridos e dos recursos executados nos nós híbridos adicionando o seguinte `relabel_configs` a cada `scrape_configs` na configuração CRD do coletor ADOT.

```
relabel_configs:
  - action: keep
    regex: hybrid
    source_labels:
    - __meta_kubernetes_node_label_eks_amazonaws_com_compute_type
```

O complemento ADOT tem como pré-requisito a instalação do `cert-manager` para os certificados TLS usados pelo webhook do operador ADOT. O `cert-manager` também executa webhooks e você pode configurá-lo para executar em nós na Nuvem AWS com a configuração de valores do Helm a seguir.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
startupapicheck:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
```

`cert-manager`

O complemento `cert-manager` também executa webhooks, e você pode configurá-lo para execução em nós na Nuvem AWS com a configuração de valores do Helm a seguir.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
startupapicheck:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
```

```
            - hybrid
```

Configurar proxy para nós híbridos

Caso esteja usando um servidor proxy no ambiente on-premises para tráfego que sai do data center ou do ambiente de borda, você precisa configurar em separado os nós e o cluster para usar o servidor proxy.

Cluster

No cluster, você precisa configurar `kube-proxy` para usar o servidor proxy. Você deve configurar `kube-proxy` após criar o cluster do Amazon EKS.

Nós

Nos nós, você deve configurar o sistema operacional, `containerd`, `kubelet` e o agente Amazon SSM para usar o servidor proxy. Você pode fazer essas alterações durante o processo de criação para as imagens do sistema operacional ou antes de executar `nodeadm init` em cada nó híbrido.

Configuração no nível do nó

Você deve aplicar as configurações a seguir nas imagens do sistema operacional ou antes de executar `nodeadm init` em cada nó híbrido.

Configuração do proxy **`containerd`**

`containerd` é o runtime padrão de gerenciamento de contêineres para o Kubernetes. Caso esteja usando um proxy para acesso à internet, você deverá configurar o `containerd` para que ele possa extrair as imagens de contêiner exigidas pelo Kubernetes e pelo Amazon EKS.

Crie um arquivo em cada nó híbrido denominado `http-proxy.conf` no diretório `/etc/systemd/system/containerd.service.d` com os conteúdos a seguir. Substitua `proxy-domain` e `port` pelos valores do seu ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configuração do **`containerd`** dos dados do usuário

O diretório `containerd.service.d` precisará ser criado para esse arquivo. Você precisará recarregar o systemd para obter o arquivo de configuração sem reinicializar. No AL2023, o serviço provavelmente já estará em execução quando seu script for executado, então você também precisará reiniciá-lo.

```
mkdir -p /etc/systemd/system/containerd.service.d
echo '[Service]' > /etc/systemd/system/containerd.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/containerd.service.d/
http-proxy.conf
systemctl daemon-reload
systemctl restart containerd
```

Configuração do proxy **`kubelet`**

`kubelet` é o agente de nó do Kubernetes que é executado em cada nó do Kubernetes e é responsável por gerenciar os nós e pods em execução nele. Caso esteja usando um proxy no ambiente on-premises, você deverá configurar o `kubelet` para que ele possa se comunicar com os endpoints públicos ou privados do cluster do Amazon EKS.

Crie um arquivo em cada nó híbrido denominado `http-proxy.conf` no diretório `/etc/systemd/system/kubelet.service.d/` com o conteúdo a seguir. Substitua `proxy-domain` e `port` pelos valores do seu ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configuração do **`kubelet`** dos dados do usuário

O diretório `kubelet.service.d` deve ser criado para esse arquivo. Você precisará recarregar o systemd para obter o arquivo de configuração sem reinicializar. No AL2023, o serviço provavelmente já estará em execução quando seu script for executado, então você também precisará reiniciá-lo.

```
mkdir -p /etc/systemd/system/kubelet.service.d
```

```
echo '[Service]' > /etc/systemd/system/kubelet.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/kubelet.service.d/http-
proxy.conf
systemctl daemon-reload
systemctl restart kubelet
```

Configuração do proxy **`ssm`**

O `ssm` é um dos provedores de credenciais que podem ser usados para inicializar um nó híbrido. O `ssm` é responsável pela autenticação na AWS e pela geração de credenciais temporárias que são usadas pelo `kubelet`. Caso esteja usando um proxy no ambiente on-premises e usando o `ssm` como provedor de credenciais no nó, você deverá configurar o `ssm` para que ele possa se comunicar com os endpoints do serviço do Amazon SSM.

Crie um arquivo em cada nó híbrido chamado `http-proxy.conf` no caminho abaixo, dependendo do sistema operacional

- Ubuntu - `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d/http-proxy.conf`
- Amazon Linux 2023 e Red Hat Enterprise Linux: `/etc/systemd/system/amazon-ssm-agent.service.d/http-proxy.conf`

Preencha o arquivo com o seguinte conteúdo. Substitua `proxy-domain` e `port` pelos valores do seu ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configuração do **`ssm`** dos dados do usuário

O serviço `ssm` systemd deve ser criado para esse arquivo. O caminho do diretório dependerá do sistema operacional usado no nó.

- Ubuntu - `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d`
- Amazon Linux 2023 e Red Hat Enterprise Linux: `/etc/systemd/system/amazon-ssm-agent.service.d`

Substitua o nome do serviço systemd no comando restart abaixo, dependendo do sistema operacional usado no nó

- Ubuntu - `snap.amazon-ssm-agent.amazon-ssm-agent`
- Amazon Linux 2023 e Red Hat Enterprise Linux: `amazon-ssm-agent`

```
mkdir -p systemd-service-file-directory
echo '[Service]' > [.replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://[.replaceable]#proxy-domain:port"' >> systemd-
service-file-directory/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://[.replaceable]#proxy-domain:port"' >>
 [.replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> [.replaceable]#systemd-service-file-
directory/http-proxy.conf
systemctl daemon-reload
systemctl restart [.replaceable]#systemd-service-name
```

Configuração de proxy do sistema operacional

Caso esteja usando um proxy para acesso à internet, você deve configurar o sistema operacional para poder extrair as dependências dos nós híbridos do gerenciador de pacotes do sistema operacional.

Ubuntu

1. Configure `snap` para usar o proxy com os seguintes comandos:

```
sudo snap set system proxy.https=http://proxy-domain:port
sudo snap set system proxy.http=http://proxy-domain:port
```

2. Para habilitar o proxy para `apt`, crie um arquivo denominado `apt.conf` no diretório `/etc/apt/`. Substitua proxy-domain e a porta pelos valores do seu ambiente.

```
Acquire::http::Proxy "http://proxy-domain:port";
Acquire::https::Proxy "http://proxy-domain:port";
```

Amazon Linux 2023

1. Configure `dnf` para usar o proxy. Crie um arquivo `/etc/dnf/dnf.conf` com os valores de proxy-domain e port para seu ambiente.

```
proxy=http://proxy-domain:port
```

Red Hat Enterprise Linux

1. Configure `yum` para usar o proxy. Crie um arquivo `/etc/yum.conf` com os valores de proxy-domain e port para seu ambiente.

```
proxy=http://proxy-domain:port
```

Configuração de proxies do IAM Roles Anywhere

O serviço de provedor de credenciais do IAM Roles Anywhere é responsável por atualizar as credenciais ao usar o IAM Roles Anywhere com o sinalizador `enableCredentialsFile` (consulte the section called “Atendente de Identidade de Pods do EKS”). Caso esteja usando um proxy no ambiente on-premises, você deverá configurar o serviço para que ele possa se comunicar com os endpoints do IAM Roles Anywhere.

No diretório `/etc/systemd/system/aws_signing_helper_update.service.d/`, crie um arquivo denominado `http-proxy.conf` com o conteúdo a seguir. Substitua `proxy-domain` e `port` pelos valores do seu ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configuração de todo o cluster

As configurações nesta seção devem ser aplicadas depois de você criar o cluster do Amazon EKS e antes de executar `nodeadm init` em cada nó híbrido.

Configuração do proxy kube-proxy

O Amazon EKS instala automaticamente o `kube-proxy` em cada nó híbrido como um DaemonSet quando os nós híbridos se unem ao cluster. O `kube-proxy` permite o roteamento entre serviços que são apoiados por pods nos clusters do Amazon EKS. Para configurar cada host, o `kube-proxy` requer resolução de DNS para o endpoint de cluster do Amazon EKS.

1. Editar o DaemonSet do `kube-proxy` com o comando a seguir

   ```
   kubectl -n kube-system edit ds kube-proxy
   ```

   Isso abrirá a definição do DaemonSet do `kube-proxy` no editor configurado.

2. Adicione variáveis de ambiente para `HTTP_PROXY` e `HTTPS_PROXY`. Observe que a variável de ambiente `NODE_NAME` já deve existir na configuração. Substitua `proxy-domain` e `port` pelos valores do seu ambiente.

   ```
   containers:
     - command:
       - kube-proxy
       - --v=2
       - --config=/var/lib/kube-proxy-config/config - --hostname-override=$(NODE_NAME)
       env:
       - name: HTTP_PROXY
         value: http://proxy-domain:port
       - name: HTTPS_PROXY
         value: http://proxy-domain:port
       - name: NODE_NAME
         valueFrom:
           fieldRef:
             apiVersion: v1
             fieldPath: spec.nodeName
   ```

Configurar o BGP do Cilium para nós híbridos

Este tópico descreve como configurar o Protocolo de Gateway da Borda (BGP) do Cilium para o Amazon EKS Hybrid Nodes. A funcionalidade BGP do Cilium é chamada de Ambiente de gerenciamento BGP do Cilium e pode ser usada para anunciar endereços de pod e serviços em sua rede on-premises. Para obter métodos alternativos de tornar os CIDRs de pods roteáveis na rede on-premises, consulte the section called “CIDRs de pods remotos roteáveis”.

Configurar o BGP do Cilium

Pré-requisitos

- Cilium instalado seguindo as instruções em the section called “Configurar CNI”.

Procedimento

1. Para usar o BGP com o Cilium para anunciar endereços de pods ou serviços na sua rede on-premises, você deve ter o Cilium instalado com `bgpControlPlane.enabled: true`. Caso esteja habilitando o BGP para uma implantação existente do Cilium, você deverá reiniciar o operador do Cilium para aplicar a configuração do BGP caso ele não tenha sido habilitado anteriormente. Você pode definir `operator.rollOutPods` como `true` nos seus valores do Helm para reiniciar o operador do Cilium como parte do processo de instalação e atualização do Helm.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  --set bgpControlPlane.enabled=true
```

2. Confirme se o operador e os atendentes do Cilium foram reiniciados e estão funcionando.

```
kubectl -n kube-system get pods --selector=app.kubernetes.io/part-of=cilium
```

```
NAME                               READY   STATUS    RESTARTS   AGE
cilium-grwlc                       1/1     Running   0          4m12s
cilium-operator-68f7766967-5nnbl   1/1     Running   0          4m20s
cilium-operator-68f7766967-7spfz   1/1     Running   0          4m20s
cilium-pnxcv                       1/1     Running   0          6m29s
cilium-r7qkj                       1/1     Running   0          4m12s
```

```
cilium-wxhfn                       1/1     Running   0          4m1s
cilium-z7hlb                       1/1     Running   0          6m30s
```

3. Crie um arquivo chamado `cilium-bgp-cluster.yaml` com uma definição `CiliumBGPClusterConfig`. Talvez seja necessário obter as informações a seguir do seu administrador de rede.
   - Configure o `localASN` com o ASN dos nós que executam o Cilium.
   - Configure o `peerASN` com o ASN do roteador on-premises.
   - Configure o `peerAddress` com o IP do roteador on-premises com o qual cada nó que executa o Cilium fará emparelhamento.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPClusterConfig
metadata:
  name: cilium-bgp
spec:
  nodeSelector:
    matchExpressions:
    - key: eks.amazonaws.com/compute-type
      operator: In
      values:
      - hybrid
  bgpInstances:
  - name: "rack0"
    localASN: NODES_ASN
    peers:
    - name: "onprem-router"
      peerASN: ONPREM_ROUTER_ASN
      peerAddress: ONPREM_ROUTER_IP
      peerConfigRef:
        name: "cilium-peer"
```

4. Aplique a configuração do cluster do BGP do Cilium ao seu cluster.

```
kubectl apply -f cilium-bgp-cluster.yaml
```

5. Crie um arquivo `cilium-bgp-peer.yaml` com o nome do recurso `CiliumBGPPeerConfig` que define a configuração de pares do BGP. Vários pares podem compartilhar a mesma configuração e fornecer referência ao recurso comum de `CiliumBGPPeerConfig`. Consulte a configuração de pares do BGP na documentação do Cilium para obter uma lista completa das opções de configuração.

Os valores das seguintes configurações de pares do Cilium devem corresponder aos do roteador on-premises com o qual você está fazendo o emparelhamento.

- Configure `holdTimeSeconds`, que determina quanto tempo um par do BGP espera por uma mensagem de manutenção ou atualização antes de declarar a sessão inativa. O padrão é de 90 segundos.
- Configure `keepAliveTimeSeconds`, que determina se um par do BGP ainda está acessível e se a sessão do BGP está ativa. O padrão é 30 segundos.
- Configure `restartTimeSeconds`, que determina o tempo em que se espera que o ambiente de gerenciamento do BGP do Cilium restabeleça a sessão do BGP após uma reinicialização. O padrão é 120 segundos.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPPeerConfig
metadata:
  name: cilium-peer
spec:
  timers:
    holdTimeSeconds: 90
    keepAliveTimeSeconds: 30
  gracefulRestart:
    enabled: true
    restartTimeSeconds: 120
  families:
    - afi: ipv4
      safi: unicast
      advertisements:
        matchLabels:
          advertise: "bgp"
```

6. Aplique a configuração de pares do BGP do Cilium ao seu cluster.

```
kubectl apply -f cilium-bgp-peer.yaml
```

7. Crie um arquivo `cilium-bgp-advertisement-pods.yaml` com o nome de um recurso `CiliumBGPAdvertisement` para anunciar os CIDRs do pod em sua rede on-premises.
   - O recurso `CiliumBGPAdvertisement` é usado para definir tipos e atributos de anúncios associados a eles. O exemplo abaixo configura o Cilium para anunciar somente CIDRs de pod. Veja os exemplos em the section called “Tipo de serviço LoadBalancer” e the section

called “Balanceamento de carga no cluster do Cilium” para obter mais informações sobre como configurar o Cilium para anunciar endereços de serviços.

- Cada nó híbrido que executa o atendente do Cilium emparelha com o roteador upstream habilitado para BGP. Cada nó anuncia a faixa de pod CIDR que possui quando `advertisementType` do Cilium está configurada como `PodCIDR` no exemplo abaixo. Consulte a configuração de anúncios do BGP na documentação do Cilium para obter mais informações.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisement-pods
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "PodCIDR"
```

8. Aplique a configuração de anúncios do BGP do Cilium ao seu cluster.

```
kubectl apply -f cilium-bgp-advertisement-pods.yaml
```

9. Você pode confirmar que o emparelhamento do BGP funcionou com a CLI do Cilium usando o comando `cilium bgp peers`. Você deve ver os valores corretos na saída do ambiente e no estado da sessão como `established`. Consulte o Guia de solução de problemas e operações na documentação do Cilium para obter mais informações sobre a solução de problemas.

   Nos exemplos abaixo, há cinco nós híbridos executando o atendente do Cilium e cada nó está anunciando o intervalo de CIDR do Pod que possui.

```
cilium bgp peers
```

```
Node                   Local AS    Peer AS              Peer Address        Session
 State   Uptime     Family         Received   Advertised
mi-026d6a261e355fba7   NODES_ASN
                   ONPREM_ROUTER_ASN
                   ONPREM_ROUTER_IP    established     1h18m58s   ipv4/unicast   1
      2
mi-082f73826a163626e   NODES_ASN
                   ONPREM_ROUTER_ASN
```

```
                  ONPREM_ROUTER_IP    established     1h19m12s   ipv4/unicast   1
      2
mi-09183e8a3d755abf6   NODES_ASN
                  ONPREM_ROUTER_ASN
                  ONPREM_ROUTER_IP    established     1h18m47s   ipv4/unicast   1
      2
mi-0d78d815980ed202d   NODES_ASN
                  ONPREM_ROUTER_ASN
                  ONPREM_ROUTER_IP    established     1h19m12s   ipv4/unicast   1
      2
mi-0daa253999fe92daa   NODES_ASN
                  ONPREM_ROUTER_ASN
                  ONPREM_ROUTER_IP    established     1h18m58s   ipv4/unicast   1
      2
```

```
cilium bgp routes
```

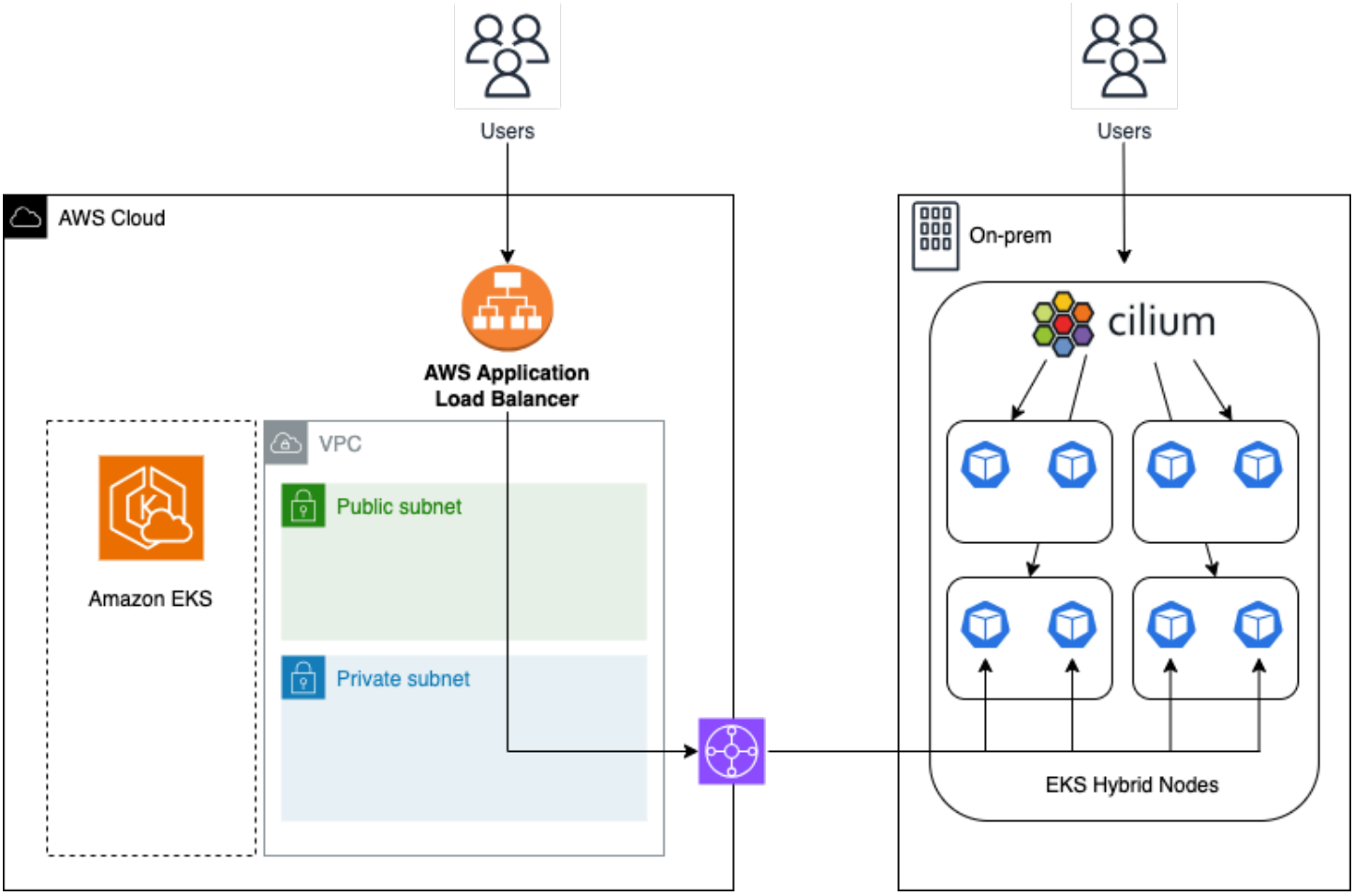
```
Node                   VRouter       Prefix           NextHop   Age         Attrs
mi-026d6a261e355fba7   NODES_ASN     10.86.2.0/26     0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-082f73826a163626e   NODES_ASN     10.86.2.192/26   0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-09183e8a3d755abf6   NODES_ASN     10.86.2.64/26    0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-0d78d815980ed202d   NODES_ASN     10.86.2.128/26   0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-0daa253999fe92daa   NODES_ASN     10.86.3.0/26     0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
```

Configurar o Ingress do Kubernetes para nós híbridos

Este tópico descreve como configurar o Ingress do Kubernetes para workloads em execução no Amazon EKS Hybrid Nodes. O Ingress do Kubernetes expõe rotas HTTP e HTTPS de fora do cluster para os serviços dentro do cluster. Para usar os recursos do Ingress, é necessário um controlador do Ingress do Kubernetes para configurar a infraestrutura de rede e os componentes que atendem ao tráfego da rede.

A AWS oferece suporte para o AWS Application Load Balancer (ALB) e Cilium para Ingress do Kubernetes para workloads em execução no EKS Hybrid Nodes. A decisão de usar o ALB ou o Cilium para o Ingress é baseada na origem do tráfego da aplicação. Se o tráfego da aplicação for

originário de uma região da AWS, a AWS recomenda usar o AWS ALB e o AWS Load Balancer Controller. Se o tráfego da aplicação for originado do ambiente on-premises ou da borda, a AWS recomenda usar os recursos de entrada integrados do Cilium, que podem ser usados com ou sem a infraestrutura de balanceador de carga em seu ambiente.



Application Load Balancer da AWS

Você pode usar o AWS Load Balancer Controller e o Application Load Balancer (ALB) com o tipo de destino `ip` para workloads executadas em nós híbridos. Ao usar o tipo de destino `ip`, o ALB encaminha o tráfego diretamente para os pods, ignorando o caminho de rede da camada de serviço. Para que o ALB alcance os destinos IP do pod on-premises em nós híbridos, o CIDR do pod on-premises deverá ser roteável em sua rede on-premises. Além disso, o AWS Load Balancer Controller usa webhooks e requer comunicação direta do ambiente de gerenciamento EKS. Para obter mais informações, consulte the section called “Configurar webhooks”.

Considerações

- Consulte the section called “Balanceador de carga da aplicação” e the section called “Instalação usando o Helm” para obter mais informações sobre o AWS Application Load Balancer e o AWS Load Balancer Controller.
- Consulte Práticas recomendadas para balanceamento de carga para obter informações sobre como escolher entre o AWS Application Load Balancer e o AWS Network Load Balancer.
- Consulte Anotações do Ingress do AWS Load Balancer Controller para ver a lista de anotações que podem ser configuradas para recursos do Ingress com o AWS Application Load Balancer.

Pré-requisitos

- Cilium instalado seguindo as instruções em the section called “Configurar CNI”.
- O ambiente de gerenciamento BGP do Cilium é habilitado seguindo as instruções em the section called “Configurar o BGP”. Se você não quiser usar o BGP, deverá usar um método alternativo para tornar os CIDRs do pod on-premises roteáveis em sua rede on-premises. Se você não tornar seus CIDRs de pod on-premises roteáveis, o ALB não poderá registrar ou contatar seus destinos IP de pod.
- O Helm instalado em seu ambiente de linha de comando. Consulte as Instruções de configuração do Helm para obter mais informações.
- O eksctl instalado em seu ambiente de linha de comando. Consulte as instruções de instalação do eksctl para obter mais informações.

Procedimento

1. Baixe uma política do IAM para o AWS Load Balancer Controller que permita que ele faça chamadas para APIs da AWS em seu nome.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/refs/heads/main/docs/install/iam_policy.json
```

2. Crie uma política do IAM usando a política obtida por download na etapa anterior.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

3. Substitua os valores do nome do cluster (`CLUSTER_NAME`), região da AWS (`AWS_REGION`) e ID da conta da AWS (`AWS_ACCOUNT_ID`) pelas suas configurações e execute o comando a seguir.

```
eksctl create iamserviceaccount \
    --cluster=CLUSTER_NAME \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws:iam::AWS_ACCOUNT_ID:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
    --region AWS_REGION \
    --approve
```

4. Adicione o repositório de chart do Helm do eks-charts e atualize o repositório Helm local para garantir que você tenha os gráficos mais recentes.

```
helm repo add eks https://aws.github.io/eks-charts
```

```
helm repo update eks
```

5. Instale o AWS Load Balancer Controller. Substitua os valores do nome do cluster (`CLUSTER_NAME`), região da AWS (`AWS_REGION`), ID da VPC (`VPC_ID`) e versão do chart do Helm do AWS Load Balancer Controller (`AWS_LBC_HELM_VERSION`) pelas suas configurações e execute o comando a seguir. Se você estiver executando um cluster em modo misto com nós híbridos e nós na Nuvem AWS, você pode executar o AWS Load Balancer Controller nos nós da nuvem seguindo as instruções em the section called “AWS Load Balancer Controller”.

   - Você pode encontrar a versão mais recente do chart do Helm executando `helm search repo eks/aws-load-balancer-controller --versions`.

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --version AWS_LBC_HELM_VERSION \
  --set clusterName=CLUSTER_NAME \
  --set region=AWS_REGION \
  --set vpcId=VPC_ID \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

6. Verifique se o AWS Load Balancer Controller foi instalado com êxito.

```
kubectl get -n kube-system deployment aws-load-balancer-controller
```

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

7. Crie uma aplicação de exemplo. O exemplo abaixo usa a aplicação de microsserviços de amostra Istio Bookinfo.

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
```

8. Crie um arquivo denominado `my-ingress-alb.yaml` com os conteúdos a seguir.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  namespace: default
  annotations:
    alb.ingress.kubernetes.io/load-balancer-name: "my-ingress-alb"
    alb.ingress.kubernetes.io/target-type: "ip"
    alb.ingress.kubernetes.io/scheme: "internet-facing"
    alb.ingress.kubernetes.io/healthcheck-path: "/details/1"
spec:
  ingressClassName: alb
  rules:
  - http:
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

9. Aplique a configuração do Ingress ao seu cluster.

```
kubectl apply -f my-ingress-alb.yaml
```

10. O provisionamento do ALB para o recurso do Ingress pode demorar alguns minutos. Depois que o ALB for provisionado, seu recurso do Ingress terá um endereço atribuído a ele que corresponde ao nome DNS da implantação do ALB. O endereço terá o formato `<alb-name>-<random-string>.<region>.elb.amazonaws.com`.

```
kubectl get ingress my-ingress
```

```
NAME         CLASS   HOSTS   ADDRESS
    PORTS   AGE
my-ingress   alb     *       my-ingress-alb-<random-
string>.<region>.elb.amazonaws.com   80      23m
```

11. Acesse o serviço usando o endereço do ALB.

```
curl -s http//my-ingress-alb-<random-string>.<region>.elb.amazonaws.com:80/details/1
 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
  "details": "This is the details page"
}
```

Visão geral do Ingress do Cilium e do Gateway do Cilium

Os recursos de entrada do Cilium são incorporados à arquitetura do Cilium e podem ser gerenciados com a API do Ingress do Kubernetes ou a API do Gateway. Se você não tem recursos do Ingress existentes, a AWS recomenda começar com a API do Gateway, pois é uma forma mais expressiva e flexível de definir e gerenciar os recursos de rede do Kubernetes. A API do Gateway do Kubernetes visa padronizar como os recursos de rede para Ingress, balanceamento de carga e malha de serviços são definidos e gerenciados nos clusters do Kubernetes.

Quando você habilita atributos do Ingress ou do Gateway do Cilium, o operador do Cilium reconcilia os objetos do Ingress ou Gateway no cluster e os proxies Envoy em cada nó processam o tráfego de rede da Camada 7 (L7). O Cilium não provisiona diretamente a infraestrutura Ingress ou Gateway, como balanceadores de carga. Se você planeja usar o Ingress do Cilium/Gateway com um balanceador de carga, deve usar as ferramentas do balanceador de carga, geralmente um controlador Ingress ou Gateway, para implantar e gerenciar a infraestrutura do balanceador de carga.

Para tráfego de Ingress ou Gateway, o Cilium lida com o tráfego da rede principal e a aplicação da política L3/L4, e os proxies Envoy integrados processam o tráfego de rede de L7. Com o Ingress do Cilium/Gateway, o Envoy é responsável por aplicar regras, políticas e manipulação de solicitações de roteamento L7, gerenciamento avançado de tráfego, como divisão e espelhamento de tráfego, e terminação e originação de TLS. Os proxies Envoy do Cilium são implantados como um DaemonSet (`cilium-envoy`) separado por padrão, o que permite que o Envoy e o atendente do Cilium sejam atualizados, escalados e gerenciados separadamente.

Para obter mais informações sobre como o Ingress do Cilium e o Gateway do Cilium funcionam, consulte as páginas Ingress do Cilium e Gateway do Cilium na documentação do Cilium.

Comparação entre Ingress e Gateway do Cilium

A tabela abaixo resume os recursos do Ingress do Cilium e do Gateway do Cilium a partir da versão 1.17.x do Cilium.

Recurso	Ingress	Gateway
Tipo de serviço LoadBalancer	Sim	Sim
Tipo de serviço NodePort	Sim	Não[1]
Rede do host	Sim	Sim
Balanceador de carga compartilhado	Sim	Sim
Balanceador de carga dedicado	Sim	Número[2]
Políticas de rede	Sim	Sim
Protocolos	Camada 7 (HTTP(S), gRPC)	Camada 7 (HTTP(S), gRPC)[3]

Recurso	Ingress	Gateway
Passagem do TLS	Sim	Sim
Gerenciamento de tráfego	Roteamento de caminhos e hosts	Roteamento de caminho e host, redirecionamento e reescrita de URL, divisão de tráfego, modificação de cabeçalho

[1] O suporte do Gateway do Cilium para serviços NodePort está planejado para a versão 1.18.x do Cilium (#27273)

[2] Suporte do Gateway do Cilium para balanceadores de carga dedicados (#25567)

[3] Suporte do Gateway do Cilium para TCP/UDP (#21929)

Instale o Gateway do Cilium

Considerações

- O Cilium deve ser configurado com `nodePort.enabled` definido como `true`, conforme mostrado nos exemplos abaixo. Se você estiver usando o atributo de substituição de kube-proxy do Cilium, não precisará definir `nodePort.enabled` como `true`.
- O Cilium deve ser configurado com `envoy.enabled` definido como `true`, conforme mostrado nos exemplos abaixo.
- O Gateway do Cilium pode ser implantado no balanceador de carga (padrão) ou no modo de rede host.
- Ao usar o Gateway do Cilium no modo balanceador de carga, a anotação `service.beta.kubernetes.io/aws-load-balancer-type: "external"` deve ser definida no recurso do Gateway para evitar que o provedor de, nuvem AWS herdado crie um Classic Load Balancer para o serviço do tipo LoadBalancer que o Cilium cria para o recurso do Gateway.
- Ao usar o Gateway do Cilium no modo de rede do host, o serviço do tipo LoadBalancer é desabilitado. O modo de rede do host é útil para ambientes que não têm infraestrutura de balanceador de carga, consulte the section called “Rede do host” para obter mais informações.

Pré-requisitos

1. O Helm foi instalado em seu ambiente de linha de comando. Consulte as Instruções de configuração do Helm.
2. Cilium instalado seguindo as instruções em the section called “Configurar CNI”.

Procedimento

1. Instale as definições de recursos personalizados (CRDs) da API do Gateway do Kubernetes.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_gatewayclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_gateways.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_httproutes.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_referencegrants.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_grpcroutes.yaml
```

2. Crie um arquivo chamado `cilium-gateway-values.yaml` com o conteúdo a seguir. O exemplo abaixo configura o Gateway do Cilium para usar o modo padrão do balanceador de carga e usar um DaemonSet `cilium-envoy` separado para proxies Envoy configurados para serem executados somente em nós híbridos.

```
gatewayAPI:
  enabled: true
  # uncomment to use host network mode
  # hostNetwork:
  #   enabled: true
nodePort:
  enabled: true
envoy:
  enabled: true
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
```

```
            operator: In
            values:
            - hybrid
```

3. Aplique o arquivo de valores do Helm ao seu cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  --values cilium-gateway-values.yaml
```

4. Confirme se o operador, o atendente e os pods do Envoy do Cilium estão funcionando.

```
kubectl -n kube-system get pods --selector=app.kubernetes.io/part-of=cilium
```

```
NAME                               READY   STATUS    RESTARTS   AGE
cilium-envoy-5pgnd                 1/1     Running   0          6m31s
cilium-envoy-6fhg4                 1/1     Running   0          6m30s
cilium-envoy-jskrk                 1/1     Running   0          6m30s
cilium-envoy-k2xtb                 1/1     Running   0          6m31s
cilium-envoy-w5s9j                 1/1     Running   0          6m31s
cilium-grwlc                       1/1     Running   0          4m12s
cilium-operator-68f7766967-5nnbl   1/1     Running   0          4m20s
cilium-operator-68f7766967-7spfz   1/1     Running   0          4m20s
cilium-pnxcv                       1/1     Running   0          6m29s
cilium-r7qkj                       1/1     Running   0          4m12s
cilium-wxhfn                       1/1     Running   0          4m1s
cilium-z7hlb                       1/1     Running   0          6m30s
```

Configurar o Gateway do Cilium

O Gateway do Cilium é habilitado em objetos do Gateway definindo `gatewayClassName` como `cilium`. O serviço que o Cilium cria para recursos do Gateway pode ser configurado com campos no objeto Gateway. As anotações comuns usadas pelos controladores do Gateway para configurar a infraestrutura do balanceador de carga podem ser configuradas com o campo `infrastructure` do objeto do Gateway. Ao usar o IPAM de LoadBalancer do Cilium (veja o exemplo em the section called “Tipo de serviço LoadBalancer”), o endereço IP a ser usado para o serviço do tipo LoadBalancer pode ser configurado no campo `addresses` do objeto do Gateway. Para obter mais

informações sobre a configuração do Gateway, consulte a especificação da API do Gateway do Kubernetes.

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: my-gateway
spec:
  gatewayClassName: cilium
  infrastructure:
    annotations:
      service.beta.kubernetes.io/...
      service.kuberentes.io/...
  addresses:
  - type: IPAddress
    value: <LoadBalancer IP address>
  listeners:
  ...
```

O Cilium e a especificação do Kubernetes Gateway oferecem suporte aos recursos GatewayClass, Gateway, HTTPRoute, GRPCRoute e ReferenceGrant.

- Consulte as especificações HTTPRoute e GRPCRoute para ver a lista de campos disponíveis.
- Veja os exemplos na seção the section called “Implantar o Gateway do Cilium” abaixo e os exemplos na documentação do Cilium para saber como usar e configurar esses recursos.

Implantar o Gateway do Cilium

1. Crie uma aplicação de exemplo. O exemplo abaixo usa a aplicação de microsserviços de amostra Istio Bookinfo.

   ```
   kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
   samples/bookinfo/platform/kube/bookinfo.yaml
   ```

2. Verifique se a aplicação está sendo executada com êxito.

   ```
   kubectl get pods
   ```

   ```
   NAME                              READY   STATUS    RESTARTS   AGE
   details-v1-766844796b-9965p       1/1     Running   0          81s
   ```

```
productpage-v1-54bb874995-jmc8j   1/1     Running   0          80s
ratings-v1-5dc79b6bcd-smzxz       1/1     Running   0          80s
reviews-v1-598b896c9d-vj7gb       1/1     Running   0          80s
reviews-v2-556d6457d-xbt8v        1/1     Running   0          80s
reviews-v3-564544b4d6-cpmvq       1/1     Running   0          80s
```

3. Crie um arquivo denominado `my-gateway.yaml` com os conteúdos a seguir. O exemplo abaixo usa a anotação `service.beta.kubernetes.io/aws-load-balancer-type: "external"` para impedir que o provedor de nuvem AWS herdado crie um Classic Load Balancer para o serviço do tipo LoadBalancer que o Cilium cria para o recurso do Gateway.

```
---
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: my-gateway
spec:
  gatewayClassName: cilium
  infrastructure:
    annotations:
      service.beta.kubernetes.io/aws-load-balancer-type: "external"
  listeners:
  - protocol: HTTP
    port: 80
    name: web-gw
    allowedRoutes:
      namespaces:
        from: Same
---
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: http-app-1
spec:
  parentRefs:
  - name: my-gateway
    namespace: default
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /details
    backendRefs:
```

```
    - name: details
      port: 9080
```

4. Aplique o recurso do Gateway ao seu cluster.

```
kubectl apply -f my-gateway.yaml
```

5. Confirme se o recurso do Gateway e o serviço correspondente foram criados. Nesse estágio, espera-se que o campo ADDRESS do recurso do Gateway não seja preenchido com um endereço IP ou nome de host e que o serviço do tipo LoadBalancer para o recurso do Gateway também não tenha um endereço IP ou nome de host atribuído.

```
kubectl get gateway my-gateway
```

```
NAME         CLASS    ADDRESS   PROGRAMMED   AGE
my-gateway   cilium             True         10s
```

```
kubectl get svc cilium-gateway-my-gateway
```

```
NAME                         TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
    AGE
cilium-gateway-my-gateway   LoadBalancer   172.16.227.247   <pending>     80:30912/
TCP   24s
```

6. Continue para the section called “Tipo de serviço LoadBalancer” para configurar o recurso do Gateway para usar um endereço IP alocado pelo IPAM do balanceador de carga do Cilium, e the section called “Tipo de serviço NodePort” ou the section called “Rede do host” para configurar o recurso do Gateway para usar endereços de rede NodePort ou host.

Instale o Ingress do Cilium

Considerações

- O Cilium deve ser configurado com `nodePort.enabled` definido como `true`, conforme mostrado nos exemplos abaixo. Se você estiver usando o atributo de substituição de kube-proxy do Cilium, não precisará definir `nodePort.enabled` como `true`.
- O Cilium deve ser configurado com `envoy.enabled` definido como `true`, conforme mostrado nos exemplos abaixo.

- Com `ingressController.loadbalancerMode` definido como `dedicated`, o Cilium cria serviços dedicados para cada recurso do Ingress. Com `ingressController.loadbalancerMode` definido como `shared`, o Cilium cria um serviço compartilhado do tipo LoadBalancer para todos os recursos do Ingress no cluster. Ao usar o modo `shared` do balanceador de carga, as configurações do serviço compartilhado, como `labels`, `annotations`, `type` e `loadBalancerIP`, são definidas na seção `ingressController.service` dos valores do Helm. Consulte a referência de valores Helm do Cilium para obter mais informações.
- Com `ingressController.default` definido como `true`, o Cilium é configurado como o controlador de padrão do Ingress para o cluster e criará entradas do Ingress mesmo quando `ingressClassName` não estiver especificado nos recursos do Ingress.
- O Ingress do Cilium pode ser implantado no balanceador de carga (padrão), na porta do nó ou no modo de rede host. Quando o Cilium é instalado no modo de rede host, os modos de serviço do tipo LoadBalancer e de serviço do tipo NodePort são desabilitados. Consulte the section called “Rede do host” para obter mais informações.
- Sempre defina `ingressController.service.annotations` como `service.beta.kubernetes.io/aws-load-balancer-type: "external"` nos valores do Helm para evitar que o provedor de nuvem AWS herdado crie um Classic Load Balancer para o serviço padrão `cilium-ingress` criado pelo chart do Helm do Cilium.

Pré-requisitos

1. O Helm foi instalado em seu ambiente de linha de comando. Consulte as Instruções de configuração do Helm.
2. Cilium instalado seguindo as instruções em the section called “Configurar CNI”.

Procedimento

1. Crie um arquivo chamado `cilium-ingress-values.yaml` com o conteúdo a seguir. O exemplo abaixo configura o Ingress do Cilium para usar o modo `dedicated` do balanceador de carga padrão e usar um DaemonSet separado `cilium-envoy` para Envoy, proxies configurados para serem executados somente em nós híbridos.

```
ingressController:
  enabled: true
  loadbalancerMode: dedicated
```

```
  service:
    annotations:
      service.beta.kubernetes.io/aws-load-balancer-type: "external"
nodePort:
  enabled: true
envoy:
  enabled: true
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
            - hybrid
```

2. Aplique o arquivo de valores do Helm ao seu cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  --values cilium-ingress-values.yaml
```

3. Confirme se o operador, o atendente e os pods do Envoy do Cilium estão funcionando.

```
kubectl -n kube-system get pods --selector=app.kubernetes.io/part-of=cilium
```

```
NAME                               READY   STATUS    RESTARTS   AGE
cilium-envoy-5pgnd                 1/1     Running   0          6m31s
cilium-envoy-6fhg4                 1/1     Running   0          6m30s
cilium-envoy-jskrk                 1/1     Running   0          6m30s
cilium-envoy-k2xtb                 1/1     Running   0          6m31s
cilium-envoy-w5s9j                 1/1     Running   0          6m31s
cilium-grwlc                       1/1     Running   0          4m12s
cilium-operator-68f7766967-5nnbl   1/1     Running   0          4m20s
cilium-operator-68f7766967-7spfz   1/1     Running   0          4m20s
cilium-pnxcv                       1/1     Running   0          6m29s
cilium-r7qkj                       1/1     Running   0          4m12s
cilium-wxhfn                       1/1     Running   0          4m1s
cilium-z7hlb                       1/1     Running   0          6m30s
```

Configurar o Ingress do Cilium

O Ingress do Cilium é habilitado em objetos do Ingress definindo `ingressClassName` como `cilium`. Os serviços que o Cilium cria para os recursos do Ingress podem ser configurados com anotações nos objetos do Ingress ao usar o modo `dedicated` do de balanceador de carga, assim como na configuração do Cilium ou Helm ao usar o modo `shared` do balanceador de carga. Essas anotações são comumente usadas pelos controladores do Ingress para configurar a infraestrutura do balanceador de carga ou outros atributos do serviço, como tipo de serviço, modo do balanceador de carga, portas e passagem TLS. As anotações principais são descritas abaixo. Para obter uma lista completa das anotações compatíveis, consulte as Anotações do Ingress do Cilium na documentação do Cilium.

Anotação	Descrição
`ingress.cilium.io/loadbalancer-mode`	`dedicated` : serviço dedicado do tipo LoadBalancer para cada recurso do Ingress (padrão). `shared`: serviço único do tipo LoadBalancer para todos os recursos do Ingress.
`ingress.cilium.io/service-type`	`LoadBalancer` : o serviço será do tipo LoadBalancer (padrão) `NodePort`: o serviço será do tipo NodePort.
`service.beta.kubernetes.io/aws-load-balancer-type`	`"external"` : evite que um provedor de nuvem da AWS antigo provisione o Classic Load Balancer para serviços do tipo LoadBalancer.
`lbipam.cilium.io/ips`	Lista de endereços IP a serem alocados do Cilium LoadBalancer IPAM

O Cilium e a especificação Ingress do Kubernetes oferecem suporte a regras de correspondência exatas, prefixas e específicas de implementação para caminhos de entrada. O Cilium oferece suporte para regex como sua regra de correspondência específica para a implementação. Para obter mais informações, consulte Tipos e precedência de caminhos de entrada e Exemplos de tipos de caminhos na documentação do Cilium e os exemplos na seção the section called “Implementar o Ingress do Cilium” desta página.

Um exemplo de objeto do Ingress do Cilium é mostrado abaixo.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    service.beta.kuberentes.io/...
    service.kuberentes.io/...
spec:
  ingressClassName: cilium
  rules:
  ...
```

Implementar o Ingress do Cilium

1. Crie uma aplicação de exemplo. O exemplo abaixo usa a aplicação de microsserviços de amostra Istio Bookinfo.

   ```
   kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
   samples/bookinfo/platform/kube/bookinfo.yaml
   ```

2. Verifique se a aplicação está sendo executada com êxito.

   ```
   kubectl get pods
   ```

   ```
   NAME                              READY   STATUS    RESTARTS   AGE
   details-v1-766844796b-9965p       1/1     Running   0          81s
   productpage-v1-54bb874995-jmc8j   1/1     Running   0          80s
   ratings-v1-5dc79b6bcd-smzxz       1/1     Running   0          80s
   reviews-v1-598b896c9d-vj7gb       1/1     Running   0          80s
   reviews-v2-556d6457d-xbt8v        1/1     Running   0          80s
   reviews-v3-564544b4d6-cpmvq       1/1     Running   0          80s
   ```

3. Crie um arquivo denominado `my-ingress.yaml` com os conteúdos a seguir. O exemplo abaixo usa a anotação `service.beta.kubernetes.io/aws-load-balancer-type: "external"` para impedir que o provedor de nuvem AWS herdado crie um Classic Load Balancer para o serviço do tipo LoadBalancer que o Cilium cria para o recurso do Ingress.

   ```
   apiVersion: networking.k8s.io/v1
   kind: Ingress
   ```

```
metadata:
  name: my-ingress
  namespace: default
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "external"
spec:
  ingressClassName: cilium
  rules:
  - http:
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

4. Aplique o recurso do Ingress ao seu cluster.

```
kubectl apply -f my-ingress.yaml
```

5. Confirme se o recurso de entrada e o serviço correspondente foram criados. Nesse estágio, espera-se que o campo ADDRESS do recurso do Ingress não seja preenchido com um endereço IP ou nome de host e que o serviço compartilhado ou dedicado do tipo LoadBalancer para o recurso do Ingress também não tenha um endereço IP ou nome de host atribuído.

```
kubectl get ingress my-ingress
```

```
NAME         CLASS    HOSTS   ADDRESS   PORTS   AGE
my-ingress   cilium   *                 80      8s
```

Para o modo `shared` do balanceador de carga

```
kubectl -n kube-system get svc cilium-ingress
```

```
NAME             TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
      AGE
cilium-ingress   LoadBalancer   172.16.217.48   <pending>     80:32359/TCP,443:31090/
TCP   10m
```

Para o modo `dedicated` do balanceador de carga

```
kubectl -n default get svc cilium-ingress-my-ingress
```

```
NAME                         TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
                  AGE
cilium-ingress-my-ingress    LoadBalancer   172.16.193.15   <pending>     80:32088/
TCP,443:30332/TCP   25s
```

6. Continue para the section called “Tipo de serviço LoadBalancer” para configurar o recurso do Ingress para usar um endereço IP alocado pelo IPAM do balanceador de carga do Cilium, e the section called “Tipo de serviço NodePort” ou the section called “Rede do host” para configurar o recurso do Ingress para usar endereços de rede NodePort ou host.

Tipo de serviço LoadBalancer

Infraestrutura de balanceador de carga existente

Por padrão, tanto para o Ingress do Cilium quanto para o Gateway do Cilium, o Cilium cria serviços do Kubernetes do tipo LoadBalancer para os recursos do Ingress ou Gateway. Os atributos do(s) serviço(s) que o Cilium cria podem ser configurados por meio dos recursos do Ingress e do Gateway. Quando você cria recursos de Ingress ou Gateway, o endereço IP ou os nomes de host expostos externamente para o Ingress ou o Gateway são alocados da infraestrutura do balanceador de carga, que normalmente é provisionada por um controlador de Ingress ou Gateway.

Muitos controladores Ingress e Gateway usam anotações para detectar e configurar a infraestrutura do balanceador de carga. As anotações desses controladores Ingress e Gateway são configuradas nos recursos do Ingress ou do Gateway, conforme mostrado nos exemplos anteriores acima. Consulte a documentação do controlador do Ingress ou do Gateway para ver as anotações compatíveis e consulte a documentação do Ingress do Kubernetes e a documentação do Gateway do Kubernetes para ver uma lista dos controladores populares.

> **Important**
>
> O Ingress e o Gateway do Cilium não podem ser usados com o AWS Load Balancer Controller e o AWS Network Load Balancers (NLBs) com o EKS Hybrid Nodes. A tentativa de usá-los juntos resulta em destinos não registrados, pois o NLB tenta se conectar diretamente aos IPs do pod que apoiam o serviço do tipo LoadBalancer quando o tipo `target-type` do

NLB está definido como `ip` (requisito para usar o NLB com workloads em execução no EKS Hybrid Nodes).

Não há Infraestrutura de balanceador de carga

Se você não tiver uma infraestrutura de balanceador de carga e o controlador do Ingress ou Gateway correspondente em seu ambiente, os recursos do Ingress ou Gateway e os serviços correspondentes do tipo LoadBalancer podem ser configurados para usar endereços IP alocados pelo gerenciamento de endereços IP do balanceador de carga (LB IPAM) do Cilium. O IPAM de LB do Cilium pode ser configurado com intervalos de endereços IP conhecidos de seu ambiente on-premises e pode usar o suporte integrado ao Protocolo de Gateway da Borda (BGP) do Cilium ou anúncios L2 para anunciar os endereços IP do LoadBalancer em sua rede on-premises.

O exemplo abaixo mostra como configurar o LB IPAM do Cilium com um endereço IP para usar em seus recursos do Ingress ou Gateway e como configurar o ambiente de gerenciamento BGP do Cilium para anunciar o endereço IP do LoadBalancer na rede on-premises. O atributo LB IPAM do Cilium está habilitado por padrão, mas não será habilitado até a criação de um recurso `CiliumLoadBalancerIPPool`.

Pré-requisitos

- Ingress do Cilium ou Gateway instalado seguindo as instruções em the section called “Instale o Ingress do Cilium” ou the section called “Instale o Gateway do Cilium”.
- Recursos do Ingress ou Gateway do Cilium com a aplicação de amostra implantada seguindo as instruções em the section called “Implementar o Ingress do Cilium” ou the section called “Implantar o Gateway do Cilium”.
- O ambiente de gerenciamento BGP do Cilium é habilitado seguindo as instruções em the section called “Configurar o BGP”. Se você não quiser usar o BGP, poderá ignorar esse pré-requisito, mas não poderá acessar seu recurso de entrada ou gateway até que o endereço IP do LoadBalancer alocado pelo IPAM do Cilium LB seja roteável em sua rede on-premises.

Procedimento

1. Opcionalmente, corrija o recurso do Ingress ou Gateway para solicitar um endereço IP específico para usar no serviço do tipo LoadBalancer. Se você não solicitar um endereço IP específico, o Cilium alocará um endereço IP do intervalo de endereços IP configurado no recurso

`CiliumLoadBalancerIPPool` na etapa subsequente. Nos comandos abaixo, substitua `LB_IP_ADDRESS` pelo endereço IP para solicitar o serviço do tipo LoadBalancer.

Gateway

```
kubectl patch gateway -n default my-gateway --type=merge -p '{
  "spec": {
    "addresses": [{"type": "IPAddress", "value": "LB_IP_ADDRESS"}]
  }
}'
```

Ingress

```
kubectl patch ingress my-ingress --type=merge -p '{
  "metadata": {"annotations": {"lbipam.cilium.io/ips": "LB_IP_ADDRESS"}}
}'
```

2. Crie um arquivo chamado `cilium-lbip-pool-ingress.yaml` com um recurso `CiliumLoadBalancerIPPool` para configurar o intervalo de endereços IP do balanceador de carga para seus recursos do Ingress ou Gateway.
   - Se você estiver usando o Ingress do Cilium, o Cilium aplica automaticamente o rótulo `cilium.io/ingress: "true"` aos serviços que cria para os recursos do Ingress. Você pode usar esse rótulo no campo `serviceSelector` da definição do recurso `CiliumLoadBalancerIPPool` para selecionar os serviços elegíveis para o LB IPAM.
   - Se você estiver usando o Gateway do Cilium, poderá usar o rótulo `gateway.networking.k8s.io/gateway-name` nos campos `serviceSelector` da definição do recurso `CiliumLoadBalancerIPPool` para selecionar os recursos do Gateway elegíveis para o LB IPAM.
   - Substitua `LB_IP_CIDR` pelo intervalo de endereços IP a ser usado para os endereços IP do balanceador de carga. Para selecionar um único endereço IP, use um `/32` CIDR. Para obter mais informações, consulte Gerenciamento de endereços IP do LoadBalancer na documentação do Cilium.

   ```
   apiVersion: cilium.io/v2alpha1
   kind: CiliumLoadBalancerIPPool
   metadata:
     name: bookinfo-pool
   spec:
     blocks:
   ```

```
  - cidr: "LB_IP_CIDR"
  serviceSelector:
    # if using Cilium Gateway
    matchExpressions:
      - { key: gateway.networking.k8s.io/gateway-name, operator: In, values: [ my-
gateway ] }
    # if using Cilium Ingress
    matchLabels:
      cilium.io/ingress: "true"
```

3. Aplique o recurso `CiliumLoadBalancerIPPool` ao seu cluster.

```
kubectl apply -f cilium-lbip-pool-ingress.yaml
```

4. Confirme se um endereço IP foi alocado por meio do LB IPAM do Colium para o recurso do Ingress ou Gateway.

   Gateway

```
kubectl get gateway my-gateway
```

```
NAME         CLASS    ADDRESS        PROGRAMMED   AGE
my-gateway   cilium   LB_IP_ADDRESS    True         6m41s
```

   Ingress

```
kubectl get ingress my-ingress
```

```
NAME         CLASS    HOSTS   ADDRESS        PORTS   AGE
my-ingress   cilium   *       LB_IP_ADDRESS   80      10m
```

5. Crie um arquivo nomeado como `cilium-bgp-advertisement-ingress.yaml` com um recurso `CiliumBGPAdvertisement` para anunciar o endereço IP do LoadBalancer para os recursos do Ingress ou Gateway. Se você não estiver usando o Cilium BGP, poderá ignorar esta etapa. O endereço IP do LoadBalancer usado para seu recurso do Ingress ou Gateway deve ser roteável em sua rede on-premises para que você possa consultar o serviço na próxima etapa.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
```

```
  name: bgp-advertisement-lb-ip
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - LoadBalancerIP
      selector:
        # if using Cilium Gateway
        matchExpressions:
          - { key: gateway.networking.k8s.io/gateway-name, operator: In, values:
 [ my-gateway ] }
        # if using Cilium Ingress
        matchLabels:
          cilium.io/ingress: "true"
```

6. Aplique o recurso `CiliumBGPAdvertisement` ao seu cluster.

```
kubectl apply -f cilium-bgp-advertisement-ingress.yaml
```

7. Acesse o serviço usando o endereço IP alocado pelo Cilium LB IPAM.

```
curl -s http://LB_IP_ADDRESS:80/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

Tipo de serviço NodePort

Se você não tiver infraestrutura de balanceador de carga e controlador de entrada correspondente em seu ambiente, ou se estiver autogerenciando sua infraestrutura de balanceador de carga ou usando balanceamento de carga baseado em DNS, você pode configurar o Ingress do Cilium para criar serviços do tipo NodePort para os recursos do Ingress. Ao usar o NodePort com o Ingress do Cilium, o serviço do tipo NodePort é exposto em uma porta em cada nó no intervalo de portas 30000-32767. Nesse modo, quando o tráfego chega a qualquer nó no cluster no NodePort, ele é encaminhado para um pod que faz o backup do serviço, que pode estar no mesmo nó ou em um nó diferente.

> **Note**
>
> O suporte do Gateway do Cilium para serviços NodePort está planejado para a versão 1.18.x (#27273)

Pré-requisitos

- O Ingress do Cilium foi instalado seguindo as instruções em the section called “Instale o Ingress do Cilium”.
- Recursos do Ingress do Cilium com aplicação de amostra implantada seguindo as instruções em the section called “Implementar o Ingress do Cilium”.

Procedimento

1. Corrija o recurso do Ingress existente `my-ingress` para alterá-lo do tipo de serviço LoadBalancer para NodePort.

   ```
   kubectl patch ingress my-ingress --type=merge -p '{
       "metadata": {"annotations": {"ingress.cilium.io/service-type": "NodePort"}}
   }'
   ```

   Se você não criou o recurso do Ingress, poderá fazê-lo aplicando a seguinte definição do Ingress ao seu cluster. Observe que a definição do Ingress abaixo usa a aplicação de amostra Istio Bookinfo descrita em the section called “Implementar o Ingress do Cilium”.

   ```
   apiVersion: networking.k8s.io/v1
   ```

```
kind: Ingress
metadata:
  name: my-ingress
  namespace: default
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "external"
    "ingress.cilium.io/service-type": "NodePort"
spec:
  ingressClassName: cilium
  rules:
  - http:
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

2. Confirme se o serviço do recurso do Ingress foi atualizado para usar o tipo de serviço NodePort. Observe a porta do protocolo HTTP na saída. No exemplo abaixo, essa porta HTTP é 32353, que será usada em uma etapa subsequente para fazer uma consulta sobre o serviço. A vantagem de usar o Ingress do Cilium com serviço do tipo NodePort é que você pode aplicar roteamento baseado em caminho e host, bem como políticas de rede para o tráfego de entrada, o que você não pode fazer para um serviço padrão do tipo NodePort sem o Ingress.

```
kubectl -n default get svc cilium-ingress-my-ingress
```

```
NAME                         TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)
             AGE
cilium-ingress-my-ingress   NodePort   172.16.47.153   <none>        80:32353/
TCP,443:30253/TCP   27m
```

3. Obtenha os endereços IP dos nós do seu cluster.

```
kubectl get nodes -o wide
```

```
NAME                   STATUS   ROLES    AGE   VERSION              INTERNAL-IP
 EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
```

```
mi-026d6a261e355fba7   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.150
 <none>        Ubuntu 22.04.5 LTS   5.15.0-142-generic   containerd://1.7.27
mi-082f73826a163626e   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.32
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-09183e8a3d755abf6   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.33
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0d78d815980ed202d   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.97
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0daa253999fe92daa   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.100
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

4. Acesse o serviço do tipo NodePort usando os endereços IP dos seus nós e o NodePort capturado acima. No exemplo abaixo, o endereço IP do nó usado é `10.80.146.32` e o NodePort é `32353`. Substitua-os pelos valores para o seu ambiente.

```
curl -s http://10.80.146.32:32353/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

Rede do host

Semelhante ao serviço do tipo NodePort, se você não tiver uma infraestrutura de balanceador de carga e um controlador de entrada ou gateway, ou se estiver gerenciando automaticamente seu balanceamento de carga com um balanceador de carga externo, poderá configurar o Ingress do Cilium e o Gateway do Cilium para expor os recursos do Ingress e do Gateway diretamente na rede do host. Quando o modo de rede do host está habilitado para um recurso de entrada ou gateway, o serviço dos modos LoadBalancer e NodePort é automaticamente desabilitado. O modo de rede do host é mutuamente exclusivo com esses modos alternativos para cada recurso de entrada ou gateway. Comparado ao serviço do tipo NodePort, o modo de rede host oferece flexibilidade adicional para a variedade de portas que podem ser usadas (não está restrito ao

intervalo 30000-32767 NodePort) e você pode configurar um subconjunto de nós em que os proxies Envoy são executados na rede do host.

Pré-requisitos

- Ingress do Cilium ou Gateway instalado seguindo as instruções em the section called “Instale o Ingress do Cilium” ou the section called “Instale o Gateway do Cilium”.

Procedimento

Gateway

1. Crie um arquivo chamado `cilium-gateway-host-network.yaml` com o conteúdo a seguir.

```
gatewayAPI:
  enabled: true
  hostNetwork:
    enabled: true
    # uncomment to restrict nodes where Envoy proxies run on the host network
    # nodes:
    #   matchLabels:
    #     role: gateway
```

2. Aplique a configuração do Gateway do Cilium de rede do host ao seu cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  -f cilium-gateway-host-network.yaml
```

Se você não criou o recurso do Gateway, pode criá-lo aplicando a seguinte definição de Gateway ao seu cluster. A definição de Gateway abaixo usa a aplicação de amostra Istio Bookinfo descrita em the section called “Implantar o Gateway do Cilium”. No exemplo abaixo, o recurso do Gateway está configurado para usar a porta 8111 do receptor HTTP, que é a porta compartilhada do receptor para os proxies Envoy em execução na rede do host. Se você estiver usando uma porta privilegiada (inferior a 1023) para o recurso do Gateway, consulte a documentação do Cilium para obter instruções.

```
---
```

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: my-gateway
spec:
  gatewayClassName: cilium
  listeners:
  - protocol: HTTP
    port: 8111
    name: web-gw
    allowedRoutes:
      namespaces:
        from: Same
---
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: http-app-1
spec:
  parentRefs:
  - name: my-gateway
    namespace: default
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /details
    backendRefs:
    - name: details
      port: 9080
```

Você pode observar a configuração aplicada do Cilium Envoy com o comando a seguir.

```
kubectl get cec cilium-gateway-my-gateway -o yaml
```

Você pode obter a porta do receptor Envoy para o serviço `cilium-gateway-my-gateway` com o comando a seguir. Neste exemplo, a porta compartilhada do receptor é `8111`.

```
kubectl get cec cilium-gateway-my-gateway -o jsonpath={.spec.services[0].ports[0]}
```

3. Obtenha os endereços IP dos nós do seu cluster.

```
kubectl get nodes -o wide
```

```
NAME                   STATUS   ROLES    AGE   VERSION               INTERNAL-IP
 EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
mi-026d6a261e355fba7   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.150
 <none>        Ubuntu 22.04.5 LTS   5.15.0-142-generic   containerd://1.7.27
mi-082f73826a163626e   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.32
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-09183e8a3d755abf6   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.33
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0d78d815980ed202d   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.97
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0daa253999fe92daa   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.100
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

4. Acesse o serviço usando os endereços IP dos seus nós e a porta do receptor do recurso `cilium-gateway-my-gateway`. No exemplo abaixo, o endereço IP do nó usado é `10.80.146.32` e a porta do receptor é `8111`. Substitua-os pelos valores para o seu ambiente.

```
curl -s http://10.80.146.32:8111/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

Ingress

Devido a um problema de upstream do Cilium (#34028), o Ingress do Cilium no modo de rede do host requer o uso de `loadbalancerMode: shared`, o que cria um único serviço do tipo ClusterIP para todos os recursos do Ingress no cluster. Se você estiver usando uma porta privilegiada (inferior a 1023) para o recurso do Ingress, consulte a documentação do Cilium para obter instruções.

1. Crie um arquivo chamado `cilium-ingress-host-network.yaml` com o conteúdo a seguir.

```
ingressController:
  enabled: true
  loadbalancerMode: shared
  # This is a workaround for the upstream Cilium issue
  service:
    externalTrafficPolicy: null
    type: ClusterIP
  hostNetwork:
    enabled: true
    # ensure the port does not conflict with other services on the node
    sharedListenerPort: 8111
    # uncomment to restrict nodes where Envoy proxies run on the host network
    # nodes:
    #   matchLabels:
    #     role: ingress
```

2. Aplique a configuração do Ingress do Cilium em modo de rede do host ao seu cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  -f cilium-ingress-host-network.yaml
```

Se você não criou o recurso do Ingress, poderá fazê-lo aplicando a seguinte definição do Ingress ao seu cluster. A definição do Ingress abaixo usa a aplicação de amostra Istio Bookinfo descrita em the section called “Implementar o Ingress do Cilium”.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  namespace: default
spec:
  ingressClassName: cilium
  rules:
  - http:
      paths:
      - backend:
          service:
```

```
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

Você pode observar a configuração aplicada do Cilium Envoy com o comando a seguir.

```
kubectl get cec -n kube-system cilium-ingress -o yaml
```

Você pode obter a porta do receptor Envoy para o serviço `cilium-ingress` com o comando a seguir. Neste exemplo, a porta compartilhada do receptor é 8111.

```
kubectl get cec -n kube-system cilium-ingress -o
 jsonpath={.spec.services[0].ports[0]}
```

3. Obtenha os endereços IP dos nós do seu cluster.

```
kubectl get nodes -o wide
```

```
NAME                   STATUS   ROLES    AGE   VERSION               INTERNAL-IP
 EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
mi-026d6a261e355fba7   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.150
 <none>        Ubuntu 22.04.5 LTS   5.15.0-142-generic   containerd://1.7.27
mi-082f73826a163626e   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.32
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-09183e8a3d755abf6   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.33
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0d78d815980ed202d   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.97
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0daa253999fe92daa   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.100
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

4. Acesse o serviço usando os endereços IP dos seus nós e a `sharedListenerPort` para o recurso `cilium-ingress`. No exemplo abaixo, o endereço IP do nó usado é `10.80.146.32` e a porta do receptor é 8111. Substitua-os pelos valores para o seu ambiente.

```
curl -s http://10.80.146.32:8111/details/1 | jq
```

```
{
```

```
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

Configurar serviços do tipo LoadBalancer para nós híbridos

Este tópico descreve como configurar o balanceamento de carga de camada 4 (L4) para aplicações executadas no Amazon EKS Hybrid Nodes. Os serviços Kubernetes do tipo LoadBalancer são usados para expor aplicações do Kubernetes externos ao cluster. Serviços do tipo LoadBalancer são comumente usados com a infraestrutura física do balanceador de carga na nuvem ou no ambiente on-premises para atender ao tráfego da workload. Essa infraestrutura de balanceador de carga geralmente é provisionada com um controlador específico do ambiente.

A AWS oferece suporte para o AWS Network Load Balancer (NLB) e o Cilium para serviços do tipo LoadBalancer em execução no EKS Hybrid Nodes. A decisão de usar o NLB ou o Cilium é baseada na origem do tráfego da aplicação. Se o tráfego da aplicação for originário de uma região da AWS, a AWS recomenda usar o AWS NLB e o AWS Load Balancer Controller. Se o tráfego da aplicação for originado do ambiente on-premises ou de borda, a AWS recomenda usar os recursos de balanceamento de carga integrados do Cilium, que podem ser usados com ou sem a infraestrutura de balanceador de carga em seu ambiente.

Para balanceamento de carga de tráfego de aplicações de camada 7 (L7), consulte the section called “Configurar o Ingress”. Para obter informações gerais sobre balanceamento de carga com EKS, consulte Melhores práticas para balanceamento de carga.

AWS Network Load Balancer

Você pode usar o AWS Load Balancer Controller e o NLB com o tipo de destino `ip` para workloads executadas em nós híbridos. Ao usar o tipo de destino `ip`, o NLB encaminha o tráfego diretamente para os pods, ignorando o caminho de rede da camada de serviço. Para que o NLB alcance as metas de IP do pod em nós híbridos, os CIDRs de pods on-premises deverão ser roteáveis em sua rede on-premises roteáveis em sua rede on-premises roteáveis. Além disso, o AWS Load Balancer

Controller usa webhooks e requer comunicação direta do ambiente de gerenciamento EKS. Para obter mais informações, consulte the section called “Configurar webhooks”.

- Consulte the section called “Balanceamento de carga da rede” os requisitos de configuração de sub-rede, the section called “Instalação usando o Helm” e as melhores práticas para balanceamento de carga para obter informações adicionais sobre o AWS Network Load Balancer e o AWS Load Balancer Controller.
- Consulte Configurações de NLB do AWS Load Balancer Controller para ver as configurações que podem ser aplicadas a serviços do tipo `LoadBalancer` com o AWS Network Load Balancer.

Pré-requisitos

- Cilium instalado seguindo as instruções em the section called “Configurar CNI”.
- O ambiente de gerenciamento BGP do Cilium é habilitado seguindo as instruções em the section called “Configurar o BGP”. Se não quiser usar o BGP, você deverá usar um método alternativo para tornar seus CIDRs de pods on-premises roteáveis em sua rede on-premises. Consulte the section called “CIDRs de pods remotos roteáveis” para obter mais informações.
- O Helm foi instalado em seu ambiente de linha de comando. Consulte as Instruções de configuração do Helm.
- O eksctl foi instalado em seu ambiente de linha de comando. Consulte as Instruções de configuração do eksctl.

Procedimento

1. Baixe uma política do IAM para o AWS Load Balancer Controller que permita que ele faça chamadas para APIs da AWS em seu nome.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/refs/heads/main/docs/install/iam_policy.json
```

2. Crie uma política do IAM usando a política obtida por download na etapa anterior.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

3. Substitua os valores do nome do cluster (`CLUSTER_NAME`), região da AWS (`AWS_REGION`) e ID da conta da AWS (`AWS_ACCOUNT_ID`) pelas suas configurações e execute o comando a seguir.

```
eksctl create iamserviceaccount \
    --cluster=CLUSTER_NAME \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws:iam::AWS_ACCOUNT_ID:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
    --region AWS_REGION \
    --approve
```

4. Adicione o repositório de chart do Helm eks-charts. A AWS mantém esse repositório no GitHub.

```
helm repo add eks https://aws.github.io/eks-charts
```

5. Atualize o repositório local do Helm para confirmar que você tem os gráficos mais recentes.

```
helm repo update eks
```

6. Instale o AWS Load Balancer Controller. Substitua os valores do nome do cluster (`CLUSTER_NAME`), região da AWS (`AWS_REGION`), ID da VPC (`VPC_ID`) e a versão do chart do Helm do AWS Load Balancer Controller (`AWS_LBC_HELM_VERSION`) pelas suas configurações. Você pode encontrar a versão mais recente do chart do Helm executando `helm search repo eks/aws-load-balancer-controller --versions`. Se você estiver executando um cluster em modo misto com nós híbridos e nós na Nuvem AWS, você pode executar o AWS Load Balancer Controller nos nós da nuvem seguindo as instruções em the section called “AWS Load Balancer Controller”.

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --version AWS_LBC_HELM_VERSION \
  --set clusterName=CLUSTER_NAME \
  --set region=AWS_REGION \
  --set vpcId=VPC_ID \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

7. Verifique se o AWS Load Balancer Controller foi instalado com êxito.

```
kubectl get -n kube-system deployment aws-load-balancer-controller
```

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

8. Defina uma aplicação de amostra em um arquivo chamado `tcp-sample-app.yaml`. O exemplo abaixo usa uma implantação simples do NGINX com uma porta TCP.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tcp-sample-app
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

9. Aplique a implantação ao seu cluster.

```
kubectl apply -f tcp-sample-app.yaml
```

10. Defina um serviço do tipo LoadBalancer para a implantação em um arquivo chamado `tcp-sample-service.yaml`.

```
apiVersion: v1
kind: Service
metadata:
  name: tcp-sample-service
```

```
  namespace: default
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

11 Aplicar a configuração do serviço ao seu cluster.

```
kubectl apply -f tcp-sample-service.yaml
```

12 O provisionamento do NLB para o serviço pode levar alguns minutos. Depois que o NLB for provisionado, o serviço terá um endereço atribuído a ele que corresponde ao nome DNS da implantação do NLB.

```
kubectl get svc tcp-sample-service
```

```
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP
                                                PORT(S)        AGE
tcp-sample-service   LoadBalancer   172.16.115.212   k8s-default-tcpsampl-xxxxxxxxxx-
xxxxxxxxxxxxxxxx.elb.<region>.amazonaws.com   80:30396/TCP   8s
```

13 Acesse o serviço usando o endereço do NLB.

```
curl k8s-default-tcpsampl-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.<region>.amazonaws.com
```

Um exemplo de saída está abaixo.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

14Limpe os recursos que você criou.

```
kubectl delete -f tcp-sample-service.yaml
kubectl delete -f tcp-sample-app.yaml
```

Balanceamento de carga no cluster do Cilium

O Cilium pode ser usado como um balanceador de carga no cluster para workloads em execução no EKS Hybrid Nodes, o que pode ser útil para ambientes que não têm infraestrutura de balanceador de carga. Os recursos de balanceamento de carga do Cilium são baseados em uma combinação de atributos do Cilium, incluindo a substituição do kube-proxy, o Gerenciador de endereços IP (IPAM) do balanceador de carga e o ambiente de gerenciamento BGP. As responsabilidades desses atributos estão detalhadas abaixo:

- Substituição do Cilium kube-proxy: manipula o tráfego do serviço de roteamento para pods de backend.
- IPAM do Cilium Load Balancer: gerencia endereços IP que podem ser atribuídos a serviços do tipo `LoadBalancer`.
- Ambiente de gerenciamento do Cilium BGP: anuncia endereços IP alocados para IPAM do balanceador de carga para a rede on-premises.

Se você não estiver usando o substituto kube-proxy do Cilium, ainda poderá usar o IPAM do balanceador de carga do Cilium e o ambiente de gerenciamento do BGP para alocar e atribuir endereços IP para serviços do tipo LoadBalancer. Se você não estiver usando o substituto kube-proxy do Cilium, o balanceamento de carga dos serviços para pods de backend é feito pelas regras kube-proxy e iptables por padrão no EKS.

Pré-requisitos

- O Cilium foi instalado seguindo as instruções the section called “Configurar CNI” com ou sem a substituição do kube-proxy ativada. A substituição do kube-proxy do Cilium requer a execução de um sistema operacional com um kernel Linux pelo menos tão recente quanto v4.19.57, v5.1.16 ou v5.2.0. Todas as versões recentes dos sistemas operacionais compatíveis para uso com nós híbridos atendem a esses critérios, com exceção do Red Hat Enterprise Linux (RHEL) 8.x.
- O ambiente de gerenciamento BGP do Cilium é habilitado seguindo as instruções em the section called “Configurar o BGP”. Se não quiser usar o BGP, você deverá usar um método alternativo

para tornar seus CIDRs de pods on-premises roteáveis em sua rede on-premises. Consulte the section called “CIDRs de pods remotos roteáveis” para obter mais informações.

- O Helm foi instalado em seu ambiente de linha de comando. Consulte as Instruções de configuração do Helm.

Procedimento

1. Crie um arquivo chamado `cilium-lbip-pool-loadbalancer.yaml` com um recurso `CiliumLoadBalancerIPPool` para configurar o intervalo de endereços IP do balanceador de carga para seus serviços do tipo LoadBalancer.
   - Substitua `LB_IP_CIDR` pelo intervalo de endereços IP a ser usado para os endereços IP do balanceador de carga. Para selecionar um único endereço IP, use um `/32` CIDR. Para obter mais informações, consulte Gerenciamento de endereços IP do LoadBalancer na documentação do Cilium.
   - O campo `serviceSelector` está configurado para corresponder ao nome do serviço que você criará em uma etapa subsequente. Com essa configuração, os IPs desse pool serão alocados somente aos serviços com o nome `tcp-sample-service`.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumLoadBalancerIPPool
metadata:
  name: tcp-service-pool
spec:
  blocks:
  - cidr: "LB_IP_CIDR"
  serviceSelector:
    matchLabels:
      io.kubernetes.service.name: tcp-sample-service
```

2. Aplique o recurso `CiliumLoadBalancerIPPool` ao seu cluster.

```
kubectl apply -f cilium-lbip-pool-loadbalancer.yaml
```

3. Confirme se há pelo menos um endereço IP disponível no pool.

```
kubectl get ciliumloadbalancerippools.cilium.io
```

```
NAME              DISABLED   CONFLICTING   IPS AVAILABLE   AGE
```

```
tcp-service-pool   false      False         1              24m
```

4. Crie um arquivo `cilium-bgp-advertisement-loadbalancer.yaml` com o nome de um recurso `CiliumBGPAdvertisement` para anunciar o endereço IP do balanceador de carga para o serviço que você criará na próxima etapa. Se você não estiver usando o Cilium BGP, poderá ignorar esta etapa. O endereço IP do balanceador de carga usado para seu serviço deve ser roteável em sua rede on-premises para que você possa consultar o serviço na etapa final.
   - O campo `advertisementType` está definido como `Service` e `service.addresses` está definido como `LoadBalancerIP` para anunciar somente o `LoadBalancerIP` para serviços do tipo `LoadBalancer`.
   - O campo `selector` está configurado para corresponder ao nome do serviço que você criará em uma etapa subsequente. Com essa configuração, somente o `LoadBalancerIP` para os serviços com o nome `tcp-sample-service` serão anunciados.

   ```
   apiVersion: cilium.io/v2alpha1
   kind: CiliumBGPAdvertisement
   metadata:
     name: bgp-advertisement-tcp-service
     labels:
       advertise: bgp
   spec:
     advertisements:
       - advertisementType: "Service"
         service:
           addresses:
             - LoadBalancerIP
         selector:
           matchLabels:
             io.kubernetes.service.name: tcp-sample-service
   ```

5. Aplique o recurso `CiliumBGPAdvertisement` ao seu cluster. Se você não estiver usando o Cilium BGP, poderá ignorar esta etapa.

   ```
   kubectl apply -f cilium-bgp-advertisement-loadbalancer.yaml
   ```

6. Defina uma aplicação de amostra em um arquivo chamado `tcp-sample-app.yaml`. O exemplo abaixo usa uma implantação simples do NGINX com uma porta TCP.

   ```
   apiVersion: apps/v1
   kind: Deployment
   ```

```
metadata:
  name: tcp-sample-app
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

7. Aplique a implantação ao seu cluster.

```
kubectl apply -f tcp-sample-app.yaml
```

8. Defina um serviço do tipo LoadBalancer para a implantação em um arquivo chamado `tcp-sample-service.yaml`.
   - Você pode solicitar um endereço IP específico do pool IP do balanceador de carga com a anotação `lbipam.cilium.io/ips` no objeto do serviço. Você pode remover essa anotação se não quiser solicitar um endereço IP específico para o serviço.
   - O campo de especificação `loadBalancerClass` é obrigatório para evitar que o provedor de Nuvem AWS herdado crie um Classic Load Balancer para o serviço. No exemplo abaixo, isso está configurado `io.cilium/bgp-control-plane` para usar o ambiente de gerenciamento BGP do Cilium como a classe do balanceador de carga. Como alternativa, esse campo pode ser configurado para que `io.cilium/l2-announcer` use o atributo L2 Announcements do Cilium (atualmente em versão beta e não compatível com a AWS).

```
apiVersion: v1
kind: Service
metadata:
  name: tcp-sample-service
  namespace: default
```

```
  annotations:
    lbipam.cilium.io/ips: "LB_IP_ADDRESS"
spec:
  loadBalancerClass: io.cilium/bgp-control-plane
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

9. Aplique o serviço ao seu cluster. O serviço será criado com um endereço IP externo que você poderá usar para acessar a aplicação.

```
kubectl apply -f tcp-sample-service.yaml
```

10. Verifique se o serviço foi criado com sucesso e tem um IP atribuído a ele a partir do `CiliumLoadBalancerIPPool` criado na etapa anterior.

```
kubectl get svc tcp-sample-service
```

```
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)
 AGE
tcp-sample-service   LoadBalancer   172.16.117.76   LB_IP_ADDRESS   80:31129/TCP
 14m
```

11. Se você estiver usando o Cilium no modo de substituição do kube-proxy, você pode confirmar que o Cilium está lidando com o balanceamento de carga do serviço executando o comando a seguir. Na saída abaixo, os endereços `10.86.2.x` são os endereços IP dos pods de backend para o serviço.

```
kubectl -n kube-system exec ds/cilium -- cilium-dbg service list
```

```
ID   Frontend              Service Type   Backend
...
41   LB_IP_ADDRESS:80/TCP   LoadBalancer   1 => 10.86.2.76:80/TCP (active)
                                           2 => 10.86.2.130:80/TCP (active)
                                           3 => 10.86.2.141:80/TCP (active)
```

12. Confirme se o Cilium está anunciando o endereço IP para a rede on-premises via BGP. No exemplo abaixo, há cinco nós híbridos, cada um anunciando o `LB_IP_ADDRESS` para o serviço de `tcp-sample-service` na rede on-premises.

```
Node                    VRouter      Prefix             NextHop   Age     Attrs
mi-026d6a261e355fba7   NODES_ASN
                   LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-082f73826a163626e   NODES_ASN
                   LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-09183e8a3d755abf6   NODES_ASN
                   LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-0d78d815980ed202d   NODES_ASN
                   LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-0daa253999fe92daa   NODES_ASN
                   LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
```

13. Acesse o serviço usando o endereço IP do balanceador de carga atribuído.

```
curl LB_IP_ADDRESS
```

Um exemplo de saída está abaixo.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

14. Limpe os recursos que você criou.

```
kubectl delete -f tcp-sample-service.yaml
kubectl delete -f tcp-sample-app.yaml
kubectl delete -f cilium-lb-ip-pool.yaml
kubectl delete -f cilium-bgp-advertisement.yaml
```

Configurar políticas de rede do Kubernetes para nós híbridos

A AWS oferece suporte para políticas de rede do Kubernetes (camada 3/camada 4) para tráfego de entrada e saída de pod ao usar o Cilium como a CNI com o EKS Hybrid Nodes. Se você estiver executando clusters EKS com nós na Nuvem AWS, a AWS oferece suporte às políticas de rede CNI da Amazon VPC para Kubernetes.

Este tópico aborda como configurar as políticas de rede do Cilium e do Kubernetes com o EKS Hybrid Nodes. Para obter informações detalhadas sobre as políticas de rede do Kubernetes, consulte Políticas de rede do Kubernetes na documentação do Kubernetes.

Configure as políticas de rede

Considerações

- A AWS oferece suporte para as políticas de rede upstream do Kubernetes e a especificação para entrada e saída de pods. No momento, a AWS não oferce suporte para `CiliumNetworkPolicy` ou `CiliumClusterwideNetworkPolicy`.
- O valor `policyEnforcementMode` do Helm pode ser usado para controlar o comportamento padrão de aplicação da política do Cilium. O comportamento padrão permite todo o tráfego de entrada e saída. Quando um endpoint é selecionado por uma política de rede, ele passa para um estado de negação padrão, em que somente o tráfego explicitamente permitido é permitido. Consulte a documentação do Cilium para obter mais informações sobre o modo de política padrão</img> e os modos de aplicação de políticas.
- Caso esteja mudando `policyEnforcementMode` para uma instalação existente do Cilium, você deverá reiniciar o DaemonSet do atendente do Cilium para aplicar o novo modo de imposição de políticas.
- Use `namespaceSelector` e `podSelector` para permitir ou negar tráfego de/para namespaces e pods com rótulos correspondentes. O `namespaceSelector` e `podSelector` podem ser usados com `matchLabels` ou `matchExpressions` para selecionar namespaces e pods com base em seus rótulos.
- Use `ingress.ports` e `egress.ports` para permitir ou negar tráfego de/para portas e protocolos.
- O campo `ipBlock` não pode ser usado para permitir ou negar seletivamente o tráfego de/para endereços IP do pod (#9209). O uso de seletores `ipBlock` para IPs de nós é um atributo beta do Cilium e não é compatível com a AWS.

- Consulte o recurso NetworkPolicy na documentação do Kubernetes para obter informações sobre os campos disponíveis para as políticas de rede do Kubernetes.

Pré-requisitos

- Cilium instalado seguindo as instruções em the section called “Configurar CNI”.
- O Helm foi instalado em seu ambiente de linha de comando. Consulte as Instruções de configuração do Helm.

Procedimento

O procedimento a seguir configura políticas de rede para uma aplicação de microsserviços de exemplo, de forma que os componentes só possam se comunicar com outros componentes necessários para o funcionamento da aplicação. O procedimento usa a aplicação de microsserviços de amostra Istio Bookinfo.

A aplicação Bookinfo consiste em quatro microsserviços separados com os seguintes relacionamentos:

- productpage. O microsserviço productpage chama os microsserviços de detalhes e resenhas para preencher a página.
- detalhes. O microsserviço de detalhes contém informações sobre o livro.
- análises. O microsserviço de resenhas contém resenhas de livros. Também chama o microsserviço de classificações.
- classificações. O microsserviço de classificações contém informações de classificação de livros que acompanham uma resenha de livro.

1. Crie as aplicações de exemplo.

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
```

2. Confirme se a aplicação estiver sendo executado com êxito e anote o endereço IP do pod do microsserviço productpage. Você usará esse endereço IP do pod para fazer uma consulta sobre cada microsserviço nas etapas subsequentes.

```
kubectl get pods -o wide
```

```
NAME                              READY   STATUS    RESTARTS   AGE   IP
 NODE
details-v1-766844796b-9wff2       1/1     Running   0          7s    10.86.3.7
 mi-0daa253999fe92daa
productpage-v1-54bb874995-lwfgg   1/1     Running   0          7s    10.86.2.193
 mi-082f73826a163626e
ratings-v1-5dc79b6bcd-59njm       1/1     Running   0          7s    10.86.2.232
 mi-082f73826a163626e
reviews-v1-598b896c9d-p2289       1/1     Running   0          7s    10.86.2.47
 mi-026d6a261e355fba7
reviews-v2-556d6457d-djktc        1/1     Running   0          7s    10.86.3.58
 mi-0daa253999fe92daa
reviews-v3-564544b4d6-g8hh4       1/1     Running   0          7s    10.86.2.69
 mi-09183e8a3d755abf6
```

3. Crie um pod que será usado para testar as políticas de rede. Observe que o pod é criado no namespace `default` com o rótulo `access: true`.

```
kubectl run curl-pod --image=curlimages/curl -i --tty --labels=access=true --
namespace=default --overrides='{"spec": { "nodeSelector": {"eks.amazonaws.com/
compute-type": "hybrid"}}}' -- /bin/sh
```

4. Teste o acesso ao microsserviço productpage. No exemplo abaixo, usamos o endereço IP do pod productpage (`10.86.2.193`) para consultar o microsserviço. Substitua isso pelo endereço IP do pod de productpage em seu ambiente.

```
curl -s http://10.86.2.193:9080/productpage | grep -o "<title>.*</title>"
```

```
<title>Simple Bookstore App</title>
```

5. Você pode sair do pod test curl digitando `exit` e reconectar-se ao pod executando o seguinte comando.

```
kubectl attach curl-pod -c curl-pod -i -t
```

6. Para demonstrar os efeitos das políticas de rede nas etapas a seguir, primeiro criamos uma política de rede que nega todo o tráfego dos microsserviços da BookInfo. Crie um arquivo chamado `network-policy-deny-bookinfo.yaml` que defina a política de negação de rede.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-bookinfo
  namespace: default
spec:
  podSelector:
    matchExpressions:
    - key: app
      operator: In
      values: ["productpage", "details", "reviews", "ratings"]
  policyTypes:
  - Ingress
  - Egress
```

7. Aplique a política de negação de rede ao seu cluster.

```
kubectl apply -f network-policy-default-deny-bookinfo.yaml
```

8. Teste o acesso à aplicação BookInfo. No exemplo abaixo, usamos o endereço IP do pod productpage (`10.86.2.193`) para consultar o microsserviço. Substitua isso pelo endereço IP do pod de productpage em seu ambiente.

```
curl http://10.86.2.193:9080/productpage --max-time 10
```

```
curl: (28) Connection timed out after 10001 milliseconds
```

9. Crie um arquivo chamado `network-policy-productpage.yaml` que defina a política de rede de productpage. A política estabelece as seguintes regras:
   - permite o tráfego de entrada de pods com o rótulo `access: true` (o pod curl criado na etapa anterior)
   - permite o tráfego TCP de saída na porta `9080` para os microsserviços de detalhes, avaliações e classificações
   - permite o tráfego TCP/UDP de saída na porta `53` para o CoreDNS, que é executado no namespace `kube-system`

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
```

```
  name: productpage-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: productpage
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          access: "true"
  egress:
  - to:
    - podSelector:
        matchExpressions:
        - key: app
          operator: In
          values: ["details", "reviews", "ratings"]
    ports:
    - port: 9080
      protocol: TCP
  - to:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: kube-system
      podSelector:
        matchLabels:
          k8s-app: kube-dns
    ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
```

10. Aplique a política de rede do productpage ao seu cluster.

```
kubectl apply -f network-policy-productpage.yaml
```

11. Conecte-se ao pod curl e teste o acesso à aplicação Bookinfo. O acesso ao microsserviço productpage agora é permitido, mas os outros microsserviços ainda estão negados porque

ainda estão sujeitos à política de rede de negação. Nos exemplos abaixo, usamos o endereço IP do pod de productpage (`10.86.2.193`) para consultar o microsserviço. Substitua isso pelo endereço IP do pod de productpage em seu ambiente.

```
kubectl attach curl-pod -c curl-pod -i -t
```

```
curl -s http://10.86.2.193:9080/productpage | grep -o "<title>.*</title>"
<title>Simple Bookstore App</title>
```

```
curl -s http://10.86.2.193:9080/api/v1/products/1
{"error": "Sorry, product details are currently unavailable for this book."}
```

```
curl -s http://10.86.2.193:9080/api/v1/products/1/reviews
{"error": "Sorry, product reviews are currently unavailable for this book."}
```

```
curl -s http://10.86.2.193:9080/api/v1/products/1/ratings
{"error": "Sorry, product ratings are currently unavailable for this book."}
```

12. Crie um arquivo chamado `network-policy-details.yaml` que defina os detalhes da política de rede. A política permite somente o tráfego de entrada do microsserviço productpage.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: details-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: details
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: productpage
```

13. Crie um arquivo chamado `network-policy-reviews.yaml` que defina a política de rede de avaliações. A política permite somente o tráfego de entrada do microsserviço productpage e somente o tráfego de saída para o microsserviço de classificações e o CoreDNS.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: reviews-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: reviews
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: productpage
  egress:
  - to:
    - podSelector:
        matchLabels:
          app: ratings
  - to:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: kube-system
      podSelector:
        matchLabels:
          k8s-app: kube-dns
    ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
```

14. Crie um arquivo chamado `network-policy-ratings.yaml` que defina a política de rede de classificações. A política permite somente o tráfego de entrada dos microsserviços productpage e avaliações.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ratings-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: ratings
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchExpressions:
        - key: app
          operator: In
          values: ["productpage", "reviews"]
```

15. Aplique as políticas de rede de detalhes, avaliações e classificações ao seu cluster.

```
kubectl apply -f network-policy-details.yaml
kubectl apply -f network-policy-reviews.yaml
kubectl apply -f network-policy-ratings.yaml
```

16. Conecte-se ao pod curl e teste o acesso à aplicação Bookinfo. Nos exemplos abaixo, usamos o endereço IP do pod de productpage (`10.86.2.193`) para consultar o microsserviço. Substitua isso pelo endereço IP do pod de productpage em seu ambiente.

```
kubectl attach curl-pod -c curl-pod -i -t
```

Teste o microsserviço de detalhes.

```
curl -s http://10.86.2.193:9080/api/v1/products/1
```

```
{"id": 1, "author": "William Shakespeare", "year": 1595, "type": "paperback",
 "pages": 200, "publisher": "PublisherA", "language": "English", "ISBN-10":
 "1234567890", "ISBN-13": "123-1234567890"}
```

Teste o microsserviço de avaliações.

```
curl -s http://10.86.2.193:9080/api/v1/products/1/reviews
```

```
{"id": "1", "podname": "reviews-v1-598b896c9d-p2289", "clustername": "null",
 "reviews": [{"reviewer": "Reviewer1", "text": "An extremely entertaining play
 by Shakespeare. The slapstick humour is refreshing!"}, {"reviewer": "Reviewer2",
 "text": "Absolutely fun and entertaining. The play lacks thematic depth when
 compared to other plays by Shakespeare."}]}
```

Teste o microsserviço de classificações.

```
curl -s http://10.86.2.193:9080/api/v1/products/1/ratings
```

```
{"id": 1, "ratings": {"Reviewer1": 5, "Reviewer2": 4}}
```

17 Limpe os recursos que você criou nesse processo.

```
kubectl delete -f network-policy-deny-bookinfo.yaml
kubectl delete -f network-policy-productpage.yaml
kubectl delete -f network-policy-details.yaml
kubectl delete -f network-policy-reviews.yaml
kubectl delete -f network-policy-ratings.yaml
kubectl delete -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
kubectl delete pod curl-pod
```

Conceitos de nós híbridos

Com o Amazon EKS Hybrid Nodes, você une máquinas físicas ou virtuais em execução em ambientes on-premises ou de borda aos clusters do Amazon EKS executados na Nuvem AWS. Essa abordagem traz muitos benefícios, mas também introduz novos conceitos e arquiteturas de rede para aqueles que estão familiarizados com a execução de clusters do Kubernetes em um único ambiente de rede.

As seções a seguir se aprofundam nos conceitos do Kubernetes e de rede para o EKS Hybrid Nodes e detalham como o tráfego flui pela arquitetura híbrida. Essas seções exigem que você esteja familiarizado com o conhecimento básico de rede do Kubernetes, como os conceitos de pods, nós, serviços, ambiente de gerenciamento do Kubernetes, kubelet e kube-proxy.

Recomendamos a leitura dessas páginas em ordem, começando com the section called “Conceitos de rede”, depois the section called “Conceitos do Kubernetes” e, finalmente, the section called “Fluxos de tráfego”.

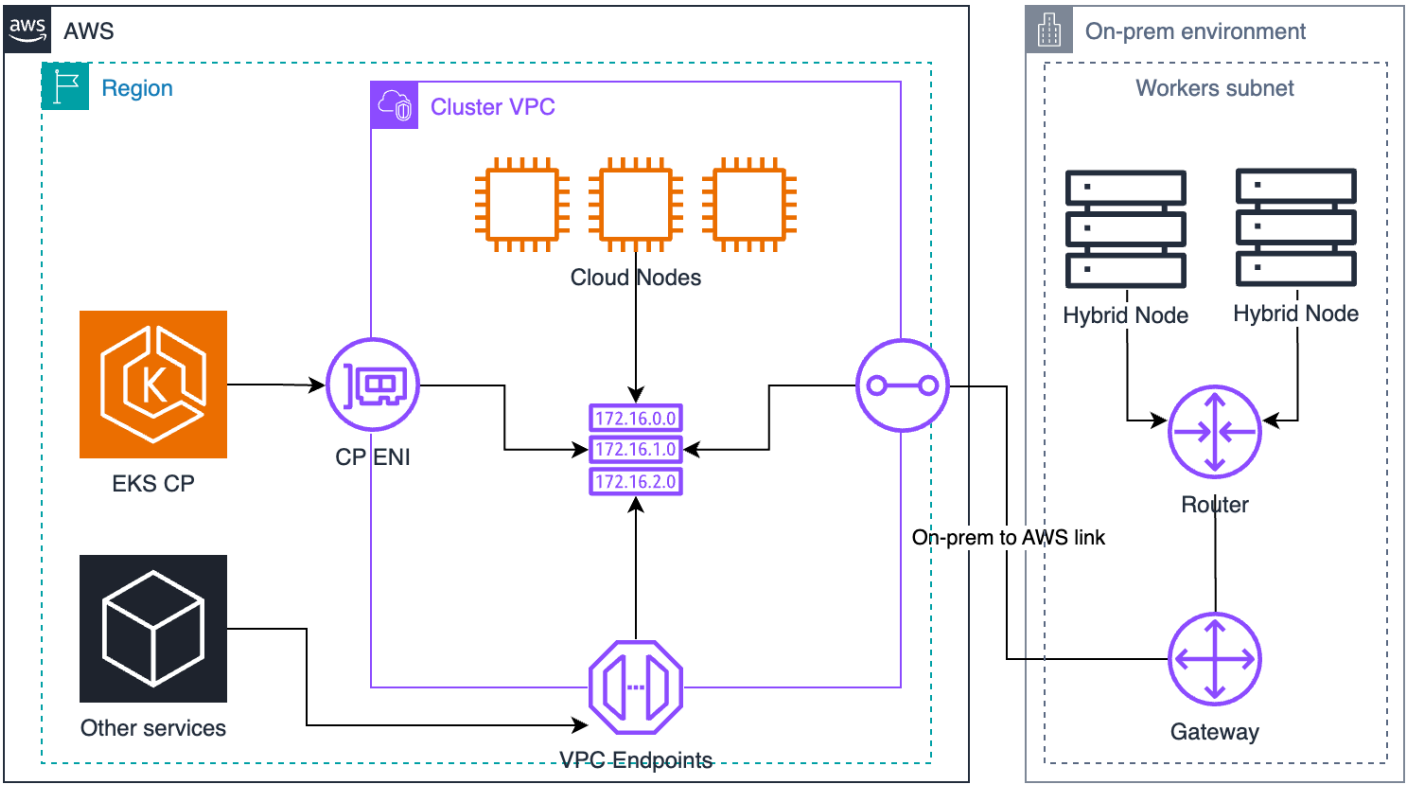
Tópicos

- Conceitos de rede para nós híbridos
- Conceitos do Kubernetes para nós híbridos
- Fluxos de tráfego de rede para nós híbridos

Conceitos de rede para nós híbridos

Esta seção detalha os principais conceitos de rede e as restrições que você deve considerar ao projetar sua topologia de rede para o EKS Hybrid Nodes.

Conceitos de rede para o EKS Hybrid Nodes



VPC como o hub de rede

Todo o tráfego que cruza os limites da nuvem passa pela sua VPC. Isso inclui o tráfego entre o ambiente de gerenciamento do EKS ou os pods em execução na AWS e os nós híbridos ou os

pods em execução neles. Você pode considerar a VPC do cluster como o hub de rede entre os nós híbridos e o restante do cluster. Essa arquitetura oferece controle total do tráfego e de seu roteamento, mas também faz com que você seja responsável por configurar corretamente rotas, grupos de segurança e firewalls para a VPC.

Ambiente de gerenciamento do EKS para a VPC

O ambiente de gerenciamento do EKS anexa interfaces de rede elástica (ENIs) à VPC. Essas ENIs processam o tráfego de e para o servidor de API do EKS. Você controla o posicionamento das ENIs do ambiente de gerenciamento do EKS ao configurar o cluster, pois o EKS anexa ENIs às sub-redes que você passa durante a criação do cluster.

O EKS associa grupos de segurança às ENIs que o EKS anexa às sub-redes. Esses grupos de segurança permitem o tráfego de e para o ambiente de gerenciamento do EKS por meio das ENIs. Isso é importante para o EKS Hybrid Nodes porque você deve permitir o tráfego dos nós híbridos e dos pods executados nele para as ENIs do ambiente de gerenciamento do EKS.

Redes de nós remotos

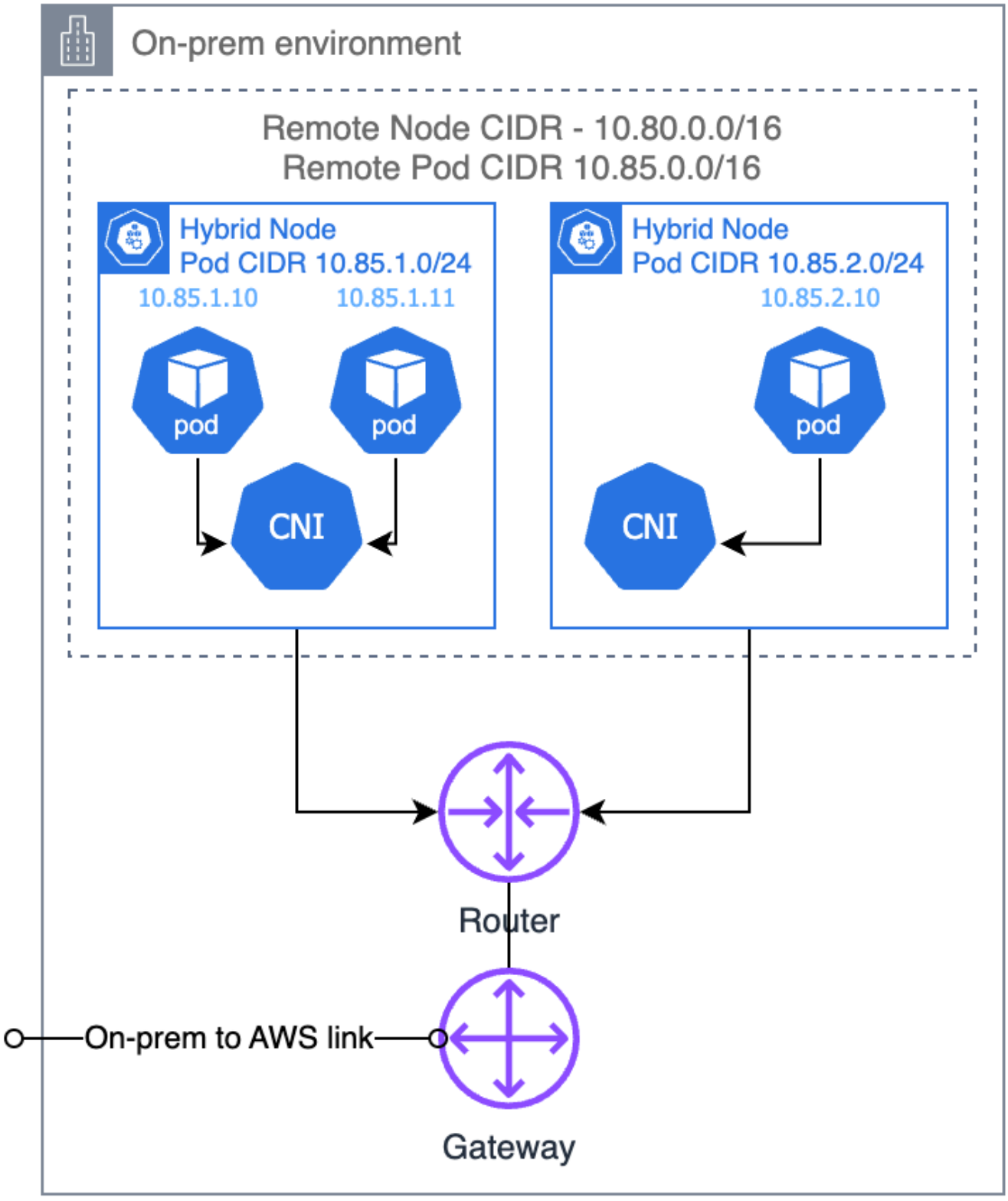
As redes de nós remotos, especificamente os CIDRs de nós remotos, são os intervalos de IPs atribuídos às máquinas que você usa como nós híbridos. Quando você provisiona nós híbridos, eles residem em seu data center on-premises ou local da borda, que é um domínio de rede diferente do ambiente de gerenciamento do EKS e da VPC. Cada nó híbrido tem um endereço IP, ou endereços, de um CIDR de nó remoto que é distinto das sub-redes na VPC.

Você configura o cluster do EKS com esses CIDRs de nós remotos para que o EKS saiba rotear todo o tráfego destinado aos IPs dos nós híbridos por meio da VPC do cluster, como solicitações para a API do kubelet. As conexões com a API do `kubelet` são usadas nos comandos `kubectl attach`, `kubectl cp`, `kubectl exec`, `kubectl logs` e `kubectl port-forward`.

Redes de pods remotos

As redes de pods remotos são os intervalos de IPs atribuídos aos pods em execução nos nós híbridos. Geralmente, você configura a CNI com esses intervalos, e a funcionalidade de gerenciamento de endereços IP (IPAM) da CNI se encarrega de atribuir uma fatia desses intervalos a cada nó híbrido. Quando você cria um pod, a CNI atribui um IP ao pod da fatia alocada para o nó em que o pod foi programado.

Você configura o cluster do EKS com esses CIDRs de pods remotos para que o ambiente de gerenciamento do EKS saiba rotear todo o tráfego destinado aos pods em execução nos nós híbridos por meio da VPC do cluster, como a comunicação com webhooks.



On-premises para a VPC

A rede on-premises que você usa para nós híbridos deve ser roteada para a VPC que você usa para o cluster do EKS. Há várias opções de conectividade de rede para a Amazon VPC disponíveis para conectar a rede on-premises a uma VPC. Você também pode usar sua própria solução de VPN.

É importante que você configure o roteamento corretamente no lado da Nuvem AWS na VPC e na rede on-premises para que ambas as redes roteiem o tráfego certo por meio da conexão das duas redes.

Na VPC, todo o tráfego que vai para o nó remoto e para as redes de pods remotos deve ser roteado pela conexão com a rede on-premises (denominada “gateway”). Se algumas das sub-redes tiverem tabelas de rotas diferentes, você deverá configurar cada tabela de rotas com as rotas dos nós híbridos e os pods em execução neles. Isso vale para as sub-redes às quais as ENIs do ambiente de gerenciamento do EKS estão anexadas e para as sub-redes que contêm nós ou pods do EC2 que precisam se comunicar com nós híbridos.

Na rede on-premises, você deve configurar a rede para permitir tráfego de e para a VPC do cluster do EKS e os outros serviços da AWS necessários para nós híbridos. O tráfego do cluster do EKS atravessa o gateway em ambas as direções.

Restrições de rede

Rede totalmente roteada

A principal restrição é que o ambiente de gerenciamento do EKS e todos os nós, da nuvem ou híbridos, precisam formar uma rede totalmente roteada. Isso significa que todos os nós devem ser capazes de acessar uns aos outros na camada três, por endereço IP.

O ambiente de gerenciamento do EKS e os nós da nuvem já podem ser acessados uns nos outros porque estão em uma rede simples (a VPC). Os nós híbridos, no entanto, estão em outro domínio de rede. É por isso que você precisa configurar um roteamento adicional na VPC e na rede on-premises para rotear o tráfego entre os nós híbridos e o restante do cluster. Se os nós híbridos puderem ser acessados entre si e pela VPC, os nós híbridos poderão estar em uma única rede simples ou em várias redes segmentadas.

CIDRs de pods remotos roteáveis

Para que o ambiente de gerenciamento do EKS se comunique com os pods executados em nós híbridos (por exemplo, webhooks ou o servidor de métricas), ou para que os pods executados em nós da nuvem se comuniquem com pods executados nos nós híbridos (comunicação leste-oeste de

workload), o CIDR de pod remoto deve ser roteável da VPC. Isso significa que a VPC deve ser capaz de rotear o tráfego para os CIDRs de pods por meio do gateway para sua rede on-premises, e que a rede on-premises deve ser capaz de rotear o tráfego de um pod para o nó correto.

É importante observar a distinção entre os requisitos de roteamento de pods na VPC e on-premises. A VPC só precisa saber que qualquer tráfego que vá para um pod remoto deve passar pelo gateway. Caso tenha apenas um CIDR de pod remoto, você precisará de apenas uma rota.

Esse requisito é válido para todos os saltos na rede on-premises até o roteador local na mesma sub-rede dos nós híbridos. Esse é o único roteador que precisa estar ciente da fatia do CIDR do pod atribuída a cada nó, garantindo que o tráfego de um determinado pod seja entregue ao nó em que o pod foi programado.

Você pode optar por propagar essas rotas para os CIDRs de pods on-premises do seu roteador local para as tabelas de rotas da VPC, mas isso não é necessário. Se seus CIDRs de pods on-premises mudarem com frequência e suas tabelas de rotas da VPC precisarem ser atualizadas para refletir os CIDRs de pods em mudança, recomendamos que você propague os CIDRs de pods on-premises para as tabelas de rotas de VPC, mas isso é incomum.

Observe que a restrição para tornar seus CIDRs de pods on-premises roteáveis é opcional. Caso você não precise executar webhooks nos nós híbridos ou fazer com que os pods em nós da nuvem se comuniquem com os pods nos nós híbridos, não será necessário configurar o roteamento para os CIDRs de pods na rede on-premises.

Por que os CIDRs de pods on-premises precisam ser roteáveis com nós híbridos?

Ao usar o EKS com a CNI da VPC para os nós da nuvem, a CNI da VPC atribui IPs diretamente da VPC aos pods. Isso significa que não há necessidade de nenhum roteamento especial, pois tanto os pods da nuvem quanto o ambiente de gerenciamento do EKS podem acessar diretamente os IPs dos pods.

Ao serem executados on-premises (e com outras CNIs na nuvem), os pods normalmente são executados em uma rede de sobreposição isolada, e a CNI se encarrega de fornecer tráfego entre os pods. Isso geralmente é feito por meio do encapsulamento: a CNI converte o tráfego pod-to-pod em tráfego node-to-node, cuidando do encapsulamento e do desencapsulamento em ambas as extremidades. Dessa forma, não há necessidade de configuração extra nos nós e nos roteadores.

A rede com nós híbridos é exclusiva porque apresenta uma combinação de ambas as topologias: o ambiente de gerenciamento do EKS e os nós da nuvem (com a CNI da VPC) esperam uma rede

simples incluindo nós e pods, enquanto os pods executados em nós híbridos estão em uma rede de sobreposição usando VXLAN para encapsulamento (por padrão, no Cilium). Os pods executados em nós híbridos podem acessar o ambiente de gerenciamento do EKS e os pods executados em nós da nuvem, supondo que a rede on-premises possa ser roteada para a VPC. No entanto, sem o roteamento dos CIDRs de pods na rede on-premises, qualquer tráfego que retorne para um IP de pod on-premises será, por fim, descartado se a rede não souber como acessar a rede de sobreposição e como rotear para os nós corretos.

Conceitos do Kubernetes para nós híbridos

Esta página detalha os principais conceitos do Kubernetes que fundamentam a arquitetura do sistema do EKS Hybrid Nodes.

Ambiente de gerenciamento do EKS na VPC

Os IPs das ENIs do ambiente de gerenciamento do EKS são armazenados no objeto de `Endpoints` do `kubernetes` no namespace `default`. Quando o EKS cria novas ENIs ou remove as mais antigas, o EKS atualiza esse objeto para que a lista de IPs esteja sempre atualizada.

Você pode usar esses endpoints por meio do serviço do `kubernetes`, e também no namespace `default`. Esse serviço, do tipo `ClusterIP`, sempre tem atribuído o primeiro IP do CIDR de serviço do cluster. Por exemplo, para o CIDR do serviço `172.16.0.0/16`, o IP do serviço será `172.16.0.1`.

Geralmente, é assim que os pods (independentemente de serem executados na nuvem ou em nós híbridos) acessam o servidor de API do Kubernetes para EKS. Os pods usam o IP do serviço como o IP de destino, que é convertido para os IPs reais de uma das ENIs do ambiente de gerenciamento do EKS. A principal exceção é o `kube-proxy`, pois ele configura a conversão.

Endpoint do servidor de API do EKS

O IP do serviço do `kubernetes` não é a única maneira de acessar o servidor de API do EKS. O EKS também criará um nome DNS do Route 53 quando você criar o cluster. Este é o campo do `endpoint` do seu cluster do EKS ao chamar a ação da API `DescribeCluster` do EKS.

```
{
    "cluster": {
        "endpoint": "https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com",
```

```
        "name": "my-cluster",
        "status": "ACTIVE"
    }
}
```

Em um acesso ao endpoint público ou em um cluster de acesso ao endpoint público e privado, seus nós híbridos resolverão esse nome DNS para um IP público por padrão, roteável pela internet. Em um cluster de acesso de endpoint privado, o nome DNS é resolvido para os IPs privados das ENIs do ambiente de gerenciamento do EKS.

É assim que o `kubelet` e o `kube-proxy` acessam o servidor de API do Kubernetes. Se você quiser que todo o tráfego do cluster do Kubernetes flua pela VPC, você precisa configurar o cluster no modo de acesso privado ou modificar o servidor DNS on-premises para resolver o endpoint do cluster do EKS para os IPs privados das ENIs do ambiente de gerenciamento do EKS.

Endpoint do **`kubelet`**

O `kubelet` expõe vários endpoints REST, permitindo que outras partes do sistema interajam e coletem informações de cada nó. Na maioria dos clusters, a maior parte do tráfego para o servidor do `kubelet` vem do ambiente de gerenciamento, mas determinados agentes de monitoramento também podem interagir com ele.

Por meio dessa interface, o `kubelet` processa várias solicitações: busca de logs (`kubectl logs`), execução de comandos dentro de contêineres (`kubectl exec`) e tráfego de encaminhamento de portas (`kubectl port-forward`). Cada uma dessas solicitações interage com o runtime do contêiner subjacente por meio do `kubelet`, parecendo perfeita para administradores e desenvolvedores de clusters.

O consumidor mais comum dessa API é o servidor de API do Kubernetes. Quando você usa qualquer um dos comandos do `kubectl` mencionados anteriormente, o `kubectl` faz uma solicitação de API ao servidor de API, que então chama a API do `kubelet` do nó em que o pod está sendo executado. Este é o principal motivo pelo qual o IP do nó precisa estar acessível no ambiente de gerenciamento do EKS e por que, mesmo que seus pods estejam em execução, você não conseguirá acessar seus logs ou `exec` se a rota do nó estiver configurada incorretamente.

IPs de nós

Quando o ambiente de gerenciamento do EKS se comunica com um nó, ele usa um dos endereços informados no status do objeto do `Node` (`status.addresses`).

Com os nós de nuvem do EKS, é comum que o kubelet informe o IP privado da instância do EC2 como um `InternalIP` durante o registro do nó. Esse IP é então validado pelo Cloud Controller Manager (CCM), certificando-se de que ele pertença à instância do EC2. Além disso, o CCM normalmente adiciona os IPs públicos (como `ExternalIP`) e os nomes DNS (`InternalDNS` e `ExternalDNS`) da instância ao status do nó.

No entanto, não há um CCM para nós híbridos. Quando você registra um nó híbrido com a CLI do EKS Hybrid Nodes (`nodeadm`), ele configura o kubelet para informar o IP da máquina diretamente no status do nó, sem o CCM.

```
apiVersion: v1
kind: Node
metadata:
  name: my-node-1
spec:
  providerID: eks-hybrid:///us-west-2/my-cluster/my-node-1
status:
  addresses:
  - address: 10.1.1.236
    type: InternalIP
  - address: my-node-1
    type: Hostname
```

Caso a máquina tenha vários IPs, o kubelet selecionará um deles seguindo sua própria lógica. Você pode controlar o IP selecionado com o sinalizador `--node-ip`, que pode ser passado no config `nodeadm` em `spec.kubelet.flags`. Somente o IP informado no objeto do `Node` precisa de uma rota da VPC. As máquinas podem ter outros IPs que não podem ser acessados pela nuvem.

`kube-proxy`

O `kube-proxy` é responsável por implementar a abstração do serviço na camada de rede de cada nó. Ele atua como um proxy de rede e um balanceador de carga para o tráfego destinado aos serviços do Kubernetes. Ao observar continuamente o servidor de API do Kubernetes em busca de alterações relacionadas a serviços e endpoints, o `kube-proxy` atualiza dinamicamente as regras de rede do host subjacente para garantir que o tráfego seja direcionado adequadamente.

No modo `iptables`, o `kube-proxy` programa várias cadeias de `netfilter` para processar o tráfego do serviço. As regras formam a seguinte hierarquia:

1. Cadeia KUBE-SERVICES: o ponto de entrada para todo o tráfego de serviços. Ele tem regras que correspondem a cada porta e `ClusterIP` do serviço.

2. Cadeias KUBE-SVC-XXX: cadeias específicas de serviços que têm regras de balanceamento de carga para cada serviço.
3. Cadeias KUBE-SEP-XXX: cadeias específicas de endpoints que têm as regras reais de DNAT.

Vamos examinar o que acontece com um serviço de `test-server` no namespace `default`:
* ClusterIP do serviço: `172.16.31.14` * Porta de serviço: `80` * Pods de backup: `10.2.0.110`, `10.2.1.39` e `10.2.2.254`

Quando inspecionamos as regras de `iptables` (usando `iptables-save 0 grep -A10 KUBE-SERVICES`):

1. Na cadeia KUBE-SERVICES, encontramos uma regra correspondente ao serviço:

```
-A KUBE-SERVICES -d 172.16.31.14/32 -p tcp -m comment --comment "default/test-server
 cluster IP" -m tcp --dport 80 -j KUBE-SVC-XYZABC123456
```

   - Essa regra corresponde aos pacotes destinados a 172.16.31.14:80
   - O comentário indica para que serve essa regra: `default/test-server cluster IP`
   - Pacotes correspondentes vão para a cadeia `KUBE-SVC-XYZABC123456`
2. A cadeia KUBE-SVC-XYZABC123456 tem regras de balanceamento de carga baseadas em probabilidade:

```
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.33333333349 -j
 KUBE-SEP-POD1XYZABC
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.50000000000 -j
 KUBE-SEP-POD2XYZABC
-A KUBE-SVC-XYZABC123456 -j KUBE-SEP-POD3XYZABC
```

   - Primeira regra: 33,3% de chance de ir para `KUBE-SEP-POD1XYZABC`
   - Segunda regra: 50% de chance de o tráfego restante (33,3% do total) ir para `KUBE-SEP-POD2XYZABC`
   - Última regra: todo o tráfego restante (33,3% do total) vai para `KUBE-SEP-POD3XYZABC`
3. As cadeias KUBE-SEP-XXX individuais executam o DNAT (NAT de destino):

```
-A KUBE-SEP-POD1XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.0.110:80
-A KUBE-SEP-POD2XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.1.39:80
-A KUBE-SEP-POD3XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.2.254:80
```

- Essas regras de DNAT regravam o IP e a porta de destino para direcionar o tráfego para pods específicos.
- Cada regra processa cerca de 33,3% do tráfego, fornecendo um balanceamento de carga uniforme entre `10.2.0.110`, `10.2.1.39` e `10.2.2.254`.

Essa estrutura de cadeia de vários níveis permite que o `kube-proxy` implemente com eficiência o balanceamento e o redirecionamento da carga de serviço por meio da manipulação de pacotes no nível do kernel, sem exigir um processo de proxy no caminho dos dados.

Impacto nas operações do Kubernetes

Uma falha no `kube-proxy` em um nó impede que esse nó roteie o tráfego do serviço adequadamente, causando tempos limite ou falhas nas conexões dos pods que dependem dos serviços do cluster. Isso pode ser especialmente disruptivo quando um nó é registrado pela primeira vez. A CNI precisa se comunicar com o servidor de API do Kubernetes para obter informações, como o CIDR do pod do nó, antes de poder configurar qualquer rede de pods. Para fazer isso, ela usa o IP do serviço do `kubernetes`. No entanto, se o `kube-proxy` não pôde iniciar ou não conseguiu definir as regras corretas de `iptables`, as solicitações que irão para o IP do serviço do `kubernetes` não serão convertidas para os IPs reais das ENIs do ambiente de gerenciamento do EKS. Como consequência, a CNI entrará em um loop de falha e nenhum dos pods será executado corretamente.

Sabemos que os pods usam o IP do serviço do `kubernetes` para se comunicar com o servidor de API do Kubernetes, mas o `kube-proxy` precisa primeiro definir as regras de `iptables` para que isso funcione.

Como o `kube-proxy` se comunica com o servidor de API?

O `kube-proxy` deve ser configurado para usar os IPs reais do servidor de API do Kubernetes ou um nome DNS que os determine. No caso do EKS, ele configura o `kube-proxy` padrão para apontar para o nome DNS do Route 53 que o EKS cria quando você cria o cluster. Você pode ver esse valor no ConfigMap do `kube-proxy` no namespace do `kube-system`. O conteúdo desse ConfigMap é um `kubeconfig` que é injetado no pod do `kube-proxy`, portanto, localize o campo `clusters0.cluster.server`. Esse valor corresponderá ao campo do `endpoint` do cluster do EKS (ao chamar a API `DescribeCluster` do EKS).

```
apiVersion: v1
data:
```

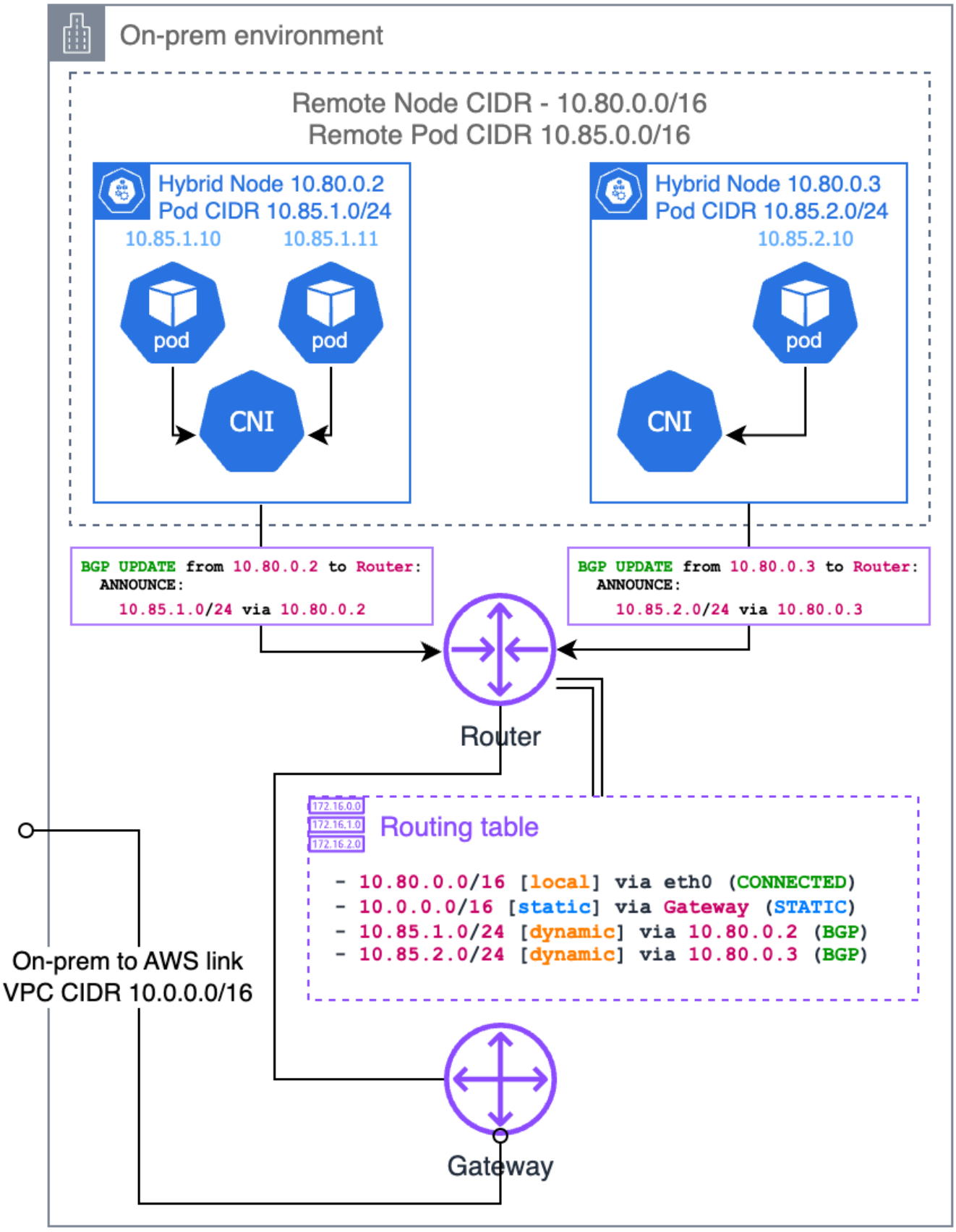
```
  kubeconfig: |-
    kind: Config
    apiVersion: v1
    clusters:
    - cluster:
        certificate-authority: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        server: https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com
      name: default
    contexts:
    - context:
        cluster: default
        namespace: default
        user: default
      name: default
    current-context: default
    users:
    - name: default
      user:
        tokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
kind: ConfigMap
metadata:
  name: kube-proxy
  namespace: kube-system
```

CIDRs de pods remotos roteáveis

A página the section called “Conceitos de rede” detalha os requisitos para executar webhooks em nós híbridos ou para que os pods executados em nós de nuvem se comuniquem com os pods executados em nós híbridos. O principal requisito é que o roteador on-premises precisa saber qual nó é responsável por um determinado IP de pod. Há várias maneiras de fazer isso, incluindo o Protocolo de Gateway da Borda (BGP), rotas estáticas e o proxy Address Resolution Protocol (ARP). Essas formas são abordadas nas próximas seções.

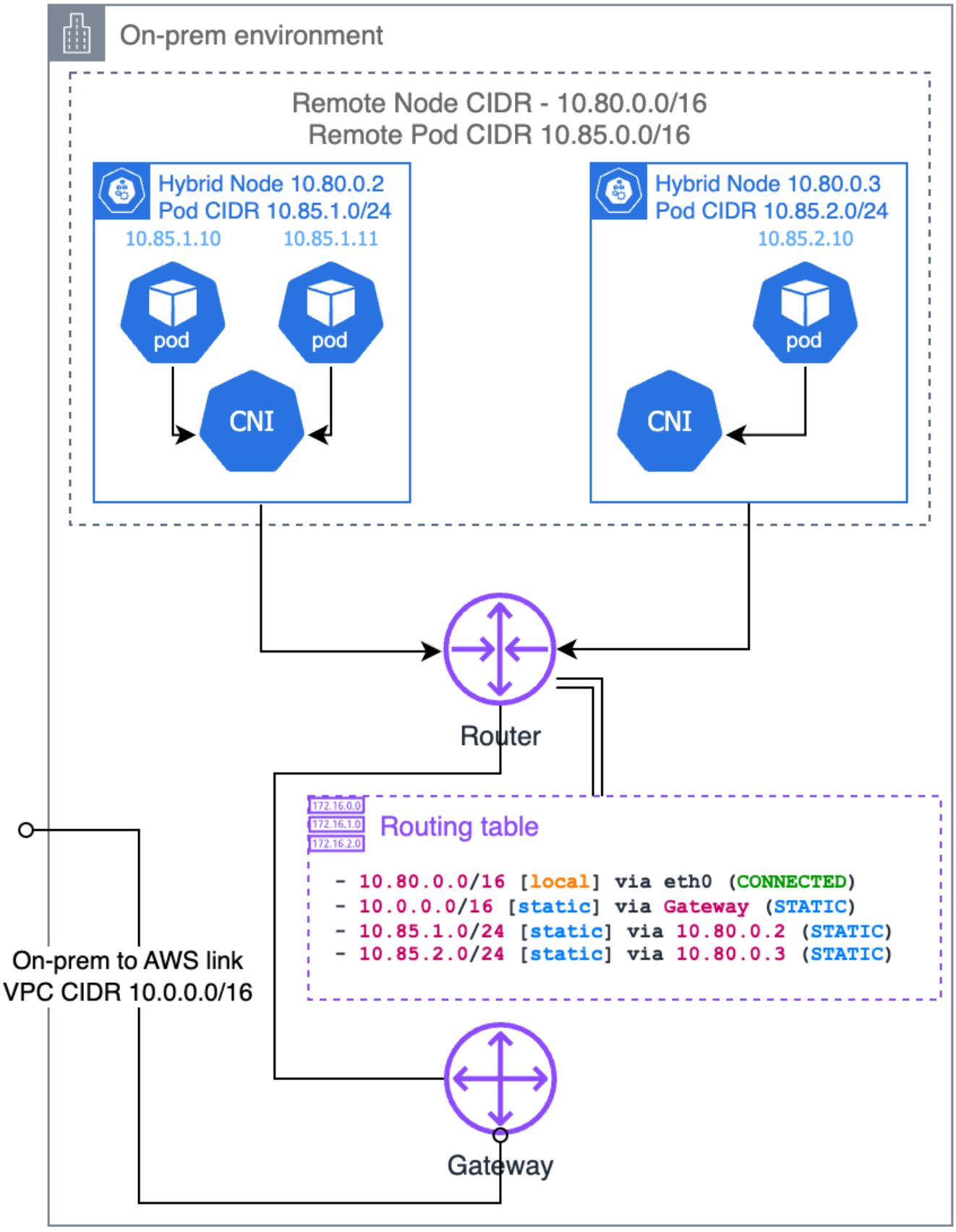
Protocolo de Gateway da Borda (BGP)

Se a CNI for compatível (como Cilium e Calico), você poderá usar o modo BGP da CNI para propagar rotas para os CIDRs de pods por nó dos nós para o roteador local. Ao usar o modo BGP da CNI, a CNI atua como um roteador virtual, então o roteador local acha que o CIDR do pod pertence a outra sub-rede e seu nó é o gateway para essa sub-rede.


On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node 10.80.0.2
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node 10.80.0.3
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
BGP UPDATE from 10.80.0.2 to Router:
ANNOUNCE:
10.85.1.0/24 via 10.80.0.2
BGP UPDATE from 10.80.0.3 to Router:
ANNOUNCE:
10.85.2.0/24 via 10.80.0.3
Router
172.16.0.0
172.16.1.0
172.16.2.0
Routing table
- 10.80.0.0/16 [local] via eth0 (CONNECTED)
- 10.0.0.0/16 [static] via Gateway (STATIC)
- 10.85.1.0/24 [dynamic] via 10.80.0.2 (BGP)
- 10.85.2.0/24 [dynamic] via 10.80.0.3 (BGP)
On-prem to AWS link
VPC CIDR 10.0.0.0/16
Gateway

Rotas estáticas

Como alternativa, é possível configurar rotas estáticas no roteador local. Esta é a maneira mais simples de rotear o CIDR do pod on-premises para a VPC, mas também é a mais propensa a erros e difícil de manter. Você precisa garantir que as rotas estejam sempre atualizadas com os nós existentes e seus CIDRs de pods atribuídos. Se o número de nós for pequeno e a infraestrutura for estática, esta é uma opção viável e elimina a necessidade de suporte do BGP em seu roteador. Se você optar por isso, recomendamos configurar a CNI com a parcela do CIDR do pod que você deseja atribuir a cada nó, em vez de deixar que o IPAM decida.


On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node 10.80.0.2
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node 10.80.0.3
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
Router
172.16.0.0
172.16.1.0
172.16.2.0
Routing table
- 10.80.0.0/16 [local] via eth0 (CONNECTED)
- 10.0.0.0/16 [static] via Gateway (STATIC)
- 10.85.1.0/24 [static] via 10.80.0.2 (STATIC)
- 10.85.2.0/24 [static] via 10.80.0.3 (STATIC)
On-prem to AWS link
VPC CIDR 10.0.0.0/16
Gateway

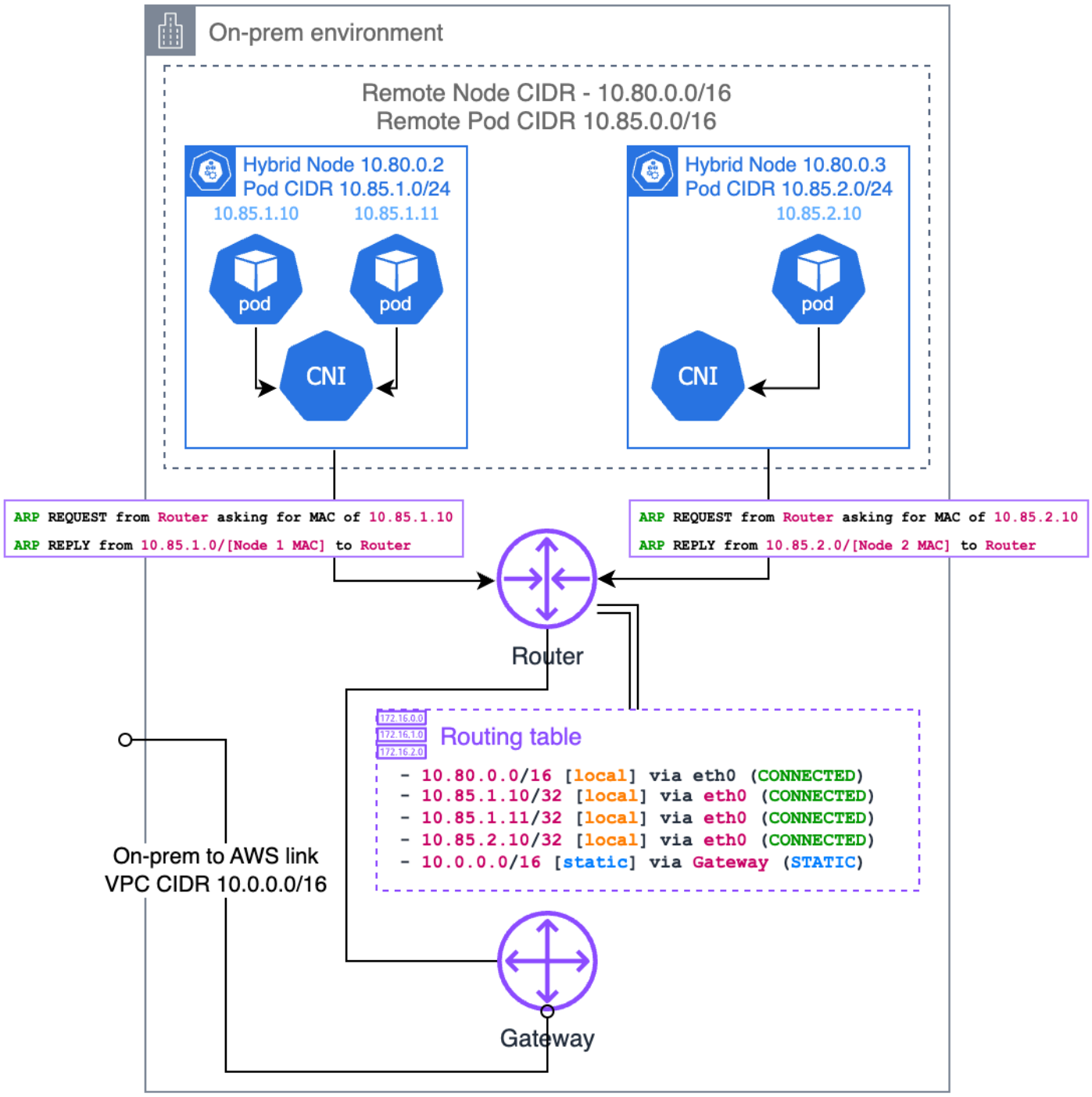
Proxy Address Resolution Protocol (ARP)

O proxy ARP é outra abordagem para tornar os IPs de pods on-premises roteáveis, particularmente útil quando os nós híbridos estão na mesma rede de camada 2 do roteador local. Com o proxy ARP habilitado, um nó responde às solicitações de ARP dos IPs de pods que ele hospeda, mesmo que esses IPs pertençam a outra sub-rede.

Quando um dispositivo em sua rede local tenta acessar um IP de pod, ele primeiro envia uma solicitação ao ARP perguntando: “quem tem este IP?”. O nó híbrido que hospeda esse pod responderá com seu próprio endereço MAC, dizendo: “eu posso processar o tráfego desse IP”. Isso cria um caminho direto entre os dispositivos na rede local e os pods sem precisar da configuração do roteador.

Para que isso funcione, a CNI deve ser compatível com a funcionalidade do proxy ARP. O Cilium tem suporte integrado para o proxy ARP que você pode habilitar por meio da configuração. A principal consideração é que o CIDR do pod não deve se sobrepor a nenhuma outra rede em seu ambiente, pois isso pode causar conflitos de roteamento.

Essa abordagem traz diversas vantagens: * Não é necessário configurar o roteador com BGP ou manter rotas estáticas * Funciona bem em ambientes em que você não tem controle sobre a configuração do roteador



Encapsulamento pod-to-pod

Em ambientes on-premises, as CNIs normalmente usam protocolos de encapsulamento para criar redes de sobreposição que possam operar na rede física sem a necessidade de reconfigurá-la. Esta seção explica como esse encapsulamento funciona. Observe que alguns dos detalhes podem variar dependendo da CNI que você está usando.

O encapsulamento compacta os pacotes de rede do pod original dentro de outro pacote de rede que pode ser roteado pela rede física subjacente. Isso permite que os pods se comuniquem entre os nós que executam a mesma CNI sem exigir que a rede física saiba como rotear esses CIDRs de pod.

O protocolo de encapsulamento mais comum usado com o Kubernetes é o Virtual Extensible LAN (VXLAN), embora outros (como o `Geneve`) também estejam disponíveis, dependendo da sua CNI.

Encapsulamento VXLAN

O VXLAN encapsula os quadros do Layer 2 Ethernet em pacotes do UDP. Quando um pod envia tráfego para outro pod em um nó diferente, a CNI faz o seguinte:

1. A CNI intercepta pacotes do pod A
2. A CNI compacta o pacote original em um cabeçalho VXLAN
3. Esse pacote compactado é então enviado pela pilha de rede regular do nó para o nó de destino
4. A CNI no nó de destino descompacta o pacote e o entrega ao pod B

Verifique o que acontece com a estrutura do pacote durante o encapsulamento VXLAN:

Pacote original de pod-to-pod:

```
+-----------------+---------------+-------------+-----------------+
| Ethernet Header | IP Header     | TCP/UDP     | Payload         |
| Src: Pod A MAC  | Src: Pod A IP | Src Port    |                 |
| Dst: Pod B MAC  | Dst: Pod B IP | Dst Port    |                 |
+-----------------+---------------+-------------+-----------------+
```

Após o encapsulamento VXLAN:

```
+-----------------+-------------+--------------+------------
+---------------------------+
| Outer Ethernet  | Outer IP    | Outer UDP    | VXLAN      | Original Pod-to-Pod
 |
| Src: Node A MAC | Src: Node A | Src: Random  | VNI: xx    | Packet (unchanged
 |
| Dst: Node B MAC | Dst: Node B | Dst: 4789    |            | from above)
 |
+-----------------+-------------+--------------+------------
+---------------------------+
```

O VXLAN Network Identifier (VNI) distingue entre diferentes redes de sobreposição.

Cenários de comunicação do pod

Pods no mesmo nó híbrido

Quando pods no mesmo nó híbrido se comunicam, normalmente nenhum encapsulamento é necessário. A CNI configura rotas locais que direcionam o tráfego entre os pods por meio das interfaces virtuais internas do nó:

```
Pod A -> veth0 -> node's bridge/routing table -> veth1 -> Pod B
```

O pacote nunca sai do nó e não requer encapsulamento.

Pods em diferentes nós híbridos

A comunicação entre pods em diferentes nós híbridos requer encapsulamento:

```
Pod A -> CNI -> [VXLAN encapsulation] -> Node A network -> router or gateway -> Node B
 network -> [VXLAN decapsulation] -> CNI -> Pod B
```

Isso permite que o tráfego do pod atravesse a infraestrutura física da rede sem exigir que a rede física saiba o roteamento IP do pod.

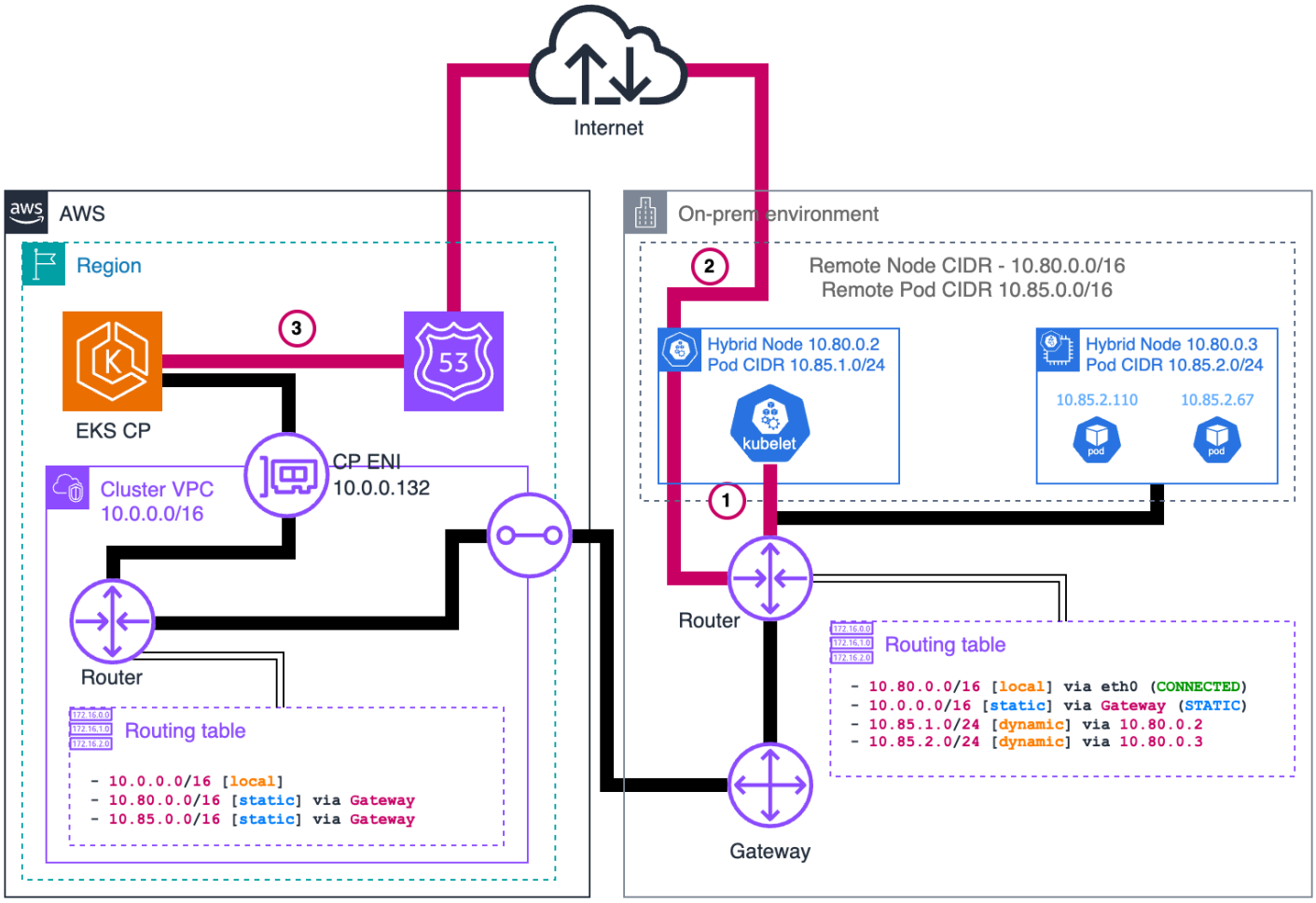
Fluxos de tráfego de rede para nós híbridos

Esta página detalha os fluxos de tráfego de rede para o EKS Hybrid Nodes com diagramas que mostram os caminhos de rede de ponta a ponta para os diferentes tipos de tráfego.

Os seguintes fluxos de tráfego são abrangidos:

- the section called “`kubelet` de nó híbrido para o ambiente de gerenciamento do EKS”
- the section called “Ambiente de gerenciamento do EKS para nó híbrido (servidor do `kubelet`)”
- the section called “Pods em execução em nós híbridos para o ambiente de gerenciamento do EKS”
- the section called “Ambiente de gerenciamento do EKS para pods executados em um nó híbrido (webhooks)”
- the section called “Pod-to-Pod em execução em nós híbridos”
- the section called “Pods em nós da nuvem para pods em nós híbridos (tráfego leste-oeste)”

`kubelet` de nó híbrido para o ambiente de gerenciamento do EKS



Solicitação

1**`kubelet`**. Inicia a solicitação

Quando o `kubelet` em um nó híbrido precisa se comunicar com o ambiente de gerenciamento do EKS (por exemplo, para informar o status do nó ou obter especificações do pod), ele usa o arquivo `kubeconfig` fornecido durante o registro do nó. Esse arquivo `kubeconfig` tem o URL do endpoint do servidor de API (o nome DNS do Route 53) em vez de endereços IP diretos.

O `kubelet` executa uma pesquisa de DNS para o endpoint (por exemplo, `https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-west-2.eks.amazonaws.com`). Em um cluster de acesso público, isso se resolve para um endereço IP público (por exemplo, `54.239.118.52`) que pertence ao serviço do EKS executado na AWS. O `kubelet` então cria uma solicitação HTTPS segura para esse endpoint. Veja como seria o pacote inicial:

```
+-------------------+--------------------+-----------------+
| IP Header         | TCP Header         | Payload         |
| Src: 10.80.0.2    | Src: 52390 (random) |                 |
| Dst: 54.239.118.52 | Dst: 443           |                 |
+-------------------+--------------------+-----------------+
```

2. Roteamento de roteador local

Como o IP de destino é um endereço IP público e não faz parte da rede local, o `kubelet` envia esse pacote para o gateway padrão (o roteador on-premises local). O roteador examina o IP de destino e determina que é um endereço IP público.

Para tráfego público, o roteador normalmente encaminha o pacote para um gateway da internet ou roteador de borda que processa o tráfego de saída para a internet. Isso é omitido no diagrama e dependerá de como sua rede on-premises está configurada. O pacote atravessa a infraestrutura de rede on-premises e, por fim, chega à rede do provedor de serviços de Internet.

3. Entrega para o ambiente de gerenciamento do EKS

O pacote trafega pela internet pública e pelas redes de trânsito até chegar à rede da AWS. A rede da AWS roteia o pacote para o endpoint do serviço do EKS na região apropriada. Quando o pacote chega ao serviço do EKS, ele é encaminhado para o ambiente de gerenciamento real do EKS do cluster.

Esse roteamento pela internet pública é diferente do caminho privado roteado por VPC que veremos em outros fluxos de tráfego. A principal diferença é que, ao usar o modo de acesso público, o tráfego do `kubelet` on-premises (embora não dos pods) para o ambiente de gerenciamento do EKS não passa pela VPC. Em vez disso, ele usa a infraestrutura global da internet.

Resposta

Depois que o ambiente de gerenciamento do EKS processa a solicitação do `kubelet`, ele envia uma resposta de volta:

3. O ambiente de gerenciamento do EKS envia uma resposta

O ambiente de gerenciamento do EKS cria um pacote de resposta. Esse pacote tem o IP público como origem e o IP do nó híbrido como destino:

```
+-------------------+--------------------+-----------------+
| IP Header         | TCP Header         | Payload         |
```

```
| Src: 54.239.118.52 | Src: 443           |                 |
| Dst: 10.80.0.2     | Dst: 52390         |                 |
+--------------------+--------------------+-----------------+
```

2. Roteamento da internet

O pacote de resposta viaja de volta pela internet, seguindo o caminho de roteamento determinado pelos provedores de serviços de Internet, até chegar ao roteador de borda da rede on-premises.

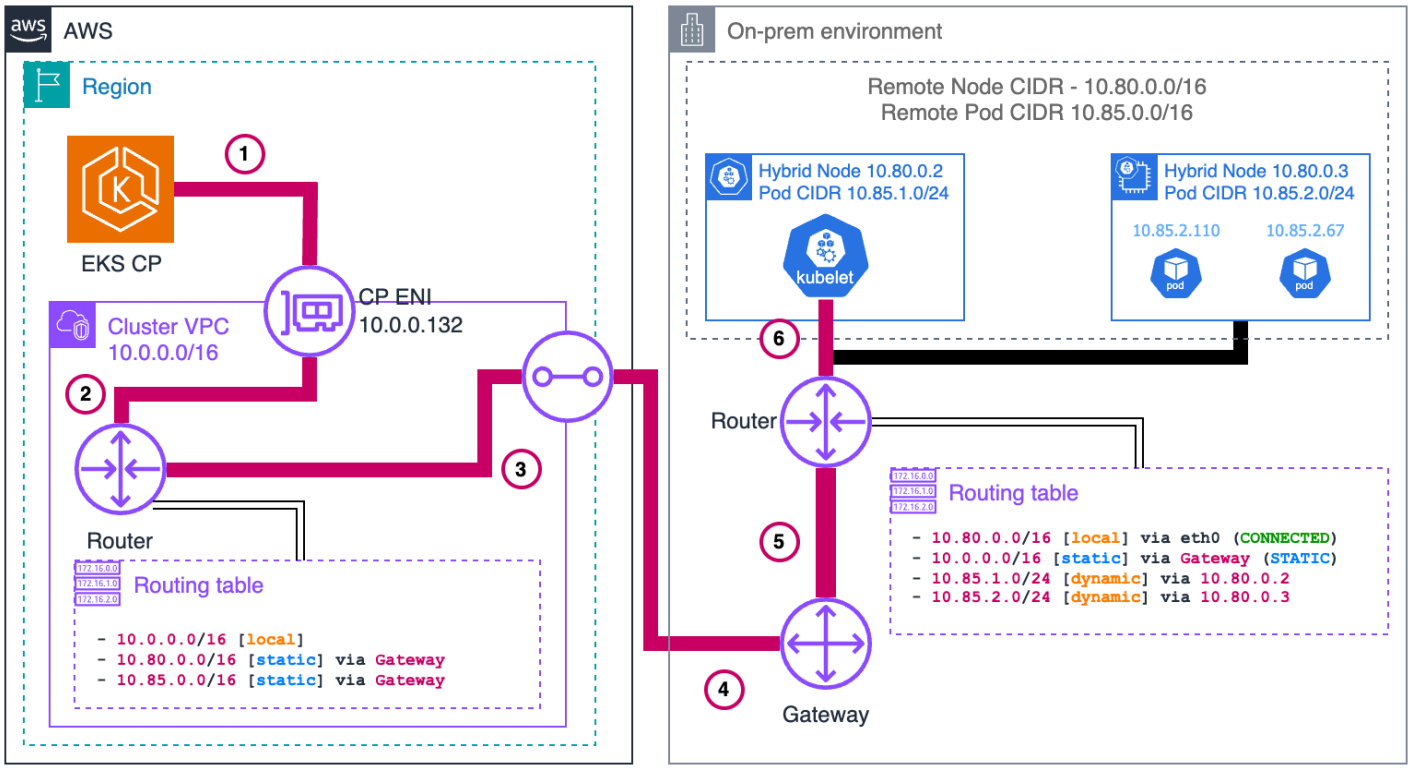
1. Entrega local

O roteador on-premises recebe o pacote e reconhece o IP de destino (`10.80.0.2`) como pertencente à rede local. Ele encaminha o pacote por meio da infraestrutura de rede local até chegar ao nó híbrido de destino, onde o `kubelet` recebe e processa a resposta.

`kube-proxy` de nó híbrido para o ambiente de gerenciamento do EKS

Se você habilitar o acesso ao endpoint público para o cluster, o tráfego de retorno usará a internet pública. Esse tráfego é proveniente de `kube-proxy` no nó híbrido e se destina ao ambiente de gerenciamento do EKS e segue o mesmo caminho do tráfego do `kubelet` até o ambiente de gerenciamento do EKS.

Ambiente de gerenciamento do EKS para nó híbrido (servidor do **`kubelet`**)



Solicitação

1. O servidor de API do EKS Kubernetes inicia a solicitação

O servidor de API do EKS Kubernetes recupera o endereço IP do nó (`10.80.0.2`) do status do objeto do nó. Em seguida, ele encaminha essa solicitação por meio da ENI na VPC, pois o IP de destino pertence ao CIDR do nó remoto configurado (`10.80.0.0/16`). Veja como seria o pacote inicial:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 67493 (random) |                 |
| Dst: 10.80.0.2  | Dst: 10250          |                 |
+-----------------+---------------------+-----------------+
```

2. Processamento de rede da VPC

O pacote sai da ENI e entra na camada de rede da VPC, onde é direcionado para o gateway da sub-rede para roteamento adicional.

3. Pesquisa da tabela de rotas da VPC

A tabela de rotas da VPC para a sub-rede que contém a ENI do ambiente de gerenciamento do EKS tem uma rota específica (a segunda no diagrama) para o CIDR do nó remoto. Com base nessa regra de roteamento, o pacote é direcionado para o gateway de VPC para on-premises.

4. Trânsito entre limites

O gateway transfere o pacote além do limite de nuvem por meio de sua conexão estabelecida (como Direct Connect ou VPN) para a rede on-premises.

5. Recepção de rede on-premises

O pacote chega ao roteador on-premises local que processa o tráfego da sub-rede em que os nós híbridos estão localizados.

6. Entrega final

O roteador local identifica que o endereço IP de destino (`10.80.0.2`) pertence à rede conectada diretamente e encaminha o pacote diretamente para o nó híbrido de destino, onde o `kubelet` recebe e processa a solicitação.

Resposta

Depois que o `kubelet` do nó híbrido processa a solicitação, ele envia de volta uma resposta seguindo o mesmo caminho no sentido inverso:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 10250          |                 |
| Dst: 10.0.0.132 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

6. **`kubelet`** Envia resposta

O `kubelet` no nó híbrido (`10.80.0.2`) cria um pacote de resposta com o IP de origem inicial como o destino. O destino não pertence à rede local, então é enviado para o gateway padrão do host, que é o roteador local.

5. Roteamento de roteador local

O roteador determina que o IP de destino (`10.0.0.132`) pertence a `10.0.0.0/16`, que tem uma rota apontando para o gateway que se conecta à AWS.

4. Retorno entre limites

O pacote viaja de volta pela mesma conexão on-premises para a VPC (como Direct Connect ou VPN), cruzando o limite da nuvem na direção inversa.

3. Roteamento da VPC

Quando o pacote chega à VPC, as tabelas de rotas identificam que o IP de destino pertence a um CIDR da VPC. O pacote é roteado dentro da VPC.

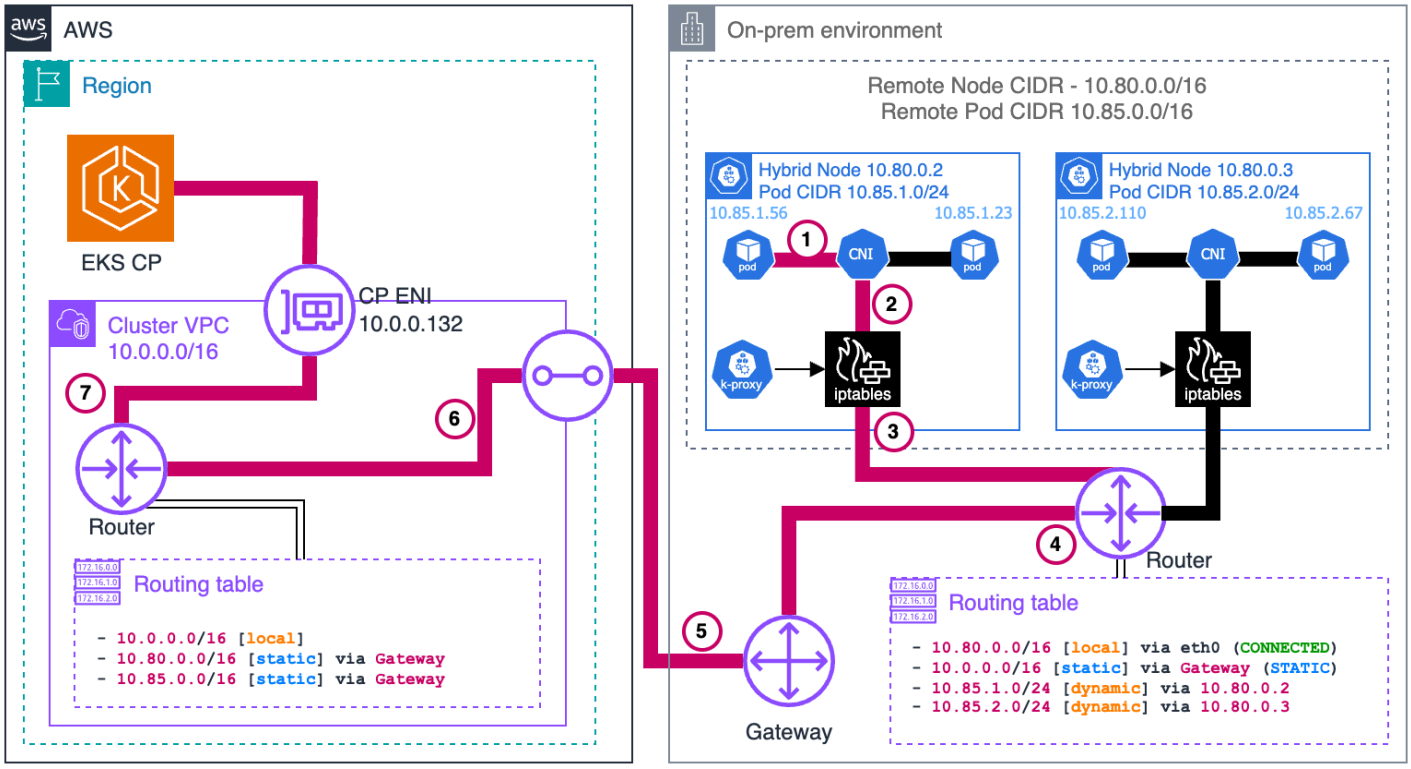
2. Entrega da rede da VPC

A camada de rede da VPC encaminha o pacote para a sub-rede com a ENI do ambiente de gerenciamento do EKS (`10.0.0.132`).

1. Recepção da ENI

O pacote chega à ENI do ambiente de gerenciamento do EKS anexado ao servidor de API do Kubernetes, completando a viagem de ida e volta.

Pods em execução em nós híbridos para o ambiente de gerenciamento do EKS



Sem CNI NAT

Solicitação

Os pods geralmente se comunicam com o servidor de API do Kubernetes por meio do serviço do `kubernetes`. O IP do serviço é o primeiro IP do CIDR de serviço do cluster. Essa convenção permite que os pods que precisam ser executados antes que o CoreDNS esteja disponível cheguem ao servidor de API, por exemplo, a CNI. As solicitações saem do pod com o IP do serviço como destino. Por exemplo, se o CIDR de serviço for `172.16.0.0/16`, o IP do serviço será `172.16.0.1`.

1. Pod inicia a solicitação

O pod envia uma solicitação para o IP de serviço (`172.16.0.1`) do `kubernetes` na porta do servidor de API (443) de uma porta de origem aleatória. O pacote seria assim:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. Processamento da CNI

A CNI detecta que o IP de destino não pertence a nenhum CIDR de pod que ele gerencia. Como o NAT de saída está desabilitado, a CNI passa o pacote para a pilha da rede do host sem modificá-lo.

3. Processamento de rede de nós

O pacote entra na pilha de rede do nó, onde os hooks `netfilter` acionam as regras de `iptables` definidas pelo kube-proxy. Várias regras se aplicam na seguinte ordem:

1. O pacote atinge primeiro a cadeia de `KUBE-SERVICES`, que contém regras que correspondem ao ClusterIP e à porta de cada serviço.
2. A regra de correspondência salta para a cadeia `KUBE-SVC-XXX` do serviço do `kubernetes` (pacotes destinados a `172.16.0.1:443`), que contém regras de balanceamento de carga.
3. A regra de balanceamento de carga seleciona aleatoriamente uma das cadeias `KUBE-SEP-XXX` para a ENI do ambiente de gerenciamento de IPs (`10.0.0.132` ou `10.0.1.23`).
4. A cadeia `KUBE-SEP-XXX` selecionada tem a regra real que altera o IP de destino do IP do serviço para o IP selecionado. Isso é chamado de conversão de endereços de rede de destino (DNAT).

Depois que essas regras são aplicadas, supondo que o IP da ENI do ambiente de gerenciamento do EKS selecionado seja `10.0.0.132`, o pacote fica assim:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

O nó encaminha o pacote para seu gateway padrão porque o IP de destino não está na rede local.

4. Roteamento de roteador local

O roteador local determina que o IP de destino (`10.0.0.132`) pertence ao CIDR da VPC (`10.0.0.0/16`) e o encaminha para o gateway conectado à AWS.

5. Trânsito entre limites

O pacote viaja pela conexão estabelecida (como Direct Connect ou VPN) além do limite da nuvem até a VPC.

6. Entrega da rede da VPC

A camada de rede da VPC roteia o pacote para a sub-rede correta em que a ENI do ambiente de gerenciamento do EKS (`10.0.0.132`) está localizado.

7. Recepção da ENI

O pacote chega à ENI do ambiente de gerenciamento do EKS anexado ao servidor de API do Kubernetes.

Resposta

Depois que o ambiente de gerenciamento do EKS processa a solicitação, ele envia uma resposta de volta ao pod:

7. Servidor de API envia resposta

O servidor de API do EKS Kubernetes cria um pacote de resposta com o IP de origem inicial como o destino. O pacote seria assim:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Como o IP de destino pertence ao CIDR do pod remoto configurado (`10.85.0.0/16`), ele o envia por meio da ENI na VPC com o roteador da sub-rede como o próximo salto.

6. Roteamento da VPC

A tabela de rotas da VPC contém uma entrada para o CIDR do pod remoto (`10.85.0.0/16`), direcionando esse tráfego para o gateway de VPC para on-premises.

5. Trânsito entre limites

O gateway transfere o pacote além do limite de nuvem por meio de sua conexão estabelecida (como Direct Connect ou VPN) para a rede on-premises.

4. Recepção de rede on-premises

O pacote chega ao seu roteador on-premises local.

3. Entrega ao nó

A tabela do roteador tem uma entrada para `10.85.1.0/24` com `10.80.0.2` como o próximo salto, entregando o pacote para o nosso nó.

2. Processamento de rede de nós

À medida que o pacote é processado pela pilha de rede do nó, `conntrack` (uma parte de `netfilter`) combina o pacote com a conexão estabelecida inicialmente pelo pod. Como o DNAT foi aplicado originalmente, `conntrack` reverte o DNAT reescrevendo o IP de origem do IP do ENI do plano de controle do EKS para o IP do serviço `kubernetes`:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1. Processamento da CNI

A CNI identifica que o IP de destino pertence a um pod em sua rede e entrega o pacote ao namespace correto da rede do pod.

Esse fluxo mostra por que os CIDRs de pods remotos devem ser roteáveis adequadamente da VPC até o nó específico que hospeda cada pod. Todo o caminho de retorno depende do roteamento adequado dos IPs do pod nas redes na nuvem e on-premises.

Com CNI NAT

Esse fluxo é muito semelhante ao fluxo sem CNI NAT, mas com uma diferença fundamental: a CNI aplica o NAT de origem (SNAT) ao pacote antes de enviá-lo para a pilha de rede do nó. Isso altera o IP de origem do pacote para o IP do nó, permitindo que o pacote seja roteado de volta para o nó sem exigir configuração adicional de roteamento.

Solicitação

1. Pod inicia a solicitação

O pod envia uma solicitação para o IP de serviço (`172.16.0.1`) do `kubernetes` na porta do servidor de API do EKS Kubernetes (443) de uma porta de origem aleatória. O pacote seria assim:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. Processamento da CNI

A CNI detecta que o IP de destino não pertence a nenhum CIDR de pod que ele gerencia. Como o NAT de saída está habilitado, a CNI aplica o SNAT ao pacote, alterando o IP de origem para o IP do nó antes de passá-lo para a pilha de rede do nó:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Observação: a CNI e as `iptables` são mostradas no exemplo como blocos separados para maior clareza, mas, na prática, é possível que algumas CNIs usem as `iptables` para aplicar o NAT.

3. Processamento de rede de nós

Aqui, as regras de `iptables` definidas pelo `kube-proxy` se comportam da mesma forma que no exemplo anterior, balanceando a carga do pacote para uma das ENIs do ambiente de gerenciamento do EKS. O pacote agora ficaria assim:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

O nó encaminha o pacote para seu gateway padrão porque o IP de destino não está na rede local.

4. Roteamento de roteador local

O roteador local determina que o IP de destino (`10.0.0.132`) pertence ao CIDR da VPC (`10.0.0.0/16`) e o encaminha para o gateway conectado à AWS.

5. Trânsito entre limites

O pacote viaja pela conexão estabelecida (como Direct Connect ou VPN) além do limite da nuvem até a VPC.

6. Entrega da rede da VPC

A camada de rede da VPC roteia o pacote para a sub-rede correta em que a ENI do ambiente de gerenciamento do EKS (`10.0.0.132`) está localizado.

7. Recepção da ENI

O pacote chega à ENI do ambiente de gerenciamento do EKS anexado ao servidor de API do Kubernetes.

Resposta

Depois que o ambiente de gerenciamento do EKS processa a solicitação, ele envia uma resposta de volta ao pod:

7. Servidor de API envia resposta

O servidor de API do EKS Kubernetes cria um pacote de resposta com o IP de origem inicial como o destino. O pacote seria assim:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Como o IP de destino pertence ao CIDR do nó remoto configurado (`10.80.0.0/16`), ele o envia por meio da ENI na VPC com o roteador da sub-rede como o próximo salto.

6. Roteamento da VPC

A tabela de rotas da VPC contém uma entrada para o CIDR do nó remoto (`10.80.0.0/16`), direcionando esse tráfego para o gateway de VPC para on-premises.

5. Trânsito entre limites

O gateway transfere o pacote além do limite de nuvem por meio de sua conexão estabelecida (como Direct Connect ou VPN) para a rede on-premises.

4. Recepção de rede on-premises

O pacote chega ao seu roteador on-premises local.

3. Entrega ao nó

O roteador local identifica que o endereço IP de destino (`10.80.0.2`) pertence à sua rede conectada diretamente e encaminha o pacote diretamente para o nó híbrido de destino.

2. Processamento de rede de nós

À medida que o pacote é processado pela pilha de rede do nó, o `conntrack` (uma parte de `netfilter`) combina o pacote com a conexão que o pod estabeleceu inicialmente e, como o DNAT foi aplicado originalmente, ele reverte isso regravando o IP de origem do IP da ENI do ambiente de gerenciamento do EKS para o IP do serviço `kubernetes`:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1. Processamento da CNI

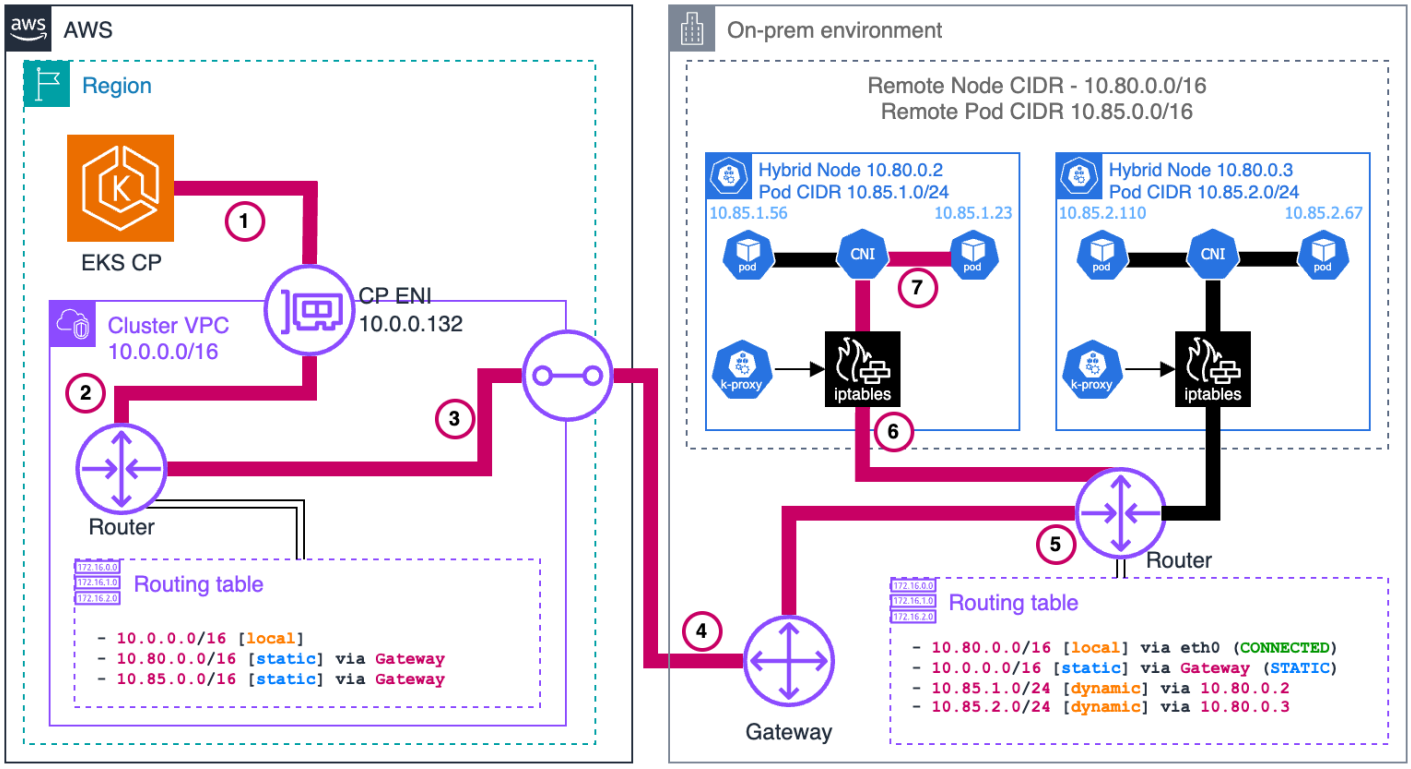
A CNI identifica que esse pacote pertence a uma conexão em que ele aplicou anteriormente o SNAT. Ele reverte o SNAT, alterando o IP de destino de volta para o IP do pod:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

A CNI detecta que o IP de destino pertence a um pod em sua rede e entrega o pacote ao namespace correto da rede do pod.

Esse fluxo mostra como o CNI NAT pode simplificar a configuração, permitindo que os pacotes sejam roteados de volta ao nó sem exigir roteamento adicional para os CIDRs do pod.

Ambiente de gerenciamento do EKS para pods executados em um nó híbrido (webhooks)



Esse padrão de tráfego é mais comum em webhooks, em que o ambiente de gerenciamento do EKS precisa iniciar conexões diretamente com servidores de webhook executados em pods em nós híbridos. Os exemplos incluem a validação e a mutação de webhooks de admissão, que são chamados pelo servidor de API durante os processos de validação ou mutação de recursos.

Solicitação

1. O servidor de API do EKS Kubernetes inicia a solicitação

Quando um webhook é configurado no cluster e uma operação de API relevante o aciona, o servidor de API do EKS Kubernetes precisa fazer uma conexão direta com o pod do servidor do webhook. O servidor de API primeiro consulta o endereço IP do pod no recurso do serviço ou do endpoint associado ao webhook.

Supondo que o pod do webhook esteja sendo executado em um nó híbrido com o IP `10.85.1.23`, o servidor de API do EKS Kubernetes criará uma solicitação HTTPS para o endpoint do webhook. O pacote inicial é enviado pela ENI do ambiente de gerenciamento do EKS na VPC porque o IP de destino `10.85.1.23` pertence ao CIDR do pod remoto configurado (`10.85.0.0/16`). O pacote seria assim:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 41892 (random) |                 |
| Dst: 10.85.1.23 | Dst: 8443           |                 |
+-----------------+---------------------+-----------------+
```

2. Processamento de rede da VPC

O pacote sai da ENI do ambiente de gerenciamento do EKS e entra na camada de rede da VPC com o roteador da sub-rede como o próximo salto.

3. Pesquisa da tabela de rotas da VPC

A tabela de rotas da VPC para a sub-rede que contém a ENI do ambiente de gerenciamento do EKS contém uma rota específica para o CIDR do pod remoto (`10.85.0.0/16`). Essa regra de roteamento direciona o pacote para o gateway de VPC para on-premises (por exemplo, um gateway privado virtual para conexões Direct Connect ou VPN):

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

4. Trânsito entre limites

O gateway transfere o pacote além do limite de nuvem por meio de sua conexão estabelecida (como Direct Connect ou VPN) para a rede on-premises. O pacote mantém seus endereços IP iniciais de origem e de destino à medida que atravessa essa conexão.

5. Recepção de rede on-premises

O pacote chega ao seu roteador on-premises local. O roteador consulta sua tabela de roteamento para determinar como acessar o endereço 10.85.1.23. Para que isso funcione, sua rede on-premises deve ter rotas para os CIDRs dos pods que direcionem o tráfego para o nó híbrido apropriado.

Nesse caso, a tabela de rotas do roteador contém uma entrada indicando que a sub-rede `10.85.1.0/24` pode ser acessada por meio do nó híbrido com o IP `10.80.0.2`:

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

6. Entrega ao nó

Com base na entrada da tabela de roteamento, o roteador encaminha o pacote para o nó híbrido (`10.80.0.2`). Quando o pacote chega ao nó, ele está igual a quando o servidor de API do EKS Kubernetes o enviou, com o IP de destino ainda sendo o IP do pod.

7. Processamento da CNI

A pilha de rede do nó recebe o pacote e, vendo que o IP de destino não é o IP do próprio nó, o passa para a CNI para processamento. A CNI identifica que o IP de destino pertence a um pod executado localmente nesse nó e encaminha o pacote para o pod correto por meio das interfaces virtuais apropriadas:

```
Original packet -> node routing -> CNI -> Pod's network namespace
```

O servidor do webhook no pod recebe a solicitação e a processa.

Resposta

Depois que o pod do webhook processa a solicitação, ele envia de volta uma resposta seguindo o mesmo caminho no sentido inverso:

7. Pod envia resposta

O pod do webhook cria um pacote de resposta com seu próprio IP como a origem e o solicitante original (a ENI do ambiente de gerenciamento do EKS) como o destino:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.23 | Src: 8443           |                 |
| Dst: 10.0.0.132 | Dst: 41892          |                 |
+-----------------+---------------------+-----------------+
```

A CNI identifica que esse pacote foi enviado para uma rede externa (e não para um pod local) e passa o pacote para a pilha de rede do nó com o IP de origem original preservado.

6. Processamento de rede de nós

O nó determina que o IP de destino (`10.0.0.132`) não está na rede local e encaminha o pacote para seu gateway padrão (o roteador local).

5. Roteamento de roteador local

O roteador local consulta sua tabela de roteamento e determina que o IP de destino (`10.0.0.132`) pertence ao CIDR da VPC (`10.0.0.0/16`). Ele encaminha o pacote para o gateway conectado à AWS.

4. Trânsito entre limites

O pacote viaja de volta pela mesma conexão on-premises para a VPC, cruzando o limite da nuvem na direção inversa.

3. Roteamento da VPC

Quando o pacote chega à VPC, as tabelas de rotas identificam que o IP de destino pertence a uma sub-rede dentro da VPC. O pacote é roteado adequadamente dentro da VPC.

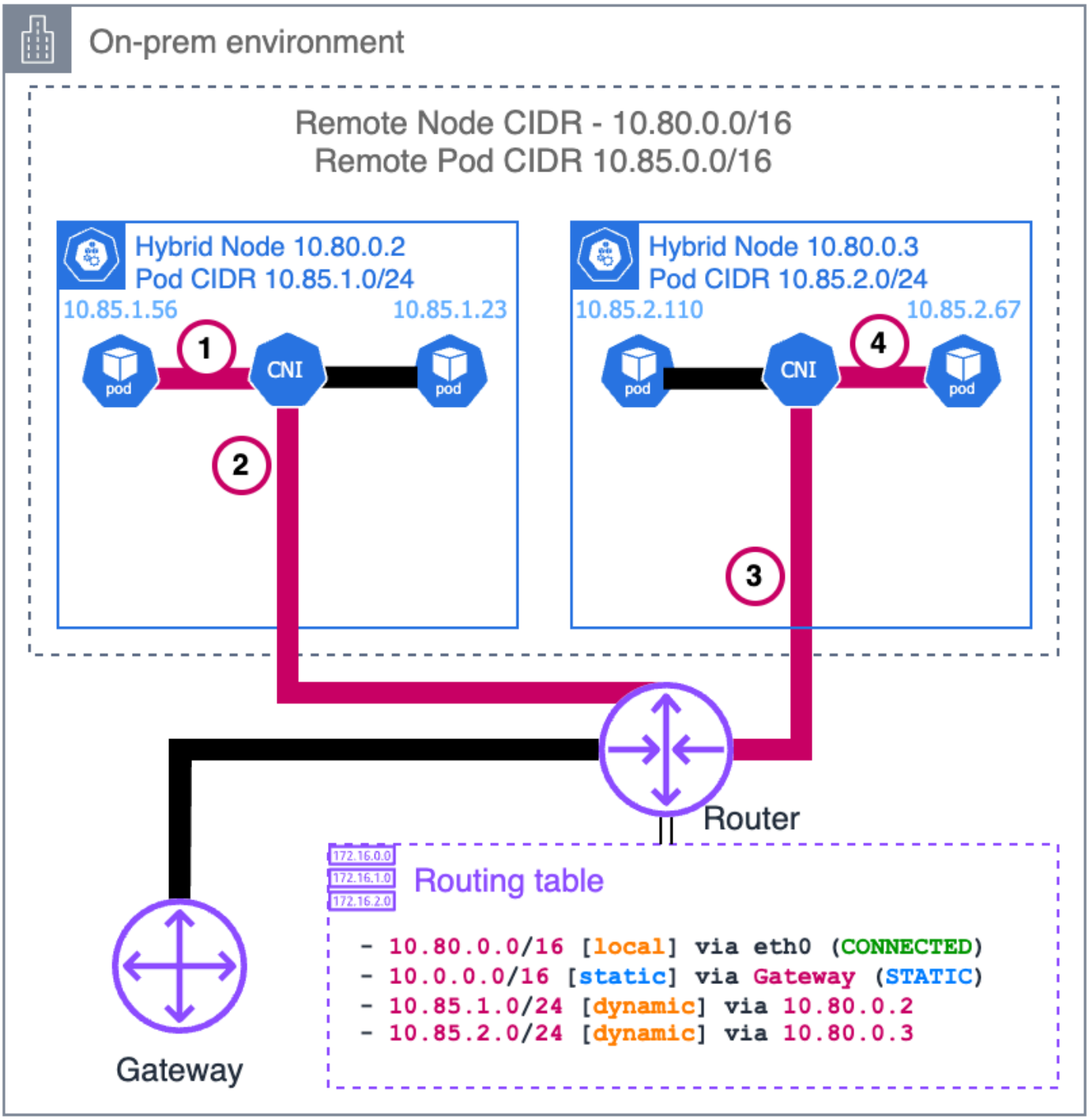
2. e 1. Recepção da ENI do ambiente de gerenciamento do EKS

O pacote chega à ENI anexada ao servidor de API do EKS Kubernetes, completando a viagem de ida e volta. O servidor de API recebe a resposta do webhook e continua processando a solicitação original da API com base nessa resposta.

Esse fluxo de tráfego demonstra por que os CIDRs de pods remotos devem ser configurados e roteados adequadamente:

- A VPC deve ter rotas para os CIDRs dos pods remotos apontando para o gateway on-premises
- Sua rede on-premises deve ter rotas para CIDRs de pods que direcionem o tráfego para os nós específicos que hospedam esses pods
- Sem essa configuração de roteamento, os webhooks e outros serviços similares executados em pods em nós híbridos não seriam acessíveis no ambiente de gerenciamento do EKS.

Pod-to-Pod em execução em nós híbridos



Esta seção explica como os pods executados em diferentes nós híbridos se comunicam entre si. Este exemplo pressupõe que a CNI usa VXLAN para encapsulamento, o que é comum para CNIs

como Cilium ou Calico. O processo geral é semelhante para outros protocolos de encapsulamento, como Geneve ou IP-in-IP.

Solicitação

1. Pod A inicia a comunicação

O pod A (`10.85.1.56`) no nó 1 deseja enviar tráfego para o pod B (`10.85.2.67`) no nó 2. Veja como seria o pacote inicial:

```
+------------------+-----------------+-------------+-----------------+
| Ethernet Header  | IP Header       | TCP/UDP     | Payload         |
| Src: Pod A MAC   | Src: 10.85.1.56 | Src: 43721  |                 |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080   |                 |
+------------------+-----------------+-------------+-----------------+
```

2. CNI intercepta e processa o pacote

Quando o pacote do pod A deixa seu namespace de rede, a CNI o intercepta. A CNI consulta sua tabela de roteamento e determina: - O IP de destino (`10.85.2.67`) pertence ao CIDR do pod - Este IP não está no nó local, mas pertence ao nó 2 (`10.80.0.3`) - O pacote precisa ser encapsulado com VXLAN.

A decisão de encapsular é fundamental porque a rede física subjacente não sabe como rotear os CIDRs de pods diretamente, ela só sabe como rotear o tráfego entre os IPs dos nós.

A CNI encapsula todo o pacote original dentro de um quadro VXLAN. Isso cria efetivamente um “pacote dentro de um pacote” com novos cabeçalhos:

```
+-----------------+---------------+-------------+-----------
+--------------------------+
| Outer Ethernet  | Outer IP       | Outer UDP    | VXLAN      | Original Pod-to-Pod
    |
| Src: Node1 MAC  | Src: 10.80.0.2 | Src: Random  | VNI: 42    | Packet (unchanged
    |
| Dst: Router MAC | Dst: 10.80.0.3 | Dst: 8472    |            | from above)
    |
+-----------------+---------------+-------------+-----------
+--------------------------+
```

Pontos-chave sobre esse encapsulamento: - O pacote externo é endereçado do nó 1 (`10.80.0.2`) para o nó 2 (`10.80.0.3`) - A porta UDP `8472` é a porta VXLAN usada por padrão pelo Cilium - O

VXLAN Network Identifier (VNI) identifica a qual rede de sobreposição esse pacote pertence - Todo o pacote original (com o IP do pod A como origem e o IP do pod B como destino) é preservado intacto por dentro

O pacote encapsulado agora entra na pilha de rede regular do nó 1 e é processado da mesma forma que qualquer outro pacote:

1. Processamento de rede de nós: a pilha de rede do nó 1 roteia o pacote com base em seu destino (`10.80.0.3`)
2. Entrega de rede local:
   - Se os dois nós estiverem na mesma rede de camada 2, o pacote será enviado diretamente para o nó 2
   - Se estiverem em sub-redes diferentes, o pacote será encaminhado primeiro para o roteador local
3. Manipulação do roteador: o roteador encaminha o pacote com base em sua tabela de roteamento, entregando-o ao nó 2

3. Processamento do nó de recebimento

Quando o pacote encapsulado chega ao nó 2 (`10.80.0.3`):

1. A pilha de rede do nó o recebe e o identifica como um pacote VXLAN (porta UDP `4789`)
2. O pacote é passado para a interface VXLAN da CNI para processamento

4. Desencapsulamento VXLAN

A CNI no nó 2 processa o pacote VXLAN:

1. Ela remove os cabeçalhos externos (Ethernet, IP, UDP e VXLAN)
2. Ela extrai o pacote interno original
3. O pacote agora está de volta à sua forma original:

```
+------------------+-----------------+-------------+-----------------+
| Ethernet Header  | IP Header       | TCP/UDP     | Payload         |
| Src: Pod A MAC   | Src: 10.85.1.56 | Src: 43721  |                 |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080   |                 |
```

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

A CNI no nó 2 examina o IP de destino (`10.85.2.67`) e:

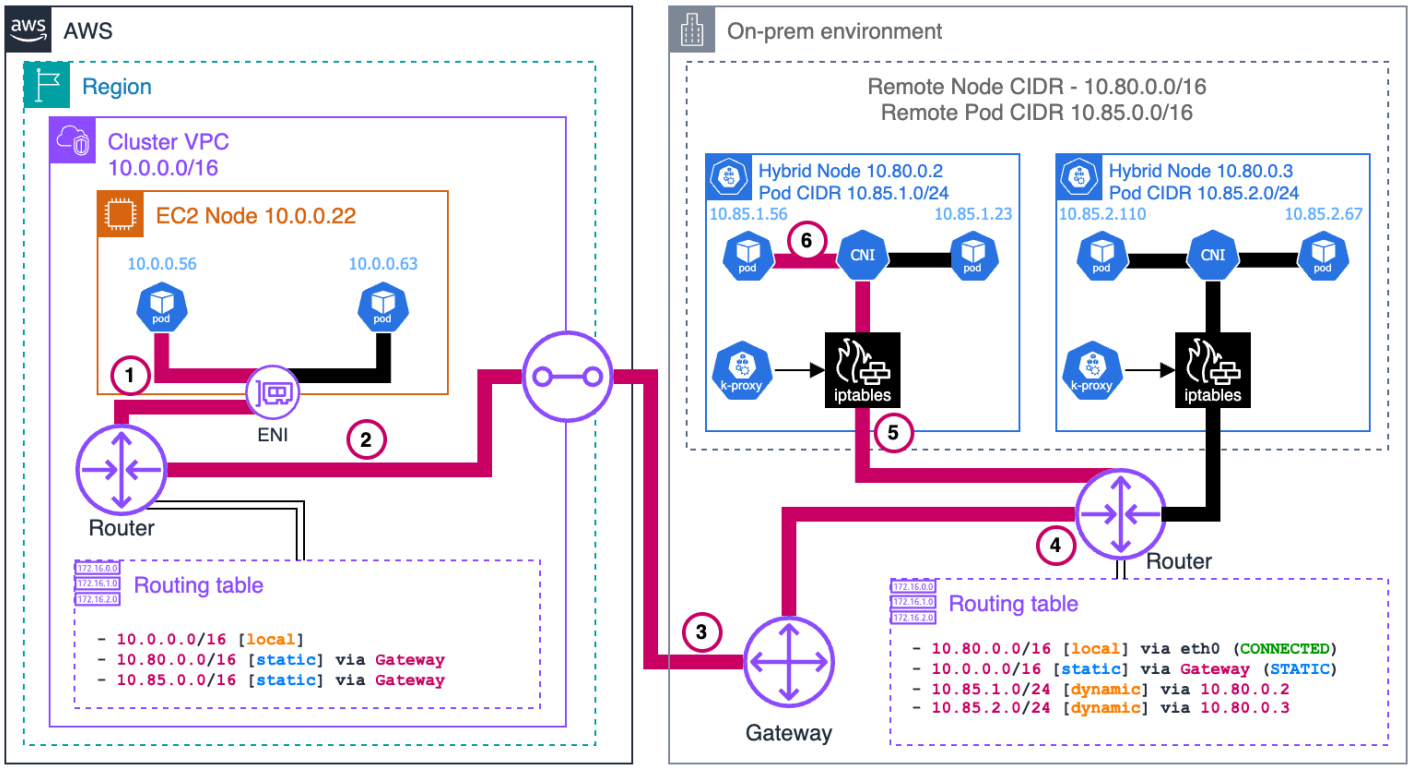
1. Identifica que esse IP pertence a um pod local
2. Encaminha o pacote pelas interfaces virtuais apropriadas
3. Entrega o pacote ao namespace de rede do pod B

Resposta

Quando o pod B responde ao pod A, todo o processo acontece no sentido inverso:

4. O pod B envia um pacote para o pod A (`10.85.1.56`)
5. A CNI do nó 2 o encapsula com VXLAN, definindo o destino para o nó 1 (`10.80.0.2`)
6. O pacote encapsulado é entregue ao nó 1
7. A CNI do nó 1 o desencapsula e fornece a resposta original ao pod A

Pods em nós da nuvem para pods em nós híbridos (tráfego leste-oeste)



Solicitação

1. Pod A inicia a comunicação

O pod A (`10.0.0.56`) no nó EC2 deseja enviar tráfego para o pod B (`10.85.1.56`) no nó híbrido. Veja como seria o pacote inicial:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.56  | Src: 52390 (random) |                 |
| Dst: 10.85.1.56 | Dst: 8080           |                 |
+-----------------+---------------------+-----------------+
```

Com a CNI da VPC, o pod A tem um IP do CIDR da VPC e está diretamente anexado a uma ENI na instância do EC2. O namespace de rede do pod está conectado à rede da VPC, então o pacote entra diretamente na infraestrutura de roteamento da VPC.

2. Roteamento da VPC

A tabela de rotas da VPC contém uma rota específica para o CIDR do pod remoto (`10.85.0.0/16`), direcionando esse tráfego para o gateway de VPC para on-premises:

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

Com base nessa regra de roteamento, o pacote é direcionado para o gateway que se conecta à rede on-premises.

3. Trânsito entre limites

O gateway transfere o pacote além do limite de nuvem por meio de sua conexão estabelecida (como Direct Connect ou VPN) para a rede on-premises. O pacote mantém seus endereços IP iniciais de origem e destino durante o trânsito.

4. Recepção de rede on-premises

O pacote chega ao seu roteador on-premises local. O roteador consulta a tabela de roteamento para determinar o próximo salto para acessar o endereço 10.85.1.56. Seu roteador on-premises deve ter rotas para os CIDRs dos pods que direcionem o tráfego para o nó híbrido apropriado.

A tabela do roteador tem uma entrada indicando que a sub-rede `10.85.1.0/24` pode ser acessada por meio do nó híbrido com IP `10.80.0.2`:

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

5. Processamento de rede de nós

O roteador encaminha o pacote para o nó híbrido (`10.80.0.2`). Quando o pacote chega ao nó, ele ainda tem o IP do pod A como a origem e o IP do pod B como o destino.

6. Processamento da CNI

A pilha de rede do nó recebe o pacote e, vendo que o IP de destino não é seu, o passa para a CNI para processamento. A CNI identifica que o IP de destino pertence a um pod executado localmente nesse nó e encaminha o pacote para o pod correto por meio das interfaces virtuais apropriadas:

```
Original packet -> node routing -> CNI -> Pod B's network namespace
```

O pod B recebe o pacote e o processa conforme necessário.

Resposta

6. Pod B envia resposta

O pod B cria um pacote de resposta com seu próprio IP como a origem e o IP do pod A como o destino:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 8080           |                 |
| Dst: 10.0.0.56  | Dst: 52390          |                 |
+-----------------+---------------------+-----------------+
```

A CNI identifica que esse pacote é destinado a uma rede externa e o passa para a pilha de rede do nó.

5. Processamento de rede de nós

O nó determina que o IP de destino (`10.0.0.56`) não pertence à rede local e encaminha o pacote para seu gateway padrão (o roteador local).

4. Roteamento de roteador local

O roteador local consulta sua tabela de roteamento e determina que o IP de destino (`10.0.0.56`) pertence ao CIDR da VPC (`10.0.0.0/16`). Ele encaminha o pacote para o gateway conectado à AWS.

3. Trânsito entre limites

O pacote viaja de volta pela mesma conexão on-premises para a VPC, cruzando o limite da nuvem na direção inversa.

2. Roteamento da VPC

Quando o pacote chega à VPC, o sistema de roteamento identifica que o IP de destino pertence a uma sub-rede dentro da VPC. O pacote é roteado pela rede da VPC em direção à instância do EC2 que hospeda o pod A.

1. Pod A recebe resposta

O pacote chega à instância do EC2 e é entregue diretamente ao pod A por meio da ENI anexada. Como a CNI da VPC não usa rede de sobreposição para pods na VPC, nenhum desencapsulamento adicional é necessário. O pacote chega com os cabeçalhos originais intactos.

Esse fluxo de tráfego leste-oeste demonstra por que os CIDRs de pods remotos devem ser configurados adequadamente e roteáveis em ambas as direções:

- A VPC deve ter rotas para os CIDRs dos pods remotos apontando para o gateway on-premises
- Sua rede on-premises deve ter rotas para CIDRs de pods que direcionem o tráfego para os nós específicos que hospedam esses pods.

Referência do **`nodeadm`** para nós híbridos

A CLI (`nodeadm`) do Amazon EKS Hybrid Nodes simplifica a instalação, a configuração, o registro e a desinstalação dos componentes dos nós híbridos. Você pode incluir o `nodeadm` nas imagens do sistema operacional para automatizar o bootstrap de nós híbridos. Consulte the section called “Preparar o sistema operacional” para obter mais informações.

A versão do `nodeadm` para nós híbridos difere da versão do `nodeadm` usada para inicializar instâncias do Amazon EC2 como nós nos clusters do Amazon EKS. Siga a documentação e as referências para a versão apropriada do `nodeadm`. Esta página da documentação é para a versão do `nodeadm` para nós híbridos.

O código-fonte do nodeadm para nós híbridos é publicado no repositório https://github.com/aws/eks-hybrid do GitHub.

> **Important**
>
> Você deve executar nodeadm com um usuário que tenha privilégios root/sudo.

Baixar **nodeadm**

A versão de nós híbridos do nodeadm está hospedada no Amazon S3 com o Amazon CloudFront na frente. Para instalar nodeadm em cada host on-premises, você pode executar o comando a seguir nos hosts on-premises.

Para hosts x86_64

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/
nodeadm'
```

Para hosts ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/
nodeadm'
```

Adicione uma permissão de arquivo executável ao binário baixado em cada host.

```
chmod +x nodeadm
```

`nodeadm install`

O comando `nodeadm install` é usado para instalar os artefatos e dependências necessários para executar e unir nós híbridos a um cluster do Amazon EKS. O comando `nodeadm install` pode ser executado individualmente em cada nó híbrido ou durante os pipelines de criação de imagens para pré-instalar as dependências dos nós híbridos nas imagens do sistema operacional.

Usage

```
nodeadm install [KUBERNETES_VERSION] [flags]
```

Argumentos posicionais

(Obrigatório) `KUBERNETES_VERSION` A versão major.minor do Kubernetes do EKS a ser instalada, por exemplo, `1.32`

Sinalizadores

Nome	Obrigatório	Descrição
`-p`, `--credential-provider`	TRUE	Provedor de credenciais a ser instalado. Os valores compatíveis são `iam-ra` e `ssm`. Consulte the section called “Preparar credenciais” para obter mais informações.
`-s`, `--containerd-source`	FALSE	Fonte para `containerd` . O `nodeadm` é compatível com a instalação de `containerd` do distro do sistema operacional, os pacotes do Docker e a omissão da instalação de `containerd` . Valores `distro`: este é o valor padrão. O `nodeadm` instalará o pacote `containerd` mais recente distribuído pelo sistema operacional do nó que seja compatível com a versão do Kubernetes do EKS. O `distro` não é um valor compatível com sistemas operacionais do Red Hat Enterprise Linux (RHEL). `docker`: o `nodeadm` instalará o pacote `containerd`

Nome	Obrigatório	Descrição
		mais recente desenvolvido e distribuído pelo Docker que seja compatível com a versão do Kubernetes do EKS. O `docker` não é um valor compatível com o Amazon Linux 2023. `none`: o `nodeadm` não vai instalar o pacote `containerd` . Você deve instalar `containerd` manualmente antes de executar `nodeadm init`.
`-r`, `--region`	FALSE	Especifica a região da AWS para baixar artefatos, como o Agente SSM. O padrão é `us-west-2` .
`-t`, `--timeout`	FALSE	Duração máxima do comando de instalação. A entrada segue o formato de duração. Por exemplo, . `1h23m`. O tempo limite de download padrão para o comando de instalação é definido como 20 minutos.
`-h`, `--help`	FALSE	Exibe a mensagem de ajuda com os parâmetros de sinalização, subcomando e valor posicional disponíveis.

Exemplos

Instale a versão `1.32` do Kubernetes com o AWS Systems Manager (SSM) como o provedor de credenciais

```
nodeadm install 1.32 --credential-provider ssm
```

Instale a versão `1.32` do Kubernetes com o AWS Systems Manager (SSM) como o provedor de credenciais, e o Docker como a fonte do containerd, com um tempo limite de download de 20 minutos.

```
nodeadm install 1.32 --credential-provider ssm --containerd-source docker --timeout 20m
```

Instale a versão `1.32` do Kubernetes com o AWS IAM Roles Anywhere como o provedor de credenciais

```
nodeadm install 1.32 --credential-provider iam-ra
```

`nodeadm config check`

O comando `nodeadm config check` verifica se há erros na configuração do nó fornecido. Esse comando pode ser usado para verificar e validar a exatidão de um arquivo de configuração de nó híbrido.

Usage

```
nodeadm config check [flags]
```

Sinalizadores

Nome	Obrigatório	Descrição
`-c`, `--config-source`	TRUE	Fonte da configuração do nodeadm. Para nós híbridos, a entrada deve seguir um URI com esquema de arquivo.
`-h`, `--help`	FALSE	Exibe a mensagem de ajuda com os parâmetros de

Nome	Obrigatório	Descrição
		sinalização, subcomando e valor posicional disponíveis.

Exemplos

```
nodeadm config check -c file://nodeConfig.yaml
```

`nodeadm init`

O comando `nodeadm init` inicia e conecta o nó híbrido com o cluster configurado do Amazon EKS. Consulte the section called “Configuração de nó para ativações híbridas do SSM” ou the section called “Configuração de nó para o IAM Roles Anywhere” para obter detalhes sobre como configurar o arquivo `nodeConfig.yaml`.

Usage

```
nodeadm init [flags]
```

Sinalizadores

Nome	Obrigatório	Descrição
`-c`, `--config-source`	TRUE	Fonte de configuração do `nodeadm`. Para nós híbridos, a entrada deve seguir um URI com esquema de arquivo.
`-s`, `--skip`	FALSE	Fases de `init` a serem ignoradas. Não é recomendá vel pular nenhuma das fases, a menos que isso ajude a corrigir um problema. Valores

Nome	Obrigatório	Descrição
		`install-validation` vai ignorar a verificação se o comando de instalação anterior tiver sido executado com êxito. A `cni-validation` vai ignorar a verificação se as portas VXLAN da CNI do Cilium ou do Calico estiverem abertas, caso o firewall esteja habilitado no nó `node-ip-validation` ignorará a verificação se o IP do nó estiver dentro de um CIDR nas redes de nós remotos
`-h`, `--help`	FALSE	Exibe a mensagem de ajuda com os parâmetros de sinalização, subcomando e valor posicional disponíveis.

Exemplos

```
nodeadm init -c file://nodeConfig.yaml
```

`nodeadm upgrade`

O comando `nodeadm upgrade` atualiza todos os artefatos instalados para a versão mais recente e inicializa o nó para configurar os artefatos atualizados e unir o cluster do EKS na AWS. O upgrade é um comando de interrupção para as workloads em execução no nó. Mova as workloads para outro nó antes de executar o comando upgrade.

Usage

```
nodeadm upgrade [KUBERNETES_VERSION] [flags]
```

Argumentos posicionais

(Obrigatório) `KUBERNETES_VERSION` A versão major.minor do Kubernetes do EKS a ser instalada, por exemplo, `1.32`

Sinalizadores

Nome	Obrigatório	Descrição
`-c`, `--config-source`	TRUE	Fonte de configuração do `nodeadm`. Para nós híbridos, a entrada deve seguir um URI com esquema de arquivo.
`-t`, `--timeout`	FALSE	Tempo limite para download de artefatos. A entrada segue o formato de duração. Por exemplo, 1h23m. O tempo limite de download padrão para o comando de atualização é definido para dez minutos.
`-s`, `--skip`	FALSE	Fases da atualização a serem ignoradas. Não é recomendável pular nenhuma fase, a menos que isso ajude a corrigir um problema. Valores `pod-validation` ignora a verificação se todos os pods não estão em execução no nó, exceto conjuntos de daemons e pods estáticos.

Nome	Obrigatório	Descrição
		`node-validation` ignora a verificação se o nó foi isolado. `init-validation` ignora a verificação se o nó foi inicializado com sucesso antes de executar a atualização. O valor `containerd-major-version-upgrade` impede atualizações da versão principal do containerd durante a atualização do nó.
`-h, --help`	FALSE	Exibe a mensagem de ajuda com os parâmetros de sinalização, subcomando e valor posicional disponíveis.

Exemplos

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml
```

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml --timeout 20m
```

`nodeadm uninstall`

O comando `nodeadm uninstall` interrompe e remove os artefatos do `nodeadm` instalados durante a `nodeadm install`, incluindo o kubelet e o containerd. Observe que o comando uninstall não drena nem exclui os nós híbridos do cluster. Você deve executar as operações de drenagem e exclusão separadamente. Consulte the section called “Excluir nós híbridos” para obter mais informações. Por padrão, `nodeadm uninstall` não prosseguirá se houver pods restantes no nó. Da mesma forma, `nodeadm uninstall` não remove dependências da CNI ou dependências de outros complementos do Kubernetes que você executa no cluster. Para remover totalmente a

instalação da CNI do host, consulte as instruções em the section called “Configurar CNI”. Se você estiver usando ativações híbridas do AWS SSM como o provedor de credenciais on-premises, o comando `nodeadm uninstall` vai cancelar o registro dos hosts como instâncias gerenciadas pelo AWS SSM.

Usage

```
nodeadm uninstall [flags]
```

Sinalizadores

Nome	Obrigatório	Descrição
`-s`, `--skip`	FALSE	Fases de desinstalação a serem ignoradas. Não é recomendável pular nenhuma das fases, a menos que isso ajude a corrigir um problema. Valores `pod-validation` ignora a verificação se todos os pods não estão em execução no nó, exceto conjuntos de daemons e pods estáticos. `node-validation` ignora a verificação se o nó foi isolado. A `init-validation` vai ignorar a verificação se o nó tiver sido inicializado com êxito antes de executar a desinstalação.
`-h`, `--help`	FALSE	Exibe a mensagem de ajuda com os parâmetros de

Nome	Obrigatório	Descrição
		sinalização, subcomando e valor posicional disponíveis.

Nome	Obrigatório	Descrição
`-f`, `--force`	FALSE	Força a exclusão de diretório s adicionais que possam conter arquivos restantes dos componentes do Kubernetes e da CNI. AVISO Isso excluirá todo o conteúdo nos diretórios padrão do Kubernetes e da CNI (`/var/lib/cni` , `/etc/cni/net.d` , etc). Não use esse sinalizador se você armazenar seus próprios dados nesses locais. A partir do nodeadm `v1.0.9`, o comando `./nodeadm uninstall --skip node-validation,pod-validation --force` não exclui mais o diretório `/var/lib/kubelet` . Isso ocorre porque ele pode conter volumes de pod e diretórios de subcaminhos de volume que às vezes incluem o sistema de arquivos do nó montado. Dicas de manuseio seguro - A exclusão de caminhos montados pode levar à exclusão acidental do sistema de arquivos real do nó

Nome	Obrigatório	Descrição
		montado. Antes de excluir manualmente o diretório `/var/lib/kubelet` , inspecione cuidadosamente todas as montagens ativas e desmonte os volumes com segurança para evitar perda de dados.

Exemplos

```
nodeadm uninstall
```

```
nodeadm uninstall --skip node-validation,pod-validation
```

`nodeadm debug`

O comando `nodeadm debug` pode ser usado para solucionar problemas de nós híbridos não íntegros ou mal configurados. Ele valida se os requisitos a seguir estão em vigor.

- O nó tem acesso de rede às APIs da AWS necessárias para obter credenciais.
- O nó é capaz de obter credenciais da AWS para o perfil do IAM configurado de nós híbridos.
- O nó tem acesso de rede ao endpoint da API do Kubernetes do EKS e à validade do certificado do endpoint da API do Kubernetes do EKS.
- O nó é capaz de se autenticar com o cluster do EKS, sua identidade no cluster é válida e o nó tem acesso ao cluster do EKS por meio da VPC configurada para o cluster do EKS.

Se forem encontrados erros, a saída do comando vai sugerir etapas de solução de problemas. Certas etapas de validação mostram processos secundários. Se estes falharem, a saída será mostrada em uma seção stderr abaixo do erro de validação.

Usage

```
nodeadm debug [flags]
```

Sinalizadores

Nome	Obrigatório	Descrição
`-c`, `--config-source`	TRUE	Fonte de configuração do `nodeadm`. Para nós híbridos, a entrada deve seguir um URI com esquema de arquivo.
`--no-color`	FALSE	Desativa a saída de cores. Útil para automação.
`-h`, `--help`	FALSE	Exibe a mensagem de ajuda com os parâmetros de sinalização, subcomando e valor posicional disponíveis.

Exemplos

```
nodeadm debug -c file://nodeConfig.yaml
```

Locais dos arquivos do nodeadm

Instalação do nodeadm

Durante a execução da `nodeadm install`, os arquivos e as localizações de arquivos a seguir são configurados.

Artifact	Caminho
CLI do IAM Roles Anywhere	/usr/local/bin/aws_signing_helper
Binário do Kubelet	/usr/bin/kubelet
Binário do Kubectl	usr/local/bin/kubectl
Provedor de credenciais do ECR	/etc/eks/image-credential-provider/ecr-credential-provider

Artifact	Caminho
AWS IAM Authenticator	/usr/local/bin/aws-iam-authenticator
CLI de configuração do SSM	/opt/ssm/ssm-setup-cli
SSM Agent	No Ubuntu: /snap/amazon-ssm-agent/current/amazon-ssm-agent No RHEL e AL2023: /usr/bin/amazon-ssm-agent
Containerd	No Ubuntu e AL2023: /usr/bin/containerd No RHEL: /bin/containerd
Iptables	No Ubuntu e AL2023: /usr/sbin/iptables No RHEL: /sbin/iptables
Plug-ins da CNI	/opt/cni/bin
rastreador de artefatos instalado	/opt/nodeadm/tracker

Inicialização do nodeadm

Durante a execução da `nodeadm init`, os arquivos e as localizações de arquivos a seguir são configurados.

Nome	Caminho
kubeconfig do Kubelet	/var/lib/kubelet/kubeconfig
Configuração do Kubelet	/etc/kubernetes/kubelet/config.json
Unidade systemd do Kubelet	/etc/systemd/system/kubelet.service
Configuração do provedor de credenciais de imagem	/etc/eks/image-credential-provider/config.json

Nome	Caminho
Arquivo env do Kubelet	/etc/eks/kubelet/environment
Certificados do Kubelet	/etc/kubernetes/pki/ca.crt
Configuração do containerd	/etc/containerd/config.toml
Configuração dos módulos do containerd do kernel	/etc/modules-load.d/contianerd.conf
AWSArquivo de configuração da	/etc/aws/hybrid/config
Arquivo de credenciais da AWS (se habilitar arquivo de credenciais)	/eks-hybrid/.aws/credentials
Unidade do sistema auxiliar de assinatura da AWS	/etc/systemd/system/aws_signing_helper_update.service
Arquivo de configuração Sysctl	/etc/sysctl.d/99-nodeadm.conf
Ca-certificates	/etc/ssl/certs/ca-certificates.crt
Arquivo de chaves GPG	/etc/apt/keyrings/docker.asc
Arquivo de origem do repositório do Docker	/etc/apt/sources.list.d/docker.list

Configuração de nó para ativações híbridas do SSM

Veja a seguir um exemplo de `nodeConfig.yaml` ao usar ativações híbridas do AWS SSM para credenciais de nós híbridos.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
```

```
      activationId:   # SSM hybrid activation id
```

Configuração de nó para o IAM Roles Anywhere

Veja a seguir um exemplo de `nodeConfig.yaml` do AWS IAM Roles Anywhere para credenciais de nós híbridos.

Ao usar o AWS IAM Roles Anywhere como seu provedor de credenciais on-premises, o `nodeName` que você usa na configuração do `nodeadm` deve estar alinhado com as permissões que você definiu como escopo para o perfil do IAM de nós híbridos. Por exemplo, se as permissões para o perfil do IAM de nós híbridos só permitirem que o AWS IAM Roles Anywhere assuma o perfil quando o nome da sessão do perfil for igual ao CN do certificado do host, o `nodeName` na configuração do `nodeadm` deverá ser o mesmo que o CN dos certificados. O `nodeName` que você usa não pode exceder 64 caracteres. Para obter mais informações, consulte the section called “Preparar credenciais”.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:              # Name of the EKS cluster
    region:            # AWS Region where the EKS cluster resides
  hybrid:
    iamRolesAnywhere:
      nodeName:        # Name of the node
      trustAnchorArn:  # ARN of the IAM Roles Anywhere trust anchor
      profileArn:      # ARN of the IAM Roles Anywhere profile
      roleArn:         # ARN of the Hybrid Nodes IAM role
      certificatePath: # Path to the certificate file to authenticate with the IAM
 Roles Anywhere trust anchor
      privateKeyPath:  # Path to the private key file for the certificate
```

Configuração de nó para personalizar o kubelet (opcional)

Você pode passar a configuração e os sinalizadores do kubelet na configuração do `nodeadm`. Veja o exemplo abaixo para saber como adicionar um rótulo de nó `abc.amazonaws.com/test-label` e como configurar para definir `shutdownGracePeriod` como 30 segundos.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
```

```
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  kubelet:
    config:           # Map of kubelet config and values
       shutdownGracePeriod: 30s
    flags:            # List of kubelet flags
       - --node-labels=abc.company.com/test-label=true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

Configuração de nó para personalizar o containerd (opcional)

Você pode passar uma configuração personalizada do containerd na configuração do `nodeadm`. A configuração do containerd para `nodeadm` aceita TOML em linha. Veja o exemplo abaixo para saber como configurar o containerd para desabilitar a exclusão de camadas de imagens descompactadas no armazenamento de conteúdo do containerd.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  containerd:
    config: |         # Inline TOML containerd additional configuration
       [plugins."io.containerd.grpc.v1.cri".containerd]
       discard_unpacked_layers = false
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

> **Note**
>
> As versões 1.x e 2.x do containerd contam com formatos de configuração distintos. O containerd 1.x usa a versão 2 da configuração, enquanto o containerd 2.x usa a versão 3. Embora o containerd 2.x seja compatível com versões anteriores da configuração 2, a versão 3 é recomendada para obtenção da performance ideal. Confira a versão do containerd com

`containerd --version` ou analise os logs de instalação do `nodeadm`. Para obter mais detalhes sobre o versionamento da configuração, consulte https://containerd.io/releases/.

Você também pode usar a configuração do containerd para habilitar a compatibilidade com o SELinux. Com o SELinux habilitado no containerd, certifique-se de que os pods programados no nó tenham o securityContext e o seLinuxOptions adequados habilitados. Você pode encontrar mais informações sobre como configurar um contexto de segurança na documentação do Kubernetes.

Note

O Red Hat Enterprise Linux (RHEL) 8 e o RHEL 9 têm o SELinux habilitado por padrão e configurado como estrito no host. Por padrão, o Amazon Linux 2023 tem o SELinux habilitado e configurado para o modo permissivo. Quando o SELinux está configurado para o modo permissivo no host, ativá-lo no containerd não bloqueará as solicitações, mas as registrará em log de acordo com a configuração do SELinux no host.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  containerd:
    config: |         # Inline TOML containerd additional configuration
       [plugins."io.containerd.grpc.v1.cri"]
       enable_selinux = true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

Solução de problemas de nós híbridos

Este tópico aborda alguns erros comuns que você poderia encontrar ao usar o Amazon EKS Hybrid Nodes e como resolvê-los. Para obter outras informações sobre solução de problemas, consulte Solução de problemas e Conteúdo do Centro de Conhecimento para o Amazon EKS no AWS re:Post. Caso não consiga resolver o problema, entre em contato com o AWS Support.

Solução de problemas de nós com **`nodeadm debug`** Você pode executar o comando `nodeadm debug` dos nós híbridos para validar se os requisitos de rede e credenciais foram atendidos. Para obter mais informações sobre o comando `nodeadm debug`, consulte the section called “Nós híbridos nodeadm”.

Detectar problemas com os nós híbridos com insights de clusters Os insights de clusters do Amazon EKS incluem verificações de insights que detectam problemas comuns com a configuração do EKS Hybrid Nodes no cluster. Você pode ver os resultados de todas as verificações de insights no Console de gerenciamento da AWS, na AWS CLI e nos SDKs da AWS. Para obter mais informações sobre insights de clusters, consulte the section called “Insights de cluster”.

Instalar a solução de problemas de nós híbridos

Os tópicos de solução de problemas a seguir estão relacionados à instalação das dependências dos nós híbridos nos hosts com o comando `nodeadm install`.

`nodeadm` Falha no comando **`must run as root`**

O comando `nodeadm install` deve ser executado com um usuário que tenha privilégios de usuário-raiz ou `sudo` no host. Caso execute `nodeadm install` com um usuário que não tenha privilégios de usuário-raiz ou `sudo`, você verá o erro a seguir na saída de `nodeadm`.

```
"msg":"Command failed","error":"must run as root"
```

Não é possível conectar às dependências

O comando `nodeadm install` instala as dependências necessárias para os nós híbridos. As dependências dos nós híbridos incluem `containerd`, `kubelet`, `kubectl` e componentes do AWS SSM ou do AWS IAM Roles Anywhere. Você deve ter acesso do local em que está executando `nodeadm install` para baixar essas dependências. Para obter mais informações sobre a lista de locais que você deve poder acessar, consulte the section called “Preparar a rede”. Caso não tenha acesso, você verá erros semelhantes ao apresentado a seguir na saída de `nodeadm install`.

```
"msg":"Command failed","error":"failed reading file from url: ...: max retries achieved
 for http request"
```

Falha ao atualizar o gerenciador de pacotes

O comando `nodeadm install` executa `apt update` ou `yum update` ou `dnf update` antes de instalar as dependências dos nós híbridos. Se essa etapa não tiver êxito, erros semelhantes

aos abaixo poderão ocorrer. Para corrigir, é possível executar `apt update` ou `yum update` ou `dnf update` antes de executar `nodeadm install`. Também é possível tentar executar `nodeadm install` novamente.

```
failed to run update using package manager
```

Tempo limite ou prazo de contexto excedido

Ao executar `nodeadm install`, se você observar problemas em várias etapas do processo de instalação com erros que indicam que houve um tempo limite ou que o prazo do contexto foi excedido, você pode ter uma conexão lenta que está impedindo a instalação das dependências dos nós híbridos antes que os tempos limite sejam atingidos. Para contornar esses problemas, você pode tentar usar o sinalizador `--timeout` em `nodeadm` para estender a duração dos tempos limite para baixar as dependências.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --timeout 20m0s
```

Solução de problemas de conexão de nós híbridos

Os tópicos de solução de problemas nesta seção estão relacionados ao processo de conexão de nós híbridos a clusters do EKS com o comando `nodeadm init`.

Erros de operação ou esquema não compatível

Ao executar `nodeadm init`, se você encontrar erros relacionados a `operation error` ou `unsupported scheme`, verifique o `nodeConfig.yaml` para garantir que esteja formatado corretamente e tenha sido passado para `nodeadm`. Para obter mais informações sobre o formato e as opções para `nodeConfig.yaml`, consulte the section called “Nós híbridos nodeadm”.

```
"msg":"Command failed","error":"operation error ec2imds: GetRegion, request canceled,
 context deadline exceeded"
```

Perfil do IAM de nós híbridos sem permissões para a ação **`eks:DescribeCluster`**

Durante a execução do `nodeadm init`, o `nodeadm` tenta obter informações sobre seu cluster do EKS ao chamar a ação `DescribeCluster` do EKS. Se o perfil do IAM do Hybrid Nodes não tiver permissão para realizar a ação `eks:DescribeCluster`, será necessário informar o endpoint da

API do Kubernetes, o pacote da CA do cluster e o intervalo CIDR IPv4 do serviço na configuração do nó que você transmite ao `nodeadm` ao executar `nodeadm init`. Para obter mais informações sobre as permissões necessárias para o perfil do IAM de nós híbridos, consulte the section called “Preparar credenciais”.

```
"msg":"Command failed","error":"operation error EKS: DescribeCluster, https response
 error StatusCode: 403 ... AccessDeniedException"
```

Perfil do IAM de nós híbridos sem permissões para a ação **`eks:ListAccessEntries`**

Durante a execução do `nodeadm init`, o `nodeadm` tenta validar se o seu cluster do EKS contém uma entrada de acesso do tipo `HYBRID_LINUX` associada ao perfil do IAM do Hybrid Nodes ao chamar a ação `ListAccessEntries` do EKS. Se o perfil do IAM do Hybrid Nodes não tiver permissão para realizar a ação `eks:ListAccessEntries`, será necessário informar o sinalizador `--skip cluster-access-validation` ao executar o comando `nodeadm init`. Para obter mais informações sobre as permissões necessárias para o perfil do IAM de nós híbridos, consulte the section called “Preparar credenciais”.

```
"msg":"Command failed","error":"operation error EKS: ListAccessEntries, https response
 error StatusCode: 403 ... AccessDeniedException"
```

O IP do nó não está no CIDR da rede do nó remoto

Ao executar `nodeadm init`, você poderá encontrar um erro se o endereço IP do nó não estiver dentro dos CIDRs da rede do nó remoto especificado. O erro será semelhante ao seguinte exemplo:

```
node IP 10.18.0.1 is not in any of the remote network CIDR blocks [10.0.0.0/16
 192.168.0.0/16]
```

Este exemplo mostra um nó com IP 10.18.0.1 tentando ingressar em um cluster com os CIDRs de rede remota 10.0.0.0/16 e 192.168.0.0/16. O erro ocorre porque 10.18.0.1 não faz parte de nenhum dos intervalos.

Confirme se você configurou corretamente o seu `RemoteNodeNetworks` para incluir todos os endereços IP de nós. Para obter mais informações sobre a configuração da rede, consulte the section called “Preparar a rede”.

- Execute o comando a seguir na região em que seu cluster está localizado para verificar suas configurações de `RemoteNodeNetwork`. Verifique se os blocos CIDR listados na saída incluem o

intervalo de IP do seu nó e são iguais aos blocos CIDR listados na mensagem de erro. Se eles não corresponderem, confirme se o nome do cluster e a região em seu cluster do `nodeConfig.yaml` correspondem ao cluster pretendido.

```
aws eks describe-cluster --name CLUSTER_NAME --region REGION_NAME --query
 cluster.remoteNetworkConfig.remoteNodeNetworks
```

- Verifique se você está trabalhando com o nó pretendido:
  - Confirme se você está no nó correto verificando se o nome do host e o endereço IP correspondem aos que você pretende registrar no cluster.
  - Confirme se esse nó está na rede on-premises correta (aquela cujo intervalo de CIDR foi registrado como `RemoteNodeNetwork` durante a configuração do cluster).

Se o IP do seu nó ainda não for o esperado, verifique o seguinte:

- Se você estiver usando o IAM Roles Anywhere, o `kubelet` realizará uma pesquisa de DNS em `nodeName` do IAM Roles Anywhere e usará um IP associado ao nome do nó, se disponível. Se você mantiver entradas de DNS para seus nós, confirme se essas entradas apontam para IPs pertencentes aos CIDRs da rede dos nós remotos.
- Se seu nó tiver várias interfaces de rede, `kubelet` poderá selecionar uma interface com um endereço IP fora dos CIDRs de rede do nó remoto como padrão. Para usar uma interface diferente, especifique seu endereço IP usando o sinalizador `--node-ip` do `kubelet` em seu `nodeConfig.yaml`. Para obter mais informações, consulte the section called “Nós híbridos nodeadm”. É possível visualizar as interfaces de rede do seu nó e seus endereços IP executando o seguinte comando no nó:

```
ip addr show
```

Os nós híbridos não estão aparecendo no cluster do EKS

Se você executou `nodeadm init` e ele foi concluído, mas os nós híbridos não aparecem no cluster, pode haver problemas com a conexão de rede entre os nós híbridos e o ambiente de gerenciamento do EKS, você pode não ter as permissões de grupo de segurança necessárias configuradas ou você pode não ter o mapeamento necessário do perfil do IAM de nós híbridos para o controle de acesso por perfil (RBAC) do Kubernetes. Você pode iniciar o processo de depuração verificando o status do

`kubelet` e os logs do `kubelet` com os comandos a seguir. Execute os comandos a seguir nos nós híbridos que não conseguiram se unir ao cluster.

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Não é possível se comunicar com o cluster

Se o nó híbrido não conseguiu se comunicar com o ambiente de gerenciamento do cluster, talvez você veja logs semelhantes aos abaixo.

```
"Failed to ensure lease exists, will retry" err="Get ..."
```

```
"Unable to register node with API server" err="Post ..."
```

```
Failed to contact API server when waiting for CSINode publishing ... dial tcp <ip
 address>: i/o timeout
```

Se você vir essas mensagens, verifique as informações abaixo para garantir que atenda aos requisitos de nós híbridos detalhados em the section called “Preparar a rede”.

- Confirme se a VPC passada para o cluster do EKS tem rotas para o gateway de trânsito (TGW) ou o gateway privado virtual (VGW) para seu nó on-premises e, opcionalmente, CIDRs de pods.
- Confirme se você tem um grupo de segurança adicional para o cluster do EKS com regras de entrada para os CIDRs de nós on-premises e, opcionalmente, para CIDRs de pods.
- Confirme se o roteador on-premises está configurado para permitir o tráfego de e para o ambiente de gerenciamento do EKS.

Não autorizado

Se o nó híbrido conseguiu se comunicar com o ambiente de gerenciamento do EKS, mas não conseguiu se registrar, você poderá ver logs semelhantes aos a seguir. Observe que a principal diferença nas mensagens de logs abaixo é o erro `Unauthorized`. Isso indica que o nó não conseguiu realizar suas tarefas porque não tem as permissões necessárias.

```
"Failed to ensure lease exists, will retry" err="Unauthorized"
```

```
"Unable to register node with API server" err="Unauthorized"
```

```
Failed to contact API server when waiting for CSINode publishing: Unauthorized
```

Se você vir essas mensagens, verifique as informações a seguir para garantir que atenda aos requisitos de nós híbridos detalhados em the section called “Preparar credenciais” e the section called “Preparar acesso ao cluster”.

- Confirme se a identidade dos nós híbridos corresponde ao perfil do IAM de nós híbridos esperado. Isso pode ser feito executando `sudo aws sts get-caller-identity` nos nós híbridos.
- Confirme se o perfil do IAM de nós híbridos tem as permissões necessárias.
- Confirme se em seu cluster você tem uma entrada de acesso ao EKS para o perfil do IAM de nós híbridos, ou confirme se o ConfigMap `aws-auth` tem uma entrada para o perfil do IAM de nós híbridos. Se você estiver usando entradas de acesso ao EKS, confirme se a entrada de acesso do perfil do IAM de nós híbridos tem o tipo de acesso `HYBRID_LINUX`. Se você estiver usando o ConfigMap `aws-auth`, confirme se a entrada para o perfil do IAM de nós híbridos atende aos requisitos e à formatação detalhados em the section called “Preparar acesso ao cluster”.

Nós híbridos registrados no cluster do EKS, mas mostram o status **`Not Ready`**

Se os nós híbridos tiverem sido registrados com êxito no cluster do EKS, mas mostrarem o status `Not Ready`, a primeira coisa a verificar é o status da interface de rede de contêineres (CNI). Se você não instalou uma CNI, é esperado que os nós híbridos tenham o status `Not Ready`. Depois que uma CNI é instalada e executada com êxito, os nós são atualizados para o status `Ready`. Se você tentou instalar uma CNI, mas ela não está sendo executada com êxito, consulte the section called “Solução de problemas da CNI em nós híbridos” nesta página.

As solicitações de assinatura de certificado (CSRs) estão pendentes

Depois de conectar os nós híbridos ao cluster do EKS, se você perceber que há CSRs pendentes para os nós híbridos, isso significa que eles não estão atendendo aos requisitos de aprovação automática. As CSRs para nós híbridos serão aprovadas e emitidas automaticamente se tiverem sido criadas por um nó com o rótulo `eks.amazonaws.com/compute-type: hybrid` e tiverem os seguintes nomes alternativos de assunto (SANs): pelo menos um SAN DNS igual ao nome do nó e os SANs IP pertencerem aos CIDRs da rede remota de nós.

O usuário híbrido já existe

Caso tenha alterado a configuração do `nodeadm` e registrado novamente o nó com a nova configuração, você poderá ver um erro informando que o usuário híbrido já existe, mas que seu conteúdo foi alterado. Em vez de executar `nodeadm init` entre as alterações de configuração, execute `nodeadm uninstall` seguido de um `nodeadm install` e `nodeadm init`. Isso garante uma limpeza adequada com as alterações na configuração.

```
"msg":"Command failed","error":"hybrid profile already exists at /etc/aws/hybrid/config
 but its contents do not align with the expected configuration"
```

O nó híbrido falhou ao resolver a API privada

Após executar `nodeadm init`, se você observar um erro nos logs do `kubelet` que mostra falhas ao contatar o servidor da API do Kubernetes do EKS porque há `no such host`, talvez seja necessário alterar sua entrada de DNS para o endpoint da API do Kubernetes do EKS na rede on-premises ou no nível do host. Consulte Forwarding inbound DNS queries to your VPC na documentação do AWS Route 53.

```
Failed to contact API server when waiting for CSINode publishing: Get ... no such host
```

Não é possível visualizar nós híbridos no console do EKS

Se você registrou os nós híbridos, mas não consegue visualizá-los no console do EKS, verifique as permissões da entidade principal do IAM que você está usando para visualizar o console. A entidade principal do IAM que você está usando deve ter permissões mínimas específicas do IAM e do Kubernetes para visualizar os recursos no console. Para obter mais informações, consulte the section called “Acessar recursos do cluster”.

Solução de problemas de execução de nós híbridos

Se os nós híbridos registrados no cluster do EKS tinham o status `Ready` e depois passaram para o status `Not Ready`, há uma ampla gama de problemas que podem ter contribuído para o status de não íntegro, como a falta de recursos suficientes para CPU, memória ou espaço em disco disponível, ou o nó está desconectado do ambiente de gerenciamento do EKS. Você pode usar as etapas abaixo para solucionar problemas nos nós e, se não conseguir resolver o problema, entre em contato com o AWS Support.

Execute `nodeadm debug` nos nós híbridos para validar se os requisitos de rede e credenciais foram atendidos. Para obter mais informações sobre o comando `nodeadm debug`, consulte the section called “Nós híbridos nodeadm”.

Obter status do nó

```
kubectl get nodes -o wide
```

Verificar condições e eventos do nó

```
kubectl describe node NODE_NAME
```

Obter status do pod

```
kubectl get pods -A -o wide
```

Verificar condições e eventos do pod

```
kubectl describe pod POD_NAME
```

Verificar logs do pod

```
kubectl logs POD_NAME
```

Verificar os logs do **`kubectl`**

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Falha nas sondas de liveness do pod ou os webhooks não estão funcionando

Se as aplicações, os complementos ou os webhooks executados nos nós híbridos não estiverem iniciando corretamente, você poderá observar problemas de rede que bloqueiam a comunicação com os pods. Para que o ambiente de gerenciamento do EKS faça contato com os webhooks em execução nos nós híbridos, você deve configurar o cluster do EKS com uma rede remota de pods e ter rotas para o CIDR de pods on-premises na tabela de roteamento da VPC com o destino como o gateway de trânsito (TGW), o gateway privado virtual (VPW) ou outro gateway que você esteja usando para conectar a VPC à rede on-premises. Para obter mais informações sobre os requisitos de rede para nós híbridos, consulte the section called “Preparar a rede”. Além disso, você deve

permitir esse tráfego no firewall on-premises e garantir que o roteador possa rotear adequadamente para os pods. Consulte the section called “Configurar webhooks” para obter mais informações sobre os requisitos para executar webhooks em nós híbridos.

Uma mensagem comum de log de pod para esse cenário é mostrada abaixo, em que ip-address é o IP do cluster para o serviço do Kubernetes.

```
dial tcp <ip-address>:443: connect: no route to host
```

Comandos **`kubectl logs`** ou **`kubectl exec`** não estão funcionando (comandos da API do **`kubelet`**)

Se os comandos `kubectl attach`, `kubectl cp`, `kubectl exec`, `kubectl logs` e `kubectl port-forward` atingirem o tempo limite, enquanto outros comandos `kubectl` funcionam com êxito, o problema provavelmente está relacionado à configuração da rede remota. Esses comandos se conectam por meio do cluster ao endpoint `kubelet` no nó. Para obter mais informações, consulte the section called “Endpoint do `kubelet`”.

Verifique se seus IPs de nó e IPs de pod se enquadram na rede de nós remotos e nos CIDRs da rede de pods remotos configurados para seu cluster. Use os comandos abaixo para examinar as atribuições de IP.

```
kubectl get nodes -o wide
```

```
kubectl get pods -A -o wide
```

Compare esses IPs com seus CIDRs de rede remota configurados para garantir o roteamento adequado. Para requisitos de configuração de rede, consulte the section called “Preparar a rede”.

Solução de problemas da CNI em nós híbridos

Se, a princípio, você encontrar problemas ao iniciar o Cilium ou o Calico com nós híbridos, geralmente é devido a problemas de rede entre os nós híbridos ou os pods da CNI executados em nós híbridos e o ambiente de gerenciamento do EKS. Certifique-se de que seu ambiente atenda aos requisitos em Preparar a rede para nós híbridos. É útil dividir o problema em partes.

Configuração do cluster do EKS

- As configurações de RemoteNodeNetwork e RemotePodNetwork estão corretas?

Configuração de VPC

Há rotas para RemoteNodeNetwork e RemotePodNetwork na tabela de roteamento da VPC com o destino do gateway de trânsito ou do gateway privado virtual?

Configuração do security group

Existem regras de entrada e saída para RemoteNodeNetwork e RemotePodNetwork?

Rede on-premises

Existem rotas e acesso de e para o ambiente de gerenciamento do EKS e de e para os nós híbridos e os pods executados em nós híbridos?

Configuração da CNI

Se estiver usando uma rede de sobreposição, a configuração do grupo de IPs para a CNI corresponderá à RemotePodNetwork configurada para o cluster do EKS se estiver usando webhooks?

O nó híbrido tem o status **`Ready`** sem uma CNI instalada

Se os nós híbridos estiverem mostrando o status `Ready`, mas você não instalou uma CNI no cluster, é possível que haja artefatos de CNI antigos nos nós híbridos. Por padrão, quando você desinstala o Cilium e o Calico com ferramentas como o Helm, os recursos em disco não são removidos das máquinas físicas ou virtuais. Além disso, as definições de recursos personalizados (CRDs) dessas CNIs ainda podem estar presentes no cluster de uma instalação antiga. Para obter mais informações, consulte as seções Excluir Cilium e Excluir Calico de the section called “Configurar CNI”.

Solução de problemas do Cilium

Caso esteja tendo problemas ao executar o Cilium em nós híbridos, consulte as etapas de solução de problemas na documentação do Cilium. As seções abaixo abordam problemas que podem ser exclusivos da implantação do Cilium em nós híbridos.

O Cilium não está iniciando

Se os agentes do Cilium que são executados em cada nó híbrido não estiverem iniciando, verifique os logs dos pods do agente do Cilium em busca de erros. O agente do Cilium requer conectividade com o endpoint da API do Kubernetes do EKS para iniciar. A inicialização do agente do Cilium

falhará se essa conectividade não estiver configurada corretamente. Nesse caso, você verá mensagens de log semelhantes às abaixo nos logs do pod do agente do Cilium.

```
msg="Unable to contact k8s api-server"
level=fatal msg="failed to start: Get \"https://<k8s-cluster-ip>:443/api/v1/namespaces/
kube-system\": dial tcp <k8s-cluster-ip>:443: i/o timeout"
```

O agente do Cilium é executado na rede do host. O cluster do EKS deve ser configurado com `RemoteNodeNetwork` para a conectividade do Cilium. Confirme se você tem um grupo de segurança adicional para o cluster do EKS com uma regra de entrada para `RemoteNodeNetwork`, se você tem rotas na VPC para `RemoteNodeNetwork` e se a rede on-premises está configurada corretamente para permitir a conectividade com o ambiente de gerenciamento do EKS.

Se o operador do Cilium estiver em execução, e alguns dos agentes do Cilium estiverem em execução, mas não todos, confirme se você tem IPs de pods disponíveis para alocar para todos os nós do cluster. Você configura o tamanho dos CIDRs de pods alocáveis ao usar o IPAM do grupo de clusters com `clusterPoolIPv4PodCIDRList` na configuração do Cilium. O tamanho do CIDR por nó é configurado com a definição de `clusterPoolIPv4MaskSize` na configuração do Cilium. Consulte Expanding the cluster pool na documentação do Cilium para obter mais informações.

O BGP do Cilium não está funcionando

Se você estiver usando o ambiente de gerenciamento do BGP do Cilium para anunciar os endereços de pod ou serviço na rede on-premises, você poderá usar os comandos a seguir da CLI do Cilium para verificar se o BGP está anunciando as rotas para os recursos. Para obter as etapas de instalação da CLI do Cilium, consulte Install the Cilium CLI na documentação do Cilium.

Caso o BGP esteja funcionando corretamente, você deverá ver os nós híbridos com o estado da sessão `established` na saída. Talvez seja necessário trabalhar com a equipe de rede para identificar os valores corretos para o AS local, AS do par e Endereço do par do ambiente.

```
cilium bgp peers
```

```
cilium bgp routes
```

Se você estiver usando o BGP do Cilium para anunciar os IPs dos serviços com o tipo `LoadBalancer`, você deverá ter o mesmo rótulo tanto em `CiliumLoadBalancerIPPool` quanto no serviço, que deve ser usado no seletor de `CiliumBGPAdvertisement`. Um exemplo é mostrado abaixo. Observe que, se você estiver usando o BGP do Cilium para anunciar os IPs dos serviços

com o tipo LoadBalancer, as rotas do BGP poderão ser interrompidas durante a reinicialização do agente do Cilium. Para obter mais informações, consulte Failure Scenarios na documentação do Cilium.

Serviço

```
kind: Service
apiVersion: v1
metadata:
  name: guestbook
  labels:
    app: guestbook
spec:
  ports:
  - port: 3000
    targetPort: http-server
  selector:
    app: guestbook
  type: LoadBalancer
```

CiliumLoadBalancerIPPool

```
apiVersion: cilium.io/v2alpha1
kind: CiliumLoadBalancerIPPool
metadata:
  name: guestbook-pool
  labels:
    app: guestbook
spec:
  blocks:
  - cidr: <CIDR to advertise>
  serviceSelector:
    matchExpressions:
      - { key: app, operator: In, values: [ guestbook ] }
```

CiliumBGPAdvertisement

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisements-guestbook
  labels:
```

```
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - ExternalIP
          - LoadBalancerIP
      selector:
        matchExpressions:
          - { key: app, operator: In, values: [ guestbook ] }
```

Solução de problemas do Calico

Se você estiver tendo problemas ao executar o Calico em nós híbridos, consulte as etapas de solução de problemas na documentação do Calico. As seções abaixo abordam problemas que podem ser exclusivos da implantação do Calico em nós híbridos.

A tabela abaixo resume os componentes do Calico e se eles são executados na rede de nós ou de pods por padrão. Se você configurou o Calico para usar NAT para tráfego de saída de pods, a rede on-premises deve estar configurada para rotear o tráfego para o CIDR de nós on-premises, e as tabelas de roteamento da VPC devem ser configuradas com uma rota para o CIDR de nós on-premises com o gateway de trânsito (TGW) ou o gateway privado virtual (VGW) como destino. Se você não estiver configurando o Calico para usar NAT para tráfego de saída de pods, a rede on-premises deverá estar configurada para rotear o tráfego para o CIDR de pods on-premises, e as tabelas de roteamento da VPC devem ser configuradas com uma rota para o CIDR de pods on-premises com o gateway de trânsito (TGW) ou o gateway privado virtual (VGW) como destino.

Componente	Rede
Servidor da API do Calico	Nó
Calico Controllers for Kubernetes	Pod
Agente de nós do Calico	Nó
Calico `typha`	Nó
Driver de nós da CSI do Calico	Pod
Operador do Calico	Nó

Recursos do Calico são programados ou executados em nós isolados

Os recursos do Calico que não são executados como um DaemonSet têm tolerâncias flexíveis por padrão que permitem que sejam programados em nós isolados que não estão prontos para programar ou executar pods. Você pode restringir as tolerâncias para os recursos do Calico que não são DaemonSet alterando a instalação do operador para incluir o abaixo.

```
installation:
  ...
  controlPlaneTolerations:
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  calicoKubeControllersDeployment:
    spec:
      template:
        spec:
          tolerations:
          - effect: NoExecute
            key: node.kubernetes.io/unreachable
            operator: Exists
            tolerationSeconds: 300
          - effect: NoExecute
            key: node.kubernetes.io/not-ready
            operator: Exists
            tolerationSeconds: 300
  typhaDeployment:
    spec:
      template:
        spec:
          tolerations:
          - effect: NoExecute
            key: node.kubernetes.io/unreachable
            operator: Exists
            tolerationSeconds: 300
          - effect: NoExecute
            key: node.kubernetes.io/not-ready
            operator: Exists
```

```
            tolerationSeconds: 300
```

Solução de problemas de credenciais

Para ativações híbridas do AWS SSM e AWS IAM Roles Anywhere, você pode validar se as credenciais do perfil do IAM de nós híbridos estão configuradas corretamente nos nós híbridos ao executar o comando a seguir nos nós híbridos. Confirme se o nome do nó e o nome do perfil do IAM de nós híbridos são os esperados.

```
sudo aws sts get-caller-identity
```

```
{
    "UserId": "ABCDEFGHIJKLM12345678910:<node-name>",
    "Account": "<aws-account-id>",
    "Arn": "arn:aws:sts::<aws-account-id>:assumed-role/<hybrid-nodes-iam-role/<node-
name>"
}
```

AWS Solução de problemas do Systems Manager (SSM

Se você estiver usando ativações híbridas do AWS SSM para as credenciais de nós híbridos, esteja ciente dos diretórios e artefatos do SSM a seguir que são instalados nos nós híbridos pelo `nodeadm`. Para obter mais informações sobre o SSM Agente, consulte Trabalhar com o SSM Agent no Guia do usuário do AWS Systems Manager.

Descrição	Local
SSM agent	Ubuntu: `/snap/amazon-ssm-agent/current/amazon-ssm-agent` RHEL e AL2023: `/usr/bin/amazon-ssm-agent`
Registros de atendentes do SSM	`/var/log/amazon/ssm`
Credenciais AWS	`/root/.aws/credentials`
CLI de configuração do SSM	`/opt/ssm/ssm-setup-cli`

Reiniciar o SSM Agent

Alguns problemas podem ser resolvidos ao reiniciar o SSM Agent. Você pode usar os comandos abaixo para reiniciá-lo.

AL2023 e outros sistemas operacionais

```
systemctl restart amazon-ssm-agent
```

Ubuntu

```
systemctl restart snap.amazon-ssm-agent.amazon-ssm-agent
```

Verificar a conectividade com os endpoints do SSM

Confirme se você pode se conectar aos endpoints do SSM nos nós híbridos. Para obter uma lista de endpoints do SSM, consulte AWS Systems Manager endpoints and quotas. Substitua `us-west-2` no comando abaixo pela região da AWS para sua ativação híbrida do AWS SSM.

```
ping ssm.us-west-2.amazonaws.com
```

Visualizar o status da conexão das instâncias registradas do SSM

Você pode verificar o status da conexão das instâncias registradas com ativações híbridas do SSM com o comando da AWS CLI a seguir. Substitua o ID da máquina pelo ID da máquina da sua instância.

```
aws ssm get-connection-status --target mi-012345678abcdefgh
```

Incompatibilidade da soma de verificação da CLI de configuração do SSM

Caso encontre um problema com a incompatibilidade da soma de verificação `ssm-setup-cli` ao executar `nodeadm install`, você deve confirmar que não há instalações mais antigas do SSM existentes no host. Se houver instalações mais antigas do SSM no host, remova-as e execute novamente `nodeadm install` para resolver o problema.

```
Failed to perform agent-installation/on-prem registration: error while verifying
 installed ssm-setup-cli checksum: checksum mismatch with latest ssm-setup-cli.
```

do SSM **`InvalidActivation`**

Se você vir um erro ao registrar a instância com o AWS SSM, confirme se `region`, `activationCode` e `activationId` em `nodeConfig.yaml` estão corretos. A região da AWS para o cluster do EKS deve corresponder à região da ativação híbrida do SSM. Se esses valores estiverem configurados incorretamente, você verá um erro semelhante ao abaixo.

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 InvalidActivation
```

`ExpiredTokenException` do SSM: o token de segurança incluído na solicitação está expirado

Se o SSM Agent não conseguir atualizar as credenciais, você poderá observar um `ExpiredTokenException`. Nesse cenário, se você conseguir se conectar aos endpoints do SSM nos nós híbridos, talvez seja necessário reiniciar o SSM Agent para forçar uma atualização da credencial.

```
"msg":"Command failed","error":"operation error SSM: DescribeInstanceInformation, https
 response error StatusCode: 400, RequestID: eee03a9e-f7cc-470a-9647-73d47e4cf0be, api
 error ExpiredTokenException: The security token included in the request is expired"
```

Erro do SSM ao executar o comando de registro de máquina

Se você observar um erro ao registrar a máquina com o SSM, talvez seja necessário executar `nodeadm install` novamente para garantir que todas as dependências do SSM estejam instaladas corretamente.

```
"error":"running register machine command: , error: fork/exec /opt/aws/ssm-setup-cli:
 no such file or directory"
```

do SSM **`ActivationExpired`**

Durante a execução de `nodeadm init`, se você vir um erro ao registrar a instância com o SSM devido a uma ativação expirada, você precisará criar uma ativação híbrida do SSM, atualizar `nodeConfig.yaml` com `activationCode` e `activationId` da nova ativação híbrida do SSM e executar `nodeadm init` novamente.

```
"msg":"Command failed","error":"SSM activation expired. Please use a valid activation"
```

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 ActivationExpired
```

O SSM falhou ao atualizar as credenciais em cache

Se você perceber uma falha na atualização das credenciais em cache, o arquivo `/root/.aws/credentials` pode ter sido excluído do host. Primeiro, verifique a ativação híbrida do SSM e certifique-se de que ela esteja ativa e que os nós híbridos estejam configurados corretamente para usar a ativação. Verifique os logs do SSM Agent em `/var/log/amazon/ssm` e execute novamente o comando `nodeadm init` depois de resolver o problema no lado do SSM.

```
"Command failed","error":"operation error SSM: DescribeInstanceInformation, get
 identity: get credentials: failed to refresh cached credentials"
```

Limpar o SSM

Para remover o SSM Agent do host, você pode executar os comandos a seguir.

```
dnf remove -y amazon-ssm-agent
sudo apt remove --purge amazon-ssm-agent
snap remove amazon-ssm-agent
rm -rf /var/lib/amazon/ssm/Vault/Store/RegistrationKey
```

AWS Solução de problemas do IAM Roles Anywhere

Se você estiver usando o AWS IAM Roles Anywhere para as credenciais dos nós híbridos, esteja ciente dos diretórios e artefatos a seguir que são instalados nos nós híbridos pelo `nodeadm`. Para obter mais informações sobre a solução de problemas do IAM Roles Anywhere, consulte Troubleshooting AWS IAM Roles Anywhere identity and access no Guia do usuário do AWS IAM Roles Anywhere.

Descrição	Local
CLI do IAM Roles Anywhere	`/usr/local/bin/aws_signing_helper`
Local e nome padrão do certificado	`/etc/iam/pki/server.pem`
Local e nome padrão da chave	`/etc/iam/pki/server.key`

O IAM Roles Anywhere falhou ao atualizar as credenciais em cache

Se você observar uma falha na atualização das credenciais em cache, revise o conteúdo de `/etc/aws/hybrid/config` e confirme se o IAM Roles Anywhere foi configurado corretamente na sua configuração do `nodeadm`. Confirme se `/etc/iam/pki` existe. Cada nó deve ter um certificado e uma chave exclusivos. Por padrão, ao usar o IAM Roles Anywhere como provedor de credenciais, o `nodeadm` usa `/etc/iam/pki/server.pem` para o local e o nome do certificado e `/etc/iam/pki/server.key` para a chave privada. Talvez seja necessário criar os diretórios antes de colocar os certificados e as chaves nos diretórios com `sudo mkdir -p /etc/iam/pki`. Você pode verificar o conteúdo do certificado com o comando abaixo.

```
openssl x509 -text -noout -in server.pem
```

```
open /etc/iam/pki/server.pem: no such file or directory
could not parse PEM data
Command failed {"error": "... get identity: get credentials: failed to refresh cached
 credentials, process provider error: error in credential_process: exit status 1"}
```

IAM Roles Anywhere não autorizado a executar **`sts:AssumeRole`**

Nos logs do `kubelet`, se você vir um problema de acesso negado para a operação `sts:AssumeRole` ao usar o IAM Roles Anywhere, verifique a política de confiança do perfil do IAM de nós híbridos para confirmar se a entidade principal do serviço do IAM Roles Anywhere tem permissão para assumir o perfil do IAM de nós híbridos. Além disso, confirme se o ARN da âncora de confiança está configurado corretamente na política de confiança do perfil do IAM de nós híbridos, e se este foi adicionado ao seu usuário do IAM Roles Anywhere.

```
could not get token: AccessDenied: User: ... is not authorized to perform:
 sts:AssumeRole on resource: ...
```

IAM Roles Anywhere não autorizado a definir **`roleSessionName`**

Nos logs do `kubelet`, se você vir um problema de acesso negado para definir `roleSessionName`, confirme que você definiu `acceptRoleSessionName` como true para seu usuário do IAM Roles Anywhere.

```
AccessDeniedException: Not authorized to set roleSessionName
```

Solução de problemas do sistema operacional

RHEL

Falhas no registro do gerenciador de direitos ou assinaturas

Se você estiver executando `nodeadm install` e encontrar uma falha na instalação das dependências dos nós híbridos devido a problemas de registro de direitos, certifique-se de ter definido corretamente a senha e o nome de usuário do Red Hat no host.

```
This system is not registered with an entitlement server
```

Ubuntu

GLIBC não encontrado

Se estiver usando o Ubuntu como o sistema operacional e o IAM Roles Anywhere como o provedor de credenciais com nós híbridos e verificar um problema com o GLIBC não encontrado, você poderá instalar essa dependência manualmente para resolver o problema.

```
GLIBC_2.32 not found (required by /usr/local/bin/aws_signing_helper)
```

Execute os seguintes comandos para instalar a dependência:

```
ldd --version
sudo apt update && apt install libc6
sudo apt install glibc-source
```

Bottlerocket

Se o contêiner de administração do Bottlerocket estiver habilitado, você poderá acessá-lo via SSH para depuração avançada e solução de problemas com privilégios elevados. As seções a seguir contêm comandos que precisam ser executados no contexto do host do Bottlerocket. Quando estiver no contêiner de administração, você poderá executar `sheltie` para abrir um shell com privilégios totais de usuário-raiz no host do Bottlerocket.

```
sheltie
```

Você também pode executar os comandos nas seções a seguir a partir do shell do contêiner de administração prefixando cada comando com `sudo chroot /.bottlerocket/rootfs`.

```
sudo chroot /.bottlerocket/rootfs <command>
```

Usar o logdog para coletar logs

O Bottlerocket fornece o utilitário `logdog` para coletar logs e informações do sistema de forma eficiente para fins de solução de problemas.

```
logdog
```

O utilitário `logdog` reúne logs de vários locais em um host do Bottlerocket e os combina em um arquivo tarball. Por padrão, o arquivo tarball será criado em `/var/log/support/bottlerocket-logs.tar.gz` e pode ser acessado via contêineres do host em `/.bottlerocket/support/bottlerocket-logs.tar.gz`.

Acessar logs do sistema com journalctl

Você pode verificar o status dos vários serviços do sistema, como `kubelet`, `containerd`, etc., e visualizar seus logs com os comandos a seguir. A flag `-f` acompanhará os logs em tempo real.

Para verificar o status do serviço `kubelet` e recuperar logs do `kubelet`, execute:

```
systemctl status kubelet
journalctl -u kubelet -f
```

Para verificar o status do serviço `containerd` e recuperar os logs da instância orquestrada do `containerd`, execute:

```
systemctl status containerd
journalctl -u containerd -f
```

Para verificar o status do serviço `host-containerd` e recuperar os logs da instância host do `containerd`, execute:

```
systemctl status host-containerd
journalctl -u host-containerd -f
```

Para recuperar os logs dos contêineres de bootstrap e contêineres de host, execute:

```
journalctl _COMM=host-ctr -f
```

Usar armazenamento de dados de aplicações para o cluster

Você pode usar uma variedade de serviços de armazenamento da AWS com o Amazon EKS para atender às necessidades de armazenamento das aplicações. Por meio de uma variedade de drivers compatíveis com a AWS da interface de armazenamento de contêineres (CSI), você pode usar facilmente o Amazon EBS, Amazon S3, Amazon EFS, Amazon FSX e Amazon File Cache para as necessidades de armazenamento das aplicações em execução no Amazon EKS. Para gerenciar backups do cluster do Amazon EKS, consulte o suporte do AWS Backup para Amazon EKS.

Este capítulo aborda as opções de armazenamento para clusters do Amazon EKS.

Tópicos

- Usar armazenamento de volumes do Kubernetes com o Amazon EBS
- Usar armazenamento elástico do sistema de arquivos com o Amazon EFS
- Usar armazenamento de aplicações de alta performance com o Amazon FSx para Lustre
- Usar armazenamento de aplicações de alta performance com o FSx para NetApp ONTAP
- Usar armazenamento de dados com o Amazon FSx para OpenZFS
- Minimizar a latência com o Amazon File Cache
- Acessar objetos do Amazon S3 com o driver CSI do Mountpoint para Amazon S3
- Habilitar a funcionalidade de snapshot para volumes CSI

Usar armazenamento de volumes do Kubernetes com o Amazon EBS

> **Note**
>
> Novo: o Modo Automático do Amazon EKS automatiza tarefas de rotina para armazenamento em blocos. Saiba como the section called “Implantar workload com estado”.

O driver CSI (Container Storage Interface) do Amazon Elastic Block Store (Amazon EBS) gerencia o ciclo de vida dos volumes do Amazon EBS como armazenamento para os volumes do Kubernetes que você cria. O driver da CSI do Amazon EBS cria volumes do Amazon EBS para estes tipos de volumes do Kubernetes: volumes temporários e volumes persistentes genéricos.

Considerações

- Você não precisa instalar o Controlador da CSI do Amazon EBS nos clusters do Modo Automático do EKS.
- Você não pode montar volumes do Amazon EBS em pods do Fargate.
- Você pode executar o controlador da CSI do Amazon EBS em nós do Fargate, mas o `DaemonSet` do nó da CSI do Amazon EBS só pode ser executado em instâncias do Amazon EC2.
- Os volumes do Amazon EBS e o driver CSI do Amazon EBS não são compatíveis com o Amazon EKS Hybrid Nodes.
- Suporte será fornecido para a versão complementar mais recente e para uma versão anterior. Os bugs ou vulnerabilidades encontrados na versão mais recente serão corrigidos na versão anterior em uma nova versão secundária.
- O Modo Automático do EKS exige que as classes de armazenamento usem `ebs.csi.eks.amazonaws.com` como provisionador. O driver CSI padrão do Amazon EBS (`ebs.csi.aws.com`) gerencia seus próprios volumes separadamente. Para usar volumes existentes com o Modo Automático do EKS, migre-os usando snapshots de volume para uma classe de armazenamento que use o provisionador do Modo Automático.

> **Important**
>
> Para utilizar a funcionalidade de snapshot do driver da CSI do Amazon EBS, primeiro é necessário instalar o controlador de snapshots da CSI. Para obter mais informações, consulte the section called “Controlador de snapshots da CSI”.

Pré-requisitos

- Um cluster existente. Para ver a versão necessária da plataforma, execute o comando a seguir.

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- O driver CSI do EBS precisa de permissões do AWS IAM.
  - A AWS sugere o uso de Identidades de Pod do EKS. Para obter mais informações, consulte the section called “Visão geral da configuração do EKS Pod Identities”.

- Para obter informações sobre perfis do IAM para contas de serviço, consulte the section called “Provedor OIDC do IAM”.

Etapa 1: criar um perfil do IAM

O plugin CSI do Amazon EBS requer permissões do IAM para fazer chamadas para APIs da AWS em seu nome. Se você não concluir essas etapas, a tentativa de instalar o complemento e de executar `kubectl describe pvc` mostrarão `failed to provision volume with StorageClass` junto com erro `could not create volume in EC2: UnauthorizedOperation`. Para obter mais informações, consulte Configurar permissão de driver no GitHub.

> **Note**
>
> Os pods terão acesso às permissões atribuídas ao perfil do IAM, a menos que você bloqueie o acesso ao IMDS. Para obter mais informações, consulte the section called “Práticas recomendadas”.

O procedimento a seguir mostra como criar um perfil do IAM e associar a política gerenciada da AWS a ele. Para implementar esse procedimento, você pode usar uma das seguintes ferramentas:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”
- the section called “AWS CLI”

> **Note**
>
> É possível criar uma política autogerenciada com permissões de escopo reduzido. Analise a política `AmazonEBSCSIDriverPolicy` e crie uma política do IAM personalizada com permissões reduzidas.

> **Note**
>
> As etapas específicas desse procedimento foram escritas para usar o driver como um complemento do Amazon EKS. Diferentes etapas para usar o driver como um complemento autogerenciado. Para obter mais informações, consulte Set up driver permissions no GitHub.

`eksctl`

1. Criar um perfil do IAM e associar uma política. AWS mantém uma política AWS gerenciada ou você pode criar sua própria política personalizada. Você pode criar uma função IAM e anexar a política gerenciada AWS com o seguinte comando. Substitua *`my-cluster`* pelo nome do cluster. O comando implementa uma pilha do AWS CloudFormation que cria um perfil do IAM e anexa a política do IAM a ela.

```
eksctl create iamserviceaccount \
        --name ebs-csi-controller-sa \
        --namespace kube-system \
        --cluster my-cluster \
        --role-name AmazonEKS_EBS_CSI_DriverRole \
        --role-only \
        --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicy \
        --approve
```

2. Você poderá pular esta etapa se não usar uma chave do KMS personalizada. Se você usar uma para criptografia nos volumes do Amazon EBS, personalize o perfil do IAM conforme necessário. Por exemplo, faça o seguinte:

   a. Copie e cole o código a seguir em um novo arquivo `kms-key-for-encryption-on-ebs.json`. Substitua *`custom-key-arn`* pelo ARN da chave do KMS personalizada.

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
```

```
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"]
        }
      ]
    }
```

b. Crie a política. É possível alterar *KMS_Key_For_Encryption_On_EBS_Policy* para um nome diferente. Porém, se fizer isso, não se esqueça de alterá-lo também em etapas posteriores.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Anexe a política do IAM necessária à função com o comando a seguir. Substitua *111122223333* pelo ID da sua conta.

```
aws iam attach-role-policy \
      --policy-arn arn:aws:iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

Console de gerenciamento da AWS

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Na página Perfis, selecione Criar perfil.
4. Na página Select trusted entity (Selecionar entidade confiável), faça o seguinte:
   a. Na seção Tipo de entidade confiável, escolha Identidade da Web.
   b. Para Identity provider (Provedor de identidade), escolha OpenID Connect provider URL (URL do provedor OpenID Connect) para o seu cluster (conforme mostrado na guia Overview (Visão geral) do Amazon EKS).
   c. Em Público, escolha `sts.amazonaws.com`.
   d. Escolha Próximo.
5. Na página Add permissions (Adicionar permissões), faça o seguinte:
   a. Na caixa Filtrar políticas insira `AmazonEBSCSIDriverPolicy`.
   b. Marque a caixa de seleção à esquerda do `AmazonEBSCSIDriverPolicy` retornado na pesquisa.
   c. Escolha Próximo.
6. Na página Nomear, revisar e criar, faça o seguinte:
   a. Em Nome do perfil, insira um nome exclusivo para o perfil, como *`AmazonEKS_EBS_CSI_DriverRole`*.
   b. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.
   c. Selecione Criar perfil.
7. Depois que a função for criada, escolha a função no console a fim de abri-la para edição.
8. Escolha a guia Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).
9. Encontre a linha semelhante à seguinte:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

Adicione uma vírgula ao final da linha anterior e depois adicione a linha a seguir após a anterior. Substitua *region-code* pela região da AWS em que seu cluster se encontra. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* pelo ID do provedor OIDC do cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:ebs-csi-controller-sa"
```

10. Escolha Atualizar política para concluir.
11. Se você usar uma chave do KMS personalizada para criptografia em seus volumes do Amazon EBS, personalize o perfil do IAM conforme necessário. Por exemplo, faça o seguinte:
    a. No painel de navegação à esquerda, escolha Políticas.
    b. Na página Policies (Políticas), escolha Create Policy (Criar política).
    c. Na página Criar política, escolha a guia JSON.
    d. Copie e cole o código a seguir no editor, substituindo *custom-key-arn* pelo ARN da chave KMS personalizada.

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
```

```
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"]
        }
      ]
    }
```

e. Escolha Próximo: tags.

f. Na página Add tags (optional) (Adicionar etiquetas - opcional), escolha Next: Review (Próximo: revisar).

g. Em Nome, insira um nome exclusivo para sua política (por exemplo, *KMS_Key_For_Encryption_On_EBS_Policy*).

h. Escolha Criar política.

i. No painel de navegação à esquerda, selecione Perfis.

j. Escolha ***AmazonEKS_EBS_CSI_DriverRole*** no console para abri-lo para edição.

k. Na lista suspensa Add permissions (Adicionar permissões), escolha Attach policies (Anexar políticas).

l. Na caixa Políticas de filtro, insira *KMS_Key_For_Encryption_On_EBS_Policy*.

m. Marque a caixa de seleção à esquerda da *KMS_Key_For_Encryption_On_EBS_Policy* retornada na pesquisa.

n. Escolha Anexar políticas.

AWS CLI

1. Exiba a URL do provedor OIDC do cluster. Substitua *my-cluster* pelo nome do cluster. Se o resultado do comando for `None`, revise os Pré-requisitos.

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

Veja um exemplo de saída abaixo.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Crie o perfil do IAM, concedendo a ação `AssumeRoleWithWebIdentity`.

   a. Copie o conteúdo a seguir em um arquivo chamado `aws-ebs-csi-driver-trust-policy.json`. Substitua *111122223333* pelo ID da sua conta. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* e *region-code* pelos valores retornados na etapa anterior.

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
          },
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {
            "StringEquals": {
              "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
              "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:ebs-csi-
controller-sa"
            }
          }
        }
      ]
    }
```

   b. Crie a função. É possível alterar *AmazonEKS_EBS_CSI_DriverRole* para um nome diferente. Se você alterá-lo, não se esqueça de alterá-lo em etapas superiores.

```
aws iam create-role \
      --role-name AmazonEKS_EBS_CSI_DriverRole \
      --assume-role-policy-document file://"aws-ebs-csi-driver-trust-policy.json"
```

3. Associar uma política. AWS mantém uma política AWS gerenciada ou você pode criar sua própria política personalizada. Anexe a política gerenciada pela AWS função com o comando a seguir.

```
aws iam attach-role-policy \
      --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
```

```
    --role-name AmazonEKS_EBS_CSI_DriverRole
```

4. Se você usar uma chave do KMS personalizada para criptografia em seus volumes do Amazon EBS, personalize o perfil do IAM conforme necessário. Por exemplo, faça o seguinte:

   a. Copie e cole o código a seguir em um novo arquivo `kms-key-for-encryption-on-ebs.json`. Substitua *`custom-key-arn`* pelo ARN da chave do KMS personalizada.

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"]
        }
      ]
    }
```

b. Crie a política. É possível alterar *KMS_Key_For_Encryption_On_EBS_Policy* para um nome diferente. Porém, se fizer isso, não se esqueça de alterá-lo também em etapas posteriores.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Anexe a política do IAM necessária à função com o comando a seguir. Substitua *111122223333* pelo ID da sua conta.

```
aws iam attach-role-policy \
      --policy-arn arn:aws:iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

Agora que você criou o perfil do IAM do driver de CSI do Amazon EBS, poderá avançar para a próxima seção. Quando você implanta o complemento com esse perfil do IAM, ele cria e é configurado para usar uma conta de serviço chamada `ebs-csi-controller-sa`. A conta de serviço está vinculada a um `clusterrole` do Kubernetes que recebe as permissões necessárias do Kubernetes.

Etapa 2: obter o driver de CSI do Amazon EBS

Recomendamos que você instale o driver de CSI do Amazon EBS por meio do complemento do Amazon EKS para reforçar a segurança e reduzir o volume do trabalho. Para adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criação de um complemento”. Para obter mais informações sobre complementos, consulte the section called “Complementos do Amazon EKS”.

> **Important**
>
> Antes de adicionar o driver do Amazon EBS como um complemento do Amazon EKS, verifique se você não tem uma versão autogerenciada do driver instalada no seu cluster. Em caso afirmativo, consulte Uninstalling a self-managed Amazon EBS CSI driver no GitHub.

> **Note**
>
> Por padrão, o perfil de RBAC usado pela CSI do EBS tem permissões para realizar mutações em nós a fim de fornecer suporte à funcionalidade de remoção de taints. Por conta de limitações do RBAC do Kubernetes, isso também possibilita a modificação de qualquer outro nó presente no cluster. O chart do Helm conta com um parâmetro (`node.serviceAccount.disableMutation`) que desabilita permissões de RBAC para modificar nós, vinculadas ao perfil da conta de serviço ebs-csi-node. Caso habilitado, as funcionalidades do driver, por exemplo, a remoção de taints, deixarão de operar.

Como alternativa, se quiser uma instalação autogerenciada do driver CSI do Amazon EBS, consulte Installation no GitHub.

Etapa 3: implantar uma aplicação de exemplo

Você pode implantar toda uma variedade de aplicações de exemplo e modificá-las conforme necessário. Para obter mais informações, consulte Kubernetes Examples no GitHub.

Usar armazenamento elástico do sistema de arquivos com o Amazon EFS

O Amazon Elastic File System (Amazon EFS) fornece armazenamento de arquivos com tecnologia sem servidor e totalmente elástico para que você possa compartilhar dados de arquivos sem provisionar ou gerenciar a capacidade e a performance do armazenamento. O driver da Container Storage Interface (CSI) do Amazon EFS fornece uma interface CSI que permite que os clusters do Kubernetes executados na AWS gerenciem o ciclo de vida dos sistemas de arquivos do Amazon EFS. Este tópico mostra como implantar o driver da CSI do Amazon EFS no cluster do Amazon EKS.

Considerações

- O driver da CSI do Amazon EFS não é compatível com imagens de contêiner baseadas no Windows.
- Você não pode usar o provisionamento dinâmico de volumes persistentes com nós do Fargate, mas pode usar o provisionamento estático.
- O provisionamento dinâmico requer a versão 1.2 ou superior do driver. Você pode usar o provisionamento estático para volumes persistentes usando a versão `1.1` do driver em qualquer

versão de cluster do Amazon EKS compatível (consulte Versões compatíveis com o Amazon EKS).

- A versão 1.3.2 ou superior deste driver é compatível com a arquitetura Arm64, inclusive instâncias baseadas no Graviton do Amazon EC2.
- A versão 1.4.2 ou posterior desse driver é compatível com o uso de FIPS para montagem de sistemas de arquivos.
- Anote as cotas de recursos do Amazon EFS. Por exemplo, há uma cota de 1000 pontos de acesso que podem ser criados para cada sistema de arquivos do Amazon EFS. Para acessar mais informações, consulte Cotas de recursos do Amazon EFS que você não pode alterar.
- A partir da versão 2.0.0, esse driver deixou de usar `stunnel` e passou a usar `efs-proxy` para conexões TLS. Quando `efs-proxy` é usado, ele abre um número de threads igual a um mais o número de núcleos do nó em que está sendo executado.
- O driver CSI do Amazon EFS não é compatível com o Amazon EKS Hybrid Nodes.

Pré-requisitos

- O driver CSI do Amazon EFS precisa de permissões do AWS Identity and Access Management (IAM).
  - A AWS sugere o uso de Identidades de Pod do EKS. Para obter mais informações, consulte the section called “Visão geral da configuração do EKS Pod Identities”.
  - Para obter informações sobre os perfis do IAM para contas de serviço e a configuração de um provedor OpenID Connect (OIDC) do IAM para o cluster, consulte the section called “Provedor OIDC do IAM”.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à

versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.

> **Note**
>
> Um pod em execução no Fargate automaticamente monta um sistema de arquivos do Amazon EFS sem precisar das etapas de instalação manual do driver.

Etapa 1: criar um perfil do IAM

O driver da CSI do Amazon EFS exige permissões do IAM para interagir com seu sistema de arquivos. Crie um perfil do IAM de cluster e associe a política gerenciada do AWS a ele. Para implementar esse procedimento, você pode usar uma das seguintes ferramentas:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”
- the section called “AWS CLI”

> **Note**
>
> As etapas específicas desse procedimento foram escritas para usar o driver como um complemento do Amazon EKS. Para obter detalhes sobre instalações autogerenciadas, consulte Set up driver permission no GitHub.

`eksctl`

Se estiver usando Identidades de Pods

Execute os comandos a seguir para criar um perfil do IAM e a Identidade de Pods com o `eksctl`. Substitua `my-cluster` pelo nome do cluster. É possível substituir `AmazonEKS_EFS_CSI_DriverRole` por um nome diferente.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
```

```
eksctl create podidentityassociation \
    --service-account-name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --permission-policy-arns arn:aws:iam::aws:policy/service-role/
AmazonEFSCSIDriverPolicy
```

Se estiver usando perfis do IAM para contas de serviço

Execute os seguintes comandos para criar um perfil IAM com o `eksctl`. Substitua `my-cluster` pelo nome do cluster. É possível substituir `AmazonEKS_EFS_CSI_DriverRole` por um nome diferente.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
    --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

Console de gerenciamento da AWS

Execute o seguinte para criar um perfil IAM com Console de gerenciamento da AWS.

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Na página Perfis, selecione Criar perfil.
4. Na página Selecionar entidade confiável, faça o seguinte:
   a. Se estiver usando Identidade de Pods do EKS:
      i. Na seção Tipo de entidade confiável), escolha Service da AWS.

ii. No menu suspenso Serviço ou caso de uso, escolha EKS.

iii. Na seção Caso de uso, escolha EKS - Identidade de Pods.

iv. Escolha Próximo.

b. Se estiver usando perfis do IAM para contas de serviço:

i. Na seção Tipo de entidade confiável, escolha Identidade da Web.

ii. Para Identity provider (Provedor de identidade), escolha OpenID Connect provider URL (URL do provedor OpenID Connect) para o seu cluster (conforme mostrado na guia Overview (Visão geral) do Amazon EKS).

iii. Em Público, escolha `sts.amazonaws.com`.

iv. Escolha Próximo.

5. Na página Add permissions (Adicionar permissões), faça o seguinte:

a. Na caixa Filtrar políticas insira `AmazonEFSCSIDriverPolicy`.

b. Marque a caixa de seleção à esquerda do `AmazonEFSCSIDriverPolicy` retornado na pesquisa.

c. Escolha Próximo.

6. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:

a. Em Nome do perfil, insira um nome exclusivo para o perfil, como `AmazonEKS_EFS_CSI_DriverRole`.

b. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.

c. Selecione Criar perfil.

7. Depois que o perfil for criado:

a. Se estiver usando Identidade de Pods do EKS:

i. Abra o console do Amazon EKS.

ii. No painel de navegação esquerdo, selecione Clusters e depois o nome do cluster para o qual você deseja configurar a associação a Identidade de Pods do EKS.

iii. Escolha a guia Acesso.

iv. Em Associações da Identidade de Pods, escolha Criar.

v. Escolha o menu suspenso Perfil do IAM e selecione o perfil que acabou de criar.

vi. Escolha o campo Namespace Kubernetes e a entrada `kube-system`.

vii. Escolha o campo Conta de serviço do Kubernetes e a entrada `efs-csi-controller-sa`.

viii. Escolha Criar.

ix. Para obter mais informações sobre a criação de associações da Identidade de Pods, consulte the section called “Criar uma associação de identidade de pod (console do AWS)”.

b. Se estiver usando perfis do IAM para contas de serviço:

i. Escolha o perfil para abri-lo para edição.

ii. Escolha a guia Relacionamentos de confiança e, em seguida, escolha Editar política de confiança.

iii. Encontre a linha semelhante à seguinte:

```
"oidc.eks.region-code.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud":
 "sts.amazonaws.com"
```

Adicione a seguinte linha acima da linha anterior. Substitua `<region-code>` pela região da AWS em que seu cluster está localizado. Substitua `<EXAMPLED539D4633E53DE1B71EXAMPLE>` pelo ID do provedor OIDC do cluster.

```
"oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-*",
```

iv. Modifique o operador `Condition` de `"StringEquals"` para `"StringLike"`.

v. Escolha Atualizar política para concluir.

AWS CLI

Execute os seguintes comandos para criar um perfil IAM com a AWS CLI.

Se estiver usando Identidades de Pods

1. Crie o perfil do IAM que concede as ações `AssumeRole` e `TagSession` ao serviço `pods.eks.amazonaws.com`.

a. Copie o conteúdo a seguir em um arquivo denominado `aws-efs-csi-driver-trust-policy-pod-identity.json`.

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

b. Crie a função. Substitua `my-cluster` pelo nome do cluster. É possível substituir `AmazonEKS_EFS_CSI_DriverRole` por um nome diferente.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy-pod-
identity.json"
```

2. Anexe a política gerenciada pela AWS necessária ao perfil com o comando a seguir.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

3. Execute o comando a seguir para criar a associação da Identidade de Pods. Substitua `arn:aws:iam::<111122223333>:role/my-role` pelo perfil criado nas etapas anteriores.

```
aws eks create-pod-identity-association --cluster-name $cluster_name --role-arn {arn-
aws}iam::<111122223333>:role/my-role --namespace kube-system --service-account efs-
csi-controller-sa
```

4. Para obter mais informações sobre a criação de associações da Identidade de Pods, consulte the section called “Criar uma associação de identidade de pod (console do AWS)”.

Se estiver usando perfis do IAM para contas de serviço

1. Exiba a URL do provedor OIDC do cluster. Substitua `my-cluster` pelo nome do cluster. É possível substituir `AmazonEKS_EFS_CSI_DriverRole` por um nome diferente.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer"
 --output text
```

   Veja abaixo um exemplo de saída.

```
https://oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>
```

   Se o resultado do comando for `None`, revise os Pré-requisitos.

2. Crie um perfil do IAM que conceda a ação `AssumeRoleWithWebIdentity`.

   a. Copie o conteúdo a seguir em um arquivo denominado `aws-efs-csi-driver-trust-policy.json`. Substitua `<111122223333>` pelo ID da sua conta. Substitua `<EXAMPLED539D4633E53DE1B71EXAMPLE>` e `<region-code>` pelos valores retornados na etapa anterior.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:efs-csi-
*",
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
```

```
    }
  ]
}
```

b. Crie a função.

```
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy.json"
```

3. Anexe a política gerenciada pela AWS necessária ao perfil com o comando a seguir.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

Etapa 2: obter o driver de CSI do Amazon EFS

Recomendamos que você instale o driver da CSI do Amazon EFS por meio do complemento Amazon EKS. Para adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criação de um complemento”. Para obter mais informações sobre complementos, consulte the section called “Complementos do Amazon EKS”. Se você não conseguir usar o complemento do Amazon EKS, recomendamos que você envie um problema sobre o motivo pelo qual não pode usar o repositório GitHub para roteiro de contêineres.

> **Important**
>
> Antes de adicionar o driver do Amazon EFS como um complemento do Amazon EKS, verifique se você não tem uma versão autogerenciada do driver instalada no seu cluster. Em caso afirmativo, consulte Uninstalling the Amazon EFS CSI Driver no GitHub.

Como alternativa, se quiser uma instalação autogerenciada do driver CSI do Amazon EFS, consulte Installation no GitHub.

Etapa 3: criar um sistema de arquivos do Amazon EFS

Para criar um sistema de arquivos do Amazon EFS, consulte Criar um sistema de arquivos do Amazon EFS para o Amazon EKS no GitHub.

Etapa 4: implantar uma aplicação de exemplo

Você pode implantar toda uma variedade de aplicações de exemplo e modificá-las conforme necessário. Para obter mais informações, consulte Exemplos no GitHub.

Usar armazenamento de aplicações de alta performance com o Amazon FSx para Lustre

O driver da Container Storage Interface (CSI) do Amazon FSx for Lustre fornece uma interface da CSI que permite que os clusters do Amazon EKS gerenciem o ciclo de vida dos sistemas de arquivos do Amazon FSx for Lustre. Para obter mais informações, consulte o Guia do usuário do Amazon FSx para Lustre.

Para obter detalhes sobre como implantar o driver de CSI do Amazon FSx para Lustre no cluster do Amazon EKS e verificar se ele funciona, consulte the section called “Implantar o driver”.

Implantar o driver do FSx para Lustre

Este tópico mostra como implantar o driver de CSI do FSx para Lustre no cluster do Amazon EKS e verificar se ele funciona. Recomendamos o uso da versão mais recente do driver. Para ver as versões disponíveis, consulte CSI Specification Compatibility Matrix (Matriz de compatibilidade de especificações de CSI) no GitHub.

> **Note**
>
> O driver não é compatível com o Fargate ou o Amazon EKS Hybrid Nodes.

Para obter descrições detalhadas dos parâmetros disponíveis e exemplos completos que demonstram os recursos do driver, consulte o projeto FSx for Lustre Container Storage Interface (CSI) driver no GitHub.

Pré-requisitos

- Um cluster existente.
- O complemento do driver CSI do Amazon FSx para o EKS requer o agente da Identidade de Pods do EKS para autenticação. Sem esse componente, o complemento falhará com o erro `Amazon`

`EKS Pod Identity agent is not installed in the cluster`, impedindo operações de volume. Instale o atendente da Identidade de Pods antes ou depois de implantar o complemento do driver CSI do FSx. Para obter mais informações, consulte the section called “Configurar o atendente”.

- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- Versão `0.215.0` ou posterior da ferramenta de linha de comando da `eksctl` instalada no seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.

Etapa 1: criar um perfil do IAM

O plug-in do CSI para Amazon FSx requer permissões do IAM para fazer chamadas para APIs da AWS em seu nome.

> **Note**
>
> Os pods terão acesso às permissões atribuídas ao perfil do IAM, a menos que você bloqueie o acesso ao IMDS. Para obter mais informações, consulte the section called “Práticas recomendadas”.

O procedimento a seguir mostra como criar um perfil do IAM e associar a política gerenciada da AWS a ele.

1. Crie um perfil do IAM e anexe a política do IAM gerenciada pela AWS com o comando a seguir. Substitua `my-cluster` pelo nome do cluster que você deseja usar. O comando implementa uma pilha do AWS CloudFormation que cria um perfil do IAM e anexa a política do IAM a ela.

```
eksctl create iamserviceaccount \
    --name fsx-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_FSx_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/AmazonFSxFullAccess \
    --approve
```

   Você verá várias linhas de resultado à medida que a conta de serviço é criada. As últimas linhas de saída são semelhantes às seguintes:

```
[#]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/fsx-csi-controller-sa",
        create serviceaccount "kube-system/fsx-csi-controller-sa",
    } }
[#]  building iamserviceaccount stack "eksctl-my-cluster-addon-iamserviceaccount-
kube-system-fsx-csi-controller-sa"
[#]  deploying stack "eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-
controller-sa"
[#]  waiting for CloudFormation stack "eksctl-my-cluster-addon-iamserviceaccount-
kube-system-fsx-csi-controller-sa"
[#]  created serviceaccount "kube-system/fsx-csi-controller-sa"
```

   Observe o nome da pilha do AWS CloudFormation que foi implantada. No resultado do exemplo anterior, a pilha é denominada `eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa`.

Agora que criou o perfil do IAM do driver CSI do Amazon FSx, você poderá avançar para a próxima seção. Quando você implanta o complemento com esse perfil do IAM, ele cria e é configurado para usar uma conta de serviço chamada `fsx-csi-controller-sa`. A conta de serviço está vinculada a um `clusterrole` do Kubernetes que recebe as permissões necessárias do Kubernetes.

Etapa 2: instalar o driver CSI do Amazon FSx

Recomendamos que você instale o driver CSI do Amazon FSx usando o complemento do Amazon EKS para reforçar a segurança e reduzir o volume do trabalho. Para adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criação de um complemento”. Para obter mais informações sobre complementos, consulte the section called “Complementos do Amazon EKS”.

> **Important**
>
> Instalações preexistentes do driver CSI do Amazon FSx no cluster podem causar falhas na instalação do complemento. Quando você tentar instalar a versão do complemento do Amazon EKS quando houver um driver CSI do FSx que não seja para o EKS, a instalação falhará devido a conflitos de recursos. Use o sinalizador `OVERWRITE` durante a instalação para resolver esse problema.
>
> ```
> aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
>  --resolve-conflicts OVERWRITE
> ```

Como alternativa, se quiser uma instalação autogerenciada do driver CSI do Amazon FSx, consulte Installation no GitHub.

Etapa 3: implantar uma classe de armazenamento, uma solicitação de volume persistente e uma aplicação de exemplo

Este procedimento usa o repositório GitHub do driver Container Storage Interface (CSI) do FSx para Lustre para consumir um volume provisionado dinamicamente do Amazon FSx para Lustre.

1. Observe o grupo de segurança do seu cluster. Você pode vê-lo em Console de gerenciamento da AWS na seção Networking ou usando o seguinte comando da AWS CLI. Substitua `my-cluster` pelo nome do cluster que você deseja usar.

   ```
   aws eks describe-cluster --name my-cluster --query
    cluster.resourcesVpcConfig.clusterSecurityGroupId
   ```

2. Crie um grupo de segurança para seu sistema de arquivos Amazon FSx conforme os critérios mostrados em Grupos de segurança de Amazon VPC, no Guia do usuário do Amazon FSx para Lustre. Para a VPC, selecione a VPC do seu cluster, conforme mostrado na seção Networking

(Redes). Para os “grupos de segurança associados aos seus clientes Lustre”, utilize o grupo de segurança do seu cluster. É possível deixar as regras de saída sozinhas para permitir Todo o tráfego.

3. Baixe o manifesto da classe de armazenamento com o seguinte comando:

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
```

4. Edite a seção de parâmetros do arquivo `storageclass.yaml`. Substitua cada valor de exemplo por seus próprios valores.

```
parameters:
  subnetId: subnet-0eabfaa81fb22bcaf
  securityGroupIds: sg-068000ccf82dfba88
  deploymentType: PERSISTENT_1
  automaticBackupRetentionDays: "1"
  dailyAutomaticBackupStartTime: "00:00"
  copyTagsToBackups: "true"
  perUnitStorageThroughput: "200"
  dataCompressionType: "NONE"
  weeklyMaintenanceStartTime: "7:09:00"
  fileSystemTypeVersion: "2.12"
```

- **`subnetId`** – o ID de da sub-rede em que o sistema de arquivos do Amazon FSx para Lustre deve ser criado. O Amazon FSx for Lustre não conta com suporte em todas as zonas de disponibilidade. Abra o console do Amazon FSx for Lustre em https://console.aws.amazon.com/fsx/ para confirmar se a sub-rede que você deseja usar está em uma zona de disponibilidade compatível. A sub-rede pode incluir os nós ou pode ser uma sub-rede ou uma VPC diferente:
  - É possível verificar as sub-redes do nó no Console de gerenciamento da AWS, selecionando o grupo de nós na seção Compute (Computação).
  - Se a sub-rede especificada não for a mesma sub-rede em que os nós estiverem localizados, as VPCs deverão estar conectadas, e você deverá garantir que as portas necessárias estejam abertas nos grupos de segurança.
- **`securityGroupIds`** – o ID do grupo de segurança criado para o sistema de arquivos.
- **`deploymentType`** (opcional): o tipo de implantação do sistema de arquivos. Os valores válidos são `SCRATCH_1`, `SCRATCH_2`, `PERSISTENT_1` e `PERSISTENT_2`. Para obter mais informações sobre os tipos de implantação, consulte Create your Amazon FSx for Lustre file system (Criar o sistema de arquivos do Amazon FSx for Lustre).

- outros parâmetros (opcionais): para obter informações sobre outros parâmetros, consulte Editar StorageClass, no GitHub.

5. Crie o manifesto da classe de armazenamento.

```
kubectl apply -f storageclass.yaml
```

Veja um exemplo de saída abaixo.

```
storageclass.storage.k8s.io/fsx-sc created
```

6. Faça download do manifesto de declaração de volume persistente.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. (Opcional) Edite o arquivo `claim.yaml`. Altere `1200Gi` para um dos seguintes valores de incremento com base nos requisitos de armazenamento e no `deploymentType` selecionado em uma etapa anterior.

```
storage: 1200Gi
```

- `SCRATCH_2` e `PERSISTENT` a `1.2 TiB`, `2.4 TiB` ou incrementos de 2,4 TiB, com mais de 2,4 TiB.
- `SCRATCH_1` – `1.2 TiB`, `2.4 TiB`, `3.6 TiB`, a , , ou incrementos de 3,6 TiB, com mais de 3,6 TiB.

8. Crie a declaração de volume persistente.

```
kubectl apply -f claim.yaml
```

Veja um exemplo de saída abaixo.

```
persistentvolumeclaim/fsx-claim created
```

9. Confirme se o sistema de arquivos está provisionado.

```
kubectl describe pvc
```

Veja um exemplo de saída abaixo.

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
[...]
```

> **Note**
>
> O `Status` pode ser exibido como `Pending` por 5 a 10 minutos, antes de mudar para `Bound`. Não prossiga para a próxima etapa até o `Status` for `Bound`. Se `Status` mostrar `Pending` por mais de 10 minutos, use mensagens de aviso em `Events` como referência para resolver problemas.

10. Implante a aplicação de exemplo.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11. Verifique se a aplicação de exemplo está em execução.

```
kubectl get pods
```

Veja um exemplo de saída abaixo.

```
NAME      READY   STATUS              RESTARTS   AGE
fsx-app   1/1     Running             0          8s
```

12. Verifique se o sistema de arquivos está montado corretamente pela aplicação.

```
kubectl exec -ti fsx-app -- df -h
```

Veja um exemplo de saída abaixo.

```
Filesystem                 Size  Used Avail Use% Mounted on
overlay                     80G  4.0G   77G   5% /
tmpfs                       64M     0   64M   0% /dev
tmpfs                      3.8G     0  3.8G   0% /sys/fs/cgroup
192.0.2.0@tcp:/abcdef01    1.1T  7.8M  1.1T   1% /data
/dev/nvme0n1p1              80G  4.0G   77G   5% /etc/hosts
```

```
shm                              64M     0   64M   0% /dev/shm
tmpfs                           6.9G   12K  6.9G   1% /run/secrets/kubernetes.io/
serviceaccount
tmpfs                           3.8G     0  3.8G   0% /proc/acpi
tmpfs                           3.8G     0  3.8G   0% /sys/firmware
```

13. Verifique se os dados foram gravados no sistema de arquivos do FSx for Lustre pela aplicação de exemplo.

```
kubectl exec -it fsx-app -- ls /data
```

Veja um exemplo de saída abaixo.

```
out.txt
```

Essa saída de exemplo indica que a aplicação de amostra gravou com êxito o arquivo `out.txt` no sistema de arquivos.

> **Note**
>
> Antes de excluir o cluster, não esqueça de excluir o sistema de arquivos do FSx for Lustre. Para obter mais informações, consulte Limpar os recursos no Guia do usuário do FSx for Lustre.

Ajuste de performance para o FSx para Lustre

Ao usar o FSx para Lustre com o Amazon EKS, você pode otimizar a performance aplicando ajustes do Lustre durante a inicialização de nós. A abordagem recomendada é usar os dados do usuário do modelo de inicialização para garantir uma configuração consistente em todos os nós.

Esses ajustes incluem:

- Otimizações de rede e RPC
- Gerenciamento do módulo Lustre
- Ajustes da LRU (unidade de bloqueio de recursos)
- Configurações de controle de cache do cliente
- Controles de RPC para OST e MDC

Para obter instruções detalhadas sobre a implementação desses ajustes de performance:

- Para otimizar a performance de nós padrão (não EFA), consulte the section called “Otimizar (não EFA)” para obter um script completo que pode ser adicionado aos dados do usuário do modelo de inicialização.
- Para otimizar a performance de nós habilitados para o EFA, consulte the section called “Otimizar (EFA)”.

Otimizar a performance do Amazon FSx para Lustre em nós (EFA)

Este tópico descreve como configurar o ajuste do Elastic Fabric Adapter (EFA) com o Amazon EKS e o Amazon FSx para Lustre.

> **Note**
>
> - Para obter informações sobre como criar e implantar o driver CSI do FSx para Lustre, consulte the section called “Implantar o driver”.
> - Para otimizar os nós padrão sem o EFA, consulte the section called “Otimizar (não EFA)”.

Etapa 1. Criar um cluster do EKS

Crie um cluster usando o arquivo de configuração fornecido:

```
# Create cluster using efa-cluster.yaml
eksctl create cluster -f efa-cluster.yaml
```

Exemplo `efa-cluster.yaml`:

```
#efa-cluster.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: csi-fsx
  region: us-east-1
  version: "1.30"
```

```
iam:
  withOIDC: true

availabilityZones: ["us-east-1a", "us-east-1d"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
    maxSize: 1
    availabilityZones: ["us-east-1b"]
    volumeSize: 300
    privateNetworking: true
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print $1}')"
        efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//g')
        min_efa_version="2.12.1"

        if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
            sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
            tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer
            echo "Installing EFA driver"
            sudo apt-get update && apt-get upgrade -y
            sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms
            sudo ./efa_installer.sh -y
            modinfo efa
        else
            echo "Using EFA driver version $efa_version"
        fi

        echo "Installing Lustre client"
        sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-
ubuntu-public-key.asc | gpg --dearmor | sudo tee /usr/share/keyrings/fsx-ubuntu-public-
key.gpg > /dev/null
```

```
        sudo echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
 https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" > /etc/apt/
sources.list.d/fsxlustreclientrepo.list
        sudo apt update | tail
        sudo apt install -y lustre-client-modules-$(uname -r) amazon-ec2-utils | tail
        modinfo lustre

        echo "Loading Lustre/EFA modules..."
        sudo /sbin/modprobe lnet
        sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
        sudo /sbin/modprobe ksocklnd
        sudo lnetctl lnet configure

        echo "Configuring TCP interface..."
        sudo lnetctl net del --net tcp 2> /dev/null
        sudo lnetctl net add --net tcp --if $eth_intf

        # For P5 instance type which supports 32 network cards,
        # by default add 8 EFA interfaces selecting every 4th device (1 per PCI bus)
        echo "Configuring EFA interface(s)..."
        instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
        num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
        echo "Found $num_efa_devices available EFA device(s)"

        if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
 "p5e.48xlarge" ]]; then
           for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
               sudo lnetctl net add --net efa --if $intf --peer-credits 32
          done
        else
        # Other instances: Configure 2 EFA interfaces by default if the instance
 supports multiple network cards,
        # or 1 interface for single network card instances
        # Can be modified to add more interfaces if instance type supports it
            sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband | head -
n1) --peer-credits 32
            if [[ $num_efa_devices -gt 1 ]]; then
               sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband | tail
 -n1) --peer-credits 32
            fi
        fi

        echo "Setting discovery and UDSP rule"
        sudo lnetctl set discovery 1
```

```
        sudo lnetctl udsp add --src efa --priority 0
        sudo /sbin/modprobe lustre

        sudo lnetctl net show
        echo "Added $(sudo lnetctl net show | grep -c '@efa') EFA interface(s)"
```

Etapa 2. Criar grupos de nós

Crie um grupo de nós habilitado para EFA:

```
# Create node group using efa-ng.yaml
eksctl create nodegroup -f efa-ng.yaml
```

> **Important**
>
> === Ajuste esses valores para seu ambiente na seção `# 5. Mount FSx filesystem`.

```
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
```

===

Exemplo `efa-ng.yaml`:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: final-efa
  region: us-east-1

managedNodeGroups:
  - name: ng-1
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
    maxSize: 1
    availabilityZones: ["us-east-1a"]
    volumeSize: 300
```

```
    privateNetworking: true
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        exec 1> >(logger -s -t $(basename $0)) 2>&1


################################################################################################
        #                                      Configuration Section
         #

################################################################################################

        # File System Configuration
        FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
        MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
        MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.

        # Lustre Tuning Parameters
        LUSTRE_LRU_MAX_AGE=600000
        LUSTRE_MAX_CACHED_MB=64
        LUSTRE_OST_MAX_RPC=32
        LUSTRE_MDC_MAX_RPC=64
        LUSTRE_MDC_MOD_RPC=50

        # File paths
        FUNCTIONS_SCRIPT="/usr/local/bin/lustre_functions.sh"
        TUNINGS_SCRIPT="/usr/local/bin/apply_lustre_tunings.sh"
        SERVICE_FILE="/etc/systemd/system/lustre-tunings.service"

        #EFA
        MIN_EFA_VERSION="2.12.1"

        # Function to check if a command was successful
        check_success() {
            if [ $? -eq 0 ]; then
                echo "SUCCESS: $1"
            else
                echo "FAILED: $1"
                return 1
            fi
        }
```

```
        echo "********Starting FSx for Lustre configuration********"

        # 1. Install Lustre client
        if grep -q '^ID=ubuntu' /etc/os-release; then
            echo "Detected Ubuntu, proceeding with Lustre setup..."
            # Add Lustre repository
            sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/
fsx-ubuntu-public-key.asc | sudo gpg --dearmor | sudo tee /usr/share/keyrings/fsx-
ubuntu-public-key.gpg > /dev/null

            echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
 https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" | sudo tee /etc/
apt/sources.list.d/fsxlustreclientrepo.list

            sudo apt-get update
            sudo apt-get install -y lustre-client-modules-$(uname -r)
            sudo apt-get install -y lustre-client
        else
            echo "Not Ubuntu, exiting"
            exit 1
        fi

        check_success "Install Lustre client"

        # Ensure Lustre tools are in the PATH
        export PATH=$PATH:/usr/sbin

        # 2. Apply network and RPC tunings
        echo "********Applying network and RPC tunings********"
        if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/
modprobe.conf; then
            echo "options ptlrpc ptlrpcd_per_cpt_max=64" | sudo tee -a /etc/modprobe.d/
modprobe.conf
            check_success "Set ptlrpcd_per_cpt_max"
        else
            echo "ptlrpcd_per_cpt_max already set in modprobe.conf"
        fi

        if ! grep -q "options ksocklnd credits" /etc/modprobe.d/modprobe.conf; then
            echo "options ksocklnd credits=2560" | sudo tee -a /etc/modprobe.d/
modprobe.conf
            check_success "Set ksocklnd credits"
        else
```

```
            echo "ksocklnd credits already set in modprobe.conf"
        fi

        # 3. Load Lustre modules
        manage_lustre_modules() {
            echo "Checking for existing Lustre modules..."
            if lsmod | grep -q lustre; then
                echo "Existing Lustre modules found."

                # Check for mounted Lustre filesystems
                echo "Checking for mounted Lustre filesystems..."
                if mount | grep -q "type lustre"; then
                    echo "Found mounted Lustre filesystems. Attempting to unmount..."
                    mounted_fs=$(mount | grep "type lustre" | awk '{print $3}')
                    for fs in $mounted_fs; do
                        echo "Unmounting $fs"
                        sudo umount $fs
                        check_success "Unmount filesystem $fs"
                    done
                else
                    echo "No Lustre filesystems mounted."
                fi

                # After unmounting, try to remove modules
                echo "Attempting to remove Lustre modules..."
                sudo lustre_rmmod
                if [ $? -eq 0 ]; then
                    echo "SUCCESS: Removed existing Lustre modules"
                else
                    echo "WARNING: Could not remove Lustre modules. They may still be
in use."
                    echo "Please check for any remaining Lustre processes or mounts."
                    return 1
                fi
            else
                echo "No existing Lustre modules found."
            fi

            echo "Loading Lustre modules..."
            sudo modprobe lustre
            check_success "Load Lustre modules" || exit 1

            echo "Checking loaded Lustre modules:"
            lsmod | grep lustre
```

```
        }

        # Managing Lustre kernel modules
        echo "********Managing Lustre kernel modules********"
        manage_lustre_modules

        # 4. Initializing Lustre networking
        echo "********Initializing Lustre networking********"
        sudo lctl network up
        check_success "Initialize Lustre networking" || exit 1

        # 4.5 EFA Setup and Configuration
        setup_efa() {
            echo "********Starting EFA Setup********"

            # Get interface and version information
            eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print
 $1}')"
            efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//
g')
            min_efa_version=$MIN_EFA_VERSION

            # Install or verify EFA driver
            if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
                echo "Installing EFA driver..."
                sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
                tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer

                # Install dependencies
                sudo apt-get update && apt-get upgrade -y
                sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms

                # Install EFA
                sudo ./efa_installer.sh -y
                modinfo efa
            else
                echo "Using existing EFA driver version $efa_version"
            fi
        }

        configure_efa_network() {
```

```
            echo "********Configuring EFA Network********"

            # Load required modules
            echo "Loading network modules..."
            sudo /sbin/modprobe lnet
            sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
            sudo /sbin/modprobe ksocklnd

            # Initialize LNet
            echo "Initializing LNet..."
            sudo lnetctl lnet configure

            # Configure TCP interface
            echo "Configuring TCP interface..."
            sudo lnetctl net del --net tcp 2> /dev/null
            sudo lnetctl net add --net tcp --if $eth_intf

            # For P5 instance type which supports 32 network cards,
            # by default add 8 EFA interfaces selecting every 4th device (1 per PCI
bus)
            echo "Configuring EFA interface(s)..."
            instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
            num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
            echo "Found $num_efa_devices available EFA device(s)"

            if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
"p5e.48xlarge" ]]; then
                # P5 instance configuration
                for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
                    sudo lnetctl net add --net efa --if $intf --peer-credits 32
                done
            else
                # Standard configuration
                # Other instances: Configure 2 EFA interfaces by default if the
instance supports multiple network cards,
                # or 1 interface for single network card instances
                # Can be modified to add more interfaces if instance type supports it
                sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband |
head -n1) --peer-credits 32
                if [[ $num_efa_devices -gt 1 ]]; then
                    sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband |
tail -n1) --peer-credits 32
                fi
            fi
```

```
        # Configure discovery and UDSP
        echo "Setting up discovery and UDSP rules..."
        sudo lnetctl set discovery 1
        sudo lnetctl udsp add --src efa --priority 0
        sudo /sbin/modprobe lustre

        # Verify configuration
        echo "Verifying EFA network configuration..."
        sudo lnetctl net show
        echo "Added $(sudo lnetctl net show | grep -c '@efa') EFA interface(s)"
    }

    # Main execution
    setup_efa
    configure_efa_network

    # 5. Mount FSx filesystem
    if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
        echo "********Creating mount point********"
        sudo mkdir -p $MOUNT_POINT
        check_success "Create mount point"

        echo "Mounting FSx filesystem..."
        sudo mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
        check_success "Mount FSx filesystem"
    else
        echo "Skipping FSx mount as DNS or mount name is not provided"
    fi

    # 6. Applying Lustre performance tunings
    echo "********Applying Lustre performance tunings********"

    # Get number of CPUs
    NUM_CPUS=$(nproc)

    # Calculate LRU size (100 * number of CPUs)
    LRU_SIZE=$((100 * NUM_CPUS))

    #Apply LRU tunings
    echo "Apply LRU tunings"
    sudo lctl set_param ldlm.namespaces.*.lru_max_age=${LUSTRE_LRU_MAX_AGE}
    check_success "Set lru_max_age"
    sudo lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
```

```
        check_success "Set lru_size"

        # Client Cache Control
        sudo lctl set_param llite.*.max_cached_mb=${LUSTRE_MAX_CACHED_MB}
        check_success "Set max_cached_mb"

        # RPC Controls
        sudo lctl set_param osc.*OST*.max_rpcs_in_flight=${LUSTRE_OST_MAX_RPC}
        check_success "Set OST max_rpcs_in_flight"

        sudo lctl set_param mdc.*.max_rpcs_in_flight=${LUSTRE_MDC_MAX_RPC}
        check_success "Set MDC max_rpcs_in_flight"

        sudo lctl set_param mdc.*.max_mod_rpcs_in_flight=${LUSTRE_MDC_MOD_RPC}
        check_success "Set MDC max_mod_rpcs_in_flight"

        # 7. Verify all tunings
        echo "********Verifying all tunings********"

        # Function to verify parameter value
        verify_param() {
            local param=$1
            local expected=$2
            local actual=$3

            if [ "$actual" == "$expected" ]; then
                echo "SUCCESS: $param is correctly set to $expected"
            else
                echo "WARNING: $param is set to $actual (expected $expected)"
            fi
        }

        echo "Verifying all parameters:"

        # LRU tunings
        actual_lru_max_age=$(lctl get_param -n ldlm.namespaces.*.lru_max_age | head -1)
        verify_param "lru_max_age" "600000" "$actual_lru_max_age"

        actual_lru_size=$(lctl get_param -n ldlm.namespaces.*.lru_size | head -1)
        verify_param "lru_size" "$LRU_SIZE" "$actual_lru_size"

        # Client Cache
        actual_max_cached_mb=$(lctl get_param -n llite.*.max_cached_mb | grep
 "max_cached_mb:" | awk '{print $2}')
```

```
        verify_param "max_cached_mb" "64" "$actual_max_cached_mb"

        # RPC Controls
        actual_ost_rpcs=$(lctl get_param -n osc.*OST*.max_rpcs_in_flight | head -1)
        verify_param "OST max_rpcs_in_flight" "32" "$actual_ost_rpcs"

        actual_mdc_rpcs=$(lctl get_param -n mdc.*.max_rpcs_in_flight | head -1)
        verify_param "MDC max_rpcs_in_flight" "64" "$actual_mdc_rpcs"

        actual_mdc_mod_rpcs=$(lctl get_param -n mdc.*.max_mod_rpcs_in_flight | head -1)
        verify_param "MDC max_mod_rpcs_in_flight" "50" "$actual_mdc_mod_rpcs"

        # Network and RPC configurations from modprobe.conf
        actual_ptlrpc=$(grep "ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf
| awk '{print $3}')
        verify_param "ptlrpcd_per_cpt_max" "ptlrpcd_per_cpt_max=64" "$actual_ptlrpc"

        actual_ksocklnd=$(grep "ksocklnd credits" /etc/modprobe.d/modprobe.conf | awk
'{print $3}')
        verify_param "ksocklnd credits" "credits=2560" "$actual_ksocklnd"

        # 8. Setup persistence
        setup_persistence() {
            # Create functions file
            cat << EOF > $FUNCTIONS_SCRIPT
        #!/bin/bash

        apply_lustre_tunings() {
            local NUM_CPUS=\$(nproc)
            local LRU_SIZE=\$((100 * NUM_CPUS))

            echo "Applying Lustre performance tunings..."
            lctl set_param ldlm.namespaces.*.lru_max_age=$LUSTRE_LRU_MAX_AGE
            lctl set_param ldlm.namespaces.*.lru_size=\$LRU_SIZE
            lctl set_param llite.*.max_cached_mb=$LUSTRE_MAX_CACHED_MB
            lctl set_param osc.*OST*.max_rpcs_in_flight=$LUSTRE_OST_MAX_RPC
            lctl set_param mdc.*.max_rpcs_in_flight=$LUSTRE_MDC_MAX_RPC
            lctl set_param mdc.*.max_mod_rpcs_in_flight=$LUSTRE_MDC_MOD_RPC
        }
        EOF

            # Create tuning script
            cat << EOF > $TUNINGS_SCRIPT
        #!/bin/bash
```

```
        exec 1> >(logger -s -t \$(basename \$0)) 2>&1

        source $FUNCTIONS_SCRIPT

        # Function to check if Lustre is mounted
        is_lustre_mounted() {
            mount | grep -q "type lustre"
        }

        # Function to mount Lustre
        mount_lustre() {
            echo "Mounting Lustre filesystem..."
            mkdir -p $MOUNT_POINT
            mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} $MOUNT_POINT
            return \$?
        }

        # Main execution
        # Try to mount if not already mounted
        if ! is_lustre_mounted; then
            echo "Lustre filesystem not mounted, attempting to mount..."
            mount_lustre
        fi

        # Wait for successful mount (up to 5 minutes)
        for i in {1..30}; do
            if is_lustre_mounted; then
                echo "Lustre filesystem mounted, applying tunings..."
                apply_lustre_tunings
                exit 0
            fi
            echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
            sleep 10
        done

        echo "Timeout waiting for Lustre filesystem mount"
        exit 1
        EOF

        # Create systemd service

        # Create systemd directory if it doesn't exist
        sudo mkdir -p /etc/systemd/system/
```

```
            # Create service file directly for Ubuntu
            cat << EOF > $SERVICE_FILE
        [Unit]
        Description=Apply Lustre Performance Tunings
        After=network.target remote-fs.target

        [Service]
        Type=oneshot
        ExecStart=/bin/bash -c 'source $FUNCTIONS_SCRIPT && $TUNINGS_SCRIPT'
        RemainAfterExit=yes

        [Install]
        WantedBy=multi-user.target
        EOF


            # Make scripts executable and enable service
            sudo chmod +x $FUNCTIONS_SCRIPT
            sudo chmod +x $TUNINGS_SCRIPT
            systemctl enable lustre-tunings.service
            systemctl start lustre-tunings.service
        }

        echo "********Setting up persistent tuning********"
        setup_persistence

        echo "FSx for Lustre configuration completed."
```

(Opcional) Etapa 3. Verificar a configuração do EFA

SSH no nó:

```
# Get instance ID from EKS console or {aws} CLI
ssh -i /path/to/your-key.pem ec2-user@<node-internal-ip>
```

Verifique a configuração do EFA:

```
sudo lnetctl net show
```

Verifique os logs de configuração:

```
sudo cat /var/log/cloud-init-output.log
```

Aqui está um exemplo de saída esperada para `lnetctl net show`:

```
net:
    - net type: tcp
      ...
    - net type: efa
      local NI(s):
        - nid: xxx.xxx.xxx.xxx@efa
          status: up
```

Implantações de exemplo

a. Crie claim.yaml

```
#claim.yaml

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fsx-claim-efa
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 4800Gi
  volumeName: fsx-pv
```

Aplique a reivindicação:

```
kubectl apply -f claim.yaml
```

b. Crie pv.yaml

Atualize a `<replaceable-placeholders>`:

```
#pv.yaml

apiVersion: v1
kind: PersistentVolume
metadata:
```

```
  name: fsx-pv
spec:
  capacity:
    storage: 4800Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  mountOptions:
    - flock
  persistentVolumeReclaimPolicy: Recycle
  csi:
    driver: fsx.csi.aws.com
    volumeHandle: fs-<1234567890abcdef0>
    volumeAttributes:
      dnsname: fs-<1234567890abcdef0>.fsx.us-east-1.amazonaws.com
      mountname: <abcdef01>
```

Aplique o volume persistente:

```
kubectl apply -f pv.yaml
```

c. Crie pod.yaml

```
#pod.yaml

apiVersion: v1
kind: Pod
metadata:
  name: fsx-efa-app
spec:
  containers:
  - name: app
    image: amazonlinux:2
    command: ["/bin/sh"]
    args: ["-c", "while true; do dd if=/dev/urandom bs=100M count=20 > data/test_file;
 sleep 10; done"]
    resources:
      requests:
        vpc.amazonaws.com/efa: 1
      limits:
        vpc.amazonaws.com/efa: 1
    volumeMounts:
    - name: persistent-storage
```

```
      mountPath: /data
  volumes:
  - name: persistent-storage
    persistentVolumeClaim:
      claimName: fsx-claim-efa
```

Aplique o pod:

```
kubectl apply -f pod.yaml
```

Comandos de verificação adicionais

Verifique se o Pod monta e grava no sistema de arquivos:

```
kubectl exec -ti fsx-efa-app -- df -h | grep data
# Expected output:
# <192.0.2.0>@tcp:/<abcdef01>  4.5T  1.2G  4.5T   1% /data

kubectl exec -ti fsx-efa-app -- ls /data
# Expected output:
# test_file
```

SSH no nó para verificar se o tráfego está passando pelo EFA:

```
sudo lnetctl net show -v
```

A saída esperada mostrará interfaces EFA com estatísticas de tráfego.

Informações relacionadas

- the section called “Implantar o driver”
- the section called “Otimizar (não EFA)”
- Performance do Amazon FSx para Lustre
- Elastic Fabric Adapter

Otimizar a performance do Amazon FSx para Lustre em nós (não EFA)

Você pode otimizar a performance do Amazon FSx para Lustre aplicando parâmetros de ajuste durante a inicialização do nó usando dados do usuário do modelo de execução.

> **Note**
>
> - Para obter informações sobre como criar e implantar o driver CSI do FSx para Lustre, consulte the section called “Implantar o driver”. Para otimizar a performance com nós habilitados pelo EFA, consulte the section called “Otimizar (EFA)”.

Por que usar os dados do usuário do modelo de execução?

- Aplica ajustes automaticamente durante a inicialização do nó.
- Garante uma configuração consistente em todos os nós.
- Elimina a necessidade de configuração manual dos nós.

Visão geral do script de exemplo

O script de exemplo definido neste tópico executa as seguintes operações:

`# 1. Install Lustre client`

- Detecta automaticamente a versão do seu sistema operacional Amazon Linux (AL).
- Instala o pacote de cliente do Lustre apropriado.

`# 2. Apply network and RPC tunings`

- Define `ptlrpcd_per_cpt_max=64` para processamento paralelo de RPC.
- Configura `ksocklnd credits=2560` para otimizar os buffers de rede.

`# 3. Load Lustre modules`

- Remove com segurança os módulos do Lustre existentes, se houver.
- Lida com a desmontagem dos sistemas de arquivos existentes.
- Carrega módulos do Lustre novos.

`# 4. Lustre Network Initialization`

- Inicializa a configuração de rede do Lustre.

- Configura os parâmetros de rede necessários.

5. Mount FSx filesystem

- Você deve ajustar os valores para o seu ambiente nesta seção.

6. Apply tunings

- Ajustes da LRU (unidade de bloqueio de recursos):
  - `lru_max_age=600000`
  - `lru_size` calculado com base na contagem de CPU
- Controle de cache do cliente: `max_cached_mb=64`
- Controles de RPC:
  - OST `max_rpcs_in_flight=32`
  - MDC `max_rpcs_in_flight=64`
  - MDC `max_mod_rpcs_in_flight=50`

7. Verify tunings

- Verifica todos os ajustes aplicados.
- Relata êxito ou uma advertência para cada parâmetro.

8. Setup persistence

- Você também deve ajustar os valores para o seu ambiente nesta seção.
- Detecta automaticamente a versão do seu sistema operacional (AL2023) para determinar qual serviço do `Systemd` deve ser aplicado.
- O sistema é iniciado.
- O `Systemd` inicia o serviço `lustre-tunings` (devido a `WantedBy=multi-user.target`).
- O serviço executa `apply_lustre_tunings.sh`, que:
  - Verifica se o sistema de arquivos está montado.
  - Monta o sistema de arquivos se não estiver montado.
  - Aguarda a montagem bem-sucedida (até cinco minutos).

  - Aplica parâmetros de ajuste após a montagem bem-sucedida.
- As configurações permanecem ativas até a reinicialização.
- O serviço é encerrado após a conclusão do script.
  - Systemd marca o serviço como "ativo (encerrado)".
- O processo se repete na próxima reinicialização.

Criar um modelo de execução

1. Abra o console do Amazon EC2 em https://console.aws.amazon.com/ec2/.
2. Escolha Modelos de execução.
3. Escolha Criar modelo de execução.
4. Em Detalhes avançados, localize a seção Dados do usuário.
5. Cole o script abaixo, atualizando o que for necessário.

> **Important**
>
> Ajuste esses valores para seu ambiente na seção `# 5. Mount FSx filesystem` e na função `setup_persistence()` de `apply_lustre_tunings.sh` na seção `# 8. Setup persistence`:
>
> ```
> FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
> MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
> MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
> ```

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="
--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Function definitions
check_success() {
    if [ $? -eq 0 ]; then
        echo "SUCCESS: $1"
    else
```

```
        echo "FAILED: $1"
        return 1
    fi
}
apply_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age=600000"
        "ldlm.namespaces.*.lru_size=$LRU_SIZE"
        "llite.*.max_cached_mb=64"
        "osc.*OST*.max_rpcs_in_flight=32"
        "mdc.*.max_rpcs_in_flight=64"
        "mdc.*.max_mod_rpcs_in_flight=50"
    )
    for param in "${params[@]}"; do
        lctl set_param $param
        check_success "Set ${param%%=*}"
    done
}
verify_param() {
    local param=$1
    local expected=$2
    local actual=$3

    if [ "$actual" == "$expected" ]; then
        echo "SUCCESS: $param is correctly set to $expected"
    else
        echo "WARNING: $param is set to $actual (expected $expected)"
    fi
}
verify_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age:600000"
        "ldlm.namespaces.*.lru_size:$LRU_SIZE"
        "llite.*.max_cached_mb:64"
        "osc.*OST*.max_rpcs_in_flight:32"
        "mdc.*.max_rpcs_in_flight:64"
        "mdc.*.max_mod_rpcs_in_flight:50"
    )
    echo "Verifying all parameters:"
    for param in "${params[@]}"; do
```

```
        name="${param%%:*}"
        expected="${param#*:}"
        actual=$(lctl get_param -n $name | head -1)
        verify_param "${name##*.}" "$expected" "$actual"
    done
}
setup_persistence() {
    # Create functions file
    cat << 'EOF' > /usr/local/bin/lustre_functions.sh
#!/bin/bash
apply_lustre_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))

    echo "Applying Lustre performance tunings..."
    lctl set_param ldlm.namespaces.*.lru_max_age=600000
    lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
    lctl set_param llite.*.max_cached_mb=64
    lctl set_param osc.*OST*.max_rpcs_in_flight=32
    lctl set_param mdc.*.max_rpcs_in_flight=64
    lctl set_param mdc.*.max_mod_rpcs_in_flight=50
}
EOF
    # Create tuning script
    cat << 'EOF' > /usr/local/bin/apply_lustre_tunings.sh
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Source the functions
source /usr/local/bin/lustre_functions.sh
# FSx details
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
# Function to check if Lustre is mounted
is_lustre_mounted() {
    mount | grep -q "type lustre"
}
# Function to mount Lustre
mount_lustre() {
    echo "Mounting Lustre filesystem..."
    mkdir -p ${MOUNT_POINT}
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    return $?
}
```

```
# Main execution
# Try to mount if not already mounted
if ! is_lustre_mounted; then
    echo "Lustre filesystem not mounted, attempting to mount..."
    mount_lustre
fi
# Wait for successful mount (up to 5 minutes)
for i in {1..30}; do
    if is_lustre_mounted; then
        echo "Lustre filesystem mounted, applying tunings..."
        apply_lustre_tunings
        exit 0
    fi
    echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
    sleep 10
done
echo "Timeout waiting for Lustre filesystem mount"
exit 1
EOF
    # Create systemd service
    cat << 'EOF' > /etc/systemd/system/lustre-tunings.service
[Unit]
Description=Apply Lustre Performance Tunings
After=network.target remote-fs.target
StartLimitIntervalSec=0
[Service]
Type=oneshot
ExecStart=/usr/local/bin/apply_lustre_tunings.sh
RemainAfterExit=yes
Restart=on-failure
RestartSec=30
[Install]
WantedBy=multi-user.target
EOF
    chmod +x /usr/local/bin/lustre_functions.sh
    chmod +x /usr/local/bin/apply_lustre_tunings.sh
    systemctl enable lustre-tunings.service
    systemctl start lustre-tunings.service
}
echo "Starting FSx for Lustre configuration..."
# 1. Install Lustre client
if grep -q 'VERSION="2"' /etc/os-release; then
    amazon-linux-extras install -y lustre
elif grep -q 'VERSION="2023"' /etc/os-release; then
```

```
    dnf install -y lustre-client
fi
check_success "Install Lustre client"
# 2. Apply network and RPC tunings
export PATH=$PATH:/usr/sbin
echo "Applying network and RPC tunings..."
if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf; then
    echo "options ptlrpc ptlrpcd_per_cpt_max=64" | tee -a /etc/modprobe.d/
modprobe.conf
    echo "options ksocklnd credits=2560" | tee -a /etc/modprobe.d/modprobe.conf
fi
# 3. Load Lustre modules
modprobe lustre
check_success "Load Lustre modules" || exit 1
# 4. Lustre Network Initialization
lctl network up
check_success "Initialize Lustre networking" || exit 1
# 5. Mount FSx filesystem
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
    mkdir -p $MOUNT_POINT
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    check_success "Mount FSx filesystem"
fi
# 6. Apply tunings
apply_tunings
# 7. Verify tunings
verify_tunings
# 8. Setup persistence
setup_persistence
echo "FSx for Lustre configuration completed."
--==MYBOUNDARY==--
```

6. Ao criar grupos de nós do Amazon EKS, selecione este modelo de execução. Para obter mais informações, consulte the section called “Criar”.

Informações relacionadas

- the section called “Implantar o driver”
- the section called “Otimizar (EFA)”

- Performance do Amazon FSx para Lustre

Usar armazenamento de aplicações de alta performance com o FSx para NetApp ONTAP

O NetApp Trident fornece orquestração de armazenamento dinâmico usando um driver compatível com a interface de armazenamento de contêineres (CSI). Isso permite que clusters do Amazon EKS gerenciem o ciclo de vida dos volumes persistentes (PVs) com suporte por sistemas de arquivo Amazon FSx para NetApp ONTAP. Observe que o Driver CSI do Amazon FSx para NetApp ONTAP não é compatível com o Amazon EKS Hybrid Nodes. Para começar, consulte Use Trident with Amazon FSx for NetApp ONTAP na documentação do NetApp Trident.

O Amazon FSx for NetApp ONTAP é um serviço de armazenamento que permite iniciar e executar sistemas de arquivos ONTAP totalmente gerenciados na nuvem. ONTAP é a tecnologia de sistema de arquivos da NetApp que fornece um conjunto, amplamente adotado, com todos os recursos de acesso a dados e gerenciamento de dados. O FSx para ONTAP fornece os recursos, a performance e as APIs dos sistemas de arquivos on-premises da NetApp com a agilidade, a escalabilidade e a simplicidade dos serviços totalmente gerenciados da AWS. Para obter mais informações, consulte o Guia do usuário do FSx for ONTAP.

Important

Se estiver usando o Amazon FSx para NetApp ONTAP junto com o driver CSI do Amazon EBS para provisionar volumes do EBS, você deve especificar que dispositivos EBS não devem ser usados no arquivo `multipath.conf`. Para ver os métodos compatíveis, consulte Lista Negra de Arquivos de Configuração. Aqui está um exemplo.

```
defaults {
       user_friendly_names yes
       find_multipaths no
     }
     blacklist {
       device {
         vendor "NVME"
         product "Amazon Elastic Block Store"
       }
     }
```

Usar armazenamento de dados com o Amazon FSx para OpenZFS

O Amazon FSx para OpenZFS é um serviço de armazenamento de arquivos totalmente gerenciado que facilita a transferência de dados do ZFS on-premises ou de outros servidores de arquivos baseados em Linux para a AWS. É possível fazer isso sem alterar o código da aplicação ou a forma como seus dados são gerenciados. Ele oferece armazenamento de arquivos altamente confiável, escalável, eficiente e rico em recursos, construído no sistema de arquivos de código aberto OpenZFS. Ele combina esses recursos com a agilidade, a escalabilidade e a simplicidade de um serviço da AWS totalmente gerenciado. Para obter mais informações, consulte o Guia do usuário do Amazon FSx para OpenZFS.

O driver Container Storage Interface (CSI) do FSx para OpenZFS fornece uma interface de CSI que permite que os clusters do Amazon EKS gerenciem o ciclo de vida de volumes do FSx para OpenZFS. Observe que o Driver da CSI do Amazon FSx para OpenZFS não é compatível com o Amazon EKS Hybrid Nodes. Para implantar o driver FSx for OpenZFS CSI em seu cluster do Amazon EKS, consulte aws-fsx-openzfs-csi-driver no GitHub.

Minimizar a latência com o Amazon File Cache

O Amazon File Cache é um cache de alta velocidade totalmente gerenciado na AWS, que é usado para processar dados de arquivos, independentemente do local em que os dados estão armazenados. O Amazon File Cache carrega os dados no cache automaticamente quando é acessado pela primeira vez e libera os dados quando não é usado. Para obter mais informações, consulte o Guia do usuário do Amazon File Cache.

O driver da interface de armazenamento de contêiner (CSI) do Amazon File Cache fornece uma interface CSI que possibilita que os clusters do Amazon EKS gerenciem o ciclo de vida dos caches de arquivos da Amazon. Observe que o driver CSI do Amazon File Cache não é compatível com o Amazon EKS Hybrid Nodes. Para implantar o driver CSI do Amazon File Cache em seu cluster do Amazon EKS, consulte aws-file-cache-csi-driver no GitHub.

Acessar objetos do Amazon S3 com o driver CSI do Mountpoint para Amazon S3

Com o driver Container Storage Interface (CSI) do Mountpoint para Amazon S3, suas aplicações do Kubernetes podem acessar objetos do Amazon S3 por meio de uma interface de sistema de arquivos, obtendo alto throughput agregado sem alterar o código de aplicação. Desenvolvido no

Mountpoint para Amazon S3, o driver CSI apresenta um bucket do Amazon S3 como um volume que pode ser acessado por contêineres no Amazon EKS e em clusters autogerenciados do Kubernetes.

Considerações

- Atualmente, o driver CSI do Mountpoint para Amazon S3 não é compatível com imagens de contêineres baseadas no Windows.
- O driver CSI do Mountpoint para Amazon S3 não é atualmente compatível com o Amazon EKS Hybrid Nodes.
- O driver CSI do Mountpoint para Amazon S3 não é compatível com o AWS Fargate. No entanto, os contêineres em execução no Amazon EC2 (com o Amazon EKS ou com uma instalação personalizada do Kubernetes) são compatíveis.
- O driver CSI do Mountpoint para Amazon S3 é compatível somente com provisionamento estático. Não há suporte ao provisionamento dinâmico ou à criação de novos buckets.

> **Note**
>
> O provisionamento estático refere-se ao uso de um bucket do Amazon S3 existente que é especificado como o `bucketName` no `volumeAttributes` no objeto `PersistentVolume`. Para obter mais informações, consulte Static Provisioning no GitHub.

- Os volumes montados com o driver CSI do Mountpoint para Amazon S3 não são compatíveis com todos os recursos do sistema de arquivos POSIX. Para obter detalhes sobre o comportamento do sistema de arquivos, consulte Mountpoint for Amazon S3 file system behavior no GitHub.

Para obter detalhes sobre a implantação do driver, consulte the section called “Implantar o driver”. Para obter detalhes sobre como remover o driver, consulte the section called “Remover o driver”.

Implantar o driver de CSI do Mountpoint para Amazon S3

Com o driver Container Storage Interface (CSI) do Mountpoint para Amazon S3, suas aplicações do Kubernetes podem acessar objetos do Amazon S3 por meio de uma interface de sistema de arquivos, obtendo alto throughput agregado sem alterar o código de aplicação.

Este procedimento mostrará como implantar o driver do Amazon EKS de CSI do Mountpoint para Amazon S3. Antes de continuar, revise as Considerações.

Pré-requisitos

- Um provedor AWS Identity and Access Management (IAM) OpenID Connect (OIDC) para o seu cluster. Para determinar se você tem ou para criar uma, consulte the section called “Provedor OIDC do IAM”.
- Versão 2.12.3 ou posterior do AWS CLI instalada e configurada em seu dispositivo ou AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.

Etapa 1: criar uma política do IAM

O driver CSI do Mountpoint para Amazon S3 precisa de permissões do Amazon S3 para interagir com seu sistema de arquivos. Esta seção mostra como criar uma política do IAM que conceda as permissões necessárias.

O exemplo de política a seguir segue as recomendações de permissão do IAM para o Mountpoint. Como alternativa, você pode usar a política AmazonS3FullAccess gerenciada pela AWS, mas ela concede mais permissões do que as necessárias para o Mountpoint.

Para obter mais informações sobre as permissões recomendadas para o Mountpoint, consulte Mountpoint IAM permissions no GitHub.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Políticas.
3. Na página Políticas, escolha Criar política.
4. Em Editor de políticas, escolha JSON.
5. No Editor de políticas, copie e cole o seguinte:

> **Important**
>
> Substitua `amzn-s3-demo-bucket1` pelo nome do seu próprio bucket do Amazon S3.

```
{
   "Version":"2012-10-17",
   "Statement": [
        {
            "Sid": "MountpointFullBucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Sid": "MountpointFullObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
        }
   ]
}
```

Os buckets de diretório, introduzidos com a classe de armazenamento Amazon S3 Express One Zone, usam um mecanismo de autenticação diferente dos buckets de uso geral. Em vez de usar ações `s3:*`, você deve usar a ação `s3express:CreateSession`. Para obter mais informações sobre os buckets do diretório, consulte Buckets de diretório no Guia do usuário do Amazon S3.

Abaixo está um exemplo de política de privilégio mínimo que você usaria para um bucket de diretório.

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/amzn-s3-
demo-bucket1--usw2-az1--x-s3"
        }
    ]
}
```

6. Escolha Próximo.
7. Na página Analisar, atribua um nome à sua política. Esta demonstração usa o nome `AmazonS3CSIDriverPolicy`.
8. Escolha Criar política.

Etapa 2: Criar uma função do IAM

O driver CSI do Mountpoint para Amazon S3 precisa de permissões do Amazon S3 para interagir com seu sistema de arquivos. Esta seção mostra como criar um perfil do IAM para delegar essas permissões. Para criar esse perfil, você pode usar uma destas ferramentas:

- the section called “eksctl”
- the section called “Console de gerenciamento da AWS”
- the section called “AWS CLI”

> **Note**
>
> A política do IAM `AmazonS3CSIDriverPolicy` foi criada na seção anterior.

eksctl

Para criar seu perfil do IAM do driver CSI do Mountpoint para Amazon S3 com o **`eksctl`**

Para criar o perfil do IAM e a conta de serviço do Kubernetes, execute os comandos a seguir. Esses comandos também anexam a política do IAM `AmazonS3CSIDriverPolicy` ao perfil, anotam a conta de serviço do Kubernetes (`s3-csi-controller-sa`) com o nome do recurso da Amazon (ARN) do perfil do IAM e adicionam o nome da conta de serviço do Kubernetes à política de confiança para o perfil do IAM.

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

Console de gerenciamento da AWS

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Na página Perfis, selecione Criar perfil.
4. Na página Select trusted entity (Selecionar entidade confiável), faça o seguinte:
   a. Na seção Tipo de entidade confiável, escolha Identidade da Web.
   b. Para Identity provider (Provedor de identidade), escolha OpenID Connect provider URL (URL do provedor OpenID Connect) para o seu cluster (conforme mostrado na guia Overview (Visão geral) do Amazon EKS).

      Se não forem exibidos URLs, revise os Pré-requisitos.
   c. Em Público, escolha `sts.amazonaws.com`.
   d. Escolha Próximo.
5. Na página Adicionar permissões, faça o seguinte:
   a. Na caixa Filtrar políticas, insira AmazonS3CSIDriverPolicy.

      > **Note**
      >
      > Essa política foi criada na seção anterior.

   b. Marque a caixa de seleção à esquerda do resultado `AmazonS3CSIDriverPolicy` retornado na pesquisa.

c. Escolha Próximo.

6. Na página Nomear, revisar e criar, faça o seguinte:

   a. Em Nome do perfil, insira um nome exclusivo para o perfil, como AmazonEKS_S3_CSI_DriverRole.

   b. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.

   c. Selecione Criar perfil.

7. Depois que a função for criada, escolha a função no console a fim de abri-la para edição.

8. Escolha a guia Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).

9. Encontre a linha semelhante à seguinte:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

   Adicione uma vírgula ao final da linha anterior e, em seguida, adicione a linha a seguir após ela. Substitua *region-code* pela região da AWS em que seu cluster se encontra. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* pelo ID do provedor OIDC do cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:s3-csi-driver-sa"
```

10. Certifique-se de que o operador `Condition` esteja definido como `"StringEquals"`.

11. Escolha Atualizar política para concluir.

AWS CLI

1. Visualize o URL do provedor OIDC do cluster. Substitua *my-cluster* pelo nome do cluster. Se o resultado do comando for None, revise os Pré-requisitos.

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

   Veja abaixo um exemplo de saída.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Crie a função do IAM, concedendo à conta de serviço do Kubernetes a ação `AssumeRoleWithWebIdentity`.

   a. Copie o conteúdo a seguir em um arquivo denominado `aws-s3-csi-driver-trust-policy.json`. Substitua *111122223333* pelo ID da sua conta. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* e *region-code* pelos valores retornados na etapa anterior.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:s3-csi-
driver-sa",
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

   b. Crie a função. Você pode alterar o *AmazonEKS_S3_CSI_DriverRole* para um nome diferente, mas, se fizer isso, certifique-se de alterá-lo também nas etapas posteriores.

```
aws iam create-role \
  --role-name AmazonEKS_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"
```

3. Anexe a política do IAM criada anteriormente ao perfil executando o comando a seguir.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole
```

> **Note**
>
> A política do IAM `AmazonS3CSIDriverPolicy` foi criada na seção anterior.

4. Pule esta etapa se você estiver instalando o driver como um complemento do Amazon EKS.. Para instalações autogerenciadas do driver, crie contas de serviço do Kubernetes que são anotadas com o ARN do perfil do IAM que você criou.

   a. Salve o conteúdo a seguir em um arquivo denominado `mountpoint-s3-service-account.yaml`. Substitua *`111122223333`* pelo ID da sua conta.

   ```
   ---
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     labels:
       app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
     name: mountpoint-s3-csi-controller-sa
     namespace: kube-system
     annotations:
       eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/
   AmazonEKS_S3_CSI_DriverRole
   ```

   b. Crie a conta de serviço do Kubernetes no cluster: A conta de serviço do Kubernetes (`mountpoint-s3-csi-controller-sa`) é anotada com o perfil do IAM que você criou denominado *`AmazonEKS_S3_CSI_DriverRole`*.

   ```
   kubectl apply -f mountpoint-s3-service-account.yaml
   ```

   > **Note**
   >
   > Quando você implanta o plug-in nesse procedimento, ele cria e é configurado para usar uma conta de serviço chamada `s3-csi-driver-sa`.

Etapa 3: instalar o driver de CSI do Mountpoint para Amazon S3

Você pode instalar o driver CSI do Mountpoint para Amazon S3 por meio do complemento do Amazon EKS. Você pode usar as seguintes ferramentas para adicionar o complemento ao seu cluster:

- the section called “eksctl”
- the section called “Console de gerenciamento da AWS”
- the section called “AWS CLI”

Você também pode instalar o driver CSI do Mountpoint para Amazon S3 como uma instalação autogerenciada. Para obter instruções sobre como realizar uma instalação autogerenciada, consulte Installation no GitHub.

A partir da versão `v1.8.0`, é possível configurar taints para tolerar os pods do driver CSI. Para fazer isso, especifique um conjunto personalizado de taints para tolerar `node.tolerations` ou tolere todos os taints com `node.tolerateAllTaints`. Para obter mais informações, consulte Taints and Tolerations na documentação do Kubernetes.

eksctl

Para adicionar o complemento do Amazon S3 CSI usando **`eksctl`**

Execute o comando a seguir. Substitua *`my-cluster`* pelo nome do seu cluster, *`111122223333`* pelo ID da sua conta e *`AmazonEKS_S3_CSI_DriverRole`* pelo nome do perfil do IAM criado anteriormente.

```
eksctl create addon --name aws-mountpoint-s3-csi-driver --cluster my-cluster \
  --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
 --force
```

Se você remover a opção *`--force`* e alguma das configurações do complemento do Amazon EKS entrar em conflito com as configurações existentes, a atualização do complemento do Amazon EKS falhará e você receberá uma mensagem de erro para ajudá-lo a resolver o conflito. Antes de especificar essa opção, certifique-se de que o complemento Amazon EKS não gerencie as configurações que você precisa gerenciar, pois essas configurações são substituídas por essa opção. Para obter mais informações sobre outras opções para essa configuração, consulte Addons (Complementos) na documentação do `eksctl`. Para obter mais informações sobre o gerenciamento

de campos do Kubernetes do Amazon EKS, consulte the section called “Campos que você pode personalizar”.

É possível personalizar o `eksctl` por meio de arquivos de configuração. Para obter mais informações, consulte Trabalhar com valores de configuração na documentação do `eksctl`. O seguinte exemplo de mostra como tolerar todos os taints.

```
# config.yaml
...

addons:
- name: aws-mountpoint-s3-csi-driver
  serviceAccountRoleARN: arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
  configurationValues: |-
    node:
      tolerateAllTaints: true
```

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome do cluster para o qual você deseja configurar o complemento CSI do Mountpoint para Amazon S3.
4. Escolha a guia Add-ons (Complementos).
5. Escolha Obter mais complementos.
6. Na página Selecionar complementos, faça o seguinte:
   a. Na seção Complementos do Amazon EKS, marque a caixa de seleção Driver CSI do Mountpoint para Amazon S3.
   b. Escolha Próximo.
7. Na página Definir as configurações dos complementos selecionados:
   a. Selecione a Versão que deseja usar.
   b. Em Selecionar perfil do IAM, selecione o nome do perfil do IAM ao qual você anexou a política do IAM do driver CSI do Mountpoint para Amazon S3.
   c. (Opcional) Atualize o Método de resolução de conflitos após expandir as Configurações opcionais. Se você selecionar Substituir, uma ou mais configurações do complemento existente poderão ser substituídas pelas configurações do complemento do Amazon EKS. Se você não habilitar esta opção e houver um conflito com suas configurações existentes, a operação

falhará. É possível usar a mensagem de erro resultante para solucionar o conflito. Antes de selecionar essa opção, certifique-se de que o complemento do Amazon EKS não gerencie as configurações que você precisa autogerenciar.

d. (Opcional) Configure as tolerâncias no campo Valores de configuração depois de expandir as Configurações opcionais.

e. Escolha Próximo.

8. Na página Adicionar tags, escolha Criar. Depois que a instalação do complemento for concluída, você verá o complemento instalado.

AWS CLI

Para adicionar o complemento CSI do Mountpoint para Amazon S3 usando a AWS CLI

Execute o comando a seguir. Substitua *my-cluster* pelo nome do cluster, *111122223333* pelo ID da conta e *AmazonEKS_S3_CSI_DriverRole* pelo nome do perfil criada anteriormente.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
```

É possível personalizar o comando com o sinalizador `--configuration-values`. O exemplo alternativo a seguir mostra como tolerar todos os taints.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
 \
  --configuration-values '{"node":{"tolerateAllTaints":true}}'
```

Etapa 4: xonfigurar o Mountpoint para Amazon S3

Na maioria dos casos, você pode configurar o Mountpoint para Amazon S3 com apenas um nome de bucket. Para obter instruções sobre como configurar o Mountpoint para Amazon S3, consulte Configuring Mountpoint for Amazon S3 no GitHub.

Etapa 5: implantar uma aplicação de exemplo

É possível implantar provisionamento estático para o driver em um bucket do Amazon S3 existente. Para obter mais informações, consulte Static Provisioning no GitHub.

Remover o complemento Mountpoint para Amazon S3 do Amazon EKS

Você tem duas opções para remover o driver de CSI do Mountpoint para Amazon S3.

- Preserve add-on software on your cluster (Preservar software de complemento no cluster): esta opção remove o gerenciamento do Amazon EKS de qualquer configuração. Também remove a capacidade do Amazon EKS de notificar você sobre atualizações e de atualizar automaticamente o complemento do Amazon EKS depois de iniciar uma atualização. No entanto, ele preserva o software de complemento em seu cluster. Essa opção torna o complemento em uma instalação autogerenciada, em vez de um complemento do Amazon EKS. Com essa opção, não há tempo de inatividade para o complemento. Os comandos deste procedimento usam essa opção.
- Remover completamente o software do complemento do cluster: recomendamos remover o complemento do Amazon EKS do cluster se não houver recursos no cluster que dependam dele. Para fazer essa opção, exclua `--preserve` a partir do comando que você utiliza neste procedimento.

Se não houver uma conta do IAM associada ao complemento, a conta do IAM não será removida.

Você pode usar as seguintes ferramentas para remover o complemento do Amazon S3 CSI:

- the section called “eksctl”
- the section called “Console de gerenciamento da AWS”
- the section called “AWS CLI”

eksctl

Para remover o complemento do Amazon S3 CSI usando **`eksctl`**

Substitua *`my-cluster`* pelo nome do cluster, e depois execute comando a seguir.

```
eksctl delete addon --cluster my-cluster --name aws-mountpoint-s3-csi-driver --preserve
```

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome do cluster de onde você deseja remover o complemento CSI do Amazon EBS.

4. Escolha a guia Add-ons (Complementos).
5. Escolha Driver CSI do Mountpoint para Amazon S3.
6. Escolha Remover.
7. Na caixa de diálogo de confirmação Remover: aws-mountpoint-s3-csi-driver, faça o seguinte:
   a. Se quiser que o Amazon EKS pare de gerenciar as configurações do complemento, selecione Preservar no cluster. Faça isso se quiser manter o software de complemento em seu cluster. Isso serve para que você possa gerenciar todas as configurações do complemento por conta própria.
   b. Insira `aws-mountpoint-s3-csi-driver`.
   c. Selecione Remover.

AWS CLI

Para remover o complemento do Amazon S3 CSI usando a AWS CLI

Substitua *`my-cluster`* pelo nome do cluster, e depois execute comando a seguir.

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver --preserve
```

Habilitar a funcionalidade de snapshot para volumes CSI

A funcionalidade de snapshot permite fazer cópias pontuais dos seus dados. Para que esse recurso funcione no Kubernetes, você precisa de um driver CSI compatível com snapshots (como o driver CSI do Amazon EBS) e um controlador CSI de snapshots. O controlador de snapshot está disponível como complemento gerenciado do Amazon EKS ou como uma instalação autogerenciada.

Veja a seguir algumas considerações ao usar o controlador de snapshots da CSI.

- O controlador de snapshots deve ser instalado junto com um driver de CSI com funcionalidade de snapshot instantâneo. Para obter instruções sobre como instalar o driver de CSI do Amazon EBS no cluster, consulte the section called “Amazon EBS”.
- O Kubernetes não é compatível com snapshots de volumes que estão sendo fornecidos por meio da migração da CSI, como volumes do Amazon EBS que usam uma `StorageClass` com provisionador do `kubernetes.io/aws-ebs`. Os volumes devem ser criados com um `StorageClass` que faça referência ao provisionador do driver da CSI, `ebs.csi.aws.com`.

- O Modo Automático do Amazon EKS não inclui o controlador de snapshots. A capacidade de armazenamento do Modo Automático do EKS é compatível com o controlador de snapshots.

Recomendamos instalar o controlador de snapshots da CSI via complemento gerenciado do Amazon EKS. Esse complemento inclui as definições de recursos personalizados (CRDs) que são necessárias para criar e gerenciar snapshot no Amazon EKS. Para adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criação de um complemento”. Para obter mais informações sobre complementos, consulte the section called “Complementos do Amazon EKS”.

Como alternativa, se você quiser uma instalação autogerenciada do controlador CSI de snapshots, consulte Usage no Upstream Kubernetes `external-snapshotter` no GitHub.

Configurar redes para clusters do Amazon EKS

Seu cluster do Amazon EKS é criado em uma VPC. A rede de pods é fornecida pelo plug-in da interface de rede de contêineres (CNI) da Amazon VPC para nós executados na infraestrutura da AWS. Se você estiver executando nós em sua própria infraestrutura, consulte the section called "Configurar CNI". Este capítulo inclui os tópicos a seguir para aprender mais sobre redes para o cluster.

Tópicos

- Adicionar uma sub-rede da VPC existente a um cluster do Amazon EKS no console de gerenciamento
- Exibir os requisitos de rede do Amazon EKS para VPC e sub-redes
- Criar uma Amazon VPC para o cluster do Amazon EKS
- Exibir os requisitos para grupos de segurança do Amazon EKS em clusters
- Gerenciar os complementos de redes para os clusters do Amazon EKS

Adicionar uma sub-rede da VPC existente a um cluster do Amazon EKS no console de gerenciamento

1. Navegue até seu cluster no console de gerenciamento
2. Na guia Rede, selecione Gerenciar recursos de VPC
3. No menu suspenso Sub-redes, selecione sub-redes adicionais da VPC do cluster.

Para criar uma nova sub-rede da VPC:

- Analise os requisitos de sub-rede do EKS
- Consulte Criar uma sub-rede no Guia do usuário da Amazon Virtual Private Cloud.

Exibir os requisitos de rede do Amazon EKS para VPC e sub-redes

Ao criar um cluster, você especifica uma VPC e pelo menos duas sub-redes em diferentes zonas de disponibilidade. Este tópico fornece uma visão geral dos requisitos e das considerações específicos do Amazon EKS para a VPC e as sub-redes que você utiliza com o cluster. Se você não tiver uma

VPC para usar com o Amazon EKS, consulte the section called “Crie uma VPC”. Se estiver criando um cluster local ou estendido em AWS Outposts, consulte the section called “Criar uma VPC e sub-redes” em vez deste tópico. O conteúdo deste tópico se aplica aos clusters do Amazon EKS com nós híbridos. Para obter requisitos adicionais de rede para nós híbridos, consulte the section called “Preparar a rede”.

Requisitos e considerações para VPCs

Quando você cria um cluster, a VPC especificada deve atender aos requisitos e considerações a seguir:

- A VPC deve ter uma quantidade suficiente de endereços IP disponíveis para o cluster, todos os nós e outros recursos Kubernetes que você deseja criar. Se a VPC que você deseja utilizar não tiver uma quantidade suficiente de endereços IP, tente aumentar a quantidade de endereços IP disponíveis.

  Você pode fazer isso atualizando a configuração do cluster para alterar as sub-redes e grupos de segurança que o cluster utiliza. Você pode atualizar a partir do Console de gerenciamento da AWS, da versão mais recente da AWS CLI, do AWS CloudFormation e do `eksctl` versão `v0.164.0-rc.0` ou posterior. Talvez seja necessário fazer isso para fornecer às sub-redes mais endereços IP disponíveis para atualizar com sucesso uma versão de cluster.

  > **Important**
  >
  > Todas as sub-redes adicionadas devem estar no mesmo conjunto de AZs fornecido na ocasião da criação do cluster. As novas sub-redes devem atender a todos os outros requisitos, por exemplo, devem ter endereços IP suficientes.
  > Por exemplo, suponha que você tenha feito um cluster e especificado quatro sub-redes. Na ordem em que você as especificou, a primeira sub-rede está na Zona de disponibilidade `us-west-2a`, a segunda e a terceira sub-redes estão na Zona de disponibilidade `us-west-2b` e a quarta sub-rede está na Zona de disponibilidade `us-west-2c`. Se quiser alterar as sub-redes, você deverá fornecer pelo menos uma sub-rede em cada uma das três zonas de disponibilidade, e as sub-redes deverão estar na mesma VPC que as sub-redes originais.

  Se você precisar de mais endereços IP do que os blocos CIDR na VPC, você pode adicionar mais blocos CIDR associando blocos adicionais de Encaminhamento Entre Domínios Sem Classificação

(CIDR) à sua VPC. É possível associar blocos CIDR privados (RFC 1918) e públicos (não RFC 1918) à VPC antes ou depois de criar o cluster.

É possível adicionar nós que usem o novo bloco CIDR imediatamente após adicioná-lo. No entanto, como o ambiente de gerenciamento reconhece o novo bloco CIDR apenas após a conclusão da reconciliação, pode levar até uma hora para que um bloco CIDR que você associou a uma VPC seja reconhecido por um cluster. Em seguida, é possível executar os comandos `kubectl attach`, `kubectl cp`, `kubectl exec`, `kubectl logs` e `kubectl port-forward` (estes comandos usam a `kubelet API`) para nós e pods no novo bloco CIDR. Além disso, caso tenha pods que operam como um backend de webhook, você deverá aguardar a conclusão da reconciliação do ambiente de gerenciamento.

- Evite sobreposições de intervalos de endereços IP ao conectar seu cluster do EKS a outras VPCs por meio do Transit Gateway, do emparelhamento de VPC ou de outras configurações de rede. Os conflitos de CIDR ocorrem quando o CIDR de serviço do seu cluster do EKS se sobrepõe ao CIDR de uma VPC conectada. Nesses cenários, os endereços de ServiceIP têm prioridade sobre os recursos em VPCs conectadas com o mesmo endereço IP, embora o roteamento de tráfego possa se tornar imprevisível e as aplicações possam falhar em se conectar aos recursos pretendidos.

  Para evitar conflitos de CIDR, garanta que seu CIDR de serviço do EKS não se sobreponha a nenhum CIDR de VPC conectado e mantenha um registro centralizado de todas as atribuições de CIDR. Se você encontrar sobreposições de CIDR, poderá usar um gateway de trânsito com uma VPC de serviços compartilhados. Para obter mais informações, consulte Isolated VPCs with shared services (VPCs isoladas com serviços compartilhados) e Amazon EKS VPC routable IP address conservation patterns in a hybrid network (Padrões de conservação de endereço IP roteáveis da VPC no Amazon EKS em uma rede híbrida). Além disso, consulte a seção Comunicação entre VPCs da página Considerações sobre VPC e sub-rede no Guia de melhores práticas do EKS.
- Se quiser que o Kubernetes atribua endereços `IPv6` para pods e serviços, associe um bloco CIDR `IPv6` à VPC. Para obter mais informações, consulte Associar um bloco CIDR IPv6 à sua VPC no Guia do usuário da Amazon VPC. Você não pode usar endereços `IPv6` com pods e serviços executados em nós híbridos, e não pode usar nós híbridos com clusters configurados com a família de endereços IP `IPv6`.
- A VPC deve ter suporte de nomes de host de `DNS` e resolução de `DNS`. Caso contrário, os nós não poderão se registrar no seu cluster. Para mais informações, consulte Atributos de DNS para sua VPC, no Guia do usuário da Amazon VPC.

- A VPC pode exigir endpoints da VPC usando AWS PrivateLink. Para obter mais informações, consulte the section called “Requisitos e considerações para sub-redes”.

Se você criou um cluster com o Kubernetes `1.14` ou anterior, o Amazon EKS adicionou a seguinte etiqueta à sua VPC:

Chave	Valor
`kubernetes.io/cluster/` *`my-cluster`*	`owned`

Essa etiqueta foi utilizada apenas pelo Amazon EKS. É possível remover a etiqueta sem afetar os serviços. Ela não é utilizada com clusters que são da versão `1.15` ou posterior.

Requisitos e considerações para sub-redes

Quando você cria um cluster, o Amazon EKS cria de 2 a 4 interfaces de rede elástica nas sub-redes especificadas. Essas interfaces permitem a comunicação entre seu cluster e sua VPC. Essas interfaces de rede também habilitam recursos do Kubernetes que usam a `kubelet API`. As conexões com a API do `kubelet` são usadas nos comandos `kubectl attach`, `kubectl cp`, `kubectl exec`, `kubectl logs` e `kubectl port-forward`. Cada interface de rede criada pelo Amazon EKS tem o texto `Amazon EKS` *`cluster-name`* em sua descrição.

O Amazon EKS é capaz de criar suas interfaces de rede em qualquer sub-rede especificada ao criar um cluster. É possível alterar em quais sub-redes o Amazon EKS cria suas interfaces de rede depois que o cluster é criado. Quando você atualiza a versão do Kubernetes de um cluster, o Amazon EKS exclui as interfaces de rede originais criadas e cria novas. Essas interfaces podem ser criadas nas mesmas sub-redes que a das interfaces de rede originais ou em sub-redes diferentes das originais. Para controlar quais interfaces de rede de sub-redes são criadas, você poderá limitar o número de sub-redes especificadas a apenas duas ao criar um cluster ou atualizar as sub-redes depois de criar o cluster.

Requisitos de sub-redes para clusters

As sub-redes que você especifica ao criar ou atualizar um cluster dovem atender aos requisitos a seguir:

- As sub-redes devem ter no mínimo seis endereços IP para uso pelo Amazon EKS. Porém, recomendamos no mínimo 16 endereços IP.

- As sub-redes devem estar em ao menos duas diferentes zonas de disponibilidade.
- As sub-redes não podem residir no AWS Outposts ou no AWS Wavelength. Porém, se você as tiver na sua VPC, poderá implantar nós autogerenciados e recursos do Kubernetes para esses tipos de sub-redes. Para obter mais informações sobre nós autogerenciados, consulte the section called “Nós autogerenciados”.
- As sub-redes podem ser do tipo público ou privado. Convém, se possível, especificar sub-redes privadas. Uma sub-rede pública tem uma tabela de rotas que inclui uma rota para um gateway da Internet, enquanto uma sub-rede privada tem uma tabela de rotas que não inclui uma rota para um gateway da Internet.
- As sub-redes não podem residir nas seguintes zonas de disponibilidade:

Região da AWS	Nome da região	IDs de zona de disponibilidade não permitidas
`us-east-1`	Leste dos EUA (Norte da Virgínia)	`use1-az3`
`us-west-1`	Oeste dos EUA (N. da Califórnia)	`usw1-az2`
`ca-central-1`	Canadá (Central)	`cac1-az3`

Uso da família de endereços IP por componente

A tabela a seguir contém a família de endereços IP usada por cada componente do Amazon EKS. Você pode usar uma conversão de endereços de rede (NAT) ou outros sistemas de compatibilidade para se conectar a esses componentes usando endereços IP de origem em famílias com o valor "No" para uma entrada de tabela.

A funcionalidade pode diferir, dependendo da configuração da família de IP (`ipFamily`) do cluster. Essa configuração altera o tipo de endereços IP usados para o bloco CIDR que o Kubernetes atribui para serviços. Um cluster com o valor de configuração de IPv4 é referido como um cluster do IPv4, e um cluster com o valor de configuração IPv6 é referido como um cluster do IPv6.

Componente	Endereços IPv4	Endereços IPv6	Endereços de pilha dupla
Endpoint público da API EKS	Sim[1,3]	Sim[1,3]	Sim[1,3]
Endpoint da VPC da API EKS	Sim	Não	Não
Endpoint público da API do EKS Auth (EKS Pod Identity)	Sim[1]	Sim[1]	Sim[1]
Endpoint da VPC da API do EKS Auth (EKS Pod Identity)	Sim[1]	Sim[1]	Sim[1]
Endpoint público do cluster do `IPv4` do Kubernetes[2]	Sim	Não	Não
Endpoint privado do cluster do `IPv4` do Kubernetes[2]	Sim	Não	Não
Endpoint público do cluster do `IPv6` do Kubernetes[2]	Sim[1,4]	Sim[1,4]	Sim[4]
Endpoint privado do cluster do `IPv6` do Kubernetes[2]	Sim[1,4]	Sim[1,4]	Sim[4]
Sub-redes de clusters do Kubernetes	Sim[2]	Não	Sim[2]
Endereços IP primários do nó	Sim[2]	Não	Sim[2]

Componente	Endereços IPv4	Endereços IPv6	Endereços de pilha dupla
Intervalo CIDR do cluster para endereços IP de serviços	Sim[2]	Sim[2]	Não
Endereços IP de pods da CNI da VPC	Sim[2]	Sim[2]	Não
URLs do emissor do OIDC para IRSA	Sim [1,3]	Sim [1,3]	Sim [1,3]

Note

[1] O endpoint é uma pilha dupla com endereços `IPv4` e `IPv6`. As aplicações externas à AWS, os nós do cluster e os pods dentro do cluster podem obter acesso a esse endpoint usando `IPv4` ou `IPv6`.

[2] Você escolhe entre um cluster do `IPv4` e um cluster do `IPv6` na configuração da família de IP (`ipFamily`) do cluster ao criar um cluster, e isso não poderá ser alterado. Em vez disso, você deve escolher uma configuração diferente ao criar outro cluster e migrar as workloads.

[3] O endpoint de pilha dupla foi introduzido em agosto de 2024. Para usar os endpoints de pilha dupla com a AWS CLI, consulte a configuração de endpoints de pilha dupla e FIPS no Guia de referência de SDKs e ferramentas do AWS. A seguinte lista contém os novos endpoints:

Endpoint público da API EKS

```
eks.region.api.aws
```

URLs do emissor do OIDC para IRSA

```
oidc-eks.region.api.aws
```

[4] O endpoint de cluster do pilha dupla foi introduzido em outubro de 2024. O EKS cria o endpoint a seguir para novos clusters criados após essa data e que selecionam `IPv6` na configuração da família de IPs (ipFamily) do cluster:

Endpoint público/privado do cluster EKS

```
eks-cluster.region.api.aws
```

Requisitos de sub-redes para nós

É possível implantar nós e recursos Kubernetes nas mesmas sub-redes que você especifica ao criar seu cluster. Porém, isso não é necessário. Isso porque também é possível implantar nós e recursos do Kubernetes em sub-redes que você não especificou quando criou o cluster. Se você implantar nós em sub-redes diferentes, o Amazon EKS não criará interfaces de rede de cluster nelas. Qualquer sub-rede na qual você implantar nós e recursos Kubernetes deve atender aos seguintes requisitos:

- As sub-redes devem ter endereços IP disponíveis suficientes nos quais implantar todos os seus nós e recursos Kubernetes.
- Se quiser que o Kubernetes atribua endereços `IPv6` a pods e serviços, você deverá ter um bloco CIDR `IPv6` e um bloco CIDR `IPv4` associados à sub-rede. Para obter mais informações, consulte Associar um bloco CIDR IPv6 à sua sub-rede no Guia do usuário do Amazon VPC. As tabelas de rotas associadas às sub-redes devem incluir rotas para endereços `IPv4` e `IPv6`. Para obter mais informações, consulte Rotas, no Guia do usuário da Amazon VPC. Pods recebem apenas um endereço `IPv6`. Porém, as interfaces de rede que criadas pelo Amazon EKS para o seu cluster e os seus nós recebem um endereço `IPv4` e um endereço `IPv6`.
- Se precisar de acesso de entrada na internet para os pods, certifique-se de ter pelo menos uma sub-rede pública com endereços IP disponíveis suficientes nos quais implantar balanceadores de carga e entradas. Você pode implantar balanceadores de carga em sub-redes públicas. Balanceadores de carga podem balancear a carga para pods em sub-redes privadas ou públicas. Convém implantar nós em sub-redes privadas, se possível.
- Se você planeja implantar nós em uma sub-rede pública, esta deverá atribuir automaticamente endereços públicos `IPv4` ou endereços `IPv6`. Se você implantar nós em uma sub-rede privada que tenha um bloco CIDR `IPv6` associado, esta também deverá atribuir endereços `IPv6` automaticamente. Se tiver usado um modelo do AWS CloudFormation fornecido pelo Amazon EKS para implantar a VPC após 26 de março de 2020, essa configuração estará habilitada. Se você utilizou os modelos para implantar sua VPC antes dessa data ou se utilizar sua própria VPC, deverá habilitar essa configuração manualmente. Para obter o modelo, consulte the section called “Crie uma VPC”. Para obter mais informações, consulte Modify the public IPv4 addressing

attribute for your subnet e Modify the IPv6 addressing attribute for your subnet no Guia do usuário da Amazon VPC.

- Se a sub-rede na qual você implanta um nó for uma sub-rede privada e sua tabela de rotas não incluir uma rota para um dispositivo de conversão de endereços de rede (NAT) (`IPv4`) ou um gateway somente de saída (`IPv6`), adicione endpoints da VPC usando o AWS PrivateLink à sua VPC. Os endpoints da VPC são necessários para todos os serviços da AWS com os quais os nós e os pods precisam se comunicar. Os exemplos incluem o Amazon ECR, o Elastic Load Balancing, o Amazon CloudWatch, o AWS Security Token Service e o Amazon Simple Storage Service (Amazon S3). O endpoint deve incluir a sub-rede na qual os nós se encontram. Nem todos os serviços do AWS oferecem suporte a endpoints de VPC. Para obter mais informações, consulte O que é o AWS PrivateLink? e AWS serviços que se integram ao AWS PrivateLink. Para obter uma lista de mais requisitos para o Amazon EKS, consulte the section called “Clusters privados”.
- Se quiser implantar balanceadores de carga em uma sub-rede, esta deve ter a seguinte etiqueta:
  - Sub-redes privadas

Chave	Valor
`kubernetes.io/role/internal-elb`	`1`

  - Sub-redes públicas

Chave	Valor
`kubernetes.io/role/elb`	`1`

Quando um cluster do Kubernetes da versão `1.18` e anteriores foi criado, o Amazon EKS adicionou a tag a seguir a todas as sub-redes que foram especificadas.

Chave	Valor
`kubernetes.io/cluster/` *`my-cluster`*	`shared`

Quando você cria um cluster do Kubernetes agora, o Amazon EKS não adiciona a tag às sub-redes. Se a tag estava em sub-redes usadas por um cluster que era previamente uma versão anterior

à `1.19`, a tag não foi removida automaticamente das sub-redes quando o cluster foi atualizado para uma versão mais recente. A versão `2.1.1` ou anterior do AWS Load Balancer Controller exige essa tag. Se você estiver usando uma versão mais recente do controlador do balanceador de carga, poderá remover a tag sem interromper seus serviços. Para obter mais informações sobre o controlador, consulte the section called “ AWS Load Balancer Controller”.

Se você implantou uma VPC usando `eksctl` ou qualquer um dos modelos de VPC do Amazon EKS AWS CloudFormation, aplica-se o seguinte:

- Em ou após 26 de março de 2020: os endereços `IPv4` públicos serão atribuídos automaticamente por sub-redes públicas aos novos nós implantados em sub-redes públicas.
- Em ou antes de 26 de março de 2020: os endereços `IPv4` públicos não serão atribuídos automaticamente por sub-redes públicas aos novos nós implantados em sub-redes públicas.

Essa alteração afeta os novos grupos de nós implantados em sub-redes públicas das seguintes maneiras:

- Grupos de nós gerenciados: se o grupo de nós for implantado em uma sub-rede pública em ou após 22 de abril de 2020, a atribuição automática de endereços IP públicos deverá ser habilitada para a sub-rede pública. Para obter mais informações, consulte Modificar o atributo de endereçamento IPv4 público para a sub-rede.
- Grupos de nós autogerenciados de Linux, Windows ou Arm: se o grupo de nós for implantado em uma sub-rede pública em ou após 26 de março de 2020, a atribuição automática de endereços IP públicos deverá ser ativada para a sub-rede pública. Caso contrário, os nós deverão ser iniciados com um endereço IP público. Para obter mais informações, consulte Modificar o atributo de endereçamento IPv4 público para a sub-rede ou Atribuir um endereço IPv4 público durante a execução da instância.

Requisitos e considerações de sub-redes

Você pode usar o compartilhamento de VPC para compartilhar sub-redes com outras contas AWS dentro das mesmas organizações AWS. Você pode criar clusters do Amazon EKS em sub-redes compartilhadas com as seguintes considerações:

- O proprietário da sub-rede VPC deve compartilhar uma sub-rede com uma conta de participante antes que essa conta possa criar nela um cluster do Amazon EKS.

- Você não pode iniciar recursos usando o grupo de segurança padrão da VPC, pois ele pertence ao proprietário. Além disso, os participantes não podem iniciar recursos usando grupos de segurança de propriedade de outros participantes ou do proprietário.
- Em uma sub-rede compartilhada, o participante e o proprietário controlam separadamente os grupos de segurança na respectiva conta. O proprietário da sub-rede pode visualizar esses grupos de segurança criados pelos participantes, mas não pode realizar neles nenhuma ação. Se o proprietário da sub-rede quiser remover ou modificar esses grupos de segurança, o participante que criou o grupo de segurança deve realizar a ação.
- Se um cluster for criado por um participante, as seguintes considerações se aplicam:
  - A perfil do IAM do cluster e perfis do IAM do nó devem ser criados na conta. Para obter mais informações, consulte the section called “Função do IAM do cluster” e the section called “Perfil do IAM do nó”.
  - Todos os nós devem ser criados pelo mesmo participante, incluindo grupos de nós gerenciados.
- O proprietário da VPC compartilhada não pode visualizar, atualizar ou excluir um cluster criado por um participante na sub-rede compartilhada. Isso é além dos recursos da VPC aos quais cada conta tem acesso diferente. Para obter mais informações, consulte Responsabilidades e permissões para proprietários e participantes no Manual do usuário do Amazon VPC.
- Se usar o recurso de rede personalizada do plug-in CNI da Amazon VPC para Kubernetes, você precisará usar os mapeamentos de IDs das zonas de disponibilidade listados na conta do proprietário para criar cada `ENIConfig`. Para obter mais informações, consulte the section called “Redes personalizadas”.

Para obter informações sobre o compartilhamento de sub-rede da VPC, consulte Compartilhar sua VPC com outras contas no Manual do usuário do Amazon VPC.

Criar uma Amazon VPC para o cluster do Amazon EKS

É possível usar a Amazon Virtual Private Cloud (Amazon VPC) para iniciar recursos da AWS em uma rede virtual definida por você. Essa rede virtual é muito semelhante a uma rede tradicional que pode ser operada no seu próprio data center. Porém, ela vem com os benefícios do uso da infraestrutura escalável da Amazon Web Services. É recomendável ter uma profunda compreensão do serviço Amazon VPC antes de implantar clusters do Amazon EKS em ambientes de produção. Para obter mais informações, consulte o Manual do usuário da Amazon VPC.

Um cluster, nós e recursos Kubernetes do Amazon EKS são implantados em uma VPC. Se quiser utilizar uma VPC existente com o Amazon EKS, essa VPC deverá atender aos requisitos descritos

em the section called “Requisitos para sub-rede e VPC”. Este tópico descreve como criar uma VPC que atenda aos requisitos do Amazon EKS usando um modelo do AWS CloudFormation fornecido pelo Amazon EKS. Depois de implantar um modelo, você pode visualizar os recursos criados por ele para saber exatamente quais recursos foram criados e a configuração desses recursos. Se você estiver usando nós híbridos, a VPC deverá ter rotas em sua tabela de rotas para a rede on-premises. Para obter mais informações sobre os requisitos de rede para nós híbridos, consulte the section called “Preparar a rede”.

Pré-requisitos

Para criar uma VPC para o Amazon EKS, é necessário ter as permissões do IAM necessárias para criar recursos da Amazon VPC. Esses recursos são VPCs, sub-redes, grupos de segurança, tabelas e rotas de rotas e gateways da Internet e de NAT. Para obter mais informações, consulte o exemplo de política Criar uma VPC com uma sub-rede pública no Guia do usuário do Amazon VPC e a lista completa de Ações na Referência de autorização de serviço.

É possível criar uma VPC com sub-redes públicas e privadas, somente sub-redes públicas ou somente sub-redes privadas.

Sub-redes públicas e privadas

Essa VPC tem duas sub-redes públicas e duas privadas. A tabela de rotas associada a uma sub-rede pública tem uma rota para um gateway da Internet. Porém, a tabela de rotas de uma sub-rede privada não tem uma rota para um gateway da Internet. Uma sub-rede pública e uma privada são implantadas na mesma zona de disponibilidade. As outras sub-redes públicas e privadas são implantadas em uma segunda zona de disponibilidade na mesma região da AWS. Recomendamos essa opção para a maioria das implantações.

Com essa opção, é possível implantar nós em sub-redes privadas. Esta opção permite que o Kubernetes implante balanceadores de carga nas sub-redes públicas que podem balancear a carga do tráfego para pods que são executados nos nós nas sub-redes privadas. Endereços `IPv4` públicos são atribuídos automaticamente a nós implantados em sub-redes públicas, mas endereços `IPv4` públicos não são atribuídos a nós implantados a sub-redes privadas.

Você também pode atribuir endereços `IPv6` a nós em sub-redes públicas e privadas. Os nós em sub-redes privadas podem se comunicar com o cluster e outros serviços da AWS. Os pods podem se comunicar com a internet por meio de um gateway NAT usando endereços `IPv4` ou um gateway da internet somente de saída usando endereços `IPv6` implantados em cada zona de disponibilidade. É implantado um grupo de segurança com regras que negam todo o tráfego de entrada de origens

diferentes do cluster ou dos nós, mas permite todo o tráfego de saída. As sub-redes são marcadas para que o Kubernetes possa implantar load balancers nelas.

a. Abra o console do AWS CloudFormation.

b. Na barra de navegação, selecione uma região da AWS que ofereça suporte ao Amazon EKS.

c. Escolha Create stack (Criar pilha), With new resources (Com novos recursos (padrão)).

d. Em Prerequisite - Prepare template, (Pré-requisito: preparar o modelo), certifique-se de que a opção Template is ready (O modelo está pronto) esteja selecionada. Em seguida, em Specify template (Especificar modelo), selecione Amazon S3 URL. (URL do Simple Storage Service [Amazon S3]).

e. É possível criar uma VPC que ofereça suporte somente a `IPv4` ou uma VPC que ofereça suporte a `IPv4` e a `IPv6`. Cole um dos seguintes URLs na área de texto em Amazon S3 URL (URL do Simple Storage Service [Amazon S3]) e, então, escolha Next (Próximo):

   - `IPv4`

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

- `IPv4` e `IPv6`

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
```

a. Na página Specify stack details (Especificar detalhes da pilha), insira os parâmetros e escolha Next (Próximo).

   - Nome da pilha: escolha um nome para a pilha do AWS CloudFormation. Por exemplo, você pode usar o nome do modelo empregado na etapa anterior. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
   - VpcBlock: escolha um intervalo de CIDR `IPv4` para a sua VPC. Cada nó, pod e balanceador de carga implantado recebe um endereço `IPv4` desse bloco. Os valores padrão de `IPv4` fornecem endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-los. Para obter mais informações, consulte VPC and subnet

sizing (Dimensionamento da VPC e da sub-rede) no Guia do usuário da Amazon VPC. Você também pode adicionar blocos CIDR adicionais à VPC depois que ela for criada. Se estiver criando uma VPC IPv6, os intervalos CIDR IPv6 serão automaticamente atribuídos a você no espaço do endereço unicast global da Amazon.

- PublicSubnet01Block: especifique um bloco CIDR IPv4 para a sub-rede pública 1. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo. Se você estiver criando uma VPC IPv6, esse bloco será especificado para você no modelo.
- PublicSubnet02Block: especifique um bloco CIDR IPv4 para a sub-rede pública 2. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo. Se você estiver criando uma VPC IPv6, esse bloco será especificado para você no modelo.
- PrivateSubnet01Block: especifique um bloco CIDR IPv4 para a sub-rede privada 1. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo. Se você estiver criando uma VPC IPv6, esse bloco será especificado para você no modelo.
- PrivateSubnet02Block: especifique um bloco CIDR IPv4 para a sub-rede privada 2. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo. Se você estiver criando uma VPC IPv6, esse bloco será especificado para você no modelo.

b. (Opcional) Na página Configure stack options (Configurar opções de pilha), etiquete os seus recursos de pilha e, em seguida, escolha Next (Próximo).

c. Na página Review (Revisão), escolha Create stack (Criar pilha).

d. Quando a pilha estiver criada, selecione-a no console e escolha Outputs (Saídas).

e. Registre o VpcId para a VPC que foi criada. Você precisará disso ao criar seu cluster e nós.

f. Registre os SubnetIds para as sub-redes que foram criadas e se você as criou como sub-redes públicas ou privadas. É necessário pelo menos dois deles ao criar seu cluster e os nós.

g. Se você criou uma VPC IPv4, ignore essa etapa. Se você criou uma VPC IPv6, será necessário habilitar a opção de atribuição automática de endereço IPv6 para as sub-redes públicas criadas pelo modelo. Essa configuração já está habilitada para as sub-redes privadas. Conclua as etapas a seguir para habilitar as configurações:

   i. Abra o console da Amazon VPC em https://console.aws.amazon.com/vpc/.

   ii. No painel de navegação à esquerda, escolha Subnets (Sub-redes).

iii. Selecione uma de suas sub-redes públicas (o ***nome da pilha/SubnetPublic01*** ou ***o nome da pilha/SubnetPublic02*** contém a palavra public) e escolha Actions (Ações), Edit subnet settings (Editar configurações de sub-rede).

iv. Marque a caixa de seleção Enable auto-assign IPv6 address (Habilitar o endereço IPv6 de atribuição automática) e, então, escolha Save (Salvar).

v. Conclua as etapas anteriores novamente para a sua outra sub-rede pública.

Somente sub-redes públicas

Essa VPC tem três sub-redes públicas que são implantadas em diferentes zonas de disponibilidade em uma região AWS. Todos os nós recebem endereços `IPv4` públicos automaticamente e podem enviar e receber tráfego de Internet por meio de um gateway da Internet. É implantado um grupo de segurança que nega todo o tráfego de entrada e permite todo o tráfego de saída. As sub-redes são marcadas para que o Kubernetes possa implantar load balancers nelas.

a. Abra o console do AWS CloudFormation.

b. Na barra de navegação, selecione uma região da AWS que ofereça suporte ao Amazon EKS.

c. Escolha Create stack (Criar pilha), With new resources (Com novos recursos (padrão)).

d. Em Prepare template (Preparar o template), verifique se a opção Template is ready (O template está pronto) está selecionada e, em Template source (Origem do template), selecione Amazon S3 URL (URL do Amazon S3).

e. Cole o seguinte URL na área de texto em Amazon S3 URL (URL do Amazon S3) e escolha Next (Próximo):

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

a. Na página Specify Details (Especificar detalhes), insira os parâmetros e escolha Next (Próximo).

- Nome da pilha: escolha um nome para a pilha do AWS CloudFormation. Por exemplo, você pode chamá-lo de ***amazon-eks-vpc-sample***. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.

- VpcBlock: escolha um bloco CIDR para a VPC. Cada nó, pod e balanceador de carga implantado recebe um endereço `IPv4` desse bloco. Os valores padrão de `IPv4` fornecem endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-los. Para obter mais informações, consulte VPC and subnet sizing (Dimensionamento da VPC e da sub-rede) no Guia do usuário da Amazon VPC. Você também pode adicionar blocos CIDR adicionais à VPC depois que ela for criada.
- Subnet01Block: especifique um bloco CIDR para a sub-rede 1. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.
- Subnet02Block: especifique um bloco CIDR para a sub-rede 2. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.
- Subnet03Block: especifique um bloco CIDR para a sub-rede 3. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.

b. (Opcional) Na página Options (Opções), marque os recursos da pilha. Escolha Próximo.

c. Na página Revisar, escolha Criar.

d. Quando a pilha estiver criada, selecione-a no console e escolha Outputs (Saídas).

e. Registre o VpcId para a VPC que foi criada. Você precisará disso ao criar seu cluster e nós.

f. Registre os SubnetIds para as sub-redes que foram criadas. É necessário pelo menos dois deles ao criar seu cluster e os nós.

g. (Opcional) Todos os clusters que você implantar nessa VPC podem atribuir endereços `IPv4` privados aos seus pods e serviços. Se quiser implantar clusters nessa VPC para atribuir endereços `IPv6` privados aos seus pods e serviços, faça atualizações na VPC, na sub-rede, em tabelas de rotas e em grupos de segurança. Para obter mais informações, consulte Migrar VPCs existentes de IPv4 para IPv6 no Guia do usuário da Amazon VPC. O Amazon EKS exige que as suas sub-redes stejam com a opção de `Auto-assign` de endereços `IPv6` habilitada. Por padrão, ele está desabilitado.

Somente sub-redes privadas

Essa VPC tem três sub-redes privadas que são implantadas em diferentes zonas de disponibilidade na região AWS. Os recursos implantados nas sub-redes não podem acessar a Internet, nem a internet pode acessar os recursos nas sub-redes. O modelo cria endpoints de VPC usando AWS

PrivateLink para vários serviços AWS que os nós normalmente precisam acessar. Se seus nós precisarem de acesso à Internet de saída, você poderá adicionar um gateway de NAT público na Zona de disponibilidade de cada sub-rede após a criação da VPC. É criado um grupo de segurança que nega todo o tráfego de entrada, exceto de recursos implantados nas sub-redes. Um grupo de segurança também permite todo o tráfego de saída. As sub-redes são marcadas para que o Kubernetes possa implantar load balancers internos nelas. Se estiver criando uma VPC com essa configuração, consulte the section called “Clusters privados” para conhecer requisitos e considerações adicionais.

a. Abra o console do AWS CloudFormation.

b. Na barra de navegação, selecione uma região da AWS que ofereça suporte ao Amazon EKS.

c. Escolha Create stack (Criar pilha), With new resources (Com novos recursos (padrão)).

d. Em Prepare template (Preparar o template), verifique se a opção Template is ready (O template está pronto) está selecionada e, em Template source (Origem do template), selecione Amazon S3 URL (URL do Amazon S3).

e. Cole o seguinte URL na área de texto em Amazon S3 URL (URL do Amazon S3) e escolha Next (Próximo):

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-fully-private-vpc.yaml
```

a. Na página Specify Details (Especificar detalhes), insira os parâmetros e escolha Next (Próximo).

- Nome da pilha: escolha um nome para a pilha do AWS CloudFormation. Por exemplo, você pode chamá-lo de *`amazon-eks-fully-private-vpc`*. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
- VpcBlock: escolha um bloco CIDR para a VPC. Cada nó, pod e balanceador de carga implantado recebe um endereço `IPv4` desse bloco. Os valores padrão de `IPv4` fornecem endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-los. Para obter mais informações, consulte VPC and subnet sizing (Dimensionamento da VPC e da sub-rede) no Guia do usuário da Amazon VPC. Você também pode adicionar blocos CIDR adicionais à VPC depois que ela for criada.

- PrivateSubnet01Block: especifique um bloco CIDR para a sub-rede 1. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.
- PrivateSubnet02Block: especifique um bloco CIDR para a sub-rede 2. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.
- PrivateSubnet03Block: especifique um bloco CIDR para a sub-rede 3. O valor padrão fornece endereços IP suficientes para a maioria das implementações, mas se isso não acontecer, você poderá alterá-lo.

b. (Opcional) Na página Options (Opções), marque os recursos da pilha. Escolha Próximo.

c. Na página Revisar, escolha Criar.

d. Quando a pilha estiver criada, selecione-a no console e escolha Outputs (Saídas).

e. Registre o VpcId para a VPC que foi criada. Você precisará disso ao criar seu cluster e nós.

f. Registre os SubnetIds para as sub-redes que foram criadas. É necessário pelo menos dois deles ao criar seu cluster e os nós.

g. (Opcional) Todos os clusters que você implantar nessa VPC podem atribuir endereços `IPv4` privados aos seus pods e serviços. Se quiser implantar clusters nessa VPC para atribuir endereços `IPv6` privados aos seus pods e serviços, faça atualizações na VPC, na sub-rede, em tabelas de rotas e em grupos de segurança. Para obter mais informações, consulte Migrar VPCs existentes de IPv4 para IPv6 no Guia do usuário da Amazon VPC. O Amazon EKS exige que as suas sub-redes estejam com a opção de endereços `Auto-assign IPv6` habilitada (é desabilitada por padrão).

Exibir os requisitos para grupos de segurança do Amazon EKS em clusters

Este tópico descreve os requisitos de grupos de segurança de um cluster do Amazon EKS.

Grupo de segurança do cluster padrão

Ao criar um cluster, o Amazon EKS cria um grupo de segurança com o nome `eks-cluster-sg-`*`my-cluster`*`-`*`uniqueID`* . Esse grupo de segurança tem as seguintes regras padrão:

Tipo de regra	Protocolo	Portas	Origem	Destino
Entrada	Todos	Todos	Self	
Saída	Todos	Todos		0.0.0.0/0(IPv4) ou ::/0 (IPv6)
Saída	Todos	Todos		Próprio (para tráfego do EFA)

O grupo de segurança padrão inclui uma regra de saída que permite o tráfego do Elastic Fabric Adapter (EFA) com o destino sendo o mesmo grupo de segurança. Isso habilita o tráfego do EFA dentro do cluster, o que é benéfico para workloads de IA/ML e computação de alta performance (HPC). Para obter mais informações, consulte Elastic Fabric Adapter para workloads de IA/ML e HPC no Amazon EC2 no Guia do usuário do Amazon Elastic Compute Cloud.

> **Important**
>
> Se o cluster não precisar da regra de saída, você poderá removê-la. Se você a remover, ainda deverá ter as regras mínimas listadas em Restringir o tráfego do cluster. Se você remover a regra de entrada, o Amazon EKS a recriará sempre que o cluster for atualizado.

O Amazon EKS adiciona as tags a seguir ao grupo de segurança. Se você remover as tags, o Amazon EKS as adicionará novamente ao grupo de segurança sempre que o cluster for atualizado.

Chave	Valor
`kubernetes.io/cluster/ my-cluster`	`owned`
`aws:eks:cluster-name`	*my-cluster*
`Name`	`eks-cluster-sg-` *my-cluster* *-uniqueid*

O Amazon EKS associa automaticamente esse grupo de segurança aos recursos a seguir, que ele também cria:

- Duas a quatro interfaces de rede elásticas (chamadas no restante deste documento de interfaces de rede) que são criadas no momento em que você cria seu cluster.
- Interfaces de rede dos nós em qualquer grupo de nós gerenciados que você cria.

As regras padrão permitem que todo o tráfego flua livremente entre o cluster e os nós e aceitam todo o tráfego de saída para qualquer destino. Ao criar um cluster, você tem a opção de especificar seus próprios grupos de segurança. Se fizer isso, o Amazon EKS também associará os grupos de segurança especificados às interfaces de rede que ele criar para o cluster. Porém, ele não os associa a nenhum grupo de nós que você criar.

É possível determinar o ID do grupo de segurança do cluster no Console de gerenciamento da AWS, na seção Networking (Redes) do cluster. Ou você pode fazer isso executando o seguinte comando da CLI AWS.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Restringir o tráfego do cluster

Se for necessário limitar as portas abertas entre o cluster e os nós, você poderá remover as regras de saída padrão e adicionar as seguintes regras mínimas que são necessárias para o cluster. Se você remover a regra de entrada padrão, o Amazon EKS a recriará sempre que o cluster for atualizado.

Tipo de regra	Protocolo	Port (Porta)	Destino
Saída	TCP	443	Security group de cluster
Saída	TCP	10250	Security group de cluster
Saída (DNS)	TCP e UDP	53	Security group de cluster

Também é necessário adicionar regras para o seguinte tráfego:

- Qualquer protocolo e as portas que você espera que os nós usem para comunicação entre eles.
- Acesso de saída à Internet, para que os nós possam acessar as APIs do Amazon EKS para introspecção de clusters e registro de nós no momento da execução. Se os nós não tiverem acesso à Internet, consulte Implantar clusters privados com acesso limitado à Internet para conhecer considerações adicionais.
- Acesso ao nó para obter imagens de contêiner do Amazon ECR ou de outras APIs de registros de contêiner das quais eles precisam extrair imagens, como o DockerHub. Para obter mais informações, consulte Intervalos de endereços IP da AWS na Referência geral da AWS.
- Acesso do nó ao Amazon S3
- Regras distintas são necessárias para endereços `IPv4` e `IPv6`.
- Caso esteja usando nós híbridos, você deverá adicionar um grupo de segurança complementar ao cluster para permitir a comunicação com os nós e pods on-premises. Para obter mais informações, consulte the section called “Preparar a rede”.

Se estiver pensando em limitar as regras, recomendamos testar exaustivamente todos os seus pods antes de aplicar as regras modificadas a um cluster de produção.

Se você tiver implantado originalmente um cluster com o Kubernetes `1.14` e uma versão da plataforma `eks.3` ou anterior, considere o seguinte:

- Também é possível ter grupos de segurança e nós do ambiente de gerenciamento. Quando esses grupos foram criados, eles incluíam as regras restritas listadas na tabela anterior. Esses grupos de segurança não são mais necessários e podem ser removidos. Porém, você precisa garantir que seu grupo de segurança de cluster contenha as regras que esses grupos contêm.
- Se você implantou o cluster usando a API diretamente ou usou uma ferramenta como AWS CLI ou AWS CloudFormation para criar o cluster e não especificou um grupo de segurança na criação do cluster, o grupo de segurança padrão para o VPC foi aplicado às interfaces de rede do cluster que o Amazon EKS criou.

Grupos de segurança compartilhados

O Amazon EKS oferece suporte a grupos de segurança compartilhados.

- As associações de VPC de grupos de segurança associam grupos de segurança a várias VPCs na mesma conta e região.
  - Saiba como associar grupos de segurança a várias VPCs no Guia do usuário da Amazon VPC.

- Os grupos de segurança compartilhados permitem que você compartilhe grupos de segurança com outras contas da AWS. As contas devem fazer parte da mesma organização da AWS.
  - Saiba como Compartilhar grupos de segurança com organizações no Guia do usuário da Amazon VPC.
- Os grupos de segurança estão sempre limitados a uma única região da AWS.

Considerações sobre o Amazon EKS

- O EKS tem os mesmos requisitos de grupos de segurança compartilhados ou de várias VPCs que os grupos de segurança padrão.

Gerenciar os complementos de redes para os clusters do Amazon EKS

O Amazon EEKS está disponível para seu cluster do Amazon EKS.

Complementos integrados

> **Note**
>
> Quando você cria um cluster do EKS:
>
> - Usando o Console da AWS: os complementos integrados (como CoreDNS, kube-proxy etc.) são instalados automaticamente como complementos do Amazon EKS. Eles podem ser facilmente configurados e atualizados nos SDKs, na CLI ou no Console da AWS.
> - Usando outros métodos (CLI, SDKs etc.): os mesmos complementos integrados são instalados como versões autogerenciadas que são executadas como implantações regulares do Kubernetes. Eles exigem configuração e atualizações manuais, pois não podem ser gerenciados por meio de ferramentas da AWS.
>
> Recomendamos usar complementos do Amazon EKS em vez de versões autogerenciadas para simplificar o gerenciamento de complementos e permitir configurações e atualizações centralizadas por meio dos serviços da AWS.

Plug-in CNI da Amazon VPC para Kubernetes

Esse complemento da CNI cria interfaces de rede elásticas e as anexa aos nós do Amazon EC2. O complemento também atribui um endereço `IPv4` ou `IPv6` privado da VPC a cada pod e serviço. Esse complemento é instalado no cluster por padrão. Para obter mais informações, consulte the section called “CNI da Amazon VPC”. Caso esteja usando nós híbridos, a CNI da VPC ainda será instalada por padrão, mas será impedida de ser executada nos nós híbridos com uma regra de antiafinidade. Para obter mais informações sobre as opções de CNI para nós híbridos, consulte the section called “Configurar CNI”.

CoreDNS

O CoreDNS é um servidor DNS flexível e extensível que pode servir como DNS de cluster do Kubernetes. O CoreDNS fornece uma resolução de nomes para todos os pods no cluster. Esse complemento é instalado no cluster por padrão. Para obter mais informações, consulte the section called “CoreDNS”.

`kube-proxy`

Esse complemento mantém as regras de rede nos nós do Amazon EC2 e viabiliza a comunicação de rede com os pods. Esse complemento é instalado no cluster por padrão. Para obter mais informações, consulte the section called “kube-proxy”.

Complementos opcionais de rede da AWS

AWS Load Balancer Controller

Ao implantar objetos de serviço do Kubernetes do tipo `loadbalancer`, o controlador cria Network Load Balancers da AWS. Quando você cria objetos de entrada do Kubernetes, o controlador cria Application Load Balancers da AWS. Recomendamos usar esse controlador para provisionar Network Load Balancers em vez de usar o controlador legado do Cloud Provider incorporado no Kubernetes. Para obter mais informações, consulte a documentação do AWS Load Balancer Controller.

AWS Controlador de API de gateway

Esse controlador permite que você conecte serviços em vários clusters do Kubernetes usando a API de gateway do Kubernetes. O controlador conecta serviços do Kubernetes executados em instâncias, contêineres e funções sem servidor do Amazon EC2 usando o serviço do Amazon VPC Lattice. Para obter mais informações, consulte a documentação do AWS Gateway API Controller

Para obter mais informações sobre complementos, consulte the section called “Complementos do Amazon EKS”.

Atribuir IPs a pods com a CNI da Amazon VPC

> **Tip**
>
> Com o Modo automático do Amazon EKS, você não precisa instalar ou atualizar complementos de rede. O Modo automático inclui recursos de rede de pods e balanceamento de carga.
> Para obter mais informações, consulte Modo Automático do EKS.

O complemento plug-in CNI da Amazon VPC para Kubernetes é implantado em cada nó do Amazon EC2 no cluster do Amazon EKS. O complemento cria interfaces de rede elásticas e as anexa aos nós do Amazon EC2. O complemento também atribui um endereço `IPv4` ou `IPv6` privado da sua VPC a cada pod.

Uma versão do complemento é implantada com cada nó do Fargate em seu cluster, mas você não a atualiza em nós do Fargate. Há outros plug-ins de CNI compatíveis disponíveis para uso em clusters do Amazon EKS, mas esse é o único plug-in de CNI compatível com o Amazon EKS para nós que são executados na infraestrutura da AWS. Para obter mais informações sobre os outros plug-ins compatíveis da CNI, consulte the section called “Plug-ins CNI alternados”. A CNI da VPC não é compatível para uso com nós híbridos. Para obter mais informações sobre as opções de CNI para nós híbridos, consulte the section called “Configurar CNI”.

A tabela a seguir lista a versão mais recente disponível do tipo de complemento do Amazon EKS disponível para cada versão do Kubernetes.

Versões da CNI da Amazon VPC

Versão do Kubernetes	Tipo Amazon EKS de versão da CNI da VPC
1.34	v1.20.4-eksbuild.1
1.33	v1.20.4-eksbuild.1
1.32	v1.20.4-eksbuild.1

Versão do Kubernetes	Tipo Amazon EKS de versão da CNI da VPC
1.31	v1.20.4-eksbuild.1
1.30	v1.20.4-eksbuild.1
1.29	v1.20.4-eksbuild.1
1.28	v1.20.4-eksbuild.1

Important

Se você estiver gerenciando esse complemento automaticamente, as versões na tabela poderão não ser as mesmas que as versões autogerenciadas disponíveis. Para obter mais informações sobre como atualizar o tipo autogerenciado desse complemento, consulte the section called “Atualização (autogerenciada)”.

Important

Para atualizar para a CNI da VPC v1.12.0 ou posterior, primeiro é necessário fazer o upgrade para a CNI da VPC v1.7.0. Convém atualizar uma versão secundária de cada vez.

Considerações

A seguir estão algumas considerações sobre o uso do recurso.

- As versões são especificadas como `major-version.minor-version.patch-version-eksbuild.build-number`.
- Verifique a compatibilidade de versão em cada atributo. Alguns recursos de cada versão do plug-in CNI da Amazon VPC para Kubernetes exigem determinadas versões do Kubernetes. Ao utilizar diferentes recursos do Amazon EKS, se uma versão específica do complemento for necessária, então ela será anotada na documentação de recursos. A menos que você tenha um motivo específico para executar uma versão anterior, recomendamos que execute a versão mais recente.

Criação do plug-in da CNI da Amazon VPC (complemento do Amazon EKS)

Use as etapas a seguir para criar o complemento do plug-in CNI da Amazon VPC para Kubernetes para Amazon EKS.

Antes de começar, reveja as considerações. Para obter mais informações, consulte the section called "Considerações".

Pré-requisitos

Veja abaixo os pré-requisitos para o complemento do plug-in CNI da Amazon VPC para Kubernetes para Amazon EKS.

- Um cluster existente do Amazon EKS. Para implantar, consulte Conceitos básicos.
- Um provedor AWS Identity and Access Management (IAM) OpenID Connect (OIDC) para o seu cluster. Para determinar se você tem ou para criar uma, consulte the section called "Provedor OIDC do IAM".
- Um perfil do IAM com a política do IAM AmazonEKS_CNI_Policy (se o cluster usar a família `IPv4`) ou uma política IPv6 (se o seu cluster usar a família `IPv6`) anexada a ela. Para obter mais informações sobre o perfil VPC CNI, consulte the section called "Configuração para IRSA". Para obter informações sobre a política de IPv6, consulte the section called "Criar uma política do IAM para clusters que usam a família `IPv6`".

> **Important**
>
> As versões `v1.16.0` a `v1.16.1` do plug-in CNI da Amazon VPC para Kubernetes implementam a versão `v1.0.0` da especificação da CNI. Para obter mais informações sobre a versão `v1.0.0` da especificação da CNI, consulte Container Network Interface (CNI) Specification no GitHub.

Procedimento

Depois de concluir os pré-requisitos, use as etapas a seguir para criar o complemento.

1. Veja qual versão do complemento está atualmente instalada no cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Veja um exemplo de saída abaixo.

```
v1.16.4-eksbuild.2
```

2. Veja qual tipo de complemento está atualmente instalado no cluster. Dependendo da ferramenta com a qual você criou o cluster, talvez você não tenha o tipo de complemento do Amazon EKS instalado em seu cluster atualmente. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

Se um número de versão for retornado, você tem o tipo de complemento do Amazon EKS instalado no cluster, e não precisa completar as etapas restantes deste procedimento. Se um erro for retornado, você não tem o tipo de complemento do Amazon EKS instalado no cluster. Conclua as etapas restantes desse procedimento para instalá-lo.

3. Salve a configuração do complemento instalado atualmente.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. Crie o complemento usando a AWS CLI. Se você quiser usar o Console de gerenciamento da AWS ou `eksctl` para criar o complemento, consulte the section called “Criação de um complemento” e especifique `vpc-cni` como o nome do complemento. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.

   - Substitua *my-cluster* pelo nome do cluster.
   - Substitua *v1.20.3-eksbuild.1* pela versão mais recente listada na tabela das versões mais recentes da versão do seu cluster. Para obter a tabela de versões mais recente, consulte the section called “Versões da CNI da Amazon VPC”.
   - Substitua *111122223333* pelo ID da sua conta e *AmazonEKSVPCCNIRole* pelo nome do perfil do IAM existente que você criou. A especificação de um perfil requer que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte the section called “Provedor OIDC do IAM”.

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.20.3-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
```

Se você aplicou configurações personalizadas ao seu complemento atual que entrem em conflito com as configurações padrão do complemento Amazon EKS, a criação poderá falhar. Se a criação falhar, você receberá um erro que poderá ajudar a resolver o problema. Como alternativa, você pode adicionar `--resolve-conflicts OVERWRITE` ao comando anterior. Isso permite que o complemento substitua todas as configurações personalizadas existentes. Depois de criar o complemento, você pode atualizá-lo com suas configurações personalizadas.

5. Verifique se a versão mais recente do complemento para a versão do Kubernetes do cluster foi adicionada ao cluster. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

Pode levar alguns segundos para que a criação do complemento seja concluída.

Veja abaixo um exemplo de saída.

```
v1.20.3-eksbuild.1
```

6. Se você tiver feito configurações personalizadas no complemento original antes de criar o complemento do Amazon EKS, use a configuração salva em uma etapa anterior para atualizar o complemento do EKS com suas configurações personalizadas. Siga as etapas em the section called “Atualizar (complemento do EKS)”.
7. (Opcional) Instale `cni-metrics-helper` em seu cluster. Ele extrai a interface de rede elástica e as informações de endereço IP, agrega-as em um nível de cluster e publica as métricas no Amazon CloudWatch. Para obter mais informações, consulte cni-metrics-helper no GitHub.

Atualização do plug-in da CNI da Amazon VPC (complemento do Amazon EKS)

Atualize o tipo do Amazon EKS do complemento do plug-in CNI da Amazon VPC para Kubernetes. Se você não tiver adicionado o tipo de complemento do Amazon EKS ao seu cluster, poderá instalá-lo seguindo the section called “Criar”. Ou atualize o outro tipo de instalação do VPC CNI seguindo the section called “Atualização (autogerenciada)”.

1. Veja qual versão do complemento está atualmente instalada no cluster. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

Veja abaixo um exemplo de saída.

```
v1.20.0-eksbuild.1
```

Compare a versão com a tabela de versões mais recentes em the section called “Versões da CNI da Amazon VPC”. Se a versão retornada for a mesma da versão do cluster do Kubernetes na tabela de versões mais recentes, então você já tem a versão mais recente instalada no cluster e não precisa concluir o restante deste procedimento. Se você receber um erro, em vez de um número de versão no resultado, você não tem o tipo Amazon EKS do complemento instalado no cluster. Você precisa criar o complemento antes de poder atualizá-lo com este procedimento. Para criar o tipo Amazon EKS do complemento VPC CNI, você pode seguir the section called “Criar”.

2. Salve a configuração do complemento instalado atualmente.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

3. Atualize seu complemento usando a AWS CLI. Se você quiser usar o Console de gerenciamento da AWS ou `eksctl` para atualizar o complemento, consulte the section called “Atualização de um complemento”. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.
   - Substitua *`my-cluster`* pelo nome do cluster.
   - Substitua *`v1.20.0-eksbuild.1`* pela versão mais recente listada na tabela das versões mais recentes da versão do seu cluster.
   - Substitua *`111122223333`* pelo ID da sua conta e *`AmazonEKSVPCCNIRole`* pelo nome do perfil do IAM existente que você criou. Para criar um perfil do IAM para a VPC CNI, consulte the section called “Etapa 1: criar o perfil do IAM do plug-in CNI da Amazon VPC para Kubernetes”. A especificação de um perfil requer que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte the section called “Provedor OIDC do IAM”.
   - A opção `--resolve-conflicts PRESERVE` preserva os valores de configuração existentes para o complemento. Se você definiu valores personalizados para as configurações do complemento e não usar essa opção, o Amazon EKS sobrescreverá seus valores pelos valores padrão. Se você usar essa opção, recomendamos testar qualquer alteração de campo e valor

em um cluster que não seja de produção antes de atualizar o complemento no cluster do produção. Se você alterar esse valor para `OVERWRITE`, todas as configurações serão alteradas para os valores padrão do Amazon EKS. Se você definiu valores personalizados para qualquer configuração, eles poderão ser sobrescritos pelos valores padrão do Amazon EKS. Se você alterar esse valor para `none`, o Amazon EKS não alterará o valor de nenhuma configuração, mas a atualização poderá falhar. Se a atualização falhar, você receberá uma mensagem de erro para ajudar a resolver o conflito.

- Se você não estiver atualizando uma configuração, remova `--configuration-values '{`*`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`*`}'` do comando. Se você estiver atualizando uma configuração, substitua *`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`* pela configuração que você quiser definir. Neste exemplo, a variável de ambiente `AWS_VPC_K8S_CNI_EXTERNALSNAT` é definida como `true`. O valor que você especificar deve ser válido para o esquema da configuração. Caso não saiba o esquema de configuração, execute `aws eks describe-addon-configuration --addon-name vpc-cni --addon-version` *`v1.20.0-eksbuild.1`* , substituindo *`v1.20.0-eksbuild.1`* pelo número da versão do complemento cuja configuração você deseja visualizar. O esquema é retornado na saída. Se você tiver alguma configuração personalizada existente, quiser remover tudo e definir os valores de todas as configurações de volta aos padrões do Amazon EKS, remova *`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`* do comando, para que `{}` fique vazio. Para obter uma explicação de cada configuração, consulte Variáveis de configuração CNI no GitHub.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.20.3-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"env":
{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'
```

Pode levar alguns segundos para que a atualização seja concluída.

4. Confirme se a versão do complemento foi atualizada. Substitua *`my-cluster`* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

Pode levar alguns segundos para que a atualização seja concluída.

Veja abaixo um exemplo de saída.


```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.20.3-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/vpc-
cni/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
        "createdAt": "2023-04-12T18:25:19.319000+00:00",
        "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
        "serviceAccountRoleArn": "arn:aws:iam::111122223333:role/
AmazonEKSVPCCNIRole",
        "tags": {},
        "configurationValues": "{\"env\":{\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":\"true
\"}}"
    }
}
```


Atualização do plug-in da CNI da Amazon VPC (complemento autogerenciado)

> **Important**
>
> Recomendamos adicionar o tipo Amazon EKS do complemento ao seu cluster em vez de usar o tipo autogerenciado do complemento. Se você não estiver familiarizado com a diferença entre os tipos, consulte the section called “Complementos do Amazon EKS”. Para obter mais informações sobre como adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criação de um complemento”. Se você não conseguir usar o complemento do Amazon EKS, recomendamos que você envie um problema sobre o motivo pelo qual não pode usar o repositório GitHub para roteiro de contêineres.

1. Confirme que você não tem o tipo de complemento Amazon EKS instalado em seu cluster. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

Se uma mensagem erro for retornada, você não tem o tipo de complemento do Amazon EKS instalado no cluster. Para gerenciar automaticamente o complemento, conclua as etapas restantes neste procedimento para atualizar o complemento. Se receber um número de versão, você tem o tipo de complemento do Amazon EKS instalado no cluster. Para atualizá-lo, use o procedimento em the section called “Atualização de um complemento”, em vez de usar este procedimento. Se não estiver familiarizado com a diferença entre os tipos de complemento, consulte the section called “Complementos do Amazon EKS”.

2. Veja qual versão da imagem do contêiner está atualmente instalada em seu cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Veja um exemplo de saída abaixo.

```
v1.20.0-eksbuild.1
```

O resultado pode não incluir o número da compilação.

3. Faça o backup de suas configurações atuais para que você possa definir as mesmas configurações após atualizar sua versão.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

Para revisar as versões disponíveis e se familiarizar com as alterações na versão para a qual você deseja atualizar, consulte as versões no GitHub. Observe que recomendamos atualizar para a mesma versão `major.minor.patch` listada na tabela de versões mais recentes disponíveis, mesmo se versões posteriores estiverem disponíveis no GitHub. Para obter a tabela da versão mais recente disponível, consulte the section called “Versões da CNI da Amazon VPC”. As versões de compilação listadas na tabela não são especificadas nas versões autogerenciadas listadas no GitHub. Atualize a versão concluindo as tarefas em uma das seguintes opções:

- Se você não tiver nenhuma configuração personalizada para o complemento, execute o comando abaixo do cabeçalho `To apply this release:` no GitHub específico da versão para a qual você está atualizando.

- Se você tiver configurações personalizadas, baixe o manifesto com o comando a seguir. Mudança *https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.0/config/master/aws-k8s-cni.yaml* para a URL da versão no GitHub para a qual você está atualizando.

```
curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.3/config/
master/aws-k8s-cni.yaml
```

Se necessário, modifique o manifesto com as configurações personalizadas do backup que você fez em uma etapa anterior e aplique o manifesto modificado ao cluster. Se seus nós não tiverem acesso aos repositórios privados do Amazon ECR do Amazon EKS dos quais as imagens são extraídas (veja as linhas que começam com `image:` no manifesto), você precisará baixar as imagens, copiá-las para seu próprio repositório e modificar o manifesto a fim de extrair as imagens do seu repositório. Para obter mais informações, consulte the section called “Copiar uma imagem para um repositório”.

```
kubectl apply -f aws-k8s-cni.yaml
```

4. Confirme se agora a nova versão está instalada em seu cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Veja um exemplo de saída abaixo.

```
v1.20.3
```

5. (Opcional) Instale `cni-metrics-helper` em seu cluster. Ele extrai a interface de rede elástica e as informações de endereço IP, agrega-as em um nível de cluster e publica as métricas no Amazon CloudWatch. Para obter mais informações, consulte cni-metrics-helper no GitHub.

Configurar o plug-in CNI da Amazon VPC para usar IRSA

O plug-in CNI da Amazon VPC para Kubernetes é o plug-in de rede para redes de pods nos clusters do Amazon EKS. O plug-in é responsável por alocar endereços IP da VPC para pods do Kubernetes e configurar as redes necessárias para pods em cada nó.

> **Note**
>
> O plug-in da CNI da Amazon VPC também é compatível com as Identidades de Pods do Amazon EKS. Para obter mais informações, consulte the section called “Atribuir perfil do IAM”.

O plugin:

- Requer permissões do AWS Identity and Access Management (IAM). Se o seu cluster usar a família `IPv4`, as permissões serão especificadas na política `AmazonEKS_CNI_Policy` gerenciada pela AWS. Se o seu cluster usar a família `IPv6`, as permissões deverão ser adicionadas a um política do IAM que você mesmo cria. Para obter as instruções, consulte the section called “Criar uma política do IAM para clusters que usam a família `IPv6`”. É possível anexar a política a Amazon EKS node IAM role ou a uma função do IAM separada. Para obter instruções sobre como anexar a política ao perfil do IAM do nó do Amazon EKS, consulte the section called “Perfil do IAM do nó”. Recomendamos atribuir a uma função separada, conforme detalhado neste tópico.
- Cria e está configurado para usar uma conta de serviço do Kubernetes denominada `aws-node` quando for implantado. A conta de serviço está vinculada a um `clusterrole` do Kubernetes chamado `aws-node`, ao qual são atribuídas as permissões necessárias do Kubernetes.

> **Note**
>
> Os pods para o plug-in CNI da Amazon VPC para Kubernetes têm acesso às permissões atribuídas ao perfil do IAM do nó do Amazon EKS, a menos que você bloqueie o acesso ao IMDS. Para obter mais informações, consulte Restringir o acesso ao perfil de instância atribuído ao nó de processamento.

- Requer um cluster existente do Amazon EKS. Para implantar, consulte Conceitos básicos.
- Requer um provedor OpenID Connect (OIDC) do AWS Identity and Access Management (IAM) para o cluster. Para determinar se você tem ou para criar uma, consulte the section called “Provedor OIDC do IAM”.

Etapa 1: criar o perfil do IAM do plug-in CNI da Amazon VPC para Kubernetes

1. Determine a família de IP do seu cluster.

```
aws eks describe-cluster --name my-cluster | grep ipFamily
```

   Veja um exemplo de saída abaixo.

```
"ipFamily": "ipv4"
```

   Em vez disso, a saída pode retornar `ipv6`.

2. Crie o perfil do IAM. Você pode usar `eksctl` ou `kubectl` e a CLI AWS para criar seu perfil do IAM.

   eksctl

   - Crie um perfil do IAM e anexe a política do IAM a esse perfil por meio do comando que corresponde à família de IP do cluster. O comando cria e implementa uma pilha do AWS CloudFormation que cria um perfil do IAM, anexa a política que você especifica a ela e anota a conta de serviço `aws-node` existente do Kubernetes com o ARN do perfil do IAM criado.
     - `IPv4`

       Substitua *`my-cluster`* pelo seu próprio valor.

```
eksctl create iamserviceaccount \
    --name aws-node \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKSVPCCNIRole \
    --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
    --override-existing-serviceaccounts \
    --approve
```

     - `IPv6`

       Substitua *`my-cluster`* pelo seu próprio valor. Substitua *`111122223333`* pelo ID da sua conta e substitua *`AmazonEKS_CNI_IPv6_Policy`* pelo nome da sua política `IPv6`. Se não tiver uma política `IPv6`, consulte the section called “Criar uma política do IAM para clusters que usam a família `IPv6`” para criá-la. Para utilizar `IPv6` com seu cluster, este

deve atender a vários requisitos. Para obter mais informações, consulte the section called "IPv6".

```
eksctl create iamserviceaccount \
    --name aws-node \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKSVPCCNIRole \
    --attach-policy-arn arn:aws:iam::111122223333:policy/
AmazonEKS_CNI_IPv6_Policy \
    --override-existing-serviceaccounts \
    --approve
```

kubectl e a AWS CLI

i. Exiba a URL do provedor OIDC do cluster.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

Veja abaixo um exemplo de saída.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

Se nenhum resultado for retornado, você deverá criar um provedor OIDC do IAM do seu cluster.

ii. Copie o conteúdo a seguir em um arquivo chamado *vpc-cni-trust-policy.json*. Substitua *111122223333* pelo ID de sua conta e *EXAMPLED539D4633E53DE1B71EXAMPLE* pelo resultado retornado na etapa anterior. Substitua *region-code* pela região da AWS em que seu cluster se encontra.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
```

```
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
node"
                }
            }
        }
    ]
}
```

iii. Crie a função. Você pode substituir *AmazonEKSVPCCNIRole* por qualquer nome que desejar.

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

iv. Anexe a política do IAM necessária à função. Execute o comando que corresponde à família de IP do seu cluster.

- `IPv4`

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSVPCCNIRole
```

- `IPv6`

  Substitua *111122223333* pelo ID da sua conta e *AmazonEKS_CNI_IPv6_Policy* pelo nome da sua política `IPv6`. Se não tiver uma política `IPv6`, consulte the section called “Criar uma política do IAM para clusters que usam a família `IPv6`” para criá-la. Para utilizar `IPv6` com seu cluster, este deve atender a vários requisitos. Para obter mais informações, consulte the section called “IPv6”.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSVPCCNIRole
```

v. Execute o seguinte comando para anotar a conta de serviço `aws-node` com o ARN do perfil do IAM criado por você anteriormente: Substitua os valores de exemplo por seus próprios valores.

```
kubectl annotate serviceaccount \
    -n kube-system aws-node \
    eks.amazonaws.com/role-arn=arn:aws:iam::111122223333:role/
AmazonEKSVPCCNIRole
```

3. (Opcional) Configure o tipo de endpoint do AWS Security Token Service usado por sua conta de serviço do Kubernetes. Para obter mais informações, consulte the section called “Endpoints do STS”.

Etapa 2: reimplantar os pods do plug-in CNI da Amazon VPC para Kubernetes

1. Exclua e recrie os pods existentes associados à conta de serviço para aplicar as variáveis de credenciais do ambiente. A anotação não é aplicada a pods que estejam sendo executados sem a anotação. O comando a seguir exclui os pods do DaemonSet `aws-node` existentes e implanta-os com a anotação da conta de serviço.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

2. Verifique se todos os pods foram reiniciados.

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. Descreva um dos pods e verifique se as variáveis de ambiente `AWS_WEB_IDENTITY_TOKEN_FILE` e `AWS_ROLE_ARN` existem. Substitua *`cpjw7`* pelo nome de um dos pods retornados no resultado da etapa anterior.

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\|
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

Veja abaixo um exemplo de saída.

```
AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

```
      AWS_ROLE_ARN:                           arn:aws:iam::111122223333:role/
AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:            /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

Dois conjuntos de resultados duplicados são retornados porque o pod contém dois contêineres. Os dois contêineres têm os mesmos valores.

Se o seu pod estiver usando o endpoint regional da AWS, a linha a seguir também será retornada na saída anterior.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Etapa 3: Remover a política CNI do perfil do IAM do nó

Se o perfil do IAM para o nó do Amazon EKS atualmente conta com a política `AmazonEKS_CNI_Policy` do IAM (`IPv4`) ou uma política IPv6 anexada a ele, e você criou um perfil do IAM separado, anexou a política a esse novo perfil e a atribuiu à conta de serviço do Kubernetes, chamada `aws-node`, recomendamos que você remova a política do perfil do nó com o comando da AWS CLI correspondente à família de IPs do cluster. Substitua *`AmazonEKSNodeRole`* pelo nome da sua função de nó.

- `IPv4`

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn
 arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

- `IPv6`

  Substitua *`111122223333`* pelo ID da sua conta e *`AmazonEKS_CNI_IPv6_Policy`* pelo nome da sua política `IPv6`.

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn
 arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

Criar uma política do IAM para clusters que usam a família `IPv6`

Se você tiver criado um cluster que usa a família `IPv6`, e esse cluster tiver a versão `1.10.1` ou mais recente do complemento do plug-in CNI da Amazon VPC para Kubernetes configurada, será necessário criar uma política do IAM que você possa atribuir a um perfil do IAM. Se você tiver um cluster que não foi configurado com a família `IPv6` no momento da criação, será necessário criar um novo cluster para usar `IPv6`. Para obter mais informações sobre como usar o `IPv6` com o seu cluster, consulte the section called “IPv6”.

1. Copie o texto a seguir e salve-o em um arquivo chamado `vpc-cni-ipv6-policy.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. Crie a política do IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

Saiba mais sobre os modos e a configuração de CNI da VPC

O plug-in CNI da Amazon VPC para Kubernetes fornece redes para pods. Use a tabela a seguir para saber mais sobre os recursos de rede disponíveis.

Recursos de redes	Saiba mais
Configurar o cluster para atribuir endereços IPv6 a clusters, pods e serviços	the section called “IPv6”
Usar a conversão do endereço de rede de origem IPv4 para pods	the section called “Tráfego de saída”
Restringir o tráfego de rede de e para os pods	the section called “Restringir tráfego”
Personalizar a interface de rede secundária em nós	the section called “Redes personalizadas”
Aumentar os endereços IP do nó	the section called “Aumento dos endereços IP”
Usar grupos de segurança para o tráfego de rede de pods	the section called “Grupos de segurança para os pods”
Usar várias interfaces de rede para pods	the section called “Várias interfaces”

Saiba mais sobre endereços IPv6 para clusters, pods e serviços

Aplica-se a: pods com instâncias do Amazon EC2 e pods do Fargate

Por padrão, o Kubernetes atribui endereços `IPv4` aos seus pods e serviços. Em vez de atribuir endereços `IPv4` aos seus pods e serviços, você pode configurar o cluster para atribuir endereços `IPv6` a eles. O Amazon EKS não é compatível com serviços ou pods de pilha dupla, embora o Kubernetes seja. Consequentemente, você não pode atribuir endereços `IPv4` e `IPv6` aos seus pods e serviços.

Você seleciona qual família de IPs deseja usar para o seu cluster ao criá-lo. Não será possível alterar a família depois de criar o cluster.

Para obter um tutorial sobre como implantar um cluster `IPv6` do Amazon EKS, consulte the section called “Implantar”.

Confira as seguintes considerações sobre o uso do recurso:

Suporte a recursos **IPv6**

- Sem compatibilidade com o Windows: pods e serviços do Windows não são compatíveis.
- São necessários nós do EC2 baseados no Nitro: você só pode usar IPv6 com nós Amazon EC2 ou Fargate baseados no AWS Nitro.
- Suporte a nós do EC2 e Fargate: você pode usar IPv6 com the section called “Grupos de segurança para os pods” com nós do Amazon EC2 e nós Fargate.
- Não há suporte para Outposts : não é possível usar IPv6 com o Amazon EKS no AWS Outposts.
- Não há suporte para o FSx para Lustre: o the section called “Amazon FSx para Lustre” não é compatível.
- Não há suporte para redes personalizadas: Se você usou anteriormente the section called “Redes personalizadas” para ajudar a aliviar a exaustão de endereços IP, poderá usar IPv6 em vez disso. Não é possível usar redes personalizadas com IPv6. Se você usar redes personalizadas para isolar redes, talvez seja necessário continuar usando redes personalizadas e a família IPv4 para os seus clusters.

Atribuições de endereços IP

- Serviços Kubernetes: serviços do Kubernetes recebem apenas um endereço IPv6. Eles não recebem endereços IPv4.
- Pods: Os pods recebem um endereço IPv6 e um endereço IPv4 local do host. O endereço IPv4 local do host é atribuído usando um plug-in CNI local do host encadeado com o VPC CNI, e o endereço não é informado ao ambiente de gerenciamento do Kubernetes. Ele só é usado quando um pod precisa se comunicar com recursos IPv4 externos em outra Amazon VPC ou na internet. O endereço IPv4 local do host é SNATed (pelo VPC CNI) para o endereço IPv4 primário da ENI primária do nó de processamento.
- Pods e serviços: pods e serviços recebem somente endereços IPv6, não endereços IPv4. Quando os pods precisam se comunicar com endpoints IPv4 externos, eles usam NAT no próprio nó. Esse recurso NAT integrado elimina a necessidade de DNS64 e NAT64. Para tráfego que requer acesso público à internet, o tráfego do pod é um endereço de rede de origem convertido em um endereço IP público.
- Endereços de roteamento: quando um pod se comunica fora da VPC, seu endereço IPv6 original é preservado (não convertido para o endereço IPv6 do nó). Esse tráfego é roteado diretamente por meio de um gateway da internet ou de um gateway da internet somente de saída.

- Nós: Todos os nós recebem um endereço IPv4 e IPv6.
- Pods do Fargate: cada pod do Fargate recebe um endereço IPv6 do CIDR especificado para a sub-rede em que está implantado. A unidade de hardware subjacente que executa pods do Fargate obtém um endereço IPv4 e IPv6 exclusivo dos CIDRs atribuídos à sub-rede em que a unidade de hardware está implantada.

Como usar o **IPv6** com o EKS

- Criar novo cluster: Você deve criar um novo cluster e especificar que deseja usar a família IPv6 para esse cluster. Você não pode habilitar a família do IPv6 para um cluster atualizado de uma versão anterior. Para obter instruções sobre como criar um novo cluster, consulte Considerações .
- Use o VPC CNI recente: implante a versão do Amazon VPC CNI 1.10.1 ou posterior. Essa versão ou uma posterior é implantada por padrão. Depois de implantar o complemento, você não poderá fazer downgrade do seu complemento CNI da Amazon VPC para uma versão inferior a 1.10.1 sem remover primeiro todos os nós em todos os grupos de nós do seu cluster.
- Configure o VPC CNI para **IPv6** : Se você usar nós do Amazon EC2, deverá configurar o complemento Amazon VPC CNI com delegação de prefixo IP e IPv6. Se você escolher a família IPv6 ao criar o seu cluster, a versão 1.10.1 do complemento elegerá tal configuração como padrão. Esse é o caso de complementos autogerenciados ou do Amazon EKS. Para obter mais informações sobre a delegação de prefixos IP, consulte the section called “Aumento dos endereços IP”.
- Configure os endereços **IPv4** e **IPv6**: Quando você cria um cluster, a VPC e as sub-redes que você especificar devem ter um bloco CIDR IPv6 atribuído à VPC e às sub-redes que você especificar. Elas também precisam ter um bloco CIDR IPv4 atribuído a elas. Isso ocorre porque, ainda que você só queira usar o IPv6, uma VPC ainda precisa de um bloco CIDR IPv4 para funcionar. Para obter mais informações, consulte Associar um bloco CIDR IPv6 à sua VPC no Guia do usuário da Amazon VPC.
- Atribuição automática de endereços IPv6 aos nós: Ao criar seus nós, você deve especificar sub-redes que estejam configuradas para atribuir automaticamente endereços IPv6. Caso contrário, você não poderá implantar seus nós. Por padrão, essa configuração fica desativada. Para obter mais informações, consulte Modificar o atributo de endereçamento IPv6 para a sua sub-rede no Guia do usuário da Amazon VPC.
- Defina tabelas de rotas para usar **IPv6** : As tabelas de rotas atribuídas às suas sub-redes devem ter rotas para endereços IPv6. Para obter mais informações, consulte Migrar para IPv6 no Guia do usuário da Amazon VPC.

- Defina grupos de segurança para o **`IPv6`** : seus grupos de segurança devem permitir endereços `IPv6`. Para obter mais informações, consulte Migrar para IPv6 no Guia do usuário da Amazon VPC.
- Configurar o balanceador de carga: use a versão `2.3.1` ou mais recente do AWS Load Balancer Controller para balancear a carga de aplicações HTTP usando the section called “Balanceador de carga da aplicação” ou o tráfego de rede usando the section called “Balanceamento de carga da rede” para pods `IPv6` com qualquer um dos balanceadores de carga no modo IP, mas não no modo de instância. Para obter mais informações, consulte the section called “ AWS Load Balancer Controller”.
- Adicione a política do IAM **`IPv6`**: Você deve anexar uma política do IAM do `IPv6` à perfil do IAM do nó ou de IAM da CNI. Entre os dois, recomendamos que você a anexe a uma função do IAM do CNI. Para obter mais informações, consulte the section called “Criar uma política do IAM para clusters que usam a família `IPv6`” e the section called “Etapa 1: criar o perfil do IAM do plug-in CNI da Amazon VPC para Kubernetes”.
- Avalie todos os componentes: Faça uma avaliação completa de suas aplicações, dos complementos do Amazon EKS e dos serviços AWS com os quais você se integra antes de implantar os clusters `IPv6`. Isso ocorre para garantir que tudo funcione conforme o esperado com o `IPv6`.

Implantar um cluster **`IPv6`** do Amazon EKS e nós gerenciados do Amazon Linux

Neste tutorial, você implanta uma Amazon VPC `IPv6`, um cluster do Amazon EKS com a família `IPv6` e um grupo de nós gerenciados com nós do Amazon Linux do Amazon EC2. Não é possível implantar nós do Windows para Amazon EC2 em um cluster do `IPv6`. Você também pode implantar nós do Fargate em seu cluster. Para simplificar, tais instruções não são fornecidas neste tópico.

Pré-requisitos

Antes de começar o tutorial, conclua as seguintes etapas:

Instale e configure as ferramentas e recursos a seguir, necessários para criar e gerenciar um cluster do Amazon EKS.

- Recomendamos que você se familiarize com todas as configurações e implante um cluster com aquelas que atendam aos seus requisitos. Para obter mais informações, consulte the section called “Criar um cluster”, the section called “Grupos de nós gerenciados” e as Considerações sobre este tópico. Só é possível habilitar algumas configurações durante a criação do seu cluster.

- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- A entidade principal de segurança do IAM que você está usando deve ter permissões para trabalhar com perfis do IAM do Amazon EKS, perfis vinculadas a serviços, AWS CloudFormation, um VPC e recursos relacionados. Para obter mais informações, consulte Ações e uso de perfis vinculadas a serviços no Guia do Usuário do IAM.
- Caso use o eksctl, instale a versão `0.215.0` ou mais recente no seu computador. Para instalá-la ou atualizá-la, consulte Instalação na documentação do `eksctl`.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell. Se você usar o AWS CloudShell, talvez seja necessário instalar a versão 2.12.3 ou superior ou a versão 1.27.160 ou superior da AWS CLI, pois a versão padrão da CLI do AWS instalada no AWS CloudShell pode ser uma versão anterior.

Você pode usar o eksctl ou a CLI para implantar um cluster do `IPv6`.

Implantar um cluster do IPv6 com o eksctl

a. Crie o arquivo `ipv6-cluster.yaml`. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado:
   - Substitua *`my-cluster`* por um nome para seu cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
   - Substitua *`region-code`* por qualquer região AWS com suporte ao Amazon EKS. Para obter uma lista das regiões da AWS, consulte Endpoints e cotas de serviço do Amazon EKS no guia Referência geral da AWS.

- O valor para `version` com a versão do seu cluster. Para obter mais informações, consulte versões compatíveis do Amazon EKS.
- Substitua *`my-nodegroup`* por um nome para o seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.
- Substitua *`t3.medium`* por qualquer tipo de instância do AWS Nitro System.

```
cat >ipv6-cluster.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "X.XX"

kubernetesNetworkConfig:
  ipFamily: IPv6

addons:
  - name: vpc-cni
    version: latest
  - name: coredns
    version: latest
  - name: kube-proxy
    version: latest

iam:
  withOIDC: true

managedNodeGroups:
  - name: my-nodegroup
    instanceType: t3.medium
EOF
```

b. Crie o seu cluster.

```
eksctl create cluster -f ipv6-cluster.yaml
```

A criação do cluster leva muitos minutos. Não prossiga até ver a última linha de saída, que deve parecer com a seguinte.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

c. Verifique se os pods padrão estão atribuídos a endereços IPv6.

```
kubectl get pods -n kube-system -o wide
```

Veja abaixo um exemplo de saída.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                    NODE                                             NOMINATED NODE
 READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

d. Confirme se os serviços padrão são atribuídos a endereços IPv6.

```
kubectl get services -n kube-system -o wide
```

Veja abaixo um exemplo de saída.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

e. (Opcional) Implante uma aplicação de amostra ou implante AWS Load Balancer Controller e uma aplicação de amostra para realizar o balanceamento de carga de aplicações HTTP com the section called “Balanceador de carga da aplicação” ou de tráfego de rede com the section called “Balanceamento de carga da rede” para pods IPv6.

f. Após a conclusão de criação do cluster e nós para este tutorial, você deverá limpar os recursos que criou com o comando a seguir.

```
eksctl delete cluster my-cluster
```

Implante um cluster IPv6 com AWS CLI

Important

- Você deve concluir todas as etapas deste procedimento como o mesmo usuário. Execute o seguinte comando para verificar o usuário atual:

  ```
  aws sts get-caller-identity
  ```

- Você deve concluir todas as etapas deste procedimento no mesmo shell. Várias etapas usam variáveis definidas em etapas anteriores. As etapas que usam variáveis não funcionarão corretamente se os valores das variáveis forem definidos em um shell diferente. Se você usar o AWS CloudShell para concluir o procedimento a seguir, lembre-se de que, se não interagir com ele usando o teclado ou o ponteiro por aproximadamente 20 a 30 minutos, a sessão do shell será encerrada. Os processos em execução não contam como interações.
- As instruções são redigidas para o shell bash e podem precisar de ajustes em outros shells.

Substitua todos os valores de exemplo nas etapas deste procedimento pelos seus próprios valores.

a. Execute os comandos a seguir para definir algumas variáveis usadas em etapas posteriores. Substitua *region-code* pela região da AWS na qual você deseja implementar seus recursos. O valor pode ser qualquer região da AWS compatível com o Amazon EKS. Para obter uma lista das regiões da AWS, consulte Endpoints e cotas de serviço do Amazon EKS no guia Referência geral da AWS. Substitua *my-cluster* por um nome para seu cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster. Substitua *my-nodegroup* por um nome para o seu grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Substitua *111122223333* pelo ID da sua conta.

```
export region_code=region-code
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
```

b. Crie uma Amazon VPC com sub-redes públicas e privadas que atenda aos requisitos do Amazon EKS e IPv6.

   i. Execute o seguinte comando para definir uma variável para o nome da pilha do AWS CloudFormation. É possível substituir *my-eks-ipv6-vpc* por qualquer nome de sua preferência.

   ```
   export vpc_stack_name=my-eks-ipv6-vpc
   ```

   ii. Crie um VPC IPv6 usando um modelo AWS CloudFormation.

   ```
   aws cloudformation create-stack --region $region_code --stack-name $vpc_stack_name
    \
     --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
   cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
   ```

   A pilha leva alguns minutos para ser criada. Execute o comando a seguir. Não prossiga para a próxima etapa até que a saída do comando seja CREATE_COMPLETE.

   ```
   aws cloudformation describe-stacks --region $region_code --stack-name
    $vpc_stack_name --query Stacks[].StackStatus --output text
   ```

   iii. Recupere os IDs das sub-redes públicas que foram criadas.

```
aws cloudformation describe-stacks --region $region_code --stack-name
 $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
 text
```

Veja um exemplo de saída abaixo.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE
```

iv. Habilite a opção de atribuição automática de endereços IPv6 para as sub-redes públicas que foram criadas.

```
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-0a1a56c486EXAMPLE --assign-ipv6-address-on-creation
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-099e6ca77aEXAMPLE --assign-ipv6-address-on-creation
```

v. Recupere os nomes das sub-redes e grupos de segurança criados pelo modelo na pilha implantada do AWS CloudFormation e armazene-os em variáveis para usar em uma etapa posterior.

```
security_groups=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SecurityGroups`].OutputValue' --output
 text)

public_subnets=$(aws cloudformation describe-stacks --region $region_code --stack-
name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
 text)

private_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPrivate`].OutputValue' --output
 text)

subnets=${public_subnets},${private_subnets}
```

c. Crie um perfil do IAM de cluster e associe a política gerenciada do Amazon EKS IAM a ele. Os clusters do Kubernetes gerenciados pelo Amazon EKS fazem chamadas a outros serviços da AWS em seu nome para gerenciar os recursos que você usa com o serviço.

   i. Execute o seguinte comando para criar o arquivo `eks-cluster-role-trust-policy.json`.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   ii. Execute o seguinte comando para definir uma variável para o nome da sua função. É possível substituir *myAmazonEKSClusterRole* por qualquer nome de sua preferência.

```
export cluster_role_name=myAmazonEKSClusterRole
```

   iii. Crie a função.

```
aws iam create-role --role-name $cluster_role_name --assume-role-policy-document
 file://"eks-cluster-role-trust-policy.json"
```

   iv. Recupere o ARN da função do IAM e o armazene em uma variável para uma etapa posterior.

```
CLUSTER_IAM_ROLE=$(aws iam get-role --role-name $cluster_role_name --
query="Role.Arn" --output text)
```

   v. Anexe a política do IAM gerenciada pelo Amazon EKS ao perfil.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

d. Crie o seu cluster.

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.XX \
   --role-arn $CLUSTER_IAM_ROLE --resources-vpc-config subnetIds=
$subnets,securityGroupIds=$security_groups \
   --kubernetes-network-config ipFamily=ipv6
```

i. OBSERVAÇÃO: talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte the section called “Capacidade insuficiente”.

   A criação do cluster domora alguns minutos. Execute o comando a seguir. Não prossiga para a próxima etapa até que a saída do comando esteja `ACTIVE`.

   ```
   aws eks describe-cluster --region $region_code --name $cluster_name --query
    cluster.status
   ```

e. Crie ou atualize um arquivo `kubeconfig` para o seu cluster para que você possa se comunicar com ele.

   ```
   aws eks update-kubeconfig --region $region_code --name $cluster_name
   ```

   Por padrão, um arquivo `~/.kube` é criado em `config` ou o novo cluster é adicionado a um arquivo `config` existente no `~/.kube`.

f. Crie um perfil do IAM para o nó.

   i. Execute o seguinte comando para criar o arquivo `vpc-cni-ipv6-policy.json`.

   ```
   {
       "Version":"2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "ec2:AssignIpv6Addresses",
                   "ec2:DescribeInstances",
                   "ec2:DescribeTags",
                   "ec2:DescribeNetworkInterfaces",
   ```

```
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

ii. Crie a política do IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

iii. Execute o seguinte comando para criar o arquivo `node-role-trust-relationship.json`.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

iv. Execute o seguinte comando para definir uma variável para o nome da sua função. É possível substituir *`AmazonEKSNodeRole`* por qualquer nome de sua preferência.

```
export node_role_name=AmazonEKSNodeRole
```

v. Crie o perfil do IAM.

```
aws iam create-role --role-name $node_role_name --assume-role-policy-document
 file://"node-role-trust-relationship.json"
```

vi. Anexe a política do IAM à função do IAM.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy \
    --role-name $node_role_name
```

> **Important**
>
> Para simplificar este tutorial, a política é anexada a esta função do IAM. No entanto, em um cluster do produção, recomendamos que você anexe a política a uma função do IAM separada. Para obter mais informações, consulte the section called “Configuração para IRSA”.

vii.Anexe duas políticas do IAM gerenciadas necessárias à função do IAM.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
```

viiiRecupere o ARN da função do IAM e o armazene em uma variável para uma etapa posterior.

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --query="Role.Arn" --
output text)
```

g. Crie um grupo de nós gerenciado.

i. Exiba os IDs das sub-redes que você criou em uma etapa anterior.

```
echo $subnets
```

Veja um exemplo de saída abaixo.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE,subnet-0377963d69EXAMPLE,subnet-0c05f819
```

ii. Crie o grupo de nós. Substitua *0a1a56c486EXAMPLE*, *099e6ca77aEXAMPLE*, *0377963d69EXAMPLE* e *0c05f819d5EXAMPLE* pelos valores retornados na saída da etapa anterior. Certifique-se de remover as vírgulas entre IDs de sub-redes da saída anterior no comando a seguir. Você pode substituir *t3.medium* por qualquer tipo de instância do AWS Nitro System.

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --subnets subnet-0a1a56c486EXAMPLE subnet-099e6ca77aEXAMPLE
 subnet-0377963d69EXAMPLE subnet-0c05f819d5EXAMPLE \
    --instance-types t3.medium --node-role $node_iam_role
```

A criação do grupo de nós demora alguns minutos. Execute o comando a seguir. Não prossiga para a próxima etapa até que a saída recebida esteja `ACTIVE`.

```
aws eks describe-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --query nodegroup.status --output text
```

h. Verifique se os pods padrão estão atribuídos a endereços `IPv6` na coluna `IP`.

```
kubectl get pods -n kube-system -o wide
```

Veja abaixo um exemplo de saída.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                    NODE                                            NOMINATED NODE
 READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

```
kube-proxy-btpbk           1/1     Running   0          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

i. Confirme se os serviços padrão são atribuídos a endereços IPv6 na coluna IP.

```
kubectl get services -n kube-system -o wide
```

Veja abaixo um exemplo de saída.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

j. (Opcional) Implante uma aplicação de amostra ou implante AWS Load Balancer Controller e uma aplicação de amostra para realizar o balanceamento de carga de aplicações HTTP com the section called “Balanceador de carga da aplicação” ou de tráfego de rede com the section called “Balanceamento de carga da rede” para pods IPv6.

k. Após a conclusão de criação do cluster e nós para este tutorial, você deverá limpar os recursos que criou com os comandos a seguir. Certifique-se de que você não esteja usando nenhum dos recursos fora deste tutorial antes de excluí-los.

i. Se você estiver concluindo esta etapa em um shell diferente daquele em que concluiu as etapas anteriores, defina os valores de todas as variáveis usadas nas etapas anteriores, substituindo os valores de exemplo pelos valores especificados ao concluir as etapas anteriores. Se você estiver concluindo esta etapa no mesmo shell em que concluiu as etapas anteriores, pule para a próxima etapa.

```
export region_code=region-code
export vpc_stack_name=my-eks-ipv6-vpc
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
export node_role_name=AmazonEKSNodeRole
export cluster_role_name=myAmazonEKSClusterRole
```

ii. Exclua o seu grupo de nós.

```
aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name
```

A exclusão demora alguns minutos. Execute o comando a seguir. Não prossiga para a próxima etapa se receber alguma saída.

```
aws eks list-nodegroups --region $region_code --cluster-name $cluster_name --query
 nodegroups --output text
```

iii. Excluir o cluster.

```
aws eks delete-cluster --region $region_code --name $cluster_name
```

Demora alguns minutos até que o cluster seja excluído. Antes de continuar, verifique se o cluster foi excluído com o seguinte comando.

```
aws eks describe-cluster --region $region_code --name $cluster_name
```

Não prossiga para a próxima etapa até que a saída seja semelhante à seguinte saída.

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-cluster.
```

iv. Exclua os recursos do IAM que você criou. Substitua *AmazonEKS_CNI_IPv6_Policy* pelo nome que você escolheu, se o mesmo for diferente do que foi usado nas etapas anteriores.

```
aws iam detach-role-policy --role-name $cluster_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-policy --policy-arn arn:aws:iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy
aws iam delete-role --role-name $cluster_role_name
aws iam delete-role --role-name $node_role_name
```

v. Exclua a pilha do AWS CloudFormation que criou a VPC.

```
aws cloudformation delete-stack --region $region_code --stack-name $vpc_stack_name
```

Habilitar o acesso de saída à internet para pods

Aplica-se a: nós Linux `IPv4` do Fargate, nós Linux com instâncias do Amazon EC2

Se você implantou seu cluster utilizando a família `IPv6`, as informações neste tópico não serão aplicáveis a ele, pois endereços `IPv6` não são convertidos na rede. Para obter mais informações sobre como usar o `IPv6` com o seu cluster, consulte the section called “IPv6”.

Por padrão, cada pod no cluster recebe um endereço `IPv4` privado de um bloco de Encaminhamento Entre Domínios Sem Classificação (CIDR) associado à VPC em que o pod está implantado. Os pods na mesma VPC se comunicam entre si usando esses endereços IP privados como endpoints. Quando um pod se comunica com qualquer endereço `IPv4` que não esteja em um bloco CIDR associado à VPC, o plug-in CNI da Amazon VPC (para Linux ou Windows) converte o endereço `IPv4` do pod para o endereço privado primário `IPv4` da interface de rede elástica primária do nó em que o pod está sendo executado, por padrão, *.

> **Note**
>
> Para nós do Windows, existem detalhes adicionais a serem considerados. Por padrão, o plug-in VPC CNI para Windows é definido com uma configuração de rede na qual o tráfego para um destino dentro da mesma VPC é excluído para SNAT. Isso significa que a comunicação interna da VPC tem a SNAT desabilitada e que o endereço IP alocado a um pod é roteável dentro da VPC. Porém, o tráfego para um destino fora da VPC tem a SNAT do IP de origem do pod conectado ao endereço IP primário da ENI da instância. Essa configuração padrão para o Windows garante que o pod possa acessar redes fora da sua VPC da mesma forma que a instância do host.

Devido a este comportamento:

- Os pods poderão se comunicar com os recursos da internet somente se o nó em que estão sendo executados tiver um endereço IP público ou elástico atribuído a ele e se estiver em uma sub-rede pública. A tabela de rotas associada a uma sub-rede pública tem uma rota para um gateway da Internet. Convém implantar nós em sub-redes privadas, sempre que possível.

- Para versões do plug-in anteriores a `1.8.0`, os recursos que estão em redes ou VPCs conectados à VPC do cluster usando o emparelhamento da VPC, uma VPC de trânsito ou o AWS Direct Connect não podem iniciar a comunicação com os pods por meio de interfaces de rede elásticas secundárias. No entanto, os pods poderão iniciar a comunicação com esses recursos e receber respostas deles.

Se qualquer uma das declarações a seguir for verdadeira em seu ambiente, altere a configuração padrão com o comando a seguir.

- Você tem recursos em redes ou VPCs que estão conectados à VPC do seu cluster usando emparelhamento da VPC, uma VPC de trânsito ou o AWS Direct Connect que precisam iniciar a comunicação com os pods usando um endereço `IPv4` e sua versão de plug-in anterior à `1.8.0`.
- Os pods estão em uma sub-rede privada e precisam se comunicar com a saída da internet. A sub-rede tem uma rota para um gateway NAT.

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

> **Note**
>
> As variáveis de configuração `AWS_VPC_K8S_CNI_EXTERNALSNAT` e `AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS` da CNI não são aplicáveis a nós do Windows. A desabilitação da SNAT não tem suporte no Windows. Quanto à exclusão de uma lista de CIDRs `IPv4` da SNAT, você pode definir isso especificando o parâmetro `ExcludedSnatCIDRs` no script de bootstrap do Windows. Para obter mais informações sobre o uso desse parâmetro, consulte the section called “Parâmetros de configuração do script de bootstrap”.

Redes do host

* Se uma especificação de pod contiver `hostNetwork=true` (o padrão é `false`), seu endereço IP não será convertido para um endereço diferente. Este é o caso para o `kube-proxy` e o plug-in CNI da Amazon VPC dos pods do Kubernetes que são executados no cluster por padrão. Para esses pods, o endereço IP é o mesmo que o endereço IP primário do nó e, portanto, o endereço IP do pod não é convertido. Para saber mais sobre a configuração `hostNetwork` de um pod, consulte PodSpec v1 core na referência de APIs do Kubernetes.

Limitar o tráfego do pod com políticas de rede do Kubernetes

Visão geral

Por padrão, não há restrições no Kubernetes para endereços IP, portas ou conexões entre os pods do cluster ou entre os pods e os recursos de qualquer outra rede. Você pode usar uma política de rede do Kubernetes para restringir o tráfego de rede que entra e sai dos pods. Para obter mais informações, consulte Políticas de rede na documentação do Kubernetes.

Política de rede padrão

Você pode usar a política `NetworkPolicy` padrão para segmentar o tráfego entre pods no cluster. Essas políticas de rede operam nas camadas 3 e 4 do modelo de rede OSI, permitindo controlar o fluxo de tráfego no nível de endereço IP ou de porta dentro do cluster do Amazon EKS. As políticas de rede padrão têm escopo definido no nível do namespace.

Casos de uso

- Realize a segmentação do tráfego de rede entre as workloads para assegurar que a comunicação ocorra apenas entre aplicações relacionadas.
- Realize o isolamento dos locatários por namespace, usando políticas para garantir a separação de rede.

Exemplo

Na política abaixo, o tráfego de saída proveniente dos pods webapp no namespace sun é restrito.

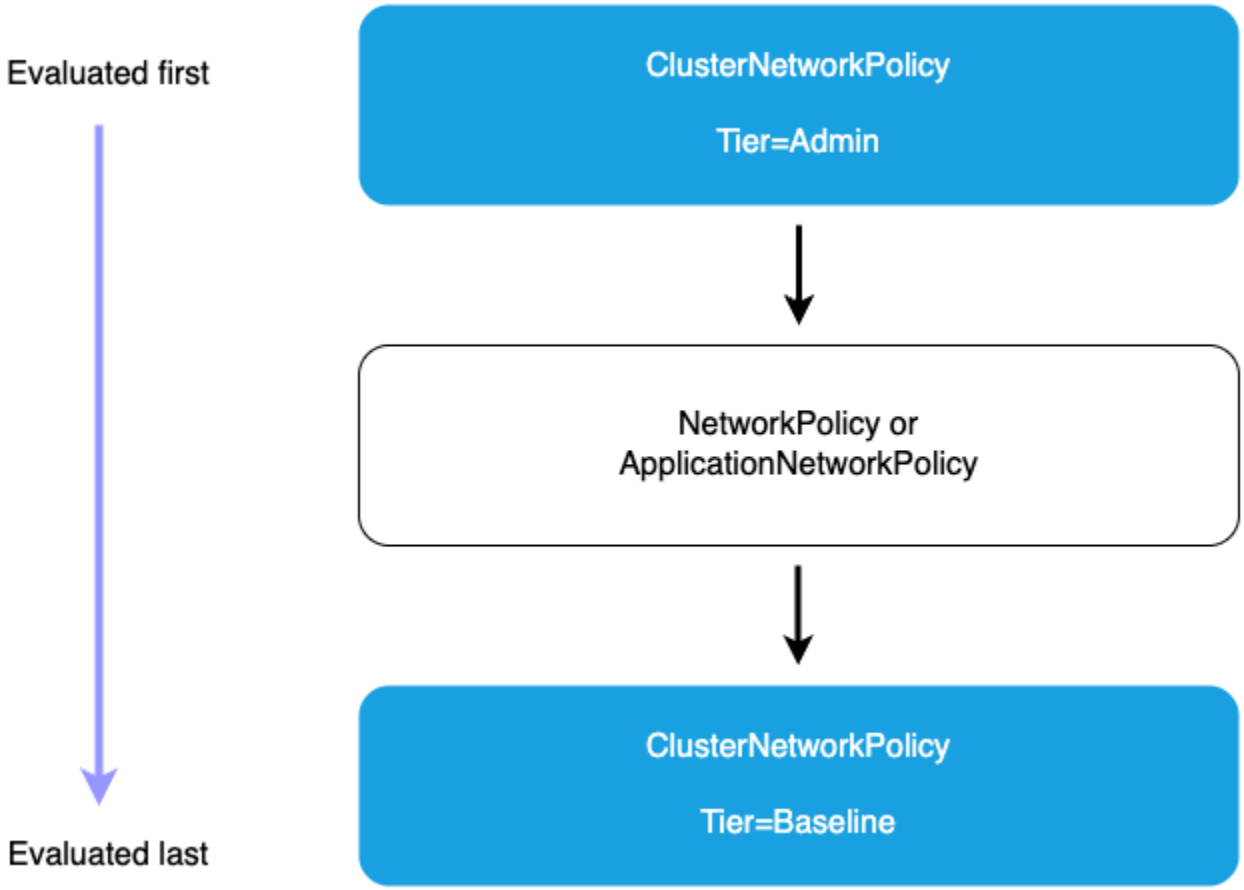
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: webapp-egress-policy
  namespace: sun
spec:
  podSelector:
    matchLabels:
      role: webapp
  policyTypes:
  - Egress
  egress:
  - to:
```

```
    - namespaceSelector:
        matchLabels:
          name: moon
      podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 8080
  - to:
    - namespaceSelector:
        matchLabels:
          name: stars
      podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 8080
```

A política é aplicada aos pods com o rótulo `role: webapp` no namespace `sun`.

- Tráfego permitido: pods com o rótulo `role: frontend` no namespace `moon` na porta TCP `8080`
- Tráfego permitido: pods com o rótulo “role: frontend” no namespace `stars` na porta TCP `8080`
- Tráfego bloqueado: todo o restante do tráfego de saída proveniente dos pods `webapp` é negado implicitamente

Política de rede do administrador (ou do cluster)



É possível usar a política `ClusterNetworkPolicy` para impor um padrão de segurança de rede válido para todo o cluster. Em vez de criar e gerenciar uma política diferente para cada namespace de forma repetitiva, é possível usar uma política única para o controle centralizado do acesso à rede de diversas workloads no cluster, independentemente do namespace em que estejam.

Casos de uso

- Gerencie de forma centralizada o acesso à rede de todas as workloads (ou de um grupo específico delas) no seu cluster do EKS.
- Defina uma postura de segurança de rede padrão em todo o cluster.
- Amplie os padrões de segurança organizacionais ao escopo do cluster com maior eficiência operacional.

Exemplo

Na política abaixo, é possível bloquear explicitamente o tráfego do cluster proveniente de outros namespaces para impedir o acesso de rede a um namespace da workload sensível.

```
apiVersion: networking.k8s.aws/v1alpha1
kind: ClusterNetworkPolicy
metadata:
  name: protect-sensitive-workload
spec:
  tier: Admin
  priority: 10
  subject:
    namespaces:
      matchLabels:
        kubernetes.io/metadata.name: earth
  ingress:
    - action: Deny
      from:
      - namespaces:
          matchLabels: {} # Match all namespaces.
      name: select-all-deny-all
```

Observações importantes

As políticas de rede no plug-in CNI da Amazon VPC para Kubernetes são compatíveis com as configurações listadas abaixo.

- Versão 1.21.0 (ou versões posteriores) do plug-in CNI da Amazon VPC para políticas de rede padrão e de administração.
- Cluster configurado para endereços `IPv4` ou `IPv6`.
- Você pode usar políticas de rede com grupos de segurança para Pods. Com as políticas de rede, você pode controlar toda a comunicação dentro do cluster. Com os grupos de segurança para pods, você pode controlar o acesso aos serviços da AWS de aplicações em um pod.
- Você pode usar políticas de rede com rede personalizada e delegação de prefixo.

Considerações

Arquitetura do

- Ao aplicar as políticas de rede do plug-in CNI da Amazon VPC para Kubernetes ao cluster com o plug-in CNI da Amazon VPC para Kubernetes, você só poderá aplicar as políticas aos nós do Linux do Amazon EC2. Você não pode aplicar as políticas aos nós do Fargate ou do Windows.
- As políticas de rede se aplicam somente a endereços `IPv4` ou `IPv6`, mas não a ambos. Em um cluster `IPv4`, a VPC CNI atribui endereços `IPv4` aos pods e aplica políticas de `IPv4`. Em um cluster `IPv6`, a VPC CNI atribui endereços `IPv6` aos pods e aplica políticas de `IPv6`. Todas as regras de política de rede `IPv4` aplicadas a um cluster `IPv6` são ignoradas. Todas as regras de política de rede `IPv6` aplicadas a um cluster `IPv4` são ignoradas.

Políticas de rede

- As políticas de rede são aplicadas somente a pods que fazem parte de uma implantação. Os pods autônomos que não têm um conjunto `metadata.ownerReferences` não podem ter políticas de rede aplicadas a eles.
- Você pode aplicar várias políticas de rede ao mesmo pod. Quando duas ou mais políticas que selecionam o mesmo pod estão configuradas, todas as políticas são aplicadas ao pod.
- O número máximo de combinações de portas e protocolos para um único intervalo de endereços IP (CIDR) é 24 em todas as suas políticas de rede. Seletores como `namespaceSelector` resolvem para um ou mais CIDRs. Se vários seletores forem resolvidos para um único CIDR ou você especificar o mesmo CIDR direto várias vezes na mesma política de rede ou em políticas de rede diferentes, todos eles contarão para esse limite.
- Para qualquer serviço do Kubernetes, a porta de serviço deve ser a mesma que porta de contêiner. Se você estiver usando portas nomeadas, use o mesmo nome na especificação do serviço.

Políticas de rede de administração

1. Políticas do nível de Administrador (avaliadas primeiro): todas as ClusterNetworkPolicies do nível de Administrador são avaliadas antes de quaisquer outras políticas. No nível de Administrador, o processamento das políticas segue a ordem de prioridade (o menor número de prioridade tem precedência). O tipo de ação determina o que acontece em seguida no processamento.
   - Ação de negação (maior precedência): quando uma política de Administrador com uma ação de negação corresponde ao tráfego, esse tráfego é bloqueado imediatamente, independentemente de quaisquer outras políticas. O processamento de quaisquer outras regras de ClusterNetworkPolicy ou NetworkPolicy é interrompido. Isso garante que os controles

de segurança de nível organizacional não possam ser substituídos por políticas de nível de namespace.

- Ação de permissão: uma vez avaliadas as regras de negação, as políticas de Administrador com a ação de permissão entram em processamento seguindo a ordem de prioridade (o menor número de prioridade tem precedência). Se o tráfego coincidir com uma regra de permissão, ele é liberado imediatamente e o processo de avaliação de políticas é encerrado. Essas políticas podem conceder acesso entre diversos namespaces com base em seletores de rótulos, oferecendo controle centralizado sobre quais workloads podem acessar recursos específicos.
- Ação de aprovação: as ações de aprovação nas políticas do nível de Administrador delegam a tomada de decisão para os níveis inferiores. Quando o tráfego corresponde a uma regra de aprovação, a avaliação ignora todas as regras restantes do nível de Administrador para esse tráfego e prossegue diretamente para o nível de NetworkPolicy. Dessa forma, os administradores conseguem transferir formalmente o controle de padrões específicos de tráfego para as equipes responsáveis pelas aplicações. Por exemplo, é possível usar regras de aprovação para delegar o gerenciamento do tráfego interno do namespace aos administradores do namespace, enquanto mantém controles rígidos sobre o acesso externo.

2. Nível de política de rede: se nenhuma política do nível de Administrador corresponder às ações de negação ou de aprovação, ou se uma ação de aprovação for acionada, os recursos de NetworkPolicy com escopo de namespace são avaliados em seguida. Essas políticas oferecem controle granular nos namespaces individuais e são gerenciadas pelas equipes responsáveis pela aplicação. As políticas com escopo de namespace conseguem apenas aumentar a restrição imposta pelas políticas de Administrador. Eles não podem substituir a decisão de negação de uma política de Administrador, mas podem restringir ainda mais o tráfego que foi permitido ou aprovado pelas políticas de Administrador.
3. Políticas de Administrador do nível de Linha de Base: se nenhuma política do nível de Administrador ou de escopo de namespace corresponder ao tráfego, as ClusterNetworkPolicies do nível de Linha de Base são avaliadas. Estas políticas fornecem posturas de segurança padrão que podem ser substituídas por políticas com escopo de namespace, permitindo que os administradores definam padrões para toda a organização, enquanto oferecem flexibilidade às equipes para realizar personalizações conforme necessário. A avaliação das políticas de Linha de Base respeita a ordem de prioridade, na qual o menor número de prioridade tem precedência.
4. Negação padrão (se nenhuma política corresponder): este comportamento de negação por padrão garante que apenas conexões explicitamente permitidas sejam autorizadas, mantendo uma postura de segurança robusta.

Migração

- Se o cluster estiver usando atualmente uma solução de terceiros para gerenciar as políticas de rede do Kubernetes, você poderá usar essas mesmas políticas com o plug-in CNI da Amazon VPC para Kubernetes. Porém, você deve remover a solução existente para que ela não gerencie as mesmas políticas.

> **Warning**
>
> Recomendamos que, depois de remover uma solução de política de rede, você substitua todos os nós que tiveram a solução de política de rede aplicada a eles. Isso ocorre porque as regras de tráfego podem ser deixadas para trás por um pod da solução se ele sair repentinamente.

Instalação

- O atributo de política de rede cria e exige uma definição de atributos personalizados (CRD) de `PolicyEndpoint` denominada `policyendpoints.networking.k8s.aws`. Os objetos de `PolicyEndpoint` do atributo personalizado são gerenciados pelo Amazon EKS. Você não deve modificar nem excluir esses recursos.
- Se você executar pods que usem as credenciais do IAM do perfil da instância ou se conectar ao IMDS do EC2, tenha o cuidado de verificar as políticas de rede que bloqueariam o acesso ao IMDS do EC2. Pode ser necessário adicionar uma política de rede para permitir o acesso ao IMDS do EC2. Para obter mais informações, consulte Metadados da instância e dados do usuário no Manual do usuário do Amazon EC2.

  Pods que usam perfis do IAM para contas de serviço ou identidade de Pods do EKS não acessam o IDMS do EC2.
- O plug-in CNI da Amazon VPC para Kubernetes não aplica políticas de rede a interfaces de rede adicionais para cada pod, somente à interface primária para cada pod (`eth0`). Isso afeta as seguintes arquiteturas:
  - Pods `IPv6` com a variável `ENABLE_V4_EGRESS` definida como `true`. Essa variável permite que o recurso de saída `IPv4` conecte os pods IPv6 a endpoints `IPv4`, como aqueles fora do cluster. O funcionamento do recurso de saída `IPv4` se dá com a criação de uma interface de rede adicional com um endereço IPv4 de loopback local.

- Ao usar plug-ins de rede encadeados, como o Multus. Como esses plug-ins adicionam interfaces de rede a cada pod, as políticas de rede não são aplicadas aos plug-ins de rede encadeados.

Restringir o tráfego de rede de pods com políticas de rede do Kubernetes

Você pode usar uma política de rede do Kubernetes para restringir o tráfego de rede que entra e sai dos pods. Para obter mais informações, consulte Políticas de rede na documentação do Kubernetes.

É necessário fazer as seguintes configurações para usar esse atributo:

1. Configure aplicação de políticas na inicialização do pod. Você faz isso no contêiner `aws-node` do `DaemonSet` CNI da VPC.
2. Habilitar o parâmetro da política de redes para o complemento.
3. Configurar o cluster para usar as políticas de rede do Kubernetes

Antes de começar, reveja as considerações. Para obter mais informações, consulte the section called "Considerações".

Pré-requisitos

A seguir estão os pré-requisitos do recurso:

Versão mínima do cluster

Um cluster existente do Amazon EKS. Para implantar, consulte Conceitos básicos. O cluster dove estar executando uma das versões do Kubernetes e versões da plataforma listadas na tabela a seguir. Observe que qualquer versão da plataforma e do Kubernetes posterior às versões listadas também é compatível. Para verificar a versão atual do Kubernetes, substitua *my-cluster* no comando a seguir pelo nome do seu cluster e execute o comando modificado:

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Versão do Kubernetes	Versão da plataforma
1.27.4	eks.5
1.26.7	eks.6

Versão mínima do VPC CNI

Para criar tanto políticas de rede padrão do Kubernetes quanto políticas de rede de administrador, é necessário executar a versão `1.21` do plug-in CNI da VPC. É possível verificar qual é a sua versão atual com o comando a seguir.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Se a versão for anterior à `1.21`, consulte the section called “Atualizar (complemento do EKS)” para atualizar para a versão `1.21` ou superior.

Versão mínima do kernel do Linux

Os nós devem ter a versão `5.10` ou superior do kernel do Linux. Você pode verificar qual é a sua versão atual com `uname -r`. Se você usa as versões mais recentes do Amazon Linux otimizado para Amazon EKS, das AMIs do Amazon Linux otimizadas para Amazon EKS e as AMIs do Bottlerocket, elas já têm a versão do kernel exigida.

A AMI do Amazon Linux acelerada otimizada para Amazon EKS versão `v20231116` ou superior possui o kernel versão `5.10`.

Etapa 1: configurar a aplicação de políticas na inicialização dos pods

O plug-in CNI da Amazon VPC para Kubernetes configura as políticas de rede para os pods em paralelo ao provisionamento de pods. Até que todas as políticas estejam configuradas para o novo pod, os contêineres no novo pod serão iniciados com uma política de permissão padrão. Isso é denominado modo padrão. Uma política de permissão padrão significa que todo o tráfego de entrada e de saída é permitido de e para os novos pods. Por exemplo, os pods não terão nenhuma regra de firewall aplicada (todo o tráfego é permitido) até que o novo pod seja atualizado com as políticas ativas.

Com a variável `NETWORK_POLICY_ENFORCING_MODE` definida como `strict`, os pods que usam o plug-in CNI da VPC são iniciados com uma política de negação padrão e, em seguida, as políticas são configuradas. Isso é denominado modo estrito. No modo estrito, você deve ter uma política de rede para cada endpoint que os pods precisam acessar no cluster. Observe que esse requisito se aplica aos pods do CoreDNS. A política de negação padrão não está configurada para pods com sistemas de redes de host.

É possível alterar a política de rede padrão ao definir a variável de ambiente `NETWORK_POLICY_ENFORCING_MODE` como `strict` no contêiner `aws-node` do `DaemonSet` do plug-in CNI da VPC.

```
env:
  - name: NETWORK_POLICY_ENFORCING_MODE
    value: "strict"
```

Etapa 2: habilitar o parâmetro da política de redes do complemento

O recurso de política de rede usa a porta `8162` no nó para métricas por padrão. Além disso, o recurso usa a porta `8163` para a realização de sondagens de integridade. Se você executar outro aplicativo nos nós ou dentro dos pods que precisa usar essas portas, o aplicativo não será executado. Na versão `v1.14.1` ou em versões posteriores da CNI da VPC, é possível alterar essas portas.

Use o procedimento a seguir para habilitar o parâmetro da política de redes do complemento.

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, selecione Clusters e o nome do cluster para o qual você deseja configurar o complemento Amazon VPC CNI.
3. Escolha a guia Add-ons (Complementos).
4. Selecione a caixa no canto superior direito da caixa do complemento e depois escolha Edit (Editar).
5. Na página Configurar **`Amazon VPC CNI`**:
   a. Selecione a versão `v1.14.0-eksbuild.3` ou superior na lista Versão.
   b. Expanda Definições de configuração opcionais.
   c. Insira a chave JSON `"enableNetworkPolicy":` e o valor `"true"` em Valores da configuração. O texto resultante deve ser um objeto JSON válido. Se esse par de chave e valor for o único dado na caixa de texto, coloque-o entre colchetes `{ }`.

      O seguinte exemplo tem a funcionalidade de política de rede ativada e as métricas e investigações de integridade estão configuradas com os números de porta padrão:

      ```
      {
      ```

```
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "healthProbeBindAddr": "8163",
        "metricsBindAddr": "8162"
    }
}
```

Helm

Se você instalou o plug-in CNI da Amazon VPC para Kubernetes por meio do `helm`, você pode atualizar a configuração para alterar as portas.

1. Execute o seguinte comando para alterar as portas. Defina o número da porta no valor da chave `nodeAgent.metricsBindAddr` ou da chave `nodeAgent.healthProbeBindAddr`, respectivamente.

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
 nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/aws-vpc-
cni
```

kubectl

1. Abra o `aws-node DaemonSet` no editor.

```
kubectl edit daemonset -n kube-system aws-node
```

2. Substitua os números da porta nos argumentos do comando a seguir no contêiner `args:` no manifesto do daemonset `aws-network-policy-agent` do CNI do `aws-node` VPC.

```
    - args:
            - --metrics-bind-addr=:8162
            - --health-probe-bind-addr=:8163
```

Etapa 3: configurar o cluster para usar as políticas de rede do Kubernetes

É possível configurar isso para um complemento do Amazon EKS ou autogerenciado.

Complemento do Amazon EKS

Usando a AWS CLI, você pode configurar o cluster para usar as políticas de rede do Kubernetes ao executar o comando a seguir. Substitua `my-cluster` pelo nome do cluster e o ARN do perfil do IAM pelo perfil que você está usando.

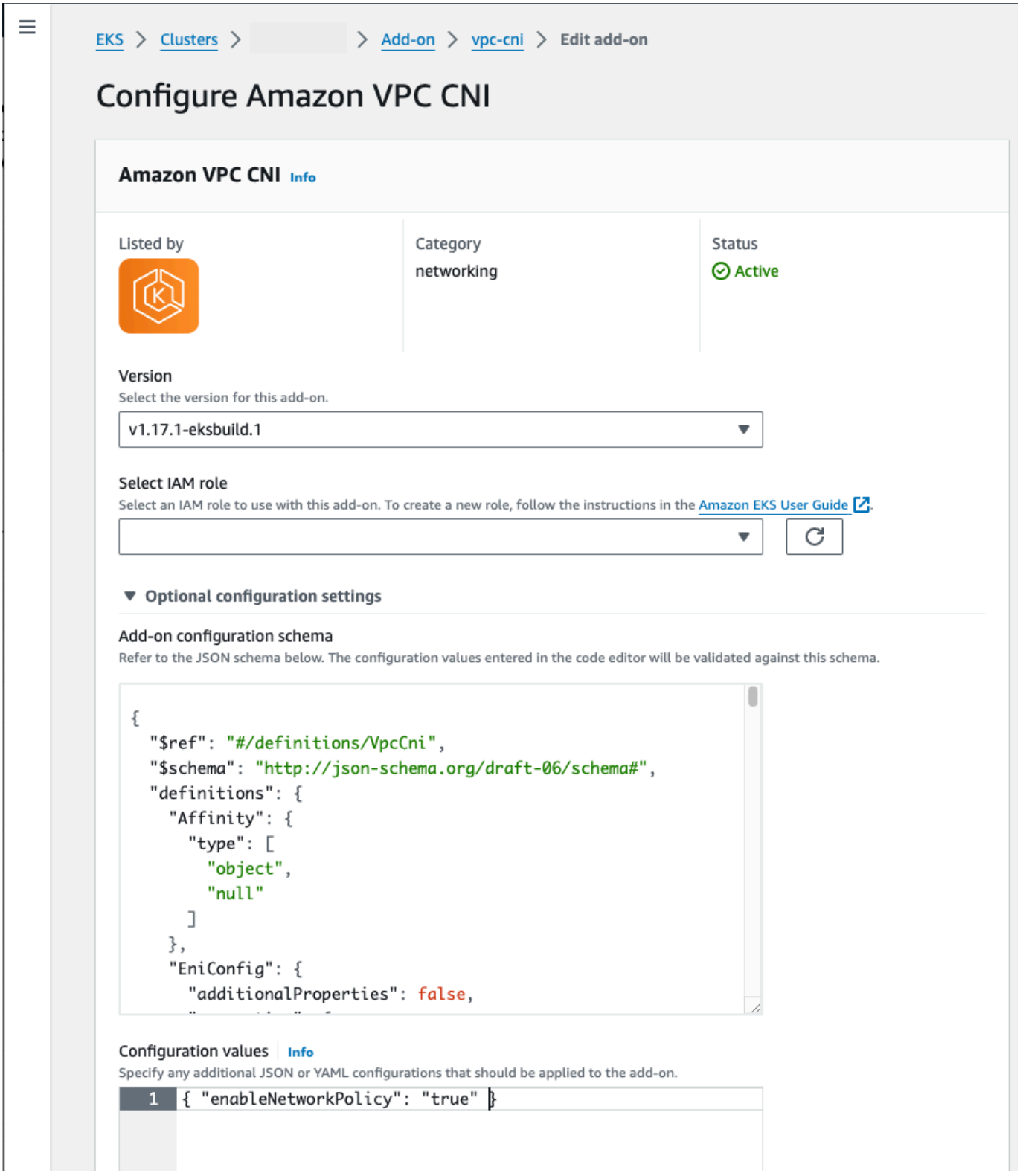
```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"enableNetworkPolicy":
 "true"}'
```

Para definir essa configuração usando o Console de Gerenciamento da AWS, siga as etapas abaixo:

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, selecione Clusters e o nome do cluster para o qual você deseja configurar o complemento Amazon VPC CNI.
3. Escolha a guia Add-ons (Complementos).
4. Selecione a caixa no canto superior direito da caixa do complemento e depois escolha Edit (Editar).
5. Na página Configurar **`Amazon VPC CNI`**:
   a. Selecione a versão `v1.14.0-eksbuild.3` ou superior na lista Versão.
   b. Expanda Definições de configuração opcionais.
   c. Insira a chave JSON `"enableNetworkPolicy":` e o valor `"true"` em Valores da configuração. O texto resultante deve ser um objeto JSON válido. Se esse par de chave e valor for o único dado na caixa de texto, coloque-o entre colchetes `{ }`. O exemplo apresentado a seguir mostra que a política de rede está habilitada:

      ```
      { "enableNetworkPolicy": "true" }
      ```

      A captura de tela a seguir mostra um exemplo desse cenário.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 { "enableNetworkPolicy": "true" }

Complemento autogerenciado

Helm

Se você instalou o plug-in CNI da Amazon VPC para Kubernetes por meio do `helm`, você pode atualizar a configuração para habilitar a política de rede.

1. Execute o comando a seguir para habilitar a política de rede.

```
helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. Abra o `amazon-vpc-cni` `ConfigMap` no editor.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

2. Adicione a linha a seguir aos `data` no `ConfigMap`.

```
enable-network-policy-controller: "true"
```

   Depois de adicionar a linha, o `ConfigMap` deverá ser semelhante ao exemplo a seguir.

```
apiVersion: v1
 kind: ConfigMap
 metadata:
  name: amazon-vpc-cni
  namespace: kube-system
 data:
  enable-network-policy-controller: "true"
```

3. Abra o `aws-node` `DaemonSet` no editor.

```
kubectl edit daemonset -n kube-system aws-node
```

   a. Substitua `false` por `true` no argumento do comando `--enable-network-policy=false` em `args:` no contêiner do `aws-network-policy-agent` no manifesto do daemonset do `aws-node` do VPC CNI.

```
    - args:
       - --enable-network-policy=true
```

Etapa 4: Próximas etapas

Depois de concluir a configuração, confirme se os pods `aws-node` estão sendo executados no cluster.

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

Veja abaixo um exemplo de saída.

```
aws-node-gmqp7                                          2/2     Running   1 (24h ago)
 24h
aws-node-prnsh                                          2/2     Running   1 (24h ago)
 24h
```

Existem dois contêineres nos pods `aws-node` na versão `1.14` e posteriores. Nas versões anteriores e quando a política de rede está desabilitada, há apenas um único contêiner nos pods do `aws-node`.

Agora você pode implantar políticas de rede do Kubernetes no cluster.

Para implementar políticas de rede do Kubernetes, você pode criar objetos `NetworkPolicy` ou `ClusterNetworkPolicy` e implantá-los em seu cluster. Os objetos `NetworkPolicy` são restritos a um namespace, enquanto objetos `ClusterNetworkPolicy` podem abranger todo o cluster ou diversos namespaces. Você implementa políticas para permitir ou negar tráfego entre os pods com base em seletores de rótulos, namespaces e intervalos de endereços IP. Para obter mais informações sobre a criação de objetos de `NetworkPolicy`, consulte Políticas de rede na documentação do Kubernetes.

A aplicação dos objetos de `NetworkPolicy` do Kubernetes é implementada usando o Extended Berkeley Packet Filter (eBPF). Em relação a implementações baseadas em `iptables`, ele oferece menor latência e características de performance, incluindo menor utilização da CPU e prevenção de consultas sequenciais. Além disso, as sondas do eBPF fornecem acesso a dados contextuais ricos que ajudam a depurar problemas complexos no nível do kernel e a melhorar a observabilidade. O Amazon EKS é compatível com um exportador baseado no eBPF que aproveita as sondas para registrar em log os resultados da política em cada nó e exportar os dados para coletores de logs

externos para auxiliar na depuração. Para obter mais informações, consulte a documentação do eBPF.

Desabilitar as políticas de rede do Kubernetes para o tráfego de rede de pods do Amazon EKS

Desabilite as políticas de rede do Kubernetes para interromper a restrição do tráfego de rede de pods do Amazon EKS

1. Liste todas as políticas de rede do Kubernetes.

```
kubectl get netpol -A
```

2. Exclua cada política de rede do Kubernetes. Você deve excluir todas as políticas de rede antes de desabilitar as políticas de rede.

```
kubectl delete netpol <policy-name>
```

3. Abra o DaemonSet aws-node em seu editor.

```
kubectl edit daemonset -n kube-system aws-node
```

4. Substitua `true` por `false` no argumento do comando `--enable-network-policy=true` em `args:` no contêiner do `aws-network-policy-agent` no manifesto do daemonset do `aws-node` do VPC CNI.

```
    - args:
       - --enable-network-policy=true
```

Solucionar problemas de políticas de rede do Kubernetes para o Amazon EKS

Este é o guia de solução de problemas do recurso de políticas de rede da CNI da Amazon VPC.

Este guia aborda:

- Informações de instalação, CRD e permissões de RBAC the section called “Novo CRD `policyendpoints` e permissões”
- Logs a serem examinados ao diagnosticar problemas de políticas de rede the section called “Logs da política de rede”
- Execução da coleção de ferramentas do SDK do eBPF para solucionar problemas

- Problemas e soluções conhecidos the section called “Problemas e soluções conhecidos”

> **Note**
>
> Observe que as políticas de rede são aplicadas somente aos pods criados pelas Implementações do Kubernetes. Para obter mais limitações das políticas de rede na CNI da VPC, consulte the section called “Considerações”.

Você pode solucionar problemas e investigar conexões de rede que usam políticas de rede lendo os logs de política de rede e executando ferramentas do SDK do eBPF.

Novo CRD **`policyendpoints`** e permissões

- CRD: `policyendpoints.networking.k8s.aws`
- API do Kubernetes: `apiservice` chamou `v1.networking.k8s.io`
- Recurso do Kubernetes: `Kind: NetworkPolicy`
- RBAC: `ClusterRole` chamou `aws-node` (CNI da VPC), `ClusterRole` chamou `eks:network-policy-controller` (controlador de políticas de rede no ambiente de gerenciamento de clusters do EKS)

Para a política de rede, a CNI da VPC cria um novo `CustomResourceDefinition` (CRD) denominado `policyendpoints.networking.k8s.aws`. A CNI da VPC deve ter permissões para criar o CRD e o CustomResources (CR) dele e do outro CRD instalado pela CNI da VPC (`eniconfigs.crd.k8s.amazonaws.com`). Ambos os CRDs estão disponíveis no arquivo `crds.yaml` no GitHub. Especificamente, a CNI da VPC deve ter as permissões das ações “get”, “list” e “watch” para `policyendpoints`.

A Política de Rede do Kubernetes faz parte da `apiservice` denominada `v1.networking.k8s.io`, e isso é `apiversion: networking.k8s.io/v1` nos seus arquivos YAML da política. A CNI da VPC `DaemonSet` precisa ter permissões para usar essa parte da API do Kubernetes.

As permissões da CNI da VPC estão em um `ClusterRole` denominado `aws-node`. Observe que os objetos `ClusterRole` não estão agrupados em namespaces. O seguinte mostra o `aws-node` de um cluster:

```
kubectl get clusterrole aws-node -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/instance: aws-vpc-cni
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: aws-node
    app.kubernetes.io/version: v1.19.4
    helm.sh/chart: aws-vpc-cni-1.19.4
    k8s-app: aws-node
  name: aws-node
rules:
- apiGroups:
  - crd.k8s.amazonaws.com
  resources:
  - eniconfigs
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
```

```
  - list
  - watch
  - get
- apiGroups:
  - ""
  - events.k8s.io
  resources:
  - events
  verbs:
  - create
  - patch
  - list
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/status
  verbs:
  - get
- apiGroups:
  - vpcresources.k8s.aws
  resources:
  - cninodes
  verbs:
  - get
  - list
  - watch
  - patch
```

Além disso, um novo controlador é executado no ambiente de gerenciamento de cada cluster do EKS. O controlador usa as permissões do `ClusterRole` denominado `eks:network-policy-controller`. O seguinte mostra o `eks:network-policy-controller` de um cluster:

```
kubectl get clusterrole eks:network-policy-controller -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/name: amazon-network-policy-controller-k8s
  name: eks:network-policy-controller
rules:
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - create
  - delete
  - get
  - list
  - patch
  - update
  - watch
```

```
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/finalizers
  verbs:
  - update
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/status
  verbs:
  - get
  - patch
  - update
- apiGroups:
  - networking.k8s.io
  resources:
  - networkpolicies
  verbs:
  - get
  - list
  - patch
  - update
  - watch
```

Logs da política de rede

Cada decisão tomada pela CNI da VPC com relação a se as conexões são permitidas ou negadas pelas políticas de uma rede está registrada nos logs de fluxo. Os logs da política de rede de cada nó incluem os logs de fluxo para todo pod que tem uma política de rede. Os logs da política de rede são armazenados em `/var/log/aws-routed-eni/network-policy-agent.log`. O seguinte exemplo é de um arquivo de `network-policy-agent.log`:

```
{"level":"info","timestamp":"2023-05-30T16:05:32.573Z","logger":"ebpf-
client","msg":"Flow Info: ","Src
IP":"192.168.87.155","Src Port":38971,"Dest IP":"64.6.160","Dest
Port":53,"Proto":"UDP","Verdict":"ACCEPT"}
```

Os logs de política de rede estão desabilitados por padrão. Para habilitar os logs de política de rede, siga estas etapas:

> **Note**
>
> Os logs de políticas de rede exigem uma vCPU adicional para o contêiner `aws-network-policy-agent` no manifesto `DaemonSet` `aws-node` da CNI da VPC.

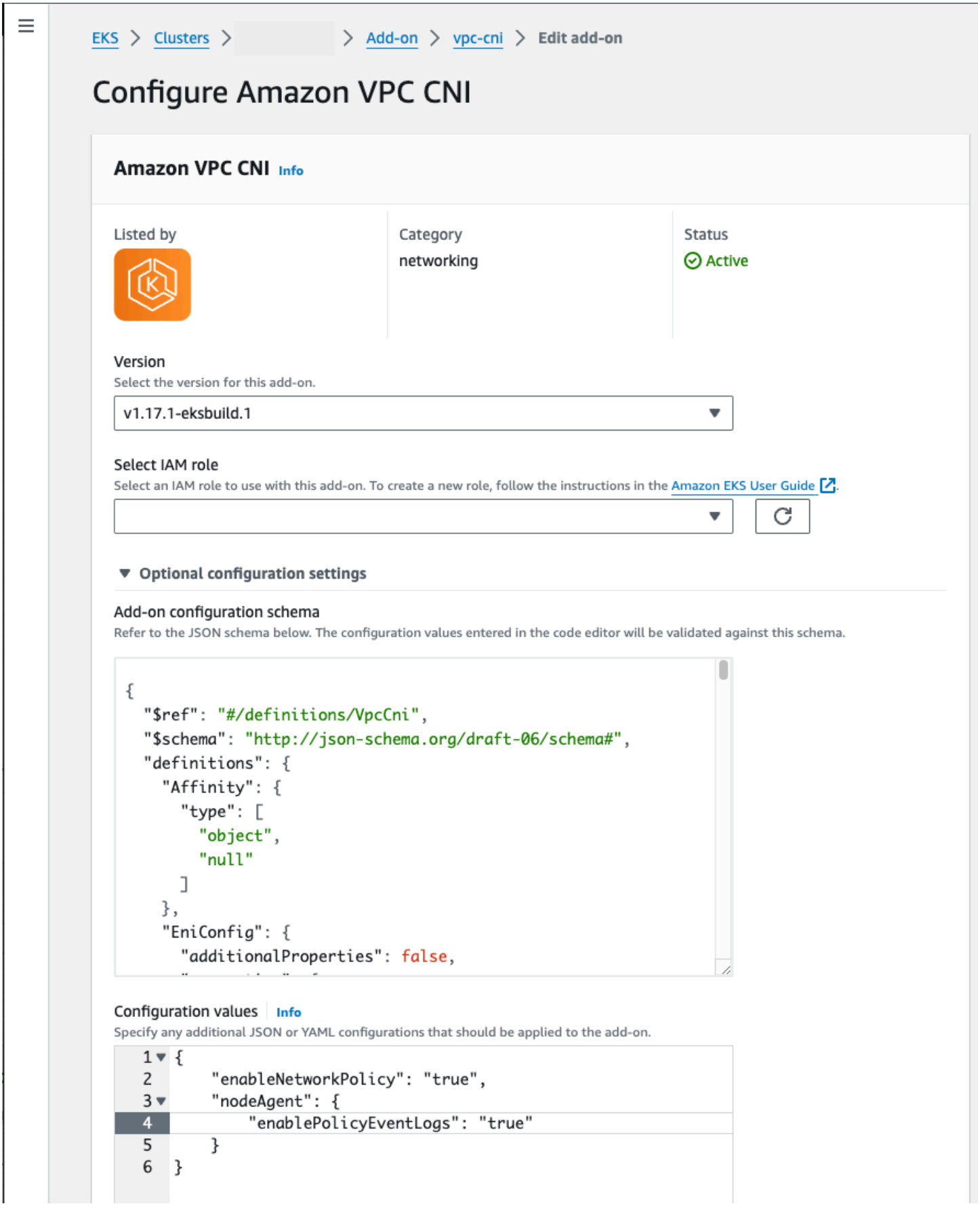
Complemento do Amazon EKS

Console de gerenciamento da AWS

a. Abra o console do Amazon EKS.
b. No painel de navegação à esquerda, selecione Clusters e o nome do cluster para o qual você deseja configurar o complemento Amazon VPC CNI.
c. Escolha a guia Add-ons (Complementos).
d. Selecione a caixa no canto superior direito da caixa do complemento e depois escolha Edit (Editar).
e. Na página Configurar ***CNI da Amazon VPC***:
   i. Selecione a versão `v1.14.0-eksbuild.3` ou posterior na lista suspensa Versão.
   ii. Expanda Definições de configuração opcionais.
   iii. Insira a chave JSON de mais alto nível `"nodeAgent":` e o valor é um objeto com uma chave `"enablePolicyEventLogs":` e valor de `"true"` em Valores da configuração. O texto resultante deve ser um objeto JSON válido. O exemplo apresentado a seguir mostra que a política de rede e os logs de política de rede estão habilitados e que os logs de política de rede são enviados para o CloudWatch Logs:

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true"
    }
}
```

A captura de tela a seguir mostra um exemplo desse cenário.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true"
5 }
6 }

AWS CLI

a. Execute o seguinte comando da AWS CLI. Substitua `my-cluster` pelo nome do cluster e o ARN do perfil do IAM pelo perfil que você está usando.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true"}}'
```

Complemento autogerenciado

Helm

Caso tenha instalado o plug-in CNI da Amazon VPC para Kubernetes por meio do `helm`, você pode atualizar a configuração para gravar os logs da política de rede.

a. Execute o comando a seguir para habilitar a política de rede.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

Caso tenha instalado o plug-in CNI da Amazon VPC para Kubernetes por meio do `kubectl`, você pode atualizar a configuração para gravar os logs da política de rede.

a. Abra o `aws-node` `DaemonSet` no editor.

```
kubectl edit daemonset -n kube-system aws-node
```

b. Substitua `false` por `true` no argumento de comando `--enable-policy-event-logs=false` em `args:`, no contêiner `aws-network-policy-agent` no manifesto `DaemonSet` `aws-node` da CNI da VPC.

```
    - args:
       - --enable-policy-event-logs=true
```

Enviar logs de política de rede para o Amazon CloudWatch Logs

Você pode monitorar os logs da política de rede usando serviços como o Amazon CloudWatch Logs. Você pode usar os métodos a seguir para enviar os logs da política de rede para o CloudWatch Logs.

Nos clusters do EKS, os logs da política podem ser encontrados em `/aws/eks/`*`cluster-name`*`/cluster/`, enquanto, para clusters autogerenciados do K8S, os logs estarão em `/aws/k8s-cluster/cluster/`.

Enviar logs da política de rede com o plug-in CNI da Amazon VPC para Kubernetes

Se você habilitar uma política de rede, um segundo contêiner será adicionado aos pods do `aws-node` para um agente do nó. Esse agente do nó pode enviar os logs da política de rede para o CloudWatch Logs.

> **Note**
>
> Somente os logs da política de rede são enviados pelo agente do nó. Outros logs feitos pelo VPC CNI não são incluídos.

Pré-requisitos

- Adicione as permissões a seguir como uma seção ou uma política separada ao perfil do IAM que você está usando para o VPC CNI.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

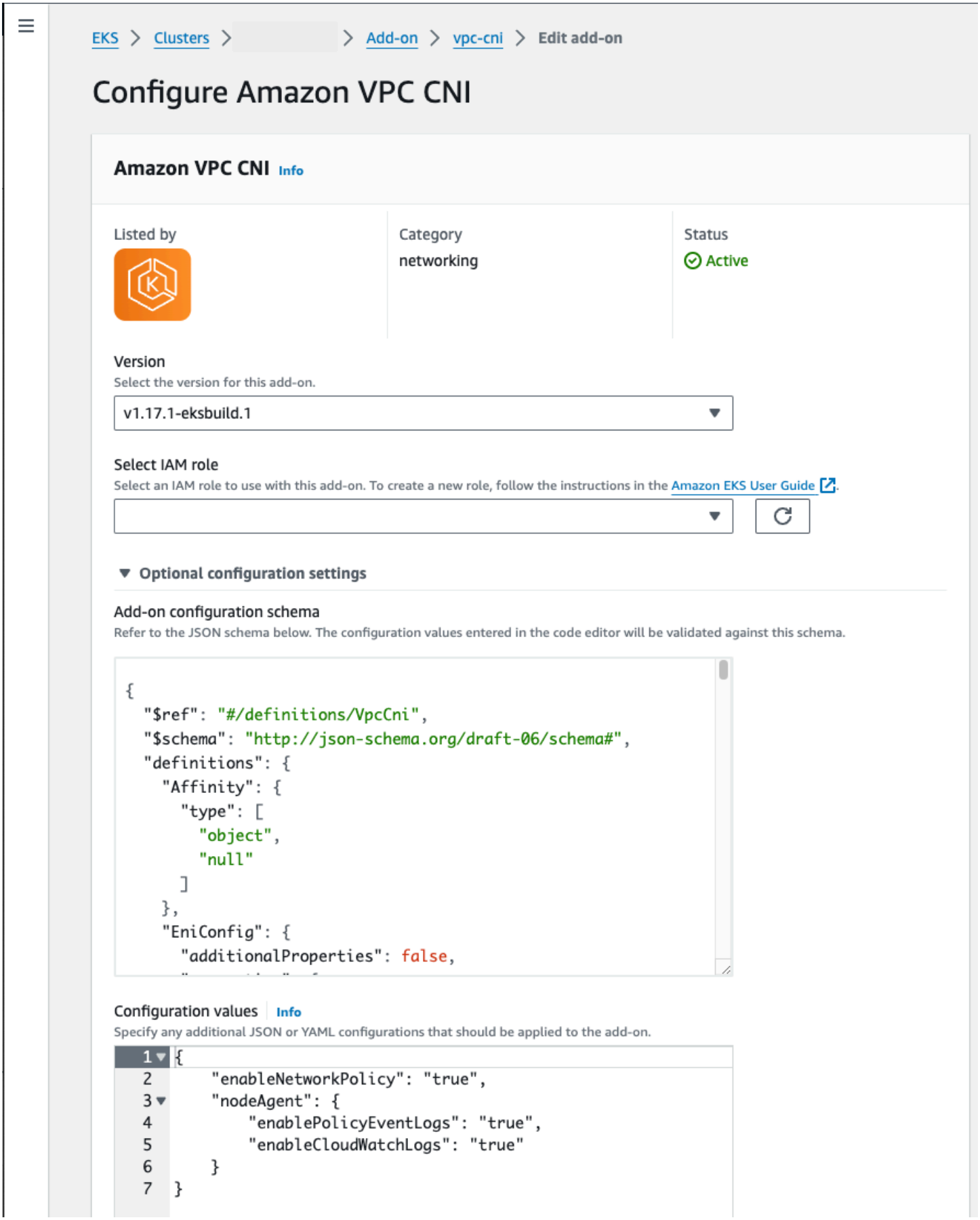
Complemento do Amazon EKS

Console de gerenciamento da AWS

a. Abra o console do Amazon EKS.
b. No painel de navegação à esquerda, selecione Clusters e o nome do cluster para o qual você deseja configurar o complemento Amazon VPC CNI.
c. Escolha a guia Add-ons (Complementos).
d. Selecione a caixa no canto superior direito da caixa do complemento e depois escolha Edit (Editar).
e. Na página Configurar ***CNI da Amazon VPC***:
   i. Selecione a versão `v1.14.0-eksbuild.3` ou posterior na lista suspensa Versão.
   ii. Expanda Definições de configuração opcionais.
   iii. Insira a chave JSON de mais alto nível `"nodeAgent":` e o valor é um objeto com uma chave `"enableCloudWatchLogs":` e valor de `"true"` em Valores da configuração. O texto resultante deve ser um objeto JSON válido. O exemplo apresentado a seguir mostra que a política de rede e os logs de política de rede estão habilitados e que os logs são enviados para o CloudWatch Logs:

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
    }
}
```

A captura de tela a seguir mostra um exemplo desse cenário.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true",
5 "enableCloudWatchLogs": "true"
6 }
7 }

AWS CLI

a. Execute o seguinte comando da AWS CLI. Substitua `my-cluster` pelo nome do cluster e o ARN do perfil do IAM pelo perfil que você está usando.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true", "enableCloudWatchLogs": "true"}}'
```

Complemento autogerenciado

Helm

Caso tenha instalado o plug-in CNI da Amazon VPC para Kubernetes por meio do `helm`, você pode atualizar a configuração para enviar os logs da política de rede para o CloudWatch Logs.

a. Execute o comando apresentado a seguir para habilitar logs de política de rede e enviá-los ao CloudWatch Logs.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set
 nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/aws-
vpc-cni
```

kubectl

a. Abra o `aws-node DaemonSet` no editor.

```
kubectl edit daemonset -n kube-system aws-node
```

b. Substitua `false` por `true` nos dois argumentos de comando `--enable-policy-event-logs=false` e `--enable-cloudwatch-logs=false` em `args:`, no contêiner `aws-network-policy-agent` no manifesto `DaemonSet aws-node` da CNI da VPC.

```
    - args:
       - --enable-policy-event-logs=true
       - --enable-cloudwatch-logs=true
```

Enviar logs de políticas de rede com um **`DaemonSet`** do Fluent Bit

Caso esteja usando o Fluent Bit em um `DaemonSet` para enviar os logs dos nós, você poderá adicionar configurações para incluir os logs das políticas de rede de políticas de rede. Você pode usar o seguinte exemplo de configuração:

```
    [INPUT]
        Name              tail
        Tag               eksnp.*
        Path              /var/log/aws-routed-eni/network-policy-agent*.log
        Parser            json
        DB                /var/log/aws-routed-eni/flb_npagent.db
        Mem_Buf_Limit     5MB
        Skip_Long_Lines   On
        Refresh_Interval  10
```

SDK do eBPF incluído

O plug-in CNI da Amazon VPC para Kubernetes instala uma coleção de ferramentas do SDK do eBPF nos nós. Você pode usar as ferramentas do SDK do eBPF para identificar problemas com as políticas de rede. Por exemplo, o comando a seguir lista os programas que estão sendo executados no nó.

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

Para executar esse comando, você pode usar qualquer método de conexão com o nó.

Problemas e soluções conhecidos

As seções a seguir descrevem problemas conhecidos com o recurso de políticas de rede da CNI da Amazon VPC e suas soluções.

Logs de políticas de rede gerados apesar de enable-policy-event-logs estar definido como false

Problema: a CNI da VPC do EKS está gerando logs de políticas de rede mesmo quando a configuração de `enable-policy-event-logs` está definida como `false`.

Solução: a configuração de `enable-policy-event-logs` desabilita apenas os logs de “decisão” da política, mas não desabilita todos os logs do agente da Política de Rede. Esse comportamento está documentado no README do aws-network-policy-agent no GitHub. Para desabilitar completamente o registro em log, você poderá precisar ajustar outras configurações de registro em log.

Problemas de limpeza do mapa de políticas de rede

Problema: problemas com a rede `policyendpoint` ainda existindo e não sendo limpa após a exclusão dos pods.

Solução: esse problema foi causado por um problema com a versão 1.19.3-eksbuild.1 do complemento da CNI da VPC. Atualize para uma versão mais recente do complemento da CNI da VPC para resolver esse problema.

As políticas de rede não são aplicadas

Problema: o recurso de política de rede está habilitado no plug-in da CNI da Amazon VPC, mas as políticas de rede não estão sendo aplicadas corretamente.

Se você criar uma política de rede `kind: NetworkPolicy` e ela não afetar o pod, verifique se o objeto policyendpoint foi criado no mesmo namespace do pod. Se não houver objetos `policyendpoint` nos namespaces, o controlador de políticas de rede (parte do cluster do EKS) não conseguiu criar regras de política de rede para o agente de política de rede (parte da CNI da VPC) a serem aplicadas.

Solução: a solução é corrigir as permissões da CNI da VPC (`ClusterRole` : `aws-node`) e do controlador de políticas de rede (`ClusterRole` : `eks:network-policy-controller`) e permitir essas ações em qualquer ferramenta de aplicação de políticas, como o Kyverno. Certifique-se de que as políticas do Kyverno não estejam bloqueando a criação de objetos `policyendpoint`. Consulte a seção anterior para obter as permissões necessárias em the section called “Novo CRD `policyendpoints` e permissões”.

Os pods não retornam ao estado de negação padrão após a exclusão da política no modo estrito

Problema: quando as políticas de rede são habilitadas no modo estrito, os pods começam com uma política de negação padrão. Depois que as políticas são aplicadas, o tráfego é permitido para os endpoints especificados. No entanto, quando as políticas são excluídas, o pod não retorna ao estado de negação padrão e, em vez disso, vai para um estado de permissão padrão.

Solução: esse problema foi corrigido na versão 1.19.3 da CNI da VPC, que incluía a versão 1.2.0 do agente de política de rede. Após a correção, com o modo estrito habilitado, depois que as políticas forem removidas, o pod voltará ao estado de negação padrão conforme o esperado.

Grupos de segurança para latência de inicialização de pods

Problema: ao usar o recurso de grupos de segurança para pods no EKS, há um aumento na latência de inicialização de pods.

Solução: a latência se deve à limitação de taxa no controlador de recursos do controle de utilização de APIs na API `CreateNetworkInterface`, que o controlador de recursos da VPC usa para criar ENIs de ramificação para os pods. Verifique os limites de APIs da sua conta para essa operação e considere solicitar um aumento de limite, se necessário.

FailedScheduling devido à insuficiência de vpc.amazonaws.com/pod-eni

Problema: os pods não são programados com o erro: `FailedScheduling 2m53s (x28 over 137m) default-scheduler 0/5 nodes are available: 5 Insufficient vpc.amazonaws.com/pod-eni. preemption: 0/5 nodes are available: 5 No preemption victims found for incoming pod.`

Solução: assim como no problema anterior, atribuir grupos de segurança a pods aumenta a latência de programação de pods e pode ultrapassar o limite de tempo da CNI para adicionar cada ENI, causando falhas na inicialização dos pods. Esse é o comportamento esperado ao usar grupos de segurança para pods. Considere as implicações de programação ao projetar sua arquitetura de workloads.

Problemas de conectividade do IPAM e falhas de segmentação

Problema: ocorrem vários erros, incluindo problemas de conectividade do IPAM, solicitações de controle de utilização e falhas de segmentação:

- `Checking for IPAM connectivity …`
- `Throttling request took 1.047064274s`
- `Retrying waiting for IPAM-D`
- `panic: runtime error: invalid memory address or nil pointer dereference`

Solução: esse problema ocorrerá se você instalar `systemd-udev` no AL2023, pois o arquivo será reescrito com uma política de violação. Isso pode acontecer ao atualizar para outro `releasever` que tenha um pacote atualizado ou ao atualizar manualmente o próprio pacote. Evite instalar ou atualizar `systemd-udev` nos nós do AL2023.

Erro ao buscar dispositivo pelo nome

Problema: mensagem de erro: `{"level":"error","ts":"2025-02-05T20:27:18.669Z","caller":"ebpf/bpf_client.go:578","msg":"failed to find device by name eni9ea69618bf0: %!w(netlink.LinkNotFoundError={0xc000115310})"}`

Solução: esse problema foi identificado e corrigido nas versões mais recentes do agente de política de rede da CNI da Amazon VPC (v1.2.0). Atualize para a versão mais recente da CNI da VPC para resolver esse problema.

Vulnerabilidades CVE na imagem Multus da CNI

Problema: o relatório CVE do EKS ImageScan aprimorado identifica vulnerabilidades na versão v4.1.4-eksbuild.2_thick da imagem Multus da CNI.

Solução: atualize para a nova versão da imagem Multus da CNI e da nova imagem do controlador de políticas de rede, que não têm vulnerabilidades. O scanner pode ser atualizado para solucionar as vulnerabilidades encontradas na versão anterior.

Veredictos de Flow Info DENY em logs

Problema: os logs das políticas de rede mostram os veredictos de DENY: `{"level":"info","ts":"2024-11-25T13:34:24.808Z","logger":"ebpf-client","caller":"events/events.go:193","msg":"Flow Info: ","Src IP":"","Src Port":9096,"Dest IP":"","Dest Port":56830,"Proto":"TCP","Verdict":"DENY"}`

Solução: esse problema foi resolvido na nova versão do controlador de políticas de rede. Atualize para a versão mais recente da plataforma do EKS para resolver problemas de registro em log.

Problemas de comunicação entre pods após a migração do Calico

Problema: depois de atualizar um cluster do EKS para a versão 1.30 e mudar do Calico para a CNI da Amazon VPC para fins de política de rede, a comunicação entre pods falha quando as políticas de rede são aplicadas. A comunicação é restaurada quando as políticas de rede são excluídas.

Solução: o agente de política de rede na CNI da VPC não pode ter tantas portas especificadas quanto o Calico. Em vez disso, use intervalos de portas nas políticas de rede. O número máximo de combinações exclusivas de portas para cada protocolo em cada seletor `ingress:` ou `egress:` em uma política de rede é 24. Use intervalos de portas para reduzir o número de portas exclusivas e evitar essa limitação.

Agente da política de rede não compatível com pods autônomos

Problema: as políticas de rede aplicadas a pods autônomos podem ter um comportamento inconsistente.

Solução: atualmente, o agente da política de rede oferece suporte apenas a pods implantados como parte de uma implantação/replicaset. Se as políticas de rede forem aplicadas a pods autônomos, poderá haver algumas inconsistências no comportamento. Isso está documentado na parte superior desta página, em the section called “Considerações”, e em aws-network-policy-agent GitHub issue #327 no GitHub. Implante pods como parte de uma implantação ou replicaset para um comportamento consistente da política de rede.

Demonstração Stars da política de redes do Amazon EKS

A demonstração cria um serviço de front-end, de back-end e de cliente no cluster do Amazon EKS. A demonstração também cria um interface gráfica do usuário de gerenciamento que mostra os caminhos de entrada e saída disponíveis entre cada serviço. Recomendamos que você conclua a demonstração em um cluster no qual você não executa workloads de produção.

Antes de você criar políticas de rede, todos os serviços podem se comunicar bidirecionalmente. Depois de aplicar as políticas de rede, você poderá ver que o cliente só pode se comunicar com o serviço de front-end, e o back-end só aceita tráfego do front-end.

1. Aplique os serviços de front-end, de back-end, de cliente e de interface do usuário de gerenciamento:

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/backend.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
```

2. Veja todos os pods no cluster.

```
kubectl get pods -A
```

Veja um exemplo de saída abaixo.

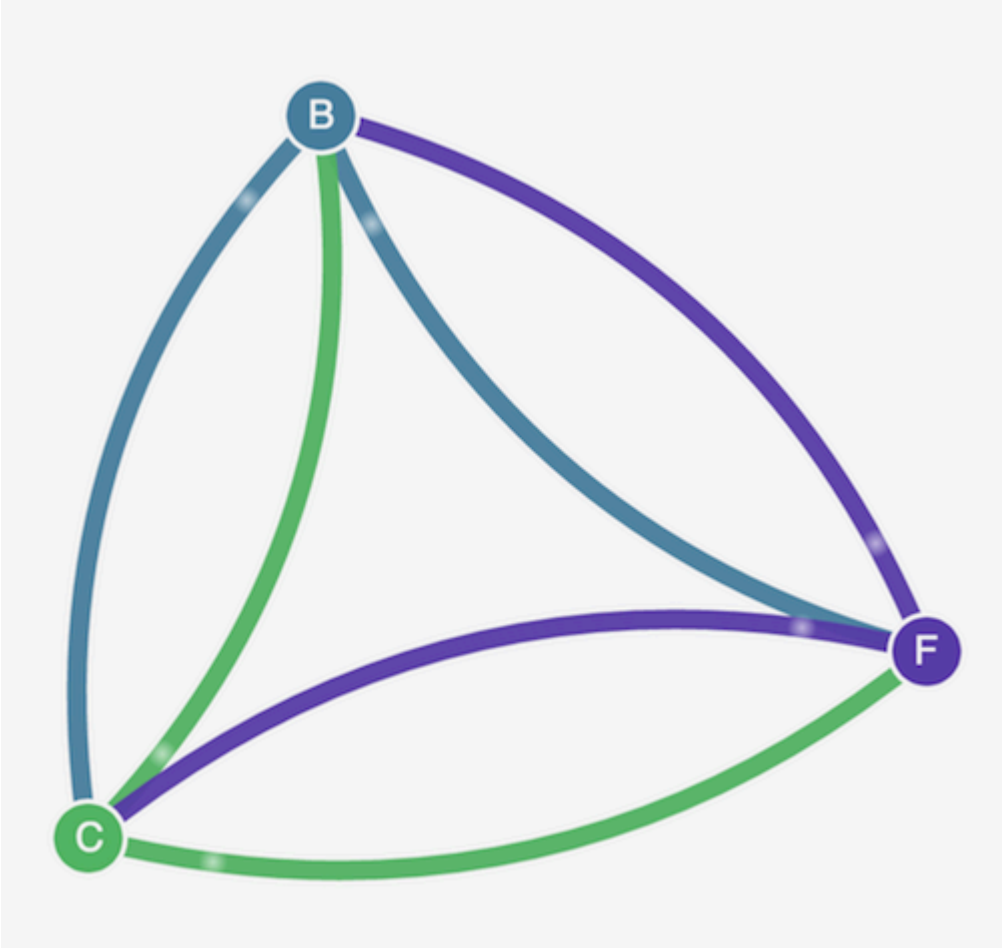
Na saída, você deve ver pods nos namespaces mostrados na saída a seguir. Os *NOMES* dos pods e o número de pods na coluna READY são diferentes dos que aparecem na saída a seguir. Não continue até ver pods com nomes semelhantes e que todos eles tenham Running na coluna STATUS.

```
NAMESPACE         NAME                                  READY   STATUS
 RESTARTS   AGE
[...]
client            client-xlffc                          1/1     Running   0
     5m19s
[...]
management-ui     management-ui-qrb2g                   1/1     Running   0
     5m24s
stars             backend-sz87q                         1/1     Running   0
     5m23s
stars             frontend-cscnf                        1/1     Running   0
     5m21s
[...]
```

3. Para se conectar à interface de usuário de gerenciamento, conecte-se ao EXTERNAL-IP do serviço em execução no cluster:

```
kubectl get service/management-ui -n management-ui
```

4. Abra o navegador no local da etapa anterior. Você deve ver a interface do usuário de gerenciamento a seguir. O nó C é o serviço de cliente, o nó F é o serviço de front-end e o nó B é o serviço de back-end. Cada nó possui acesso total à comunicação com todos os outros nós, conforme indicado pelas linhas coloridas em negrito.



5. Aplique a seguintes política de rede a ambos os namespaces `stars` e `client` para isolar os serviços um do outro:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

Você pode usar os comandos a seguir para aplicar a política a ambos os namespaces:

```
kubectl apply -n stars -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

```
kubectl apply -n client -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

6. Atualize o navegador. Você verá que a interface do usuário de gerenciamento não pode mais acessar nenhum dos nós, portanto, eles não são mostrados na interface do usuário.
7. Aplique as políticas de rede a seguir para permitir que a interface de usuário de gerenciamento acesse os serviços. Aplique essa política para permitir que a interface do usuário:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

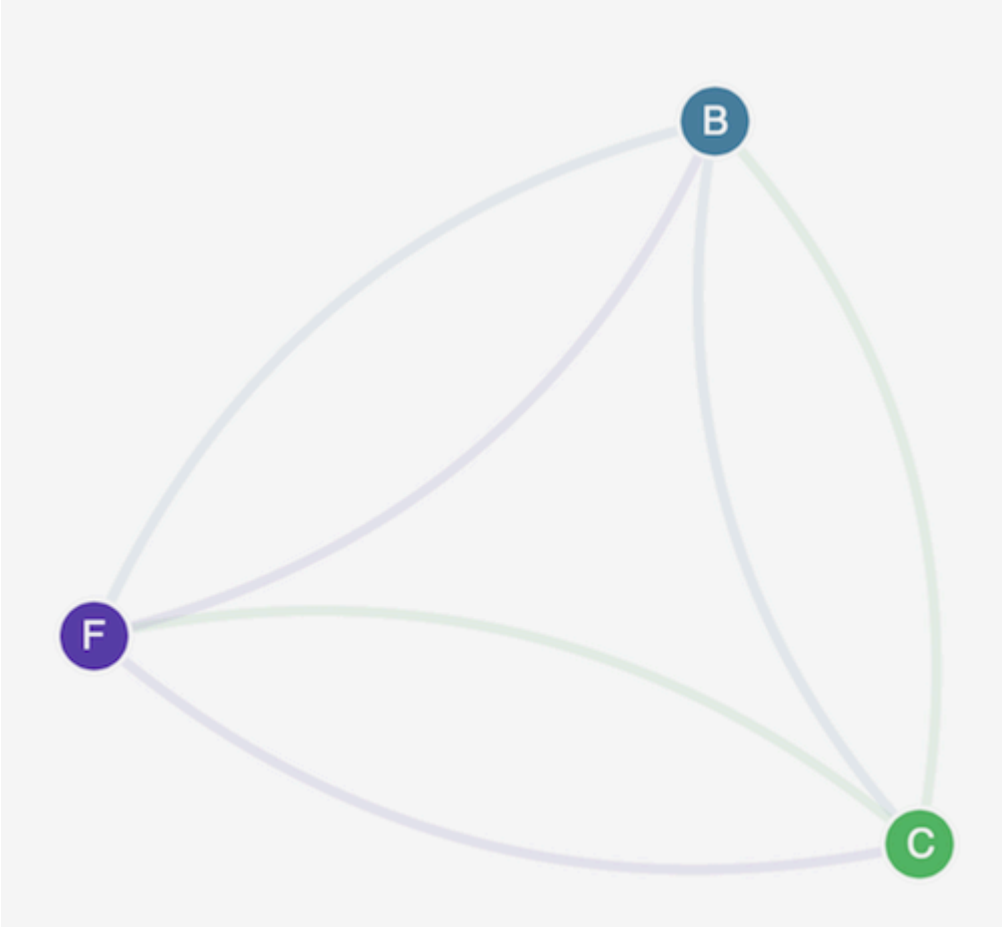
Aplique essa política para permitir que o cliente:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

Você pode usar os seguintes comandos para aplicar ambas as políticas:

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui-client.yaml
```

8. Atualize o navegador. Você verá que a interface do usuário de gerenciamento pode acessar os nós novamente, mas os nós não podem se comunicar uns com os outros.

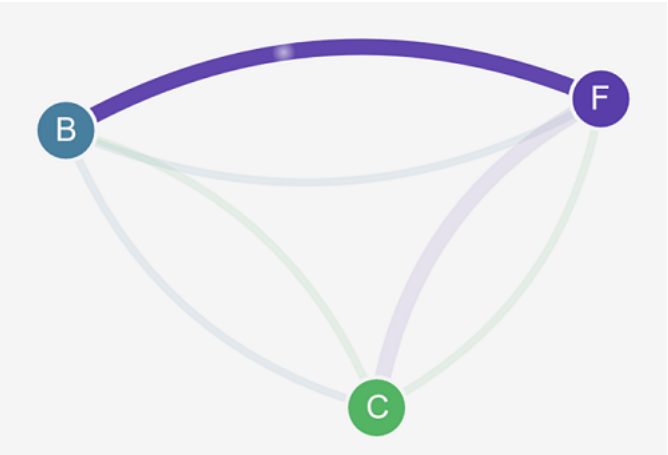


9. Aplique a seguinte política de rede para permitir o tráfego do serviço de front-end para o serviço de back-end:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: backend-policy
spec:
```

```
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
```

10 Atualize o navegador. Você vê que o front-end pode se comunicar com o back-end.

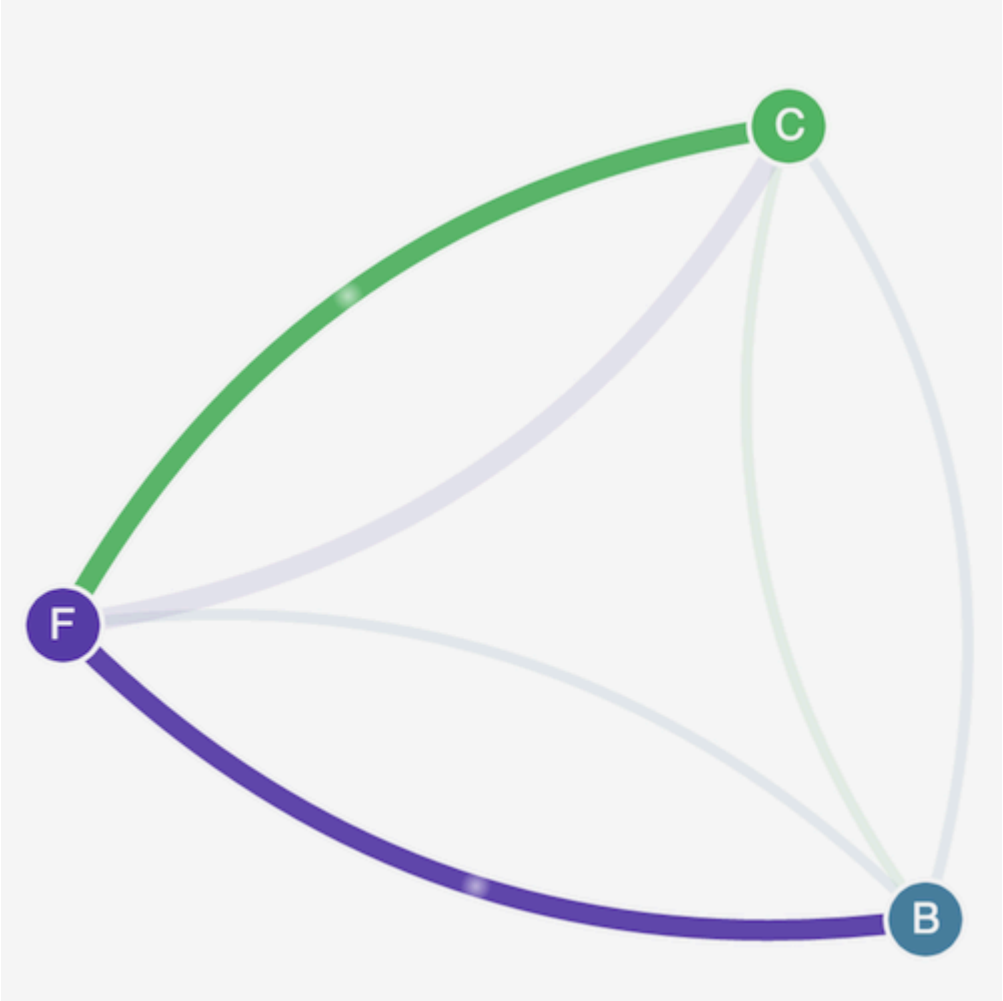


11 Aplique a política de rede a seguir para permitir tráfego do cliente para o serviço de front-end:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: client
      ports:
        - protocol: TCP
```

```
        port: 80
```

12. Atualize o navegador. Você vê que o cliente pode se comunicar com o serviço de front-end. O serviço de front-end ainda pode se comunicar com o serviço de back-end.



13. (Opcional) Ao concluir a demonstração, você poderá excluir os recursos dela.

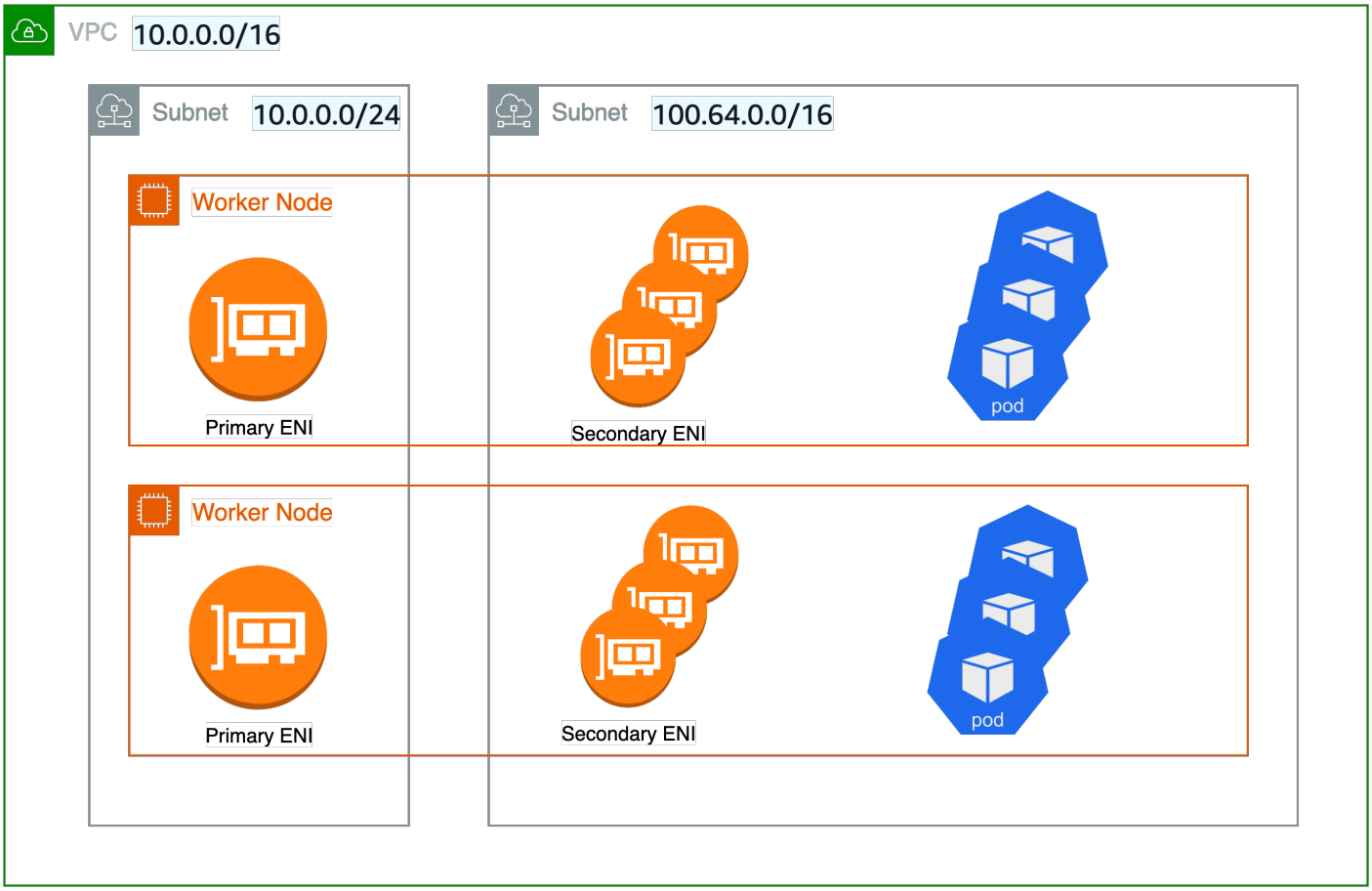
```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/backend.yaml
```

```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
```

Mesmo após a exclusão dos recursos, ainda pode haver endpoints de política de rede nos nós que podem interferir de maneiras inesperados nas conexões de rede do cluster. A única maneira garantida de remover essas regras é recarregar os nós ou encerrar todos os nós e reciclá-los. Para encerrar todos os nós, defina a contagem desejada do grupo de Auto Scaling como 0 e faça backup para o número desejado ou simplesmente encerre os nós.

Implementar pods em sub-redes alternativas com rede personalizada

Aplica-se a: nós Linux `IPv4` do Fargate, nós Linux com instâncias do Amazon EC2



Por padrão, quando o plug-in CNI da Amazon VPC para Kubernetes cria interfaces de rede elásticas secundárias (interfaces de rede) para o nó do Amazon EC2, ele as cria na mesma sub-rede que a interface de rede primária do nó. Ele também associa os mesmos grupos de segurança à interface de rede secundária que estão associados à interface de rede primária. Por uma ou mais das razões a seguir, você pode querer que o plugin crie interfaces de rede secundárias em uma sub-rede diferente ou associar grupos de segurança diferentes às interfaces de rede secundárias, ou ambos:

- Existe um número limitado de endereços `IPv4` disponíveis na sub-rede na qual a interface de rede primária se encontra. Isso pode limitar o número de pods que podem ser criados na sub-rede. Usando uma sub-rede diferente para interfaces de rede secundárias, é possível aumentar o número de endereços `IPv4` disponíveis para pods.
- Por motivos de segurança, os pods talvez precisem usar diferentes grupos de segurança ou sub-redes do que os da interface de rede primária do nó.
- Os nós são configurados em sub-redes públicas, e você deseja colocar os pods em sub-redes privadas. A tabela de rotas que está associada a uma sub-rede pública inclui uma rota para um gateway da Internet. A tabela de rotas que está associada a uma sub-rede privada não inclui uma rota para um gateway da Internet.

> **Tip**
>
> Você também pode adicionar uma sub-rede nova ou existente diretamente ao cluster do Amazon EKS, sem usar redes personalizadas. Para obter mais informações, consulte the section called “Adicionar uma sub-rede da VPC existente a um cluster do Amazon EKS no console de gerenciamento”.

Considerações

A seguir estão algumas considerações sobre o uso do recurso.

- Com redes personalizadas habilitadas, nenhum endereço IP atribuído à interface de rede primária é atribuído a pods. Apenas endereços IP de interfaces de rede secundária são atribuídos a pods.
- Se o seu cluster utilizar a família `IPv6`, não será possível utilizar redes personalizadas.
- Se você planeja utilizar redes personalizadas somente para aliviar a exaustão de endereços `IPv4`, pode criar um cluster utilizando a família `IPv6`. Para obter mais informações, consulte the section called “IPv6”.

- Mesmo que os pods implantados em sub-redes especificadas para interfaces de rede secundária possam usar sub-redes e grupos de segurança diferentes da interface de rede primária do nó, as sub-redes e os grupos de segurança devem estar na mesma VPC do nó.
- No Fargate, as sub-redes são controladas por meio do perfil do Fargate. Para obter mais informações, consulte the section called “Definir perfis”.

Personalização da interface de rede secundária nos nós do Amazon EKS

Antes de começar o tutorial, conclua as seguintes etapas:

- Revisar as considerações
- Familiaridade com como o plug-in CNI da Amazon VPC para Kubernetes cria interfaces de rede secundárias e atribui endereços IP a pods. Para saber mais, consulte Alocação de ENIs no GitHub
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Convém concluir as etapas neste tópico em um shell Bash. Se não estiver utilizando um shell Bash, alguns comandos de script, como caracteres de continuação de linha e a forma como as variáveis são definidas e utilizadas, exigirão o ajuste do seu shell. Além disso, as regras de citação e de escape do seu shell podem ser diferentes. Para obter mais informações, consulte Uso de aspas com cadeias de caracteres na AWS CLI, no Guia do usuário da AWS Command Line Interface.

Para este tutorial, convém utilizar os valores de exemplo, exceto onde indicado para substituí-los. É possível substituir qualquer valor de exemplo ao concluir as etapas para um cluster do produção. Convém que todas as etapas sejam concluídas no mesmo terminal. Isso porque variáveis são definidas e utilizadas ao longo das etapas e não existirão em terminais diferentes.

Os comandos deste tópico estão formatados de acordo com as convenções listadas em Exemplos de uso da AWS CLI. Se estiver executando comandos da linha de comandos em recursos que estão em uma região da AWS diferente da região da AWS padrão definida no perfil da AWS CLI que você está usando, será necessário adicionar `--region us-west-2` aos comandos, substituindo `us-west-2` pela sua região da AWS.

Quando quiser implantar redes personalizadas no seu cluster do produção, pule para the section called “Etapa 2: Configurar sua VPC”.

Etapa 1: Criar uma VPC de teste e um cluster

Os procedimentos a seguir ajudam a criar uma VPC de teste e um cluster e a configurar redes personalizadas para esse cluster. Não recomendamos utilizar o cluster do teste para workloads de produção, pois vários recursos não relacionados que você pode utilizar no seu cluster do produção não são abordados neste tópico. Para obter mais informações, consulte the section called “Criar um cluster”.

1. Execute o comando a seguir para definir a variável `account_id`.

```
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. Crie uma VPC.

   a. Se estiver implementando em um sistema de teste, crie um VPC usando um modelo do Amazon EKS AWS CloudFormation.

```
aws cloudformation create-stack --stack-name my-eks-custom-networking-vpc \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --parameters ParameterKey=VpcBlock,ParameterValue=192.168.0.0/24 \
  ParameterKey=PrivateSubnet01Block,ParameterValue=192.168.0.64/27 \
  ParameterKey=PrivateSubnet02Block,ParameterValue=192.168.0.96/27 \
  ParameterKey=PublicSubnet01Block,ParameterValue=192.168.0.0/27 \
  ParameterKey=PublicSubnet02Block,ParameterValue=192.168.0.32/27
```

   b. A pilha do AWS CloudFormation leva alguns minutos para ser criada. Para verificar o status de implantação da pilha, execute o comando a seguir.

```
aws cloudformation describe-stacks --stack-name my-eks-custom-networking-vpc --
query Stacks\[\].StackStatus  --output text
```

Não prossiga para a próxima etapa até que a saída do comando seja `CREATE_COMPLETE`.

c. Defina variáveis com os valores dos IDs das sub-redes privadas criadas pelo modelo.

```
subnet_id_1=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet02'].PhysicalResourceId" --output text)
```

d. Defina variáveis com as zonas de disponibilidade das sub-redes que foram recuperadas na etapa anterior.

```
az_1=$(aws ec2 describe-subnets --subnet-ids $subnet_id_1 --query
 'Subnets[*].AvailabilityZone' --output text)
az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --query
 'Subnets[*].AvailabilityZone' --output text)
```

3. Crie um perfil do IAM de cluster.

a. Execute o comando a seguir para criar um arquivo JSON de política de confiança do IAM:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Crie o perfil do IAM do cluster do Amazon EKS. Se necessário, prefixe `eks-cluster-role-trust-policy.json` com o caminho no computador no qual você gravou o arquivo na etapa anterior. O comando associa a política de confiança criada na etapa anterior à função.

Para criar um perfil do IAM, a entidade principal do IAM que estiver criando o perfil deverá ser atribuída à seguinte ação `iam:CreateRole` (permissão):

```
aws iam create-role --role-name myCustomNetworkingAmazonEKSClusterRole --assume-
role-policy-document file://"eks-cluster-role-trust-policy.json"
```

c. Anexe a política gerenciada do Amazon EKS chamada AmazonEKSClusterPolicy à perfil. Para anexar uma política do IAM a uma entidade principal do IAM a entidade principal do IAM que está anexando a política deve receber uma das seguintes ações do IAM (permissões): `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name myCustomNetworkingAmazonEKSClusterRole
```

4. Crie um cluster do Amazon EKS e configure seu dispositivo para se comunicar com ele.

a. Crie um cluster.

```
aws eks create-cluster --name my-custom-networking-cluster \
   --role-arn arn:aws:iam::$account_id:role/myCustomNetworkingAmazonEKSClusterRole
 \
   --resources-vpc-config subnetIds="$subnet_id_1","$subnet_id_2"
```

> **Note**
>
> Talvez você receba um erro porque uma das zonas de disponibilidade em sua solicitação não tem capacidade suficiente para criar um cluster do Amazon EKS. Se isso acontecer, o resultado do erro conterá as zonas de disponibilidade que são compatíveis com o novo cluster. Tente criar o cluster com pelo menos duas sub-redes que estejam localizadas nas zonas de disponibilidade compatíveis de sua conta. Para obter mais informações, consulte the section called “Capacidade insuficiente”.

b. A criação do cluster domora alguns minutos. Para verificar o status de implantação do cluster, execute o comando a seguir.

```
aws eks describe-cluster --name my-custom-networking-cluster --query
 cluster.status
```

Não prossiga para a próxima etapa até que a saída do comando seja `"ACTIVE"`.

c. Configure `kubectl` para se comunicar com o cluster.

```
aws eks update-kubeconfig --name my-custom-networking-cluster
```

Etapa 2: Configurar sua VPC

Este tutorial requer a VPC que foi criada em the section called “Etapa 1: Criar uma VPC de teste e um cluster”. Para um cluster do produção, ajuste as etapas de acordo com sua VPC, substituindo todos os valores de exemplo pelos seus próprios.

1. Verifique se o plug-in CNI da Amazon VPC instalado atualmente para Kubernetes é a versão mais recente. Para determinar a versão mais recente do tipo de complemento do Amazon EKS e atualizar sua versão para ela, consulte the section called “Atualização de um complemento”. Para determinar a versão mais recente do tipo de complemento autogerenciado e atualizar sua versão para ela, consulte the section called “CNI da Amazon VPC”.
2. Recupere o ID da VPC do seu cluster e armazene-o em uma variável para uso em etapas posteriores.

```
vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query
 "cluster.resourcesVpcConfig.vpcId" --output text)
```

3. Associe um bloco de Encaminhamento Entre Domínios Sem Classificação (CIDR) adicional à VPC do seu cluster. O bloco CIDR não pode se sobrepor a nenhum bloco CIDR existente associado.

   a. Visualize os blocos CIDR atuais que estão associados à sua VPC.

   ```
   aws ec2 describe-vpcs --vpc-ids $vpc_id \
       --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
    CidrBlockState.State}' --out table
   ```

   Veja um exemplo de saída abaixo.

   ```
   ---------------------------------
   |          DescribeVpcs          |
   +-----------------+--------------+
   |    CIDRBlock    |    State     |
   +-----------------+--------------+
   |  192.168.0.0/24 |  associated  |
   +-----------------+--------------+
   ```

b. Associe um bloco CIDR adicionais à sua VPC. Substitua o valor do bloco CIDR no comando a seguir. Para obter mais informações, consulte Associar blocos CIDR IPv4 adicionais ao seu VPC no Guia do usuário do Amazon VPC.

```
aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 192.168.1.0/24
```

c. Confirme se o novo bloco está associado.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --query
 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

Veja abaixo um exemplo de saída.

```
----------------------------------
|          DescribeVpcs          |
+-----------------+--------------+
|    CIDRBlock    |    State     |
+-----------------+--------------+
|  192.168.0.0/24 |  associated  |
|  192.168.1.0/24 |  associated  |
+-----------------+--------------+
```

Não continue na próxima etapa até que o `State` do novo bloco CIDR seja `associated`.

4. Crie quantas sub-redes quiser utilizar em cada zona de disponibilidade na qual suas sub-redes existentes estejam. Especifique um bloco CIDR que esteja dentro do bloco CIDR que você associou à sua VPC em uma etapa anterior.

a. Crie novas sub-redes. Substitua os valores do bloco CIDR no comando a seguir. As sub-redes devem ser criadas em um bloco CIDR de VPC diferente daquele em que as sub-redes existentes se encontram, mas nas mesmas zonas de disponibilidade dessas sub-redes existentes. Neste exemplo, uma sub-rede é criada no novo bloco CIDR em cada zona de disponibilidade na qual as sub-redes privadas atuais existem. Os IDs das sub-redes criadas são armazenados em variáveis para uso em etapas posteriores. Os valores de `Name` correspondem aos valores atribuídos às sub-redes criadas utilizando o modelo VPC do Amazon EKS em uma etapa anterior. Não são necessários nomes. É possível utilizar nomes diferentes.

```
new_subnet_id_1=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_1
 --cidr-block 192.168.1.0/27 \
```

```
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet01},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
new_subnet_id_2=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_2
 --cidr-block 192.168.1.32/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet02},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
```

> **Important**
>
> Por padrão, suas novas sub-redes estão implicitamente associadas a tabela de rotas principal da sua VPC. Essa tabela de rotas permite a comunicação entre todos os recursos que estão implantados na VPC. Porém, ela não permite comunicação com recursos que têm endereços IP fora dos blocos CIDR associados à sua VPC. É possível associar sua própria tabela de rotas a sub-redes para alterar esse comportamento. Para obter mais informações, consulte Tabelas de rotas de sub-rede, no Guia do usuário da Amazon VPC.

b. Visualize as sub-redes atuais na sua VPC.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" \
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

Veja um exemplo de saída abaixo.

```
-------------------------------------------------------------------
|                          DescribeSubnets                        |
+------------------+--------------------+---------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId          |
+------------------+--------------------+---------------------------+
|  us-west-2d      |  192.168.0.0/27    |     subnet-example1       |
|  us-west-2a      |  192.168.0.32/27   |     subnet-example2       |
|  us-west-2a      |  192.168.0.64/27   |     subnet-example3       |
|  us-west-2d      |  192.168.0.96/27   |     subnet-example4       |
|  us-west-2a      |  192.168.1.0/27    |     subnet-example5       |
|  us-west-2d      |  192.168.1.32/27   |     subnet-example6       |
+------------------+--------------------+---------------------------+
```

É possível ver que as sub-redes no bloco CIDR `192.168.1.0` que você criou estão nas mesmas zonas de disponibilidade que as sub-redes do bloco CIDR `192.168.0.0`.

Etapa 3: Configurar recursos do Kubernetes

1. Defina a variável de ambiente `AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG` como `true` no DaemonSet `aws-node`.

```
kubectl set env daemonset aws-node -n kube-system
 AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

2. Recupere o ID do grupo de segurança do cluster e armazene-o em uma variável para usar na próxima etapa. O Amazon EKS criará automaticamente esse grupo de segurança quando você criar seu cluster.

```
cluster_security_group_id=$(aws eks describe-cluster --name my-custom-networking-
cluster --query cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
```

3. Crie um recurso `ENIConfig` personalizado para cada sub-rede na qual você deseja implantar pods.

   a. Crie um arquivo exclusivo para cada configuração de interface de rede elástica.

   Os comandos a seguir criam arquivos `ENIConfig` separados para as duas sub-redes criadas em uma etapa anterior. O valor de `name` deve ser exclusivo. O nome é o mesmo da zona de disponibilidade na qual a sub-rede está localizada. O grupo de segurança do cluster é atribuído a `ENIConfig`.

```
cat >$az_1.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
EOF
```

```
cat >$az_2.yaml <<EOF
```

```
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_2
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_2
EOF
```

Para um cluster do produção, é possível fazer as seguintes alterações nos comandos anteriores:

- Substitua $cluster_security_group_id pelo ID de um grupo de segurança existente que você deseja usar para cada `ENIConfig`.
- Sempre que possível, recomendamos que defina seu `ENIConfigs` com o mesmo nome da zona de disponibilidade para a qual você usará o `ENIConfig`. Talvez você precise utilizar nomes para os `ENIConfigs` diferentes dos nomes das zonas de disponibilidade por vários motivos. Por exemplo, se você tiver mais de duas sub-redes na mesma zona de disponibilidade e quiser utilizá-las com redes personalizadas, precisará de vários `ENIConfigs` para a mesma zona de disponibilidade. Como cada `ENIConfig` requer um nome exclusivo, não é possível nomear mais do que um dos seus `ENIConfigs` usando o nome da zona de disponibilidade.

  Se os nomes de `ENIConfig` não forem todos iguais aos nomes da zona de disponibilidade, substitua $az_1 e $az_2 por seus próprios nomes nos comandos anteriores e anote seus nós com o ENIConfig mais adiante neste tutorial.

  > **Note**
  >
  > Se um grupo de segurança válido não for especificado para uso com um cluster do produção e você estiver usando:

- a versão `1.8.0` ou mais recente do plug-in CNI da Amazon VPC para Kubernetes, então os grupos de segurança associados à interface de rede elástica primária do nó serão utilizados.
- uma versão do plug-in CNI da Amazon VPC para Kubernetes anterior à versão `1.8.0`, então o grupo de segurança padrão da VPC será atribuído a interfaces de rede secundária.

> **Important**
>
> - `AWS_VPC_K8S_CNI_EXTERNALSNAT=false` é uma configuração padrão na configuração do plugin Amazon VPC CNI plugin para Kubernetes. Se você estiver utilizando essa configuração padrão, o tráfego destinado a endereços IP que estiverem fora de um dos blocos CIDR associados à sua VPC utilizará os grupos de segurança e as sub-redes da interface de rede primária do nó. As sub-redes e os grupos de segurança definidos em `ENIConfigs` que forem utilizados para criar interfaces de rede secundárias não serão utilizados para esse tráfego. Para obter mais informações sobre essa configuração, consulte the section called “Tráfego de saída”.
> - Se você também usar grupos de segurança para pods, o grupo de segurança especificado em uma `SecurityGroupPolicy` será utilizado no lugar do grupo de segurança especificado em `ENIConfigs`. Para obter mais informações, consulte the section called “Grupos de segurança para os pods”.

b. Aplique cada arquivo de recursos personalizados que você criou ao seu cluster com os seguintes comandos:

```
kubectl apply -f $az_1.yaml
kubectl apply -f $az_2.yaml
```

4. Confirme se as `ENIConfigs` foram criadas.

```
kubectl get ENIConfigs
```

Veja abaixo um exemplo de saída.

```
NAME         AGE
us-west-2a   117s
us-west-2d   105s
```

5. Se estiver habilitando redes personalizadas em um cluster do produção e tiver especificado para as suas `ENIConfigs` nomes diferentes dos da zona de disponibilidade para a qual você as estiver utilizando, pule para a próxima etapa para implantar nós do Amazon EC2.

   Habilite o Kubernetes para aplicar automaticamente a `ENIConfig` de uma zona de disponibilidade a todos os novos nós do Amazon EC2 que forem criados no seu cluster.

a. No caso do cluster do teste neste tutorial, pule para a próxima etapa.

   Para um cluster do produção, verifique se uma anotação com a chave `k8s.amazonaws.com/eniConfig` para a variável de ambiente `ENI_CONFIG_ANNOTATION_DEF` existe na especificação do contêiner do DaemonSet `aws-node`.

```
kubectl describe daemonset aws-node -n kube-system | grep
 ENI_CONFIG_ANNOTATION_DEF
```

   Se a saída for retornada, significa que a anotação existe. Se nenhuma saída for retornada, a variável não está definida. Para um cluster do produção, é possível utilizar essa configuração ou a configuração da etapa a seguir. Se você utilizar essa configuração, ela substituirá a configuração da etapa a seguir. Neste tutorial, é utilizada a configuração da próxima etapa.

b. Atualize o DaemonSet `aws-node` para aplicar automaticamente o `ENIConfig` de uma zona de disponibilidade a todos os novos nós do Amazon EC2 criados no cluster.

```
kubectl set env daemonset aws-node -n kube-system
 ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
```

Etapa 4: Implantar nós do Amazon EC2

1. Crie um perfil do IAM para o nó.

   a. Execute o comando a seguir para criar um arquivo JSON de política de confiança do IAM:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   b. Crie o perfil do IAM e armazene o nome do recurso da Amazon (ARN) retornado em uma variável para uso em uma etapa posterior.

```
node_role_arn=$(aws iam create-role --role-name myCustomNetworkingNodeRole --
assume-role-policy-document file://"node-role-trust-relationship.json" \
    --query Role.Arn --output text)
```

c. Anexe três políticas do IAM gerenciadas necessárias ao perfil do IAM.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name myCustomNetworkingNodeRole
```

> **Important**
>
> Para simplificar este tutorial, a política AmazonEKS_CNI_Policy é anexada à perfil IAM do nó. No entanto, em um cluster do produção, recomendamos que você anexe a política a um perfil do IAM separado que seja usado apenas com o plug-in CNI da Amazon VPC para Kubernetes. Para obter mais informações, consulte the section called “Configuração para IRSA”.

2. Crie um dos seguintes tipos de grupos de nós. Para determinar o tipo de instância a ser implantado, consulte the section called “Tipos de instância do Amazon EC2”. Neste tutorial, conclua a opção Gerenciado, Sem um modelo de inicialização ou com um modelo de inicialização sem um ID de AMI especificado. Se for usar o grupo de nós para workloads de produção, recomendamos que você se familiarize com todas as opções de grupo de nós gerenciados e grupo de nós autogerenciados antes de implantar o grupo de nós.
   - Managed (Gerenciado) – Implante seu grupo de nós usando uma das seguintes opções:
     - Sem um modelo de inicialização ou com um modelo de inicialização sem um ID de AMI especificado: execute o comando a seguir. Neste tutorial, use os valores de exemplo. Para um grupo de nós de produção, substitua todos os valores de exemplo pelos seus próprios. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres.

```
aws eks create-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup \
    --subnets $subnet_id_1 $subnet_id_2 --instance-types t3.medium --node-role
 $node_role_arn
```

- Com um modelo de inicialização com um ID de AMI especificado

  A. Determine o número máximo de pods recomendado do Amazon EKS para os nós. Siga as instruções em Pods máximos recomendados do Amazon EKS para cada tipo de instância do Amazon EC2, adicionando `--cni-custom-networking-enabled` à etapa 3 desse tópico. Anote a saída para uso na próxima etapa.

  B. No seu modelo de inicialização, especifique um ID de AMI otimizado do Amazon EKS ou uma AMI personalizada criada a partir da AMI otimizada do Amazon EKS e, em seguida implante o grupo de nós usando um modelo de inicialização e forneça os seguintes dados do usuário no modelo de lançamento: Estes dados do usuário fornecem argumentos para a especificação do `NodeConfig`. Para obter mais informações sobre o NodeConfig, consulte a Referência de API do NodeConfig. Você pode substituir `20` pelo valor da etapa anterior (recomendado) ou por seu próprio valor.

```
---
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"
--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    ...
    kubelet:
      config:
        maxPods: 20
```

  Se você criou uma AMI personalizada que não foi criada a partir da AMI otimizada do Amazon EKS, precisa criar a configuração você mesmo.

- Autogerenciado

i. Determine o número máximo de pods recomendado do Amazon EKS para os nós. Siga as instruções em the section called “O máximo de pods do Amazon EKS recomendado para cada tipo de instância do Amazon EC2”, adicionando `--cni-custom-networking-enabled` à etapa 3 nesse tópico. Anote a saída para uso na próxima etapa.

ii. Implante o grupo de nós usando as instruções em the section called “Amazon Linux”.

> **Note**
>
> Se quiser que os nós em um cluster do produção sejam compatíveis com um número significativamente maior de pods, execute o script em the section called “O máximo de pods do Amazon EKS recomendado para cada tipo de instância do Amazon EC2” novamente. Adicione também a opção `--cni-prefix-delegation-enabled` ao comando. Por exemplo, `110` é retornado para um tipo de instância `m5.large`. Para obter instruções sobre como habilitar esse recurso, consulte the section called “Aumento dos endereços IP”. É possível usar esse recurso com redes personalizadas.

3. A criação do grupo de nós demora vários minutos. Você pode verificar o status da criação de um grupo de nós gerenciados com o seguinte comando:

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name my-nodegroup --query nodegroup.status --output text
```

   Não continue na próxima etapa até que a saída recebida esteja `ACTIVE`.

4. No tutorial, é possível ignorar essa etapa.

   Para um cluster do produção, se você definir para as suas `ENIConfigs` os mesmos nomes que os da zona de disponibilidade para a qual você as estiver utilizando, deverá anotar seus nós com o nome de `ENIConfig` que deve ser utilizado com o nó. Essa etapa não será necessária se você tiver apenas uma sub-rede em cada zona de disponibilidade e tiver definido suas `ENIConfigs` com os mesmos nomes das zonas de disponibilidade. Isso ocorre porque o plug-in CNI da Amazon VPC para Kubernetes associa automaticamente o `ENIConfig` correto ao nó quando você o habilitou a fazer isso em uma etapa anterior.

   a. Obtenha a lista de nós no seu cluster.

```
kubectl get nodes
```

      Veja um exemplo de saída abaixo.

```
NAME                                           STATUS   ROLES    AGE     VERSION
ip-192-168-0-126.us-west-2.compute.internal   Ready    <none>   8m49s   v1.22.9-
eks-810597c
ip-192-168-0-92.us-west-2.compute.internal    Ready    <none>   8m34s   v1.22.9-
eks-810597c
```

b. Especifique em qual zona de disponibilidade cada nó se encontra. Execute o comando a seguir para cada nó retornado na etapa anterior, substituindo os endereços IP com base na saída anterior.

```
aws ec2 describe-instances --filters Name=network-interface.private-dns-
name,Values=ip-192-168-0-126.us-west-2.compute.internal \
--query 'Reservations[].Instances[].{AvailabilityZone: Placement.AvailabilityZone,
 SubnetId: SubnetId}'
```

Veja abaixo um exemplo de saída.

```
[
    {
        "AvailabilityZone": "us-west-2d",
        "SubnetId": "subnet-Example5"
    }
]
```

c. Anote cada nó com a `ENIConfig` que você criou para o ID da sub-rede e a zona de disponibilidade. Apenas é possível anotar um nó com uma `ENIConfig`, embora vários nós possam ser anotados com a mesma `ENIConfig`. Substitua os valores de exemplo pelos seus próprios.

```
kubectl annotate node ip-192-168-0-126.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName2
```

5. Se você tinha nós em um cluster do produção com pods em execução antes de mudar para usar o recurso de redes personalizadas, conclua as seguintes tarefas:

a. Certifique-se de ter nós disponíveis que estão utilizando o recurso de rede personalizado.

b. Isole e drene os nós para desligar os pods adequadamente. Para obter mais informações, consulte Safely Drain a Node (Drenar um nó com segurança) na documentação do Kubernetes.

c. Encerre os nós. Se os nós estiverem em um grupo de nós gerenciados existente, será possível excluir o grupo de nós. Execute o comando a seguir.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup
```

Somente os novos nós que são registrados com o rótulo `k8s.amazonaws.com/eniConfig` usam o novo recurso de redes personalizadas.

6. Verifique se os pods receberam um endereço IP de um bloco CIDR associado a uma das sub-redes criadas em uma etapa anterior.

```
kubectl get pods -A -o wide
```

Veja abaixo um exemplo de saída.

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
           NODE                                            NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-2rkn4             1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   aws-node-k96wp             1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
kube-system   coredns-657694c6f4-smcgr   1/1     Running   0          56m
 192.168.1.23    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   coredns-657694c6f4-stwv9   1/1     Running   0          56m
 192.168.1.28    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-jgshq           1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-wx9vk           1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
```

Você pode ver que os pods do CoreDNS receberam endereços IP do bloco CIDR `192.168.1.0` que você adicionou à VPC. Sem redes personalizadas, eles teriam recebido endereços do bloco CIDR `192.168.0.0`, porque este era o único bloco CIDR originalmente associado à VPC.

Se um `spec` do pod contiver `hostNetwork=true`, será atribuído o endereço IP primário do nó. Ele não receberá um endereço das sub-redes que foram adicionadas. Por padrão, esse valor é definido como `false`. Esse valor é definido como `true` para os pods do `kube-proxy` e do plug-in CNI da Amazon VPC para Kubernetes (`aws-node`) que são executados no cluster. É por esse motivo que os pods do `kube-proxy` e do `aws-node` do plug-in não receberam os endereços 192.168.1.x na saída anterior. Para saber mais sobre a configuração `hostNetwork` de um pod, consulte PodSpec v1 core na referência de APIs do Kubernetes.

Etapa 5: Excluir os recursos do tutorial

Depois de concluir o tutorial, convém excluir os recursos criados. Em seguida, ajuste as etapas para habilitar as redes personalizadas para um cluster do produção.

1. Se o grupo de nós criado foi apenas para testes, exclua-o.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name
 my-nodegroup
```

2. Mesmo após a saída da AWS CLI informar que o cluster foi excluído, o processo de exclusão pode não ter sido concluído. O processo de exclusão demora alguns minutos. Confirme se a exclusão foi feita, executando o seguinte comando:

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

   Não continue até que a saída retornada seja semelhante à seguinte:

```
An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup
 operation: No node group found for name: my-nodegroup.
```

3. Se o grupo de nós criado foi apenas para testes, exclua a perfil do IAM do nó.

   a. Desanexe as políticas da função.

```
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
```

```
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

b. Exclua o perfil.

```
aws iam delete-role --role-name myCustomNetworkingNodeRole
```

4. Excluir o cluster.

```
aws eks delete-cluster --name my-custom-networking-cluster
```

Confirme se a exclusão do cluster foi concluída, com o comando a seguir.

```
aws eks describe-cluster --name my-custom-networking-cluster --query cluster.status
 --output text
```

Quando uma saída semelhante à seguinte for retornada, significa que o cluster foi excluído com êxito:

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-custom-networking-cluster.
```

5. Exclua o perfil do IAM do cluster.

a. Desanexe as políticas da função.

```
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole --
policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
```

b. Exclua o perfil.

```
aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
```

6. Exclua as sub-redes que você criou em uma etapa anterior.

```
aws ec2 delete-subnet --subnet-id $new_subnet_id_1
aws ec2 delete-subnet --subnet-id $new_subnet_id_2
```

7. Exclua a pilha da VPC criada.

```
aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
```

Atribuir mais endereços IP aos nós do Amazon EKS com prefixos

Aplica-se a: nós do Linux e Windows com instâncias do Amazon EC2

Aplica-se a: sub-redes públicas e privadas

Cada tipo de instância do Amazon EC2 oferece suporte a um número máximo de interfaces de rede elásticas e a um número máximo de endereços IP que podem ser atribuídos a cada interface de rede. Cada nó requer um endereço IP para cada interface de rede. Todos os outros endereços IP disponíveis podem ser atribuídos a `Pods`. Cada `Pod` exige seu próprio endereço IP. Como resultado, é possível ter nós com recursos de computação e memória disponíveis, mas que não são capazes de acomodar `Pods` adicionais porque o nó ficou sem endereços IP para atribuir aos `Pods`.

É possível aumentar o número de endereços IP que os nós podem atribuir a `Pods` atribuindo prefixos IP em vez de atribuir endereços IP secundários individuais aos nós. Cada prefixo inclui vários endereços IP. Se você não configurar o cluster para atribuição de prefixos IP, ele deverá fazer mais chamadas da interface de programação de aplicações (API) do Amazon EC2 para configurar as interfaces de rede e os endereços IP necessários para a conectividade de pods. À medida que os clusters aumentam de tamanho, a frequência dessas chamadas de API podem levar a tempos de execução de instâncias e pods maiores. Isso resulta em atrasos na escalabilidade para atender à demanda de workloads grandes e com alto pico, além de adicionar custos e despesas gerais de gerenciamento, pois você precisará provisionar clusters e VPCs adicionais para atender aos requisitos de escalabilidade. Para obter mais informações, consulte Limites de escalabilidade do Kubernetes no GitHub.

Compatibilidade com recursos do plug-in CNI da Amazon VPC para Kubernetes

É possível usar prefixos IP com os seguintes atributos:

- Tradução do endereço de rede de origem IPv4: para obter mais informações, consulte the section called “Tráfego de saída”.
- Endereços IPv6 para clusters, pods e serviços: para obter mais informações, consulte the section called “IPv6”.
- Restrição do tráfego usando políticas de rede do Kubernetes: para obter mais informações, consulte the section called “Políticas do Kubernetes”.

A lista a seguir fornece informações sobre as configurações do plug-in CNI da Amazon VPC aplicáveis. Para obter mais informações sobre cada configuração, consulte amazon-vpc-cni-k8s no GitHub.

- `WARM_IP_TARGET`
- `MINIMUM_IP_TARGET`
- `WARM_PREFIX_TARGET`

Considerações

Considere o seguinte ao usar esse atributo:

- Cada tipo de instância do Amazon EC2 é compatível com um número máximo de pods. Se o grupo de nós gerenciados consistir em vários tipos de instância, o menor número de pods máximos para uma instância no cluster será aplicado a todos os nós do cluster.
- Por padrão, o número máximo de `Pods` que podem ser executados em um nó é 110, mas esse número pode ser alterado. Se você alterar o número e tiver um grupo de nós gerenciados existente, a próxima atualização da AMI ou do modelo de execução do seu grupo de nós resultará no surgimento de novos nós com o valor alterado.
- Ao fazer a transição da atribuição de endereços IP para a atribuição de prefixos IP, recomendamos criar novos grupos de nós para aumentar o número de endereços IP disponíveis, em vez de fazer uma substituição contínua dos nós existentes. A execução de pods em um nó com endereços IP e prefixos atribuídos pode causar inconsistência na capacidade anunciada do endereço IP, afetando as futuras workloads no nó. Para ver a forma recomendada de realizar a transição, consulte Modo de delegação de prefixo para Linux no Guia de práticas recomendadas do Amazon EKS.
- O escopo do grupo de segurança está no nível do nó: para obter mais informações, consulte Grupos de segurança.
- Os prefixos de IP atribuídos a uma interface de rede são compatíveis com alta densidade de pods por nó e têm o melhor tempo de execução.
- Prefixos IP e endereços IP estão associados a interfaces de rede elásticas padrão do Amazon EC2. Os pods que exigem grupos de segurança específicos recebem o endereço IP principal de uma interface de rede de filial. Você pode misturar pods recebendo endereços IP ou endereços IP de prefixos de IP com pods obtendo interfaces de rede de ramificação no mesmo nó.
- Para clusters somente com nós Linux.
  - Depois de configurar o complemento para atribuir prefixos a interfaces de rede, você não poderá fazer downgrade do complemento do plug-in CNI da Amazon VPC para Kubernetes para uma versão inferior à `1.9.0` (ou `1.10.1`) sem remover todos os nós em todos os grupos de nós do cluster.

- Se você também estiver usando grupos de segurança para pods, com `POD_SECURITY_GROUP_ENFORCING_MODE=standard` e `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`, quando seus pods se comunicarem com endpoints fora da VPC, os grupos de segurança do nó serão usados, em vez de quaisquer grupos de segurança que você possa ter atribuído a seus pods.

  Se você também estiver usando grupos de segurança para pods, com `POD_SECURITY_GROUP_ENFORCING_MODE = strict`, quando seu `Pods` se comunicar com endpoints fora de sua VPC, os grupos de segurança `Pod's` serão usados.

Aumente a quantidade de endereços IP disponíveis para o nó do Amazon EKS

É possível aumentar o número de endereços IP que os nós podem atribuir a pods atribuindo prefixos de IP em vez de endereços IP secundários individuais aos nós.

Pré-requisitos

- É necessário um cluster existente. Para implantar, consulte the section called "Criar um cluster".
- As sub-redes nas quais seus nós do Amazon EKS estão devem ter blocos de Encaminhamento Entre Domínios Sem Classificação (CIDR) `/28` (para clusters `IPv4`) ou `/80` (para clusters `IPv6`) suficientemente contíguos. Somente nós Linux podem existir em um cluster `IPv6`. O uso de prefixos IP poderá falhar se os endereços IP estiverem espalhados pelo CIDR da sub-rede. Recomendamos o seguinte:
  - Usar uma reserva CIDR de sub-rede para que, mesmo que algum endereço IP dentro do intervalo reservado ainda esteja em uso, após sua liberação, os endereços IP não sejam reatribuídos. Isso garante que os prefixos estejam disponíveis para alocação sem segmentação.
  - Usar novas sub-redes que sejam utilizadas especificamente para executar workloads às quais os prefixos IP estão atribuídos. As workloads do Linux e do Windows podem ser executadas na mesma sub-rede ao atribuir prefixos de IP.
- Para atribuir prefixos IP aos seus nós, eles devem ser baseados no AWS Nitro. As instâncias que não são baseadas em Nitro continuam alocando endereços IP secundários individuais, mas têm um número significativamente menor de endereços IP para atribuir aos pods do que as instâncias baseadas em Nitro.
- Somente para clusters com nós do Linux: se o cluster estiver configurado para a família `IPv4`, você deverá ter a versão `1.9.0` ou mais recente do complemento do plug-in CNI da Amazon VPC para Kubernetes instalada. É possível verificar a versão atual com o comando a seguir.

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/"
 -f 2
```

Se o cluster estiver configurado para a família `IPv6`, a versão `1.10.1` ou posterior do complemento deverá estar instalada. Se a versão do plug-in for anterior às versões exigidas, será necessário atualizá-lo. Para obter mais informações, consulte as seções de atualização de Atribuir IPs a pods com o Amazon VPC CNI.

- Somente para clusters com nós do Windows
  - Você deve ter o suporte do Windows habilitado para o cluster. Para obter mais informações, consulte the section called “Habilitar o suporte ao Windows”.

Atribuir prefixos de endereço IP aos nós

Configure seu cluster para atribuir prefixos de endereço IP aos nós. Conclua o procedimento que corresponda ao sistema operacional do seu nó.

Linux

1. Habilite o parâmetro para atribuir prefixos a interfaces de rede para o DaemonSet CNI da Amazon VPC. Ao implantar um cluster, a versão `1.10.1` ou mais recente do complemento do plug-in CNI da Amazon VPC para Kubernetes é implantada com ele. Se você criou o cluster com a família `IPv6`, essa configuração foi definida como `true` por padrão. Se você criou o cluster com a família `IPv4`, essa configuração foi definida como `false` por padrão.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_PREFIX_DELEGATION=true
```

> **Important**
>
> Mesmo que sua sub-rede tenha endereços IP disponíveis, se ela não tiver blocos `/28` contíguos disponíveis, você verá o erro a seguir nos logs do plug-in CNI da Amazon VPC para Kubernetes.
>
> ```
> InsufficientCidrBlocks: The specified subnet does not have enough free cidr
>  blocks to satisfy the request
> ```
>
> Isso pode acontecer devido à fragmentação dos endereços IP secundários existentes espalhados por uma sub-rede. Para resolver esse erro, crie uma sub-rede e inicie pods

lá ou use uma reserva CIDR de sub-rede do Amazon EC2 para reservar espaço em uma sub-rede para uso com atribuição de prefixo. Para obter mais informações, consulte Comportamento do endereçamento IP para sua sub-rede no Guia do usuário da Amazon VPC.

2. Caso planeje implantar um grupo de nós gerenciados sem um modelo de inicialização, ou com um modelo de inicialização no qual não especificou um ID da AMI, e esteja usando uma versão do plug-in CNI da Amazon VPC para Kubernetes igual ou posterior às versões listadas nos pré-requisitos, pule para a próxima etapa. Os grupos de nós gerenciados calculam automaticamente o número máximo de pods para você.

   Se você estiver implantando um grupo de nós autogerenciados ou um grupo de nós gerenciados com um modelo de execução em que você especificou um ID de AMI, será necessário determinar o número recomendado do Amazon EKS de pods máximos para seus nós. Siga as instruções em Pods máximos recomendados do Amazon EKS para cada tipo de instância do Amazon EC2, adicionando `--cni-prefix-delegation-enabled` à etapa 3. Anote a saída para uso em uma etapa superior.

   > **Important**
   >
   > Grupos de nós gerenciados impõem um número máximo para o valor de `maxPods`. Para instâncias com menos de 30 vCPUs, o número máximo é 110 e, para todas as outras instâncias, o número máximo é 250. Esse número máximo é aplicado quer a delegação de prefixo esteja habilitada ou não.

3. Se você estiver usando um cluster configurado para `IPv6`, pule para a próxima etapa.

   Especifique os parâmetros em uma das opções a seguir. Para determinar qual opção é a certa para você e qual valor fornecer para ela, consulte WARM_PREFIX_TARGET, WARM_IP_TARGET, and MINIMUM_IP_TARGET no GitHub.

   Você pode substituir os valores de exemplo por um valor maior que zero.

   - `WARM_PREFIX_TARGET`

     ```
     kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
     ```

   - `WARM_IP_TARGET` ou `MINIMUM_IP_TARGET` – Se qualquer um dos dois valores for definido, substitui qualquer valor definido para `WARM_PREFIX_TARGET`.

```
kubectl set env ds aws-node -n kube-system WARM_IP_TARGET=5
```

```
kubectl set env ds aws-node -n kube-system MINIMUM_IP_TARGET=2
```

4. Crie um dos tipos de grupos de nós listados a seguir, utilizando, pelo menos, uma instância do Amazon EC2 do tipo Nitro com Amazon Linux 2023. Para obter uma lista dos tipos de instância do Nitro, consulte Instâncias criadas no Sistema Nitro no Guia do usuário do Amazon EC2. Este recurso não é compatível no Windows. Para as opções que incluem *110*, substitua pelo valor da etapa 3 (recomendado) ou por seu próprio valor.

   - Autogerenciado: implante o grupo de nós usando as instruções em Criar nós autogerenciados do Amazon Linux. Antes de criar a pilha do CloudFormation, abra o arquivo de modelo e ajuste `UserData` no `NodeLaunchTemplate`, conforme mostrado abaixo.

     ```
     ...
                 apiVersion: node.eks.aws/v1alpha1
                 kind: NodeConfig
                 spec:
                   cluster:
                     name: ${ClusterName}
                     apiServerEndpoint: ${ApiServerEndpoint}
                     certificateAuthority: ${CertificateAuthorityData}
                     cidr: ${ServiceCidr}
                   kubelet:
                     config:
                       maxPods: 110
     ...
     ```

     Se estiver usando `eksctl` para criar o grupo de nós, poderá usar o comando a seguir.

     ```
     eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-node
      110
     ```

   - Managed (Gerenciado) – Implante seu grupo de nós usando uma das seguintes opções:
     - Sem um modelo de inicialização ou com um modelo de inicialização sem uma ID de AMI especificada: conclua o procedimento em Criar um grupo de nós gerenciados para o seu cluster. Grupos de nós gerenciados agora calculam automaticamente o valor recomendado de `max-pods` do Amazon EKS para você.

- Com um modelo de lançamento com um ID de AMI especificado – Em seu modelo de lançamento, especifique um ID de AMI otimizado do Amazon EKS ou uma AMI personalizada criada a partir da AMI otimizada do Amazon EKS e, em seguida implante o grupo de nós usando um modelo de lançamento e forneça os seguintes dados do usuário no modelo de lançamento. Estes dados do usuário são transferidos de um objeto `NodeConfig` para serem lidos pela ferramenta `nodeadm` no nó. Para obter mais informações sobre `nodeadm`, consulte a documentação do nodeadm.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="//"

--//
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
 cluster:
   apiServerEndpoint: [.replaceable]`my-cluster`
   certificateAuthority: [.replaceable]`LS0t...`
   cidr: [.replaceable]`10.100.0.0/16`
   name: [.replaceable]`my-cluster
 kubelet:
   config:
     maxPods: [.replaceable]`110`
--//--
```

Se estiver usando `eksctl` para criar o grupo de nós, poderá usar o comando a seguir.

```
eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
```

Se você criou uma AMI personalizada que não foi criada a partir da AMI otimizada do Amazon EKS, precisa criar a configuração você mesmo.

> **Note**
>
> Se você também quiser atribuir endereços IP aos pods de uma sub-rede diferente da sub-rede da instância, será necessário habilitar o recurso nesta etapa. Para obter mais informações, consulte the section called “Redes personalizadas”.

Windows

1. Habilite a atribuição de prefixos IP.
   a. Abra o `ConfigMap` de `amazon-vpc-cni` para edição.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. Adicione as linhas a seguir à seção `data`:

   ```
     enable-windows-prefix-delegation: "true"
   ```

   c. Salve o arquivo e feche o editor.
   d. Confirme se a anotação foi adicionada ao `ConfigMap`.

   ```
   kubectl get configmap -n kube-system amazon-vpc-cni -o "jsonpath={.data.enable-
   windows-prefix-delegation}"
   ```

   Se a saída retornada não for `true`, é possível que um erro tenha ocorrido. Tente concluir a etapa novamente.

   > **Important**
   >
   > Mesmo que sua sub-rede tenha endereços IP disponíveis, se ela não tiver blocos `/28` contíguos disponíveis, você verá o erro a seguir nos logs do plug-in CNI da Amazon VPC para Kubernetes.
   >
   > ```
   > InsufficientCidrBlocks: The specified subnet does not have enough free cidr
   >  blocks to satisfy the request
   > ```
   >
   > Isso pode acontecer devido à fragmentação dos endereços IP secundários existentes espalhados por uma sub-rede. Para resolver esse erro, crie uma sub-rede e inicie

pods lá ou use uma reserva CIDR de sub-rede do Amazon EC2 para reservar espaço em uma sub-rede para uso com atribuição de prefixo. Para obter mais informações, consulte Comportamento do endereçamento IP para sua sub-rede no Guia do usuário da Amazon VPC.

2. (Opcional) Especifique uma configuração adicional para controlar o comportamento de pré-escalabilidade e escalabilidade dinâmica do seu cluster. Para obter mais informações, consulte Opções de configuração com o modo de delegação de prefixo no Windows no GitHub.

   a. Abra o `ConfigMap` de `amazon-vpc-cni` para edição.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. Substitua os valores de exemplo por um valor maior que zero e adicione as entradas necessárias à seção `data` do `ConfigMap`. Se você definir um valor para um `warm-ip-target` ou `minimum-ip-target`, o valor substituirá qualquer valor definido para `warm-prefix-target`.

   ```
   warm-prefix-target: "1"
   warm-ip-target: "5"
   minimum-ip-target: "2"
   ```

   c. Salve o arquivo e feche o editor.

3. Crie grupos de nós do Windows com pelo menos um tipo de instância do Nitro para Amazon EC2. Para obter uma lista dos tipos de instância do Nitro, consulte Instâncias criadas no Sistema Nitro no Guia do usuário do Amazon EC2. Por padrão, o número máximo de pods que podem ser implantados em um nó é 110. Se você quiser aumentar ou diminuir esse número, especifique as informações a seguir nos dados do usuário para a configuração do bootstrap. Substitua *`max-pods-quantity`* pelo seu valor máximo de pods.

   ```
   -KubeletExtraArgs '--max-pods=max-pods-quantity'
   ```

   Se você estiver implantando grupos de nós gerenciados, essa configuração precisará ser adicionada ao modelo de inicialização. Para obter mais informações, consulte the section called “Modelos de execução”. Para obter mais informações sobre os parâmetros de configuração do script de bootstrap do Windows, consulte the section called “Parâmetros de configuração do script de bootstrap”.

Determinar o máximo de pods e endereços IP disponíveis

1. Depois que seus nós forem implantados, visualize-os no cluster.

```
kubectl get nodes
```

   Veja um exemplo de saída abaixo.

```
NAME                                              STATUS     ROLES    AGE   VERSION
ip-192-168-22-103.region-code.compute.internal    Ready      <none>   19m   v1.XX.X-
eks-6b7464
ip-192-168-97-94.region-code.compute.internal     Ready      <none>   19m   v1.XX.X-
eks-6b7464
```

2. Descreva um dos nós para determinar o valor de `max-pods` para o nó e o número de endereços IP disponíveis. Substitua *`192.168.30.193`* pelo endereço `IPv4` no nome de um do seus nós retornado na saída anterior.

```
kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|
PrivateIPv4Address'
```

   Veja abaixo um exemplo de saída.

```
pods:                                  110
vpc.amazonaws.com/PrivateIPv4Address:  144
```

   Na saída anterior, `110` é o número máximo de pods que o Kubernetes implantará no nó, mesmo que endereços IP *`144`* estejam disponíveis.

Atribuir grupos de segurança a pods individuais

Aplica-se a: nós do Linux com instâncias do Amazon EC2

Aplica-se a: sub-redes privadas

Os grupos de segurança para pods integram grupos de segurança do Amazon EC2 aos pods do Kubernetes. Você pode usar grupos de segurança do Amazon EC2 para definir regras que permitem tráfego de rede de entrada e saída de e para pods implantados em nós executados em muitos tipos

de instância do Amazon EC2 e Fargate. Para obter uma explicação detalhada sobre esse recurso, consulte o post do blog Apresentação de grupos de segurança para pods.

Compatibilidade com recursos do plug-in CNI da Amazon VPC para Kubernetes

Você pode usar os grupos de segurança dos pods com os seguintes recursos:

- Tradução do endereço de rede de origem IPv4: para obter mais informações, consulte the section called “Tráfego de saída”.
- Endereços IPv6 para clusters, pods e serviços: para obter mais informações, consulte the section called “IPv6”.
- Restrição do tráfego usando políticas de rede do Kubernetes: para obter mais informações, consulte the section called “Políticas do Kubernetes”.

Considerações

Antes de implantar grupos de segurança para pods, considere as seguintes limitações e condições:

- Os grupos de segurança para pods não podem ser usados com os nós do Windows ou com o modo automático do EKS.
- Os grupos de segurança para pods podem ser usados com clusters configurados para a família `IPv6` que contém nós do Amazon EC2 usando a versão 1.16.0 ou mais recente do plug-in CNI da Amazon VPC. Você pode usar grupos de segurança para pods com clusters configurados para a família `IPv6` que contêm apenas nós do Fargate usando a versão 1.7.7 ou mais recente do plug-in CNI da Amazon VPC. Para obter mais informações, consulte . the section called “IPv6”
- Os grupos de segurança para pods são compatíveis com a maioria das famílias de instâncias do Amazon EC2 baseadas no Nitro, mas não com todas as gerações de uma família. Por exemplo, eles são compatíveis com a família e as gerações de instâncias `m5`, `c5`, `r5`, `m6g`, `c6g` e `r6g`. Mas não são compatíveis com nenhum tipo de instância da família `t`. Para obter uma lista completa de tipos de instância compatíveis, consulte o arquivo limits.go no GitHub. Seus nós devem ser um dos tipos de instâncias listados que têm `IsTrunkingCompatible: true` nesse arquivo.
- Se você estiver usando a rede personalizada e grupos de segurança para pods juntos, o grupo de segurança especificado por grupos de segurança para pods é usado em vez do grupo de segurança especificado na `ENIConfig`.
- Se você estiver usando a versão `1.10.2` ou anterior do plug-in CNI da Amazon VPC e incluir a configuração `terminationGracePeriodSeconds` na especificação do pod, o valor dessa configuração não poderá ser zero.

- Se você estiver usando a versão `1.10` ou anterior do plug-in CNI da Amazon VPC, ou a versão `1.11` com `POD_SECURITY_GROUP_ENFORCING_MODE=strict`, que é a configuração padrão, então os serviços do Kubernetes do tipo `NodePort` e `LoadBalancer` que usam destinos de instância com a `externalTrafficPolicy` definida como `Local` não serão compatíveis com os pods aos quais você atribuir grupos de segurança. Para obter mais informações sobre como usar um balanceador de carga com destinos de instância, consulte . the section called “Balanceamento de carga da rede”.
- Se você estiver usando a versão `1.10` ou anterior do plug-in CNI da Amazon VPC ou a versão `1.11` com `POD_SECURITY_GROUP_ENFORCING_MODE=strict`, que é a configuração padrão, o NAT de origem estará desabilitado para tráfego de saída de pods com grupos de segurança atribuídos para que as regras de grupo de segurança de saída sejam aplicadas. Para acessar a internet, os pods com grupos de segurança atribuídos devem ser iniciados em nós implantados em uma sub-rede privada configurada com uma instância ou gateway NAT. Pods com grupos de segurança atribuídos implantados em sub-redes públicas não conseguem acessar a Internet.

  Se estiver usando a versão `1.11` ou mais recente do plug-in com `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, o tráfego do pod destinado para fora da VPC será convertido no endereço IP da interface de rede primária da instância. Para esse tráfego, as regras nos grupos de segurança da interface de rede primária são usadas, e não as regras nos grupos de segurança do pod.
- Para usar a política de rede do Calico com pods que têm grupos de segurança associados, é necessário usar a versão `1.11.0` ou mais recente do plug-in CNI da Amazon VPC e definir `POD_SECURITY_GROUP_ENFORCING_MODE=standard`. Do contrário, o fluxo de tráfego de e para pods com grupos de segurança associados não estará sujeito à política de rede do Calico e estará limitado apenas à imposição do grupo de segurança do Amazon EC2. Para atualizar sua versão do Amazon VPC CNI, consulte . the section called “CNI da Amazon VPC”
- Pods em execução nos nós do Amazon EC2 que usam grupos de segurança em clusters que usam NodeLocal DNSCache são compatíveis apenas com a versão `1.11.0` ou mais recente do plug-in CNI da Amazon VPC e com `POD_SECURITY_GROUP_ENFORCING_MODE=standard`. Para atualizar sua versão do plugin Amazon VPC CNI, consulte . the section called “CNI da Amazon VPC”
- Grupos de segurança nos pods podem levar a uma latência de inicialização mais alta no pod para pods com alta rotatividade. Isso se deve à limitação de taxa no controlador de recursos.
- O escopo do grupo de segurança do EC2 está no nível do pod. Para obter mais informações, consulte Grupos de segurança.

Se você tiver definido `POD_SECURITY_GROUP_ENFORCING_MODE=standard` e `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`, o tráfego destinado a endpoints fora da VPC usará os grupos de segurança do nó e não os grupos de segurança do pod.

Configurar o plug-in CNI da Amazon VPC para Kubernetes de grupos de segurança para pods do Amazon EKS

Caso use pods com instâncias do Amazon EC2, é necessário configurar o plug-in CNI da Amazon VPC para grupos de segurança do Kubernetes.

Caso use apenas pods do Fargate e não tenha nenhum nó do Amazon EC2 no cluster, consulte the section called “SecurityGroupPolicy”.

1. Verifique a versão atual do plug-in CNI da Amazon VPC para Kubernetes com o seguinte comando:

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
    cut -d : -f 3
   ```

   Veja um exemplo de saída abaixo.

   ```
   v1.7.6
   ```

   Se a versão do plug-in CNI da Amazon VPC para Kubernetes for anterior à `1.7.7`, atualize o plug-in para a versão `1.7.7` ou mais recente. Para obter mais informações, consulte the section called “CNI da Amazon VPC”.

2. Adicione a política do IAM gerenciada pelo AmazonEKSVPCResourceController ao perfil de cluster que está associado ao seu cluster do Amazon EKS. A política permite que a função gerencie interfaces de rede, seus endereços IP privados e seu anexo e desprendimento de e para instâncias de rede.

   a. Recupere o nome do perfil do IAM do cluster e armazene-o em uma variável. Substitua *my-cluster* pelo nome do cluster.

      ```
      cluster_role=$(aws eks describe-cluster --name my-cluster --query cluster.roleArn
       --output text | cut -d / -f 2)
      ```

   b. Anexe a política ao perfil.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSVPCResourceController --role-name $cluster_role
```

3. Habilite o complemento CNI da Amazon VPC para gerenciar interfaces de rede para pods, definindo a variável `ENABLE_POD_ENI` como `true` no DaemonSet `aws-node`. Depois que essa configuração estiver definida como `true`, para cada nó no cluster, o complemento cria um recurso personalizado `cninode`. O controlador de recursos da VPC cria e anexa uma interface de rede especial chamada Interface de rede com a descrição `aws-k8s-trunk-eni`.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
```

> **Note**
>
> A interface de rede de tronco está incluída no número máximo de interfaces de rede suportadas pelo tipo de instância. Para obter uma lista do número máximo de interfaces de rede com suporte por cada tipo de instância, consulte IP addresses per network interface per instance type (Endereços IP por interface de rede por tipo de instância) no Guia do usuário do Amazon EC2. Se o nó já tiver o número máximo de interfaces de rede padrão anexadas a ele, o controlador de recursos da VPC reservará um espaço. Você terá que reduzir a escala verticalmente dos pods em execução o suficiente para que o controlador desconecte e exclua uma interface de rede padrão, crie a interface de rede tronco e anexe-a à instância.

4. Você pode ver quais de seus nós possuem um recurso personalizado do `CNINode` com o comando a seguir. Se `No resources found` for retornado, aguarde alguns segundos e tente de novo. A etapa anterior requer a reinicialização do plug-in CNI da Amazon VPC para pods do Kubernetes, o que leva alguns segundos.

```
kubectl get cninode -A
     NAME FEATURES
     ip-192-168-64-141.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
     ip-192-168-7-203.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
```

   Se você estiver usando versões do VPC CNI anteriores a `1.15`, rótulos de nós foram usados em vez do recurso `CNINode` personalizado. Você pode ver quais do seus nós têm o rótulo de nó `aws-k8s-trunk-eni` definido como `true` com o comando a seguir. Se `No resources found`

for retornado, aguarde alguns segundos e tente de novo. A etapa anterior requer a reinicialização do plug-in CNI da Amazon VPC para pods do Kubernetes, o que leva alguns segundos.

```
kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
```

Depois que a interface de rede tronco for criada, os pods receberão endereços IP secundários das interfaces de rede tronco ou padrão. A interface de tronco é excluída automaticamente se o nó for excluído.

Quando você implanta um grupo de segurança para um pod em uma etapa posterior, o controlador de recursos da VPC cria uma interface de rede especial denominada interface de rede de ramificação, com uma descrição de `aws-k8s-branch-eni`, e associa os grupos de segurança a ela. As interfaces de rede da ramificação são criadas como um adicional para as interfaces de rede padrão e de tronco conectadas ao nó.

Se você estiver usando uma investigação de atividade ou de preparação, também precisará desabilitar o TCP early demux, de modo que o `kubelet` possa se conectar aos pods em interfaces de rede de ramificação usando o TCP. Para desativar o TCP early demux, execute o seguinte comando:

```
kubectl patch daemonset aws-node -n kube-system \
  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":
[{"name":"DISABLE_TCP_EARLY_DEMUX","value":"true"}],"name":"aws-vpc-cni-init"}]}}}}'
```

> **Note**
>
> Se você estiver usando o plug-in CNI `1.11.0` ou mais recente da Amazon VPC para o complemento do Kubernetes e definir `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, conforme descrito na próxima etapa, não será necessário executar o comando anterior.

5. Se o cluster usar `NodeLocal DNSCache`, ou caso deseje usar a política de rede do Calico com os pods que têm seus próprios grupos de segurança, ou caso tenha serviços do Kubernetes do tipo `NodePort` e `LoadBalancer` que usam destinos de instâncias com uma `externalTrafficPolicy` definida como `Local` para os pods aos quais deseja atribuir grupos de segurança, você deverá estar usando a versão `1.11.0` ou mais recente do complemento do plug-in CNI da Amazon VPC para Kubernetes e habilitar a seguinte configuração:

```
kubectl set env daemonset aws-node -n kube-system
 POD_SECURITY_GROUP_ENFORCING_MODE=standard
```

IMPORTANTE: as regras do grupo de segurança dos pods não são aplicadas ao tráfego entre os pods ou entre os pods e serviços, como **`kubelet`** ou **`nodeLocalDNS`**, que estão no mesmo nó. Os pods que usam grupos de segurança diferentes no mesmo nó não podem se comunicar porque estão configurados em sub-redes diferentes, e o roteamento está desativado entre essas sub-redes. O tráfego de saída dos pods para endereços fora da VPC é o endereço de rede convertido no endereço IP da interface de rede primária da instância (a menos que você também tenha definido `AWS_VPC_K8S_CNI_EXTERNALSNAT=true`). Para esse tráfego, as regras nos grupos de segurança da interface de rede primária são usadas, e não as regras nos grupos de segurança do pod. ** Para que essa configuração seja aplicada a pods existentes, é necessário reiniciar os pods ou os nós em que esses pods estejam sendo executados.

6. Para saber como usar uma política de grupos de segurança no pod, consulte the section called “SecurityGroupPolicy”.

Usar uma política para grupos de segurança para um pod do Amazon EKS

Para usar grupos de segurança em pods, é necessário ter um grupo de segurança existente. As etapas a seguir mostram como usar a política de grupo de segurança para um pod. A menos que indicado de outra forma, conclua todas as etapas no mesmo terminal, pois são utilizadas nas etapas a seguir variáveis que não persistem entre terminais.

Caso tenha um pod com instâncias do Amazon EC2, você deverá configurar o plug-in antes de usar esse procedimento. Para obter mais informações, consulte the section called “Configurar”.

1. Crie um namespace do Kubernetes no qual implantar recursos do . Você pode substituir *`my-namespace`* pelo nome de um namespace que deseje usar.

```
kubectl create namespace my-namespace
```

2. Implante uma `SecurityGroupPolicy` do Amazon EKS no cluster.
   a. Copie o conteúdo a seguir para o seu dispositivo. Você pode substituir *`podSelector`* por `serviceAccountSelector` se preferir selecionar pods com base em rótulos de conta de serviço. É preciso especificar um seletor ou outro. Um `podSelector` vazio (exemplo: `podSelector: {}`) seleciona todos os pods no namespace. Você pode alterar *`my-role`* para o nome do seu perfil. Uma `serviceAccountSelector` vazia seleciona todas as contas de

serviço no namespace. Você pode substituir *my-security-group-policy* por um nome para a SecurityGroupPolicy e *my-namespace* pelo namespace no qual você deseja criar a SecurityGroupPolicy.

Você deve substituir *my_pod_security_group_id* pelo ID de um grupo de segurança existente. Se você ainda não tiver um grupo de segurança, deverá criar um. Para obter mais informações, consulte Grupos de segurança do Amazon EC2 para instâncias do Linux no Guia do usuário do Amazon EC2. Você deve especificar de 1 a 5 IDs de grupo de segurança. Se você especificar mais de um ID, a combinação de todas as regras em todos os grupos de segurança será efetiva para os pods selecionados.

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - my_pod_security_group_id
EOF
```

Important

Os grupos de segurança que você especificar para os pods devem atender aos seguintes critérios:

- Eles devem existir. Caso não existam, então, quando você implantar um pod que corresponda ao seletor, o pod permanecerá preso no processo de criação. Se você descrever o pod, verá uma mensagem de erro semelhante à seguinte: An `error occurred (InvalidSecurityGroupID.NotFound) when calling the CreateNetworkInterface operation: The securityGroup ID 'sg-05b1d815d1EXAMPLE' does not exist`.
- Eles devem permitir comunicação de entrada do grupo de segurança aplicado aos nós (para `kubelet`) por todas as portas para as quais você configurou sondagens.

- Eles devem permitir uma comunicação de saída pelas portas 53 TCP e UDP com um grupo de segurança atribuído aos pods (ou nós em que os pods são executados) em execução no CoreDNS. O grupo de segurança dos pods do CoreDNS devem permitir tráfego de entrada na porta 53 TCP e UDP do grupo de segurança que você especificar.
- Eles devem ter regras de entrada e saída necessárias para comunicação com outros pods com os quais precisam se comunicar.
- Eles devem ter regras que permitam que os pods se comuniquem com o ambiente de gerenciamento do Kubernetes se você estiver usando esse grupo de segurança com o Fargate. A maneira mais fácil de fazer isso é especificar o grupo de segurança de cluster como um dos grupos de segurança.

As políticas de grupo de segurança só se aplicam a pods recém-agendados. Não afetam os pods em execução.

b. Implante a política.

```
kubectl apply -f my-security-group-policy.yaml
```

3. Implante uma aplicação de amostra com um rótulo que corresponda ao valor *my-role* para *podSelector* que você especificou em uma etapa anterior.

a. Copie o conteúdo a seguir para o seu dispositivo. Substitua os exemplos de valor pelos seus próprios e execute o comando modificado. Se você substituir *my-role*, certifique-se de que seja o mesmo valor que você especificou para o seletor em uma etapa anterior.

```
cat >sample-application.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 4
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
```

```
      labels:
        app: my-app
        role: my-role
    spec:
      terminationGracePeriodSeconds: 120
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
EOF
```

b. Implante a aplicação com o comando a seguir. Quando você implanta a aplicação, o plug-in CNI da Amazon VPC corresponde ao rótulo do `role`, e os grupos de segurança especificados na etapa anterior são aplicados ao pod.

```
kubectl apply -f sample-application.yaml
```

4. Visualizar os pods implantados com a aplicação de exemplo. Para o restante do tópico, esse terminal é indicado como `TerminalA`.

```
kubectl get pods -n my-namespace -o wide
```

Veja abaixo um exemplo de saída.

```
NAME                              READY   STATUS    RESTARTS   AGE     IP
  NODE                                            NOMINATED NODE   READINESS GATES
```

```
my-deployment-5df6f7687b-4fbjm   1/1     Running   0          7m51s   192.168.53.48
  ip-192-168-33-28.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-j9fl4   1/1     Running   0          7m51s   192.168.70.145
  ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-rjxcz   1/1     Running   0          7m51s   192.168.73.207
  ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-zmb42   1/1     Running   0          7m51s   192.168.63.27
  ip-192-168-33-28.region-code.compute.internal   <none>           <none>
```

Note

Tente essas dicas se houver pods presos.

- Se houver pods presos no estado `Waiting`, execute `kubectl describe pod` *`my-deployment-xxxxxxxxxx-xxxxx`* `-n` *`my-namespace`* . Se você observar `Insufficient permissions: Unable to create Elastic Network Interface.`, confirme se adicionou a política do IAM à função de cluster do IAM em uma etapa anterior.
- Se houver pods presos no estado `Pending`, verifique se o tipo de instância do nó está listado em limits.go e se o resultado da multiplicação do número máximo de interfaces de rede de ramificação compatíveis com o tipo de instância pelo número de nós do grupo de nós ainda não foi atingido. Por exemplo, um `m5.large` suporta nove interfaces de rede de ramificação. Se o grupo de nós tiver cinco nós, um máximo de 45 interfaces de rede de ramificação poderá ser criado para o grupo de nós. O 46.º pod que você tentar implantar ficará no estado `Pending` até que outro pod que tenha grupos de segurança associados seja excluído.

Se você executar `kubectl describe pod` *`my-deployment-xxxxxxxxxx-xxxxx`* `-n` *`my-namespace`* e ver uma mensagem semelhante à seguinte, ela pode ser ignorada com segurança. Esta mensagem poderá aparecer quando o plug-in CNI da Amazon VPC para Kubernetes tentar configurar a rede de host e falhar enquanto a interface de rede estiver sendo criada. O plugin CNI registra esse evento até que a interface de rede seja criada.

```
Failed to create Pod sandbox: rpc error: code = Unknown desc = failed to set up
 sandbox container "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f"
 network for Pod "my-deployment-5df6f7687b-4fbjm": networkPlugin
cni failed to set up Pod "my-deployment-5df6f7687b-4fbjm-c89wx_my-namespace" network:
 add cmd: failed to assign an IP address to container
```

Você não pode exceder o número máximo de pods que podem ser executados no tipo de instância. Para obter uma lista do número máximo de pods que você pode executar em cada tipo de instância, consulte eni-max-pods.txt no GitHub. Quando você exclui um pod que tenha grupos de segurança associados ou exclui o nó em que o pod está sendo executado, o controlador de recursos da VPC exclui a interface de rede de ramificação. Se você excluir um cluster com pods usando pods para grupos de segurança, o controlador não excluirá as interfaces de rede de ramificação, portanto, será necessário excluí-las você mesmo. Para obter mais informações sobre como excluir interfaces de rede, consulte Interfaces de rede elásticas no Guia do usuário do Amazon EC2.

5. Em um terminal à parte, faça shell em um dos pods. Para o restante do tópico, esse terminal é indicado como `TerminalB`. Substitua *`5df6f7687b-4fbjm`* pelo ID de um dos pods exibidos no resultado da etapa anterior.

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

6. No shell em `TerminalB`, confirme se a aplicação de amostra funciona.

```
curl my-app
```

Veja abaixo um exemplo de saída.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

Você recebeu a saída porque todos os pods que executam a aplicação estão associados ao grupo de segurança que você criou. Esse grupo contém uma regra que permite todo o tráfego entre todos os pods aos quais o grupo de segurança está associado. O tráfego DNS pode sair desse grupo de segurança para o grupo de segurança do cluster, que está associado aos seus nós. Os nós estão executando os pods do CoreDNS, para os quais seus pods fizeram uma pesquisa de nome.

7. No `TerminalA`, remova as regras de grupo de segurança que permitem a comunicação DNS com o grupo de segurança de cluster do grupo de segurança. Se você não tiver adicionado as regras de DNS ao grupo de segurança de cluster em uma etapa anterior, substitua

$my_cluster_security_group_id pelo ID do grupo de segurança no qual as regras foram criadas.

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_tcp_rule_id
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_udp_rule_id
```

8. No `TerminalB`, tente acessar a aplicação novamente.

```
curl my-app
```

Veja abaixo um exemplo de saída.

```
curl: (6) Could not resolve host: my-app
```

A tentativa falha porque o pod não é mais capaz de acessar os pods do CoreDNS, que têm o grupo de segurança do cluster associado. O grupo de segurança do cluster não tem mais as regras de grupo de segurança que permitem a comunicação DNS do grupo de segurança associado ao pod.

Caso tente acessar a aplicação usando os endereços IP retornados para um dos pods em uma etapa anterior, você ainda receberá uma resposta, pois todas as portas são permitidas entre os pods que têm o grupo de segurança associado a eles, e uma pesquisa de nome não é necessária.

9. Quando você terminar os testes, poderá remover o exemplo de política de grupo de segurança, a aplicação e o grupo de segurança criados. Execute os comandos a seguir no `TerminalA`.

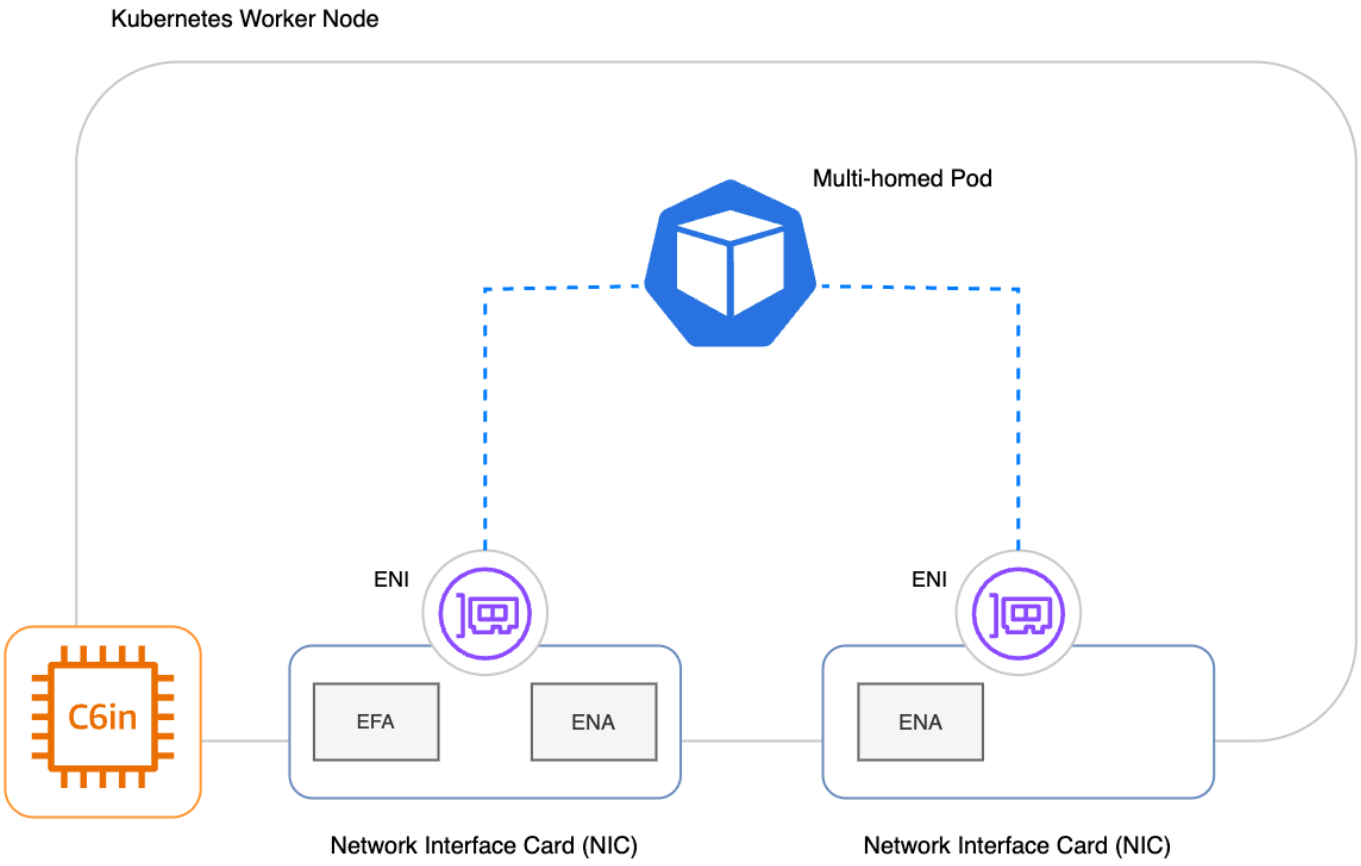
```
kubectl delete namespace my-namespace
aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
security-group-rule-ids $my_inbound_self_rule_id
wait
sleep 45s
aws ec2 delete-security-group --group-id $my_pod_security_group_id
```

Anexação de várias interfaces de rede aos pods

Por padrão, o plug-in CNI da Amazon VPC atribui um endereço IP a cada pod. Esse endereço IP é anexado a uma interface de rede elástica que gerencia todo o tráfego de entrada e saída

do pod. Para aumentar a largura de banda e a performance de taxa de pacotes por segundo, é possível usar o recurso Multi-NIC da CNI da VPC para configurar um pod com várias interfaces de rede. Um pod com várias interfaces de rede é um único pod do Kubernetes que usa múltiplas interfaces de rede (e múltiplos endereços IP). Ao executar um pod com várias interfaces de rede, é possível distribuir o tráfego da aplicação em várias interfaces de rede usando conexões simultâneas. Isso é especialmente útil em casos de uso de Inteligência Artificial (IA), Machine Learning (ML) e Computação de Alta Performance (HPC).

O diagrama a seguir mostra um pod com várias hospedagens em execução em um nó de processamento com várias placas de interface de rede (NICs) em uso.



Contexto

No Amazon EC2, uma interface de rede elástica é um componente lógico de redes em uma VPC que representa uma placa de rede virtual. Para muitos tipos de instância do EC2, as interfaces de rede compartilham uma única placa de interface de rede (NIC) no hardware. Essa única NIC tem uma largura de banda máxima e uma taxa de pacotes por segundo.

Se o recurso Multi-NIC estiver habilitado, a CNI da VPC não atribuirá endereços IP em massa, o que acontece por padrão. Em vez disso, a CNI da VPC atribuirá um endereço IP a uma interface de rede em cada placa de rede sob demanda quando um novo pod for iniciado. Esse comportamento reduz a taxa de esgotamento de endereço IP, que é aumentada com o uso de pods com várias hospedagens. Como a CNI da VPC está atribuindo endereços IP sob demanda, os pods podem levar mais tempo para serem iniciados em instâncias com o recurso multi-NIC habilitado.

Considerações

- Certifique-se de que seu cluster do Kubernetes esteja executando a versão `1.20.0` ou posterior da CNI da VPC. O recurso Multi-NIC só está disponível na versão `1.20.0` ou posterior da CNI da VPC.
- Ative a variável de ambiente `ENABLE_MULTI_NIC` no plug-in CNI da VPC. É possível executar o comando a seguir para definir a variável e iniciar uma implantação do DaemonSet.
  - `kubectl set env daemonset aws-node -n kube-system ENABLE_MULTI_NIC=true`
- Certifique-se de criar nós de processamento que tenham várias placas de interface de rede (NICs). Para obter uma lista de instâncias do EC2 que tenham várias placas de interface de rede, consulte Placas de rede no Guia do usuário do Amazon EC2.
- Se o recurso Multi-NIC estiver habilitado, a CNI da VPC não atribuirá endereços IP em massa, o que acontece por padrão. Como a CNI da VPC está atribuindo endereços IP sob demanda, os pods podem levar mais tempo para serem iniciados em instâncias com o recurso multi-NIC habilitado. Para obter mais informações, consulte a seção anterior the section called “Contexto”.
- Com o recurso Multi-NIC habilitado, os pods não têm várias interfaces de rede por padrão. Você deverá configurar cada workload para usar várias NICs. Adicione a anotação `k8s.amazonaws.com/nicConfig: multi-nic-attachment` às workloads que devam ter várias interfaces de rede.

`IPv6`Considerações do

- Política do IAM personalizada: para clusters `IPv6`, crie e use a seguinte política personalizada do IAM para a CNI da VPC. Essa política é específica para multi-NIC. Para obter mais informações gerais sobre o uso da CNI da VPC com clusters `IPv6`, consulte the section called “IPv6”.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AmazonEKSCNIPolicyIPv6MultiNIC",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:AssignIpv6Addresses",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeTags",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeInstanceTypes",
                "ec2:UnassignIpv6Addresses",
                "ec2:AttachNetworkInterface",
                "ec2:DescribeSubnets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonEKSCNIPolicyENITagIPv6MultiNIC",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        }
    ]
}
```

- Mecanismo de transição de **`IPv6`** não disponível: se você usar o recurso de várias NICs, a CNI da VPC não atribuirá um endereço `IPv4` aos pods em um cluster `IPv6`. Caso contrário, a CNI da VPC atribuirá um endereço `IPv4` local de host a cada pod para que um pod possa se comunicar com recursos externos `IPv4` em outra Amazon VPC ou na Internet.

Usage

Depois que o recurso Multi-NIC for habilitado na CNI da VPC e os pods do `aws-node` forem reiniciados, será possível configurar cada workload para ser de vários hospedeiros. A seguir há um exemplo de uma configuração de YAML com a anotação necessária:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders-deployment
```

```
  namespace: ecommerce
  labels:
    app: orders
spec:
  replicas: 3
  selector:
    matchLabels:
      app: orders
  template:
    metadata:
      annotations:
         k8s.amazonaws.com/nicConfig: multi-nic-attachment
      labels:
        app: orders
    spec:
...
```

Perguntas frequentes

1. O que é uma placa de interface de rede (NIC)?

Uma placa de interface de rede (NIC), também chamada simplesmente de placa de rede, é um dispositivo físico que permite a conectividade de rede para o hardware de computação em nuvem subjacente. Nos servidores EC2 modernos, isso se refere à placa de rede Nitro. Uma Interface de Rede Elástica (ENI) é uma representação virtual dessa placa de rede subjacente.

Alguns tipos de instâncias do EC2 têm várias NICs para maior largura de banda e performance de taxa de pacotes. Para tais instâncias, é possível atribuir ENIs secundárias às placas de rede adicionais. Por exemplo, a ENI #1 pode funcionar como a interface para a NIC conectada ao índice 0 da placa de rede, enquanto a ENI #2 pode funcionar como a interface para a NIC conectada a um índice de placa de rede separado.

2. O que é um pod com vários hospedeiros?

Um pod com vários hospedeiros é um único pod do Kubernetes com várias interfaces de rede (e, por implicação, vários endereços IP). Cada interface de rede do pod está associada a uma Interface de Rede Elástica (ENI), e essas ENIs são representações lógicas de NICs separadas no nó de processamento subjacente. Com várias interfaces de rede, um pod com vários hospedeiros tem capacidade adicional de transferência de dados, o que também aumenta sua taxa de transferência de dados.

> **Important**
>
> A CNI da VPC só pode configurar pods com vários hospedeiros em tipos de instâncias que tenham várias NICs.

3. Por que devo usar esse recurso?

Se você precisar escalar a performance da rede em suas workloads baseadas em Kubernetes, é possível usar o recurso de várias NICs para executar pods com várias interfaces que interajam com todas as NICs subjacentes que tenham um dispositivo ENA conectado a elas. A utilização de placas de rede adicionais aumenta a capacidade de largura de banda e a performance da taxa de pacotes em suas aplicações, distribuindo o tráfego de aplicações em várias conexões simultâneas. Isso é especialmente útil em casos de uso de Inteligência Artificial (IA), Machine Learning (ML) e Computação de Alta Performance (HPC).

4. Como faço para usar esse recurso?

1. Primeiro, você deve garantir que seu cluster do Kubernetes esteja usando a CNI da VPC versão 1.20 ou posterior. Para obter as etapas para atualizar a CNI da VPC como um complemento do EKS, consulte the section called “Atualizar (complemento do EKS)”.
2. Em seguida, você precisará habilitar o suporte a várias NICs na CNI do VPC usando a variável de ambiente `ENABLE_MULTI_NIC`.
3. Em seguida, você deverá se certificar de criar e unir nós que tenham várias placas de rede. Para obter uma lista de tipos de instâncias do EC2 que tenham várias placas de rede, consulte Placas de rede no Guia do usuário do Amazon EC2.
4. Por fim, você configura cada workload para usar várias interfaces de rede (pods de vários hospedeiros) ou usar uma única interface de rede.

5. Como configuro minhas workloads para usar várias NICs em um nó de processamento com suporte?

Para usar pods com vários hospedeiros, você precisa adicionar a seguinte anotação: `k8s.amazonaws.com/nicConfig: multi-nic-attachment`. Isso anexará uma ENI de cada NIC na instância subjacente ao pod (mapeamento de um para muitos entre um pod e as NICs).

Se essa anotação estiver ausente, a CNI da VPC presumirá que o seu pod requer apenas uma interface de rede e atribuirá a ele um IP de uma ENI em qualquer NIC disponível.

6. Quais adaptadores de interface de rede são compatíveis com esse recurso?

É possível usar qualquer adaptador de interface de rede se tiver pelo menos um ENA conectado à placa de rede subjacente para tráfego IP. Para obter mais informações sobre ENAs, consulte Adaptador de Rede Elástica (ENA) no Guia do usuário do Amazon EC2.

Configurações de dispositivos de rede com suporte:

- As interfaces ENA fornecem todos os atributos tradicionais de rede IP e roteamento necessários para oferecer suporte à rede IP para uma VPC. Para obter mais informações, consulte Habilitação de redes aperfeiçoadas com ENA em instâncias do EC2.
- As interfaces EFA (EFA com ENA) fornecem o dispositivo ENA para redes IP e o dispositivo EFA para comunicação de baixa latência e alto throughput.

> **Important**
>
> Se uma placa de rede tiver apenas um adaptador somente EFA conectado a ela, a CNI da VPC a ignorará ao provisionar a conectividade de rede para um pod com vários hospedeiros. No entanto, se você combinar um adaptador somente EFA com um adaptador ENA em uma placa de rede, a CNI da VPC também gerenciará as ENIs nesse dispositivo. Para usar interfaces somente EFA com clusters do EKS, consulte the section called “Configuração de treinamento de clusters com o EFA”.

7. Posso ver se um nó no meu cluster tem suporte para ENA?

Sim, é possível usar a AWS CLI ou a API do EC2 para recuperar informações de rede sobre uma instância do EC2 no seu cluster. Isso fornece detalhes sobre se a instância tem ou não suporte para ENA. No exemplo a seguir, substitua `<your-instance-id>` pelo ID de um nó da instância do EC2.

AWSExemplo da CLI de :

```
aws ec2 describe-instances --instance-ids <your-instance-id> --query
 "Reservations[].Instances[].EnaSupport"
```

Resultado do exemplo:

```
[ true ]
```

8. Posso ver os diferentes endereços IP associados a um pod?

Não, não de forma simples. No entanto, é possível usar o `nsenter` a partir do nó para executar ferramentas de rede comuns, como `ip route show`, e ver os endereços IP e as interfaces adicionais.

9. Posso controlar o número de interfaces de rede para meus pods?

Não. Quando sua workload é configurada para usar várias NICs em uma instância com suporte, um único pod tem automaticamente um endereço IP de cada placa de rede na instância. Como alternativa, os pods de único hospedeiros terão uma interface de rede conectada a uma NIC na instância.

> **Important**
>
> As placas de rede que tenham apenas um dispositivo somente EFA conectado a elas serão ignoradas pela CNI da VPC.

10. Posso configurar meus pods para usar uma NIC específica?

Não, não há suporte para isso. Se um pod tiver a anotação relevante, a CNI da VPC o configurará automaticamente para usar cada NIC com um adaptador ENA no nó de processamento.

11. Esse recurso funciona com os outros recursos de rede da CNI da VPC?

Sim, o recurso de várias NICs na CNI da VPC funciona tanto com redes personalizadas quanto com a descoberta aprimorada de sub-redes. No entanto, os pods com vários hospedeiros não usam sub-redes ou grupos de segurança personalizados. Em vez disso, a CNI da VPC atribui endereços IP e interfaces de rede aos pods com vários hospedeiros com a mesma configuração de sub-rede e grupo de segurança do nó. Para obter mais informações sobre redes personalizadas, consulte the section called “Redes personalizadas”.

O recurso de várias NICs na CNI da VPC não funciona e não pode ser combinado com grupos de segurança para pods.

12. Posso usar políticas de rede com esse recurso?

Sim, é possível usar as políticas de rede do Kubernetes com várias NICs. As políticas de rede do Kubernetes restringem o tráfego de rede que entra e sai dos seus pods. Para obter mais informações

sobre a aplicação de políticas de rede com a CNI da VPC, consulte the section called “Políticas do Kubernetes”.

13. O suporte a várias NICs está ativado no Modo Automático do EKS?

Não há suporte para várias NICs em clusters do Modo Automático do EKS.

Plug-ins CNI alternados para clusters do Amazon EKS

O plug-in CNI da Amazon VPC para Kubernetes é o único plug-in CNI compatível com o Amazon EKS com os nós do Amazon EC2. O Amazon EKS é compatível com os principais recursos do Cilium e do Calico para o Amazon EKS Hybrid Nodes. O Amazon EKS executa o Upstream Kubernetes para que você possa instalar plug-ins CNI alternativos em nós do Amazon EC2 no cluster. Se você tiver nós do Fargate no cluster, o plug-in CNI da Amazon VPC para Kubernetes já estará nos nós do Fargate. Trata-se do único plug-in de CNI que você pode usar com os nós do Fargate. Uma tentativa de instalar um plug-in alternativo de CNI em nós do Fargate falhará.

Se você quiser usar um plugin alternativo de CNI em nós do Amazon EC2, recomendamos que obtenha suporte comercial para o plug-in ou tenha a experiência interna para solucionar problemas e contribuir com correções para o projeto do plug-in de CNI.

O Amazon EKS mantém um relacionamento com uma rede de parceiros que oferecem suporte para plugins CNI compatíveis alternativos. Para obter mais detalhes sobre as versões, as qualificações e os testes realizados, consulte a documentação dos parceiros a seguir.

Parceiro	Produto	Documentação
Tigera	Calico	Instruções de instalação
Isovalent	Cilium	Instruções de instalação
Juniper	Cloud-Native Contrail Networking (CN2)	Instruções de instalação
VMware	Antrea	Instruções de instalação

O Amazon EKS tem como objetivo oferecer uma ampla variedade de opções para cobrir todos os casos de uso.

Plugins de política de rede compatíveis alternativos

O Calico é uma solução amplamente adotada para rede e segurança de contêineres. Usar o Calico no EKS fornece uma aplicação de política de rede totalmente compatível para os clusters do EKS. Além disso, você pode optar por usar a rede do Calico, que conserva os endereços IP da VPC subjacente. O Calico Cloud aprimora os recursos do Calico Open House, fornecendo recursos avançados de segurança e observabilidade.

O fluxo de tráfego de e para pods com grupos de segurança associados não está sujeito à política de rede do Calico e está limitado apenas à imposição do grupo de segurança da Amazon VPC.

Se você usar a aplicação da política de rede do Calico, recomendamos definir a variável de ambiente `ANNOTATE_POD_IP` como `true` para evitar um problema conhecido do Kubernetes. Para usar esse recurso, você deve adicionar a permissão `patch` para pods ao ClusterRole `aws-node`. Observe que adicionar permissões de aplicação de patches ao DaemonSet `aws-node` aumenta o escopo de segurança do plug-in. Para obter mais informações, consulte ANNOTATE_POD_IP no repositório do plug-in CNI da VPC no GitHub.

Considerações sobre o Modo Automático do Amazon EKS

O Modo Automático do Amazon EKS não é compatível com plug-ins CNI alternativos ou plug-ins de política de rede. Para ter mais informações, consulte Modo Automático do EKS.

Anexar várias interfaces de rede para pods com o Multus

Multus CNI é um plug-in de interface de rede de contêineres (CNI) para o Amazon EKS que permite anexar várias interfaces de rede a um pod. Para obter mais informações, consulte a documentação do Multus-CNI no GitHub.

No Amazon EKS, cada pod tem uma interface de rede atribuída pelo plug-in CNI da Amazon VPC. Com o Multus, você pode criar um pod multi-home com várias interfaces. Isso é feito com o Multus atuando como um "meta-plugin", um plugin de CNI que pode chamar vários outros plugins de CNI. O suporte da AWS ao Multus vem configurado com o plugin Amazon VPC CNI como plugin delegado padrão.

- O Amazon EKS não vai criar nem publicar plugins de CNI de virtualização de E/S raiz (SR-IOV) e Data Plane Development Kit (DPDK). Porém, você pode obter aceleração de pacotes conectando-se diretamente aos adaptadores Elastic Network Adapters (ENA) do Amazon EC2 por meio plugins de dispositivo host gerenciado e de `ipvlan` do Multus.

- O Amazon EKS suporta o Multus, que fornece um processo genérico que permite o encadeamento simples de plugins de CNI adicionais. Há suporte ao Multus e ao processo de encadeamento, mas a AWS não fornece suporte a todos os plugins de CNI compatíveis que podem ser encadeados ou para problemas que possam surgir nesses plugins de CNI que não estejam relacionados à configuração do encadeamento.
- O Amazon EKS está fornecendo suporte e gerenciamento do ciclo de vida para o plugin Multus, mas não é responsável por nenhum endereço IP ou gerenciamento adicional associado às interfaces de rede adicionais. O endereço IP e o gerenciamento da interface de rede padrão que utiliza o plugin Amazon VPC CNI permanecem inalterados.
- Somente o plugin Amazon VPC CNI recebe suporte oficial como o plugin delegado padrão. Você precisa modificar o manifesto de instalação publicado do Multus para reconfigurar o plugin delegado padrão para uma CNI alternativa se você optar por não usar o plugin Amazon VPC CNI para redes primárias.
- O suporte ao Multus só é oferecido quando a Amazon VPC CNI é usada como a CNI principal. Não oferecemos suporte à Amazon VPC CNI quando usada para interfaces de ordem superior, secundárias ou não.
- Para evitar que o plug-in CNI da Amazon VPC tente gerenciar interfaces de rede adicionais atribuídas aos pods, adicione a seguinte tag à interface de rede:

  chave

  : `node.k8s.amazonaws.com/no_manage`

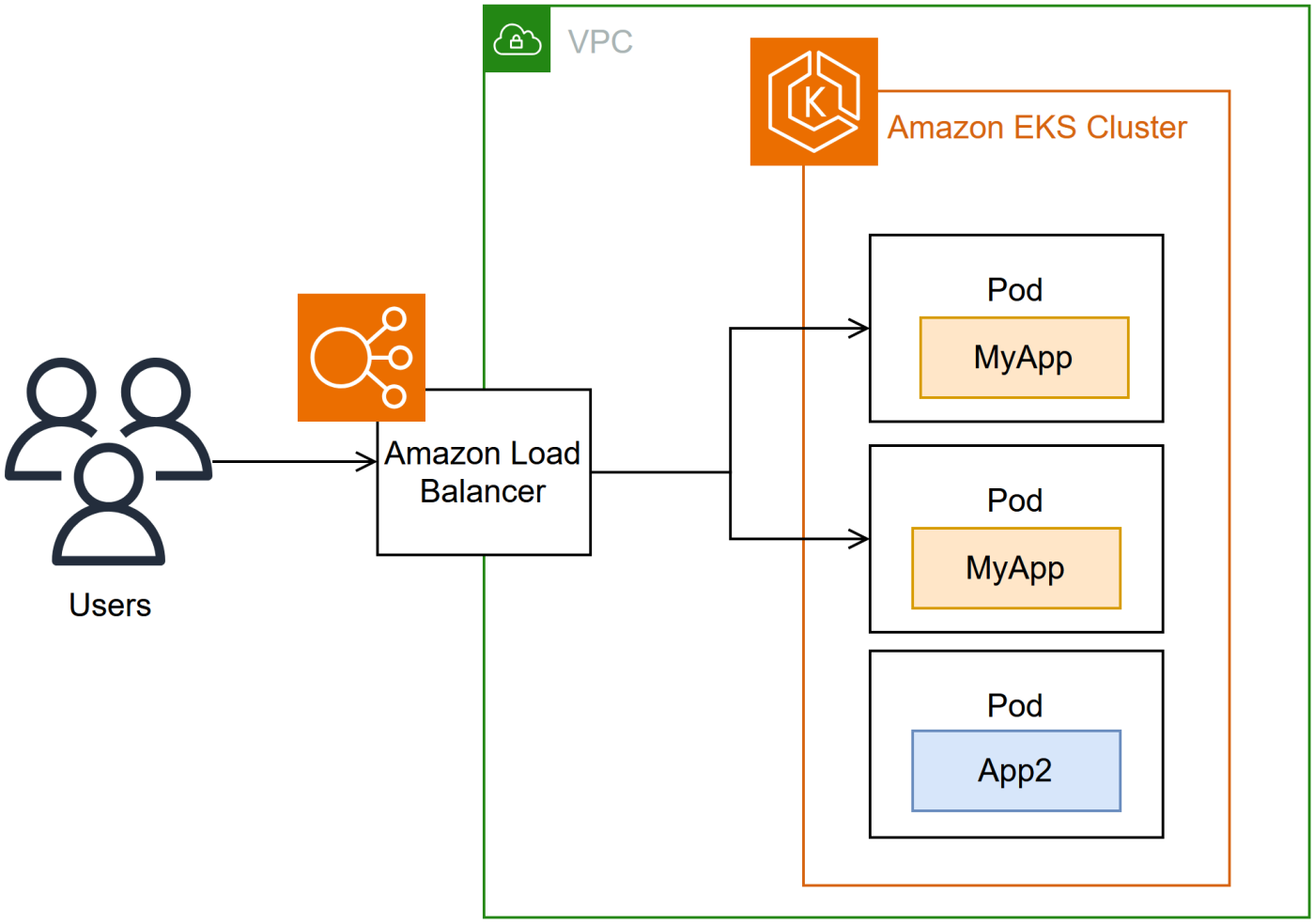
  value

  : `true`
- O Multus é compatível com políticas de rede, mas a política precisa ser enriquecida para incluir portas e endereços IP que possam fazer parte de interfaces de rede adicionais conectadas a pods.

Para uma demonstração da implementação, consulte o Guia de configuração do Multus no GitHub.

Direcionar o tráfego da Internet com o AWS Load Balancer Controller

O AWS Load Balancer Controller gerencia o AWS Elastic Load Balancers para um cluster do Kubernetes. É possível usar o controlador para expor as aplicações no cluster para a Internet. O controlador provisiona balanceadores de carga da AWS que realizam o direcionamento para os recursos Service ou Ingress do cluster. Em outras palavras, o controlador cria um único endereço IP ou nome DNS que realiza o direcionamento para múltiplos pods em seu cluster.



O controlador observa os recursos Service e Ingress do Kubernetes. Em resposta, ele cria os recursos apropriados do AWS Elastic Load Balancing. É possível configurar um comportamento específico de balanceadores de carga ao aplicar anotações aos recursos do Kubernetes. Por exemplo, você pode anexar grupos de segurança da AWS a balanceadores de carga ao usar anotações.

O controlador fornece os seguintes recursos:

Kubernetes **`Ingress`**

O LBC cria um AWS Application Load Balancer (ALB) quando você cria um Kubernetes `Ingress`. Faça uma análise das anotações que você pode aplicar a um recurso Ingress.

Serviço do Kubernetes do tipo **`LoadBalancer`**

O LBC cria um AWS Network Load Balancer (NLB) quando você cria um serviço do Kubernetes do tipo `LoadBalancer`. Faça uma análise das anotações que você pode aplicar a um recurso Service.

Antigamente, o Network Load Balancer do Kubernetes era usado para destinos de instâncias, mas usava o LBC para destinos de IPs. Com o AWS Load Balancer Controller versão `2.3.0` ou mais recente, você pode criar NLBs usando qualquer tipo de destino. Para obter mais informações sobre os tipos de destino do NLB consulte Tipo de destino no Guia do usuário do Network Load Balancer.

O controlador é um projeto de código aberto gerenciado no GitHub.

Antes de implantar o controlador, recomendamos que você analise os pré-requisitos e as considerações em Roteamento de aplicações e tráfego HTTP com Application Load Balancers e the section called “Balanceamento de carga da rede”. Nesses tópicos, você implantará uma aplicação de amostra que inclui um balanceador de carga da AWS.

API **`Gateway`** do Kubernetes

Com o AWS Load Balancer Controller na versão `2.14.0` ou em versões posteriores, o LBC cria um Application Load Balancer (ALB) da AWS quando você cria um `Gateway` do Kubernetes. O gateway do Kubernetes oferece um nível maior de padronização de configuração em comparação à entrada, que precisava de anotações personalizadas para diversas opções comuns. Analise as configurações que podem ser aplicadas a um recurso de gateway. Para obter mais informações sobre a API `Gateway`, consulte a API Gateway na documentação do Kubernetes.

Instalar o controlador

Você pode usar um dos seguintes procedimentos para instalar o AWS Load Balancer Controller:

- Se ainda não conhece o Amazon EKS, recomendamos usar o Helm para simplificar a instalação do AWS Load Balancer Controller. Para obter mais informações, consulte the section called “Instalação usando o Helm”.
- Para configurações avançadas, como clusters com acesso restrito à rede em registros de contêineres públicos, use manifestos do Kubernetes. Para obter mais informações, consulte the section called “Instalar o com manifestos”.

Migração de versões descontinuadas do controlador

- Se você tiver versões obsoletas do AWS Load Balancer Controller instaladas, consulte the section called “Migrar de descontinuado”.
- Não é possível atualizar versões descontinuadas. Essas versões devem ser removidas e uma versão atual do AWS Load Balancer Controller deve ser instalada.
- As versões descontinuadas incluem:
  - AWS ALB Ingress Controller para Kubernetes ("Ingress Controller"), um antecessor do AWS Load Balancer Controller.
  - Qualquer versão `0.1.x` do AWS Load Balancer Controller

Provedor de nuvem herdado

O Kubernetes inclui um provedor de nuvem legado para a AWS. O provedor de nuvem legado é capaz de provisionar balanceadores de carga da AWS, que são semelhantes ao AWS Load Balancer Controller. O provedor de nuvem herdado cria Classic Load Balancers. Se você não instalar o AWS Load Balancer Controller, o Kubernetes usará o provedor de nuvem legado como padrão. Você deve instalar o AWS Load Balancer Controller e evitar o uso do provedor de nuvem legado.

> **Important**
>
> Nas versões 2.5 e mais recentes, o AWS Load Balancer Controller passa a ser o controlador padrão para os recursos de serviços com o `type: LoadBalancer` e cria um AWS Network Load Balancer (NLB) para cada serviço. Ele faz isso criando um webhook mutante para serviços, que define o campo `spec.loadBalancerClass` como `service.k8s.aws/nlb` para novos serviços `type: LoadBalancer`. Você pode desativar esse recurso e voltar a usar o provedor de nuvem antigo como o controlador padrão, definindo o valor `enableServiceMutatorWebhook` do chart do Helm para `false`. A menos que você desative esse recurso, o cluster não provisionará novos Classic Load Balancers para seus serviços. Os Classic Load Balancers existentes continuarão funcionando.

Instale o AWS Load Balancer Controller com o Helm

> **Tip**
>
> Com o Modo Automático do Amazon EKS, você não precisa instalar ou atualizar complementos de rede. O Modo automático inclui recursos de rede de pods e balanceamento de carga.
> Para obter mais informações, consulte Modo Automático do EKS.

Este tópico descreve como instalar o AWS Load Balancer Controller usando o Helm, um gerenciador de pacotes para o Kubernetes, e o `eksctl`. O controlador é instalado com as opções padrão. Para obter mais informações sobre o controlador, incluindo detalhes sobre como configurá-lo com anotações, consulte a documentação do AWS Load Balancer Controller no GitHub.

Nas etapas a seguir, substitua os valores de exemplo pelos seus próprios valores.

Pré-requisitos

Antes de começar este tutorial, conclua as seguintes etapas:

- Crie um cluster do Amazon EKS. Para criar uma, consulte Conceitos básicos.
- Instale o Helm na máquina local.
- Certifique-se de que os complementos `kube-proxy`, CoreDNS e plug-in CNI da Amazon VPC para Kubernetes estejam nas versões mínimas listadas em Tokens de contas de serviços.
- Saiba mais sobre os conceitos do AWS Elastic Load Balancing. Para obter mais informações, consulte o Manual do usuário do Elastic Load Balancing.
- Saiba mais sobre os recursos de entrada e serviço do Kubernetes.

Considerações

Antes de prosseguir com as etapas de configuração nesta página, considere o seguinte:

- A política e o perfil (`AmazonEKSLoadBalancerControllerRole`) do IAM podem ser reutilizados em vários clusters do EKS na mesma conta da AWS.
- Se você estiver instalando o controlador no mesmo cluster em que o perfil (`AmazonEKSLoadBalancerControllerRole`) foi criado originalmente, vá para a Etapa 2: instalar o Load Balancer Controller depois de verificar se o perfil existe.

- Se você estiver usando perfis do IAM para contas de serviço (IRSA, da sigla em inglês), o IRSA deverá ser configurado para cada cluster, e o ARN do provedor OpenID Connect (OIDC) na política de confiança do perfil será específico para cada cluster do EKS. Além disso, se você estiver instalando o controlador em um novo cluster com um `AmazonEKSLoadBalancerControllerRole` existente, atualize a política de confiança do perfil para incluir o provedor OIDC do novo cluster e crie uma conta de serviço com a anotação de perfil apropriada. Para determinar se você já tem um provedor OIDC, ou para criar um, consulte the section called “Provedor OIDC do IAM”.

Etapa 1: criar um perfil do IAM usando a ferramenta **`eksctl`**

As etapas a seguir referem-se à versão de lançamento v2.14.1 do AWS Load Balancer Controller. Para obter mais informações sobre todas as versões, consulte a página de versões do AWS Load Balancer Controller no GitHub.

1. Baixe uma política do IAM para o AWS Load Balancer Controller que permita que ele faça chamadas para APIs da AWS em seu nome.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.14.1/docs/install/iam_policy.json
```

   - Se você for uma partição da AWS não padrão, como um governo ou região da China, consulte as políticas no GitHub e baixe a política apropriada para sua região.

2. Crie uma política do IAM usando a política obtida por download na etapa anterior.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

   > **Note**
   >
   > Se você visualizar a política no Console de gerenciamento da AWS, ele mostrará os avisos para o serviço ELB, mas não para o serviço ELB v2. Isso acontece porque algumas das ações da política existem para o ELB v2, mas não para o ELB. Você pode ignorar esses avisos para o ELB..

3. Substitua os valores do nome do cluster, código da região e ID da conta.

```
eksctl create iamserviceaccount \
    --cluster=<cluster-name> \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws:iam::<AWS_ACCOUNT_ID>:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
    --region <aws-region-code> \
    --approve
```

Etapa 2: instalar o AWS Load Balancer Controller

1. Adicione o repositório de chart do Helm `eks-charts`. A AWS mantém esse repositório no GitHub.

   ```
   helm repo add eks https://aws.github.io/eks-charts
   ```

2. Atualize o repositório local para confirmar que você tem os gráficos mais recentes.

   ```
   helm repo update eks
   ```

3. Instale o AWS Load Balancer Controller.

   Se você estiver implantando o controlador nos nós do Amazon EC2 aos quais você tem acesso restrito ao serviço de metadados de instância (IMDS) do Amazon EC2, ou se você estiver implantando no Fargate ou no Amazon EKS Hybrid Nodes, adicione os seguintes sinalizadores ao comando do `helm`:

   - `--set region=`*`region-code`*
   - `--set vpcId=`*`vpc-xxxxxxxx`*

   Substitua *`my-cluster`* pelo nome do cluster. No comando a seguir, `aws-load-balancer-controller` é a conta de serviço do Kubernetes que você criou em uma etapa anterior.

   Para obter mais informações sobre a configuração do gráfico de leme, consulte values.yaml no GitHub.

   ```
   helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
     -n kube-system \
     --set clusterName=my-cluster \
   ```

```
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --version 1.14.0
```

> **Important**
>
> O gráfico implantado não recebe atualizações de segurança automaticamente. Você precisa atualizar manualmente para um gráfico mais recente quando ele estiver disponível. Ao fazer o upgrade, altere *`install`* para `upgrade` no comando anterior.

O comando `helm install` instala automaticamente as definições de recursos personalizados (CRDs) do controlador. O comando `helm upgrade` não realiza essa instalação. Caso use `helm upgrade`,, você deverá instalar manualmente as CRDs. Execute o seguinte comando para instalar as CRDs:

```
wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-controller/crds/crds.yaml
kubectl apply -f crds.yaml
```

Etapa 3: verificar se o controlador está instalado

1. Verifique se o controlador está instalado.

   ```
   kubectl get deployment -n kube-system aws-load-balancer-controller
   ```

   Veja um exemplo de saída abaixo.

   ```
   NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
   aws-load-balancer-controller   2/2     2            2           84s
   ```

   Você recebe a saída anterior se tiver implantado usando o Helm. Se você implantou usando o manifesto do Kubernetes, só tem uma réplica.

2. Antes de usar o controlador para provisionar os recursos da AWS, o cluster doverá cumprir requisitos específicos. Para obter mais informações, consulte the section called “Balanceador de carga da aplicação” e the section called “Balanceamento de carga da rede”.

Instalar o AWS Load Balancer Controller com manifestos

> **Tip**
>
> Com o Modo Automático do Amazon EKS, você não precisa instalar ou atualizar complementos de rede. O Modo automático inclui recursos de rede de pods e balanceamento de carga.
> Para obter mais informações, consulte Modo Automático do EKS.

Este tópico descreve como instalar o controlador ao fazer download e aplicar os manifestos do Kubernetes. Você pode visualizar a documentação para o controlador no GitHub.

Nas etapas a seguir, substitua os valores de exemplo pelos seus próprios valores.

Pré-requisitos

Antes de começar este tutorial, conclua as seguintes etapas:

- Crie um cluster do Amazon EKS. Para criar uma, consulte Conceitos básicos.
- Instale o Helm na máquina local.
- Certifique-se de que os complementos `kube-proxy`, CoreDNS e plug-in CNI da Amazon VPC para Kubernetes estejam nas versões mínimas listadas em Tokens de contas de serviços.
- Saiba mais sobre os conceitos do AWS Elastic Load Balancing. Para obter mais informações, consulte o Manual do usuário do Elastic Load Balancing.
- Saiba mais sobre os recursos de entrada e serviço do Kubernetes.

Considerações

Antes de prosseguir com as etapas de configuração nesta página, considere o seguinte:

- A política e o perfil (`AmazonEKSLoadBalancerControllerRole`) do IAM podem ser reutilizados em vários clusters do EKS na mesma conta da AWS.
- Se você estiver instalando o controlador no mesmo cluster em que o perfil (`AmazonEKSLoadBalancerControllerRole`) foi criado originalmente, vá para a Etapa 2: instalar o cert-manager depois de verificar se o perfil existe.
- Se você estiver usando perfis do IAM para contas de serviço (IRSA, da sigla em inglês), o IRSA deverá ser configurado para cada cluster, e o ARN do provedor OpenID Connect

(OIDC) na política de confiança do perfil será específico para cada cluster do EKS.
Além disso, se você estiver instalando o controlador em um novo cluster com um `AmazonEKSLoadBalancerControllerRole` existente, atualize a política de confiança do perfil para incluir o provedor OIDC do novo cluster e crie uma conta de serviço com a anotação de perfil apropriada. Para determinar se você já tem um provedor OIDC, ou para criar um, consulte the section called “Provedor OIDC do IAM”.

Etapa 1: configurar o IAM

As etapas a seguir referem-se à versão de lançamento v2.14.1 do AWS Load Balancer Controller. Para obter mais informações sobre todas as versões, consulte a página de versões do AWS Load Balancer Controller no GitHub.

1. Baixe uma política do IAM para o AWS Load Balancer Controller que permita que ele faça chamadas para APIs da AWS em seu nome.

   Example

   AWS

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/install/iam_policy.json
   ```

   AWS GovCloud (US)

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/install/iam_policy_us-gov.json
   ```

   ```
   mv iam_policy_us-gov.json iam_policy.json
   ```

2. Crie uma política do IAM usando a política obtida por download na etapa anterior.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

> **Note**
>
> Se você visualizar a política no Console de gerenciamento da AWS, ele mostrará os avisos para o serviço ELB, mas não para o serviço ELB v2. Isso acontece porque algumas das ações da política existem para o ELB v2, mas não para o ELB. Você pode ignorar esses avisos para o ELB..

Example

eksctl

a. Substitua *my-cluster* pelo nome do seu cluster e *111122223333* pelo ID da conta e, então, execute o comando.

```
eksctl create iamserviceaccount \
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws:iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --approve
```

AWS CLI and kubectl

a. Recupere o ID do provedor OIDC do cluster e armazene-o em uma variável.

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

b. Determine se um provedor OIDC do IAM com o ID do cluster já está em sua conta. Você precisa do OIDC configurado para o cluster e o IAM.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Se um resultado for retornado, isso significa que você já tem um provedor OIDC do IAM para o cluster. Se nenhum resultado for retornado, você deverá criar um provedor OIDC do IAM do seu cluster. Para obter mais informações, consulte the section called “Provedor OIDC do IAM”.

c. Copie o conteúdo a seguir para o seu dispositivo. Substitua *111122223333* pelo ID da sua conta. Substitua *region-code* pela região da AWS em que seu cluster se encontra. Substitua *EXAMPLED539D4633E53DE1B71EXAMPLE* pela saída retornada na etapa anterior.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
load-balancer-controller"
                }
            }
        }
    ]
}
```

d. Crie o perfil do IAM.

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

e. Anexe a política de IAM gerenciada pelo Amazon EKS ao perfil do IAM. Substitua *111122223333* pelo ID da sua conta.

```
aws iam attach-role-policy \
```

```
  --policy-arn arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy
\
  --role-name AmazonEKSLoadBalancerControllerRole
```

f. Copie o conteúdo a seguir para o seu dispositivo. Substitua *111122223333* pelo ID da sua conta. Após substituir o texto, execute o comando modificado para criar o arquivo `aws-load-balancer-controller-service-account.yaml`.

```
cat >aws-load-balancer-controller-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/
AmazonEKSLoadBalancerControllerRole
EOF
```

g. Crie a conta de serviço do Kubernetes no cluster: Conta de serviço do Kubernetes denominada `aws-load-balancer-controller` é anotada com o perfil do IAM que você criou, chamada *AmazonEKSLoadBalancerControllerRole*..

```
kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

Etapa 2: instalar o **`cert-manager`**

Instale `cert-manager` usando um dos métodos a seguir para injetar a configuração do certificado nos webhooks. Para obter mais informações, consulte Conceitos básicos na Documentação do cert-manager.

Recomendamos usar o registro de contêiner `quay.io` para instalar `cert-manager`. Se os nós não tiverem acesso ao registro de contêiner `quay.io`, instale `cert-manager` usando o Amazon ECR (veja abaixo).

Example

Quay.io

a. Se os nós tiverem acesso ao registro do contêiner `quay.io`, instale `cert-manager` para injetar a configuração do certificado nos webhooks.

```
kubectl apply \
    --validate=false \
    -f https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-manager.yaml
```

Amazon ECR

a. Instale `cert-manager` usando um dos métodos a seguir para injetar a configuração do certificado nos webhooks. Para obter mais informações, consulte Conceitos básicos na Documentação do cert-manager.

b. Faça download do manifesto.

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-manager.yaml
```

c. Extraia as imagens a seguir e envie-as para um repositório ao qual seus nós têm acesso. Para obter mais informações sobre como extrair, etiquetar e enviar as imagens para seu próprio repositório, consulte the section called “Copiar uma imagem para um repositório”.

```
quay.io/jetstack/cert-manager-cainjector:v1.13.5
quay.io/jetstack/cert-manager-controller:v1.13.5
quay.io/jetstack/cert-manager-webhook:v1.13.5
```

d. Substitua `quay.io` no manifesto nas três imagens pelo seu próprio nome de registro. O comando a seguir pressupõe que o nome do repositório privado seja o mesmo que o repositório de origem. Substitua *`111122223333.dkr.ecr.region-code.amazonaws.com`* pelo seu registro privado.

```
sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./cert-manager.yaml
```

e. Aplique o manifesto.

```
kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

Etapa 3: instalar o AWS Load Balancer Controller

1. Faça download da especificação do controlador. Para obter mais informações sobre o controlador, consulte a documentação do GitHub.

```
curl -Lo v2_14_1_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.14.1/v2_14_1_full.yaml
```

2. Faça as edições a seguir ao arquivo.

   a. Se você tiver baixado o arquivo `v2_14_1_full.yaml`, execute o seguinte comando para remover a seção `ServiceAccount` no manifesto. Se você não remover essa seção, a anotação necessária que você fez na conta de serviço em uma etapa anterior será substituída. A remoção dessa seção também preservará a conta de serviço criada em uma etapa anterior se você excluir o controlador.

   ```
   sed -i.bak -e '764,772d' ./v2_14_1_full.yaml
   ```

   Se você tiver baixado uma versão de arquivo diferente, abra o arquivo em um editor e remova as linhas a seguir.

   ```
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     labels:
       app.kubernetes.io/component: controller
       app.kubernetes.io/name: aws-load-balancer-controller
     name: aws-load-balancer-controller
     namespace: kube-system
   ---
   ```

   b. Substitua `your-cluster-name` na seção `Deployment spec` do arquivo pelo nome do cluster substituindo *`my-cluster`* pelo nome do seu cluster.

```
sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_14_1_full.yaml
```

c. Se seus nós não tiverem acesso aos repositórios de imagens do Amazon ECR do Amazon EKS, você precisará extrair a image a seguir e enviá-la a um repositório ao qual seus nós tenham acesso. Para obter mais informações sobre como extrair, etiquetar e enviar uma imagem para seu próprio repositório, consulte the section called “Copiar uma imagem para um repositório”.

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.14.1
```

Adicione o nome do registro ao manifesto. O comando a seguir pressupõe que o nome do repositório privado seja o mesmo que o repositório de origem e adicione o nome do seu registro privado para o arquivo. Substitua *111122223333.dkr.ecr.region-code.amazonaws.com* pelo seu registro. Essa linha pressupõe que você nomeou seu repositório privado da mesma forma que o repositório de origem. Caso contrário, altere o texto de `eks/aws-load-balancer-controller` após o nome do registro privado para o nome do repositório.

```
sed -i.bak -e 's|public.ecr.aws/eks/aws-load-balancer-controller|
111122223333.dkr.ecr.region-code.amazonaws.com/eks/aws-load-balancer-
controller|' ./v2_14_1_full.yaml
```

d. (Obrigatório somente para o Fargate ou para um IMDS restrito)

Se você estiver implantando o controlador nos nós do Amazon EC2 que têm acesso restrito ao serviço de metadados de instância (IMDS) do Amazon EC2, ou se você estiver implantando no Fargate ou no Amazon EKS Hybrid Nodes, adicione os `following parameters` em `-args:`.

```
[...]
spec:
      containers:
        - args:
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code
```

```
[...]
```

3. Aplique o arquivo.

```
kubectl apply -f v2_14_1_full.yaml
```

4. Baixe os manifestos `IngressClass` e `IngressClassParams` para seu cluster.

```
curl -Lo v2.14.1_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.14.1/v2_14_1_ingclass.yaml
```

5. Aplique o manifesto ao cluster.

```
kubectl apply -f v2_14_1_ingclass.yaml
```

Etapa 4: verificar se o controlador está instalado

1. Verifique se o controlador está instalado.

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

   Veja um exemplo de saída abaixo.

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

   Você recebe a saída anterior se tiver implantado usando o Helm. Se você implantou usando o manifesto do Kubernetes, só tem uma réplica.

2. Antes de usar o controlador para provisionar os recursos da AWS, o cluster doverá cumprir requisitos específicos. Para obter mais informações, consulte the section called “Balanceador de carga da aplicação” e the section called “Balanceamento de carga da rede”.

Migrar aplicações do ALB Ingress Controller descontinuado

Este tópico descreve como migrar de versões descontinuadas do controlador. Mais especificamente, ele descreve como remover versões descontinuadas do AWS Load Balancer Controller.

- Não é possível atualizar versões descontinuadas. Você deve removê-las primeiro e depois instalar uma versão atual.
- As versões descontinuadas incluem:
  - AWS ALB Ingress Controller para Kubernetes ("Ingress Controller"), um antecessor do AWS Load Balancer Controller.
  - Qualquer versão `0.1.x` do AWS Load Balancer Controller

Remover a versão obsoleta do controlador

> **Note**
>
> É possível que você tenha instalado a versão descontinuada usando o Helm ou de forma manual com manifestos do Kubernetes. Conclua o procedimento utilizando a ferramenta com a qual ele foi originalmente instalado.

1. Se você instalou o chart do Helm `incubator/aws-alb-ingress-controller`, desinstale-o.

   ```
   helm delete aws-alb-ingress-controller -n kube-system
   ```

2. Se tiver a versão `0.1.x` do chart `eks-charts/aws-load-balancer-controller` instalado, desinstale-a. O upgrade de `0.1.x` para a versão `1.0.0` não funciona por causa de uma incompatibilidade com a versão da API do webhook.

   ```
   helm delete aws-load-balancer-controller -n kube-system
   ```

3. Verifique se o controlador está instalado no momento.

   ```
   kubectl get deployment -n kube-system alb-ingress-controller
   ```

   Esta é a saída se o controlador não estiver instalado.

   ```
   Error from server (NotFound): deployments.apps "alb-ingress-controller" not found
   ```

   Esta é a saída se o controlador estiver instalado.

   ```
   NAME                    READY UP-TO-DATE AVAILABLE AGE
   ```

```
alb-ingress-controller 1/1   1          1         122d
```

4. Insira o comando a seguir para executar o controlador.

```
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-
controller/v1.1.8/docs/examples/alb-ingress-controller.yaml
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-
controller/v1.1.8/docs/examples/rbac-role.yaml
```

Migrar para o AWS Load Balancer Controller

Para migrar do ALB Ingress Controller para Kubernetes para o AWS Load Balancer Controller, você precisa:

1. Remover o ALB Ingress Controller (veja as etapas anteriores).
2. Instale o AWS Load Balancer Controller.
3. Adicionar mais uma política ao perfil do IAM usado pelo AWS Load Balancer Controller. Esta política permite que o LBC gerencie recursos criados pelo ALB Ingress Controller para Kubernetes.
4. Faça download da política do IAM. Esta política permite que o AWS Load Balancer Controller gerencie recursos criados pelo ALB Ingress Controller para Kubernetes. Você também pode visualizar a política .

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.14.1/docs/install/iam_policy_v1_to_v2_additional.json
```

5. Se o seu cluster estiver nas regiões AWS GovCloud (US-East) ou AWS GovCloud (US-West) AWS, substitua `arn:aws:` por `arn:aws-us-gov:`.

```
sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

6. Crie a política do IAM e observe o ARN retornado.

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

7. Anexe a política do IAM ao perfil do IAM usado pelo AWS Load Balancer Controller. Substitua *`your-role-name`* pelo nome do perfil, como `AmazonEKSLoadBalancerControllerRole`.

Se você criou o perfil usando o `eksctl`, para localizar o nome do perfil criado, abra o console do AWS CloudFormation e selecione a pilha eksctl-***my-cluster***-addon-iamserviceaccount-kube-system-aws-load-balancer-controller. Selecione a guia Recursos. O nome da função está na coluna ID físico.

```
aws iam attach-role-policy \
  --role-name your-role-name \
  --policy-arn arn:aws:iam::111122223333:policy/
AWSLoadBalancerControllerAdditionalIAMPolicy
```

Gerenciar o CoreDNS para DNS em clusters do Amazon EKS

> **Tip**
>
> Com o Modo automático do Amazon EKS, você não precisa instalar ou atualizar complementos de rede. O Modo automático inclui recursos de rede de pods e balanceamento de carga.
> Para obter mais informações, consulte Modo Automático do EKS.

O CoreDNS é um servidor DNS flexível e extensível que pode servir como DNS de cluster do Kubernetes. Quando você executa um cluster do Amazon EKS com pelo menos um nó, duas réplicas da imagem do CoreDNS são implantadas por padrão, independentemente do número de nós implantados em seu cluster. Os Pods CoreDNS fornecem resolução de nomes para todos os Pods no cluster. Os pods do CoreDNS poderão ser implantados em nós do Fargate se o cluster incluir um Perfil do Fargate com um namespace que corresponda ao namespace para o CoreDNS `deployment`. Para obter mais informações sobre o CoreDNS, consulte Usar o CoreDNS para descoberta de serviços na documentação do Kubernetes.

Versões do CoreDNS

A tabela a seguir lista a versão mais recente do tipo de complemento do Amazon EKS para cada versão do Kubernetes.

Versão do Kubernetes	Versão do CoreDNS
1.34	v1.12.4-eksbuild.1
1.33	v1.12.4-eksbuild.1
1.32	v1.11.4-eksbuild.24
1.31	v1.11.4-eksbuild.24
1.30	v1.11.4-eksbuild.24
1.29	v1.11.4-eksbuild.24
1.28	v1.10.1-eksbuild.38

Important

Se você estiver gerenciando esse complemento automaticamente, as versões na tabela poderão não ser as mesmas que as versões autogerenciadas disponíveis. Para obter mais informações sobre como atualizar o tipo autogerenciado desse complemento, consulte the section called “Atualização (autogerenciada)”.

Considerações importantes sobre a atualização do CoreDNS

- As atualizações do CoreDNS usam um PodDisruptionBudget para ajudar a manter a disponibilidade do serviço DNS durante o processo de atualização.
- Para melhorar a estabilidade e a disponibilidade do CoreDNS Deployment, as versões `v1.9.3-eksbuild.6` e mais recentes e o `v1.10.1-eksbuild.3` são implantados com um `PodDisruptionBudget`. Se você implantou um `PodDisruptionBudget` existente, sua atualização para essas versões pode falhar. Se a atualização falhar, concluir uma das seguintes tarefas deve resolver o problema:
  - Ao fazer a atualização do complemento do Amazon EKS, opte por substituir as configurações existentes como sua opção de resolução de conflitos. Se você fez outras configurações personalizadas no Deployment, certifique-se de fazer backup delas antes de atualizar para poder reaplicar as demais configurações personalizadas após a atualização.

  - Remova o `PodDisruptionBudget` existente e tente fazer a atualização novamente.
- Nas versões `v1.9.3-eksbuild.3` e mais recentes e `v1.10.1-eksbuild.6` e mais recentes do complemento do EKS, o CoreDNS Deployment define `readinessProbe` para usar o endpoint `/ready`. Esse endpoint está habilitado no arquivo de configuração `Corefile` do CoreDNS.

  Caso use um `Corefile` personalizado, você deve adicionar o plug-in `ready` à configuração para que o endpoint `/ready` fique ativo no CoreDNS para a sonda ser usada.
- Nas versões complementares do EKS `v1.9.3-eksbuild.7` e superiores e `v1.10.1-eksbuild.4` e superiores, você pode alterar o `PodDisruptionBudget`. Você pode editar o complemento e alterar essas configurações nas configurações opcionais usando os campos no exemplo a seguir. Este exemplo mostra o padrão `PodDisruptionBudget`.

  ```
  {
      "podDisruptionBudget": {
          "enabled": true,
          "maxUnavailable": 1
          }
  }
  ```

  Você pode definir `maxUnavailable` ou `minAvailable`, mas não pode definir os dois em um único `PodDisruptionBudget`. Para obter mais informações sobre `PodDisruptionBudgets`, consulte Especificação de um PodDisruptionBudget na Documentação do Kubernetes.

  Observe que se você definir `enabled` como `false`, o `PodDisruptionBudget` não será removido. Depois de definir esse campo como `false`, você deve excluir o `PodDisruptionBudget` objeto. Da mesma forma, se você editar o complemento para usar uma versão mais antiga do complemento (rebaixar o complemento) depois de atualizar para uma versão com um `PodDisruptionBudget`, o não será removido `PodDisruptionBudget`. Para excluir o `PodDisruptionBudget`, você pode executar o comando a seguir:

  ```
  kubectl delete poddisruptionbudget coredns -n kube-system
  ```
- No complemento do EKS versão `v1.10.1-eksbuild.5` e superiores, altere a tolerância padrão de `node-role.kubernetes.io/master:NoSchedule` para `node-role.kubernetes.io/control-plane:NoSchedule` para manter a conformidade com o KEP 2067. Para obter mais informações sobre o KEP 2067, consulte KEP-2067: Rename the kubeadm "master" label and taint em Kubernetes Enhancement Proposals (KEPs) no GitHub.

Nas versões `v1.8.7-eksbuild.8` e mais recentes e `v1.9.3-eksbuild.9` e mais recentes do complemento do EKS, ambas as tolerâncias são configuradas para serem compatíveis com todas as versões do Kubernetes.

- Nas versões `v1.9.3-eksbuild.11` e mais recentes e `v1.10.1-eksbuild.7` e mais recentes do complemento do EKS, o CoreDNS Deployment define um valor padrão para `topologySpreadConstraints`. O valor padrão garante que os pods do CoreDNS estejam espalhados pelas zonas de disponibilidade se houver nós em várias zonas de disponibilidade disponíveis. Você pode definir um valor personalizado que será usado em vez do valor padrão. O valor padrão é:

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

Considerações sobre a atualização do CoreDNS **`v1.11`**

- No complemento do EKS versão `v1.11.1-eksbuild.4` e superiores, a imagem de contêiner é baseada em uma imagem de base mínima mantida pelo Amazon EKS Distro, que contém pacotes mínimos e não tem shells. Para obter mais informações, consulte Amazon EKS Distro. O uso e a solução de problemas da imagem do CoreDNS permanecem os mesmos.

Criar o complemento CoreDNS do Amazon EKS

Crie o complemento CoreDNS do Amazon EKS. É necessário ter um cluster antes de criar o complemento. Para ter mais informações, consulte the section called “Criar um cluster”.

1. Veja qual versão do complemento está atualmente instalada no cluster.

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
d : -f 3
```

Veja um exemplo de saída abaixo.

```
v1.10.1-eksbuild.13
```

2. Veja qual tipo de complemento está atualmente instalado no cluster. Dependendo da ferramenta com a qual você criou o cluster, talvez você não tenha o tipo de complemento do Amazon EKS instalado em seu cluster atualmente. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

   Se um número de versão for retornado, você tem o tipo de complemento do Amazon EKS instalado no cluster, e não precisa completar as etapas restantes deste procedimento. Se um erro for retornado, você não tem o tipo de complemento do Amazon EKS instalado no cluster. Conclua as etapas restantes desse procedimento para instalá-lo.

3. Salve a configuração do complemento instalado atualmente.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

4. Crie o complemento usando a AWS CLI. Se você quiser usar o Console de gerenciamento da AWS ou `eksctl` para criar o complemento, consulte the section called “Criação de um complemento” e especifique `coredns` como o nome do complemento. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.

   - Substitua *my-cluster* pelo nome do cluster.
   - Substitua *v1.11.3-eksbuild.1* pela versão mais recente listada na tabela das versões mais recentes da versão do seu cluster.

```
aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-version
 v1.11.3-eksbuild.1
```

   Se você aplicou configurações personalizadas ao seu complemento atual que entrem em conflito com as configurações padrão do complemento Amazon EKS, a criação poderá falhar. Se a criação falhar, você receberá um erro que poderá ajudar a resolver o problema. Como alternativa, você pode adicionar `--resolve-conflicts OVERWRITE` ao comando anterior. Isso permite que o complemento substitua todas as configurações personalizadas existentes. Depois de criar o complemento, você pode atualizá-lo com suas configurações personalizadas.

5. Verifique se a versão mais recente do complemento para a versão do Kubernetes do cluster foi adicionada ao cluster. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

   Pode levar alguns segundos para que a criação do complemento seja concluída.

   Veja um exemplo de saída abaixo.

```
v1.11.3-eksbuild.1
```

6. Se você fez configurações personalizadas no complemento original, antes de criar o complemento Amazon EKS, use a configuração que você salvou em uma etapa anterior para atualizar o complemento Amazon EKS com as configurações personalizadas. Para obter instruções sobre como atualizar o complemento, consulte the section called “Atualizar (complemento do EKS)”.

Atualizar o complemento CoreDNS do Amazon EKS

Atualize o tipo do Amazon EKS do complemento . Se você não adicionou o complemento do Amazon EKS ao cluster, adicione-o ou consulte the section called “Atualização (autogerenciada)”.

Antes de começar, reveja as considerações sobre o upgrade. Para ter mais informações, consulte the section called “Considerações importantes sobre a atualização do CoreDNS”.

1. Veja qual versão do complemento está atualmente instalada no cluster. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 "addon.addonVersion" --output text
```

   Veja um exemplo de saída abaixo.

```
v1.10.1-eksbuild.13
```

   Se a versão retornada for a mesma da versão do cluster do Kubernetes na tabela das versões mais recentes, então você já tem a versão mais recente instalada no cluster e não precisa concluir o restante deste procedimento. Se você receber um erro, em vez de um número de versão no

resultado, você não tem o tipo Amazon EKS do complemento instalado no cluster. Você precisa criar o complemento antes de poder atualizá-lo com este procedimento.

2. Salve a configuração do complemento instalado atualmente.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

3. Atualize seu complemento usando a AWS CLI. Se você quiser usar o Console de gerenciamento da AWS ou `eksctl` para atualizar o complemento, consulte the section called “Atualização de um complemento”. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado.
   - Substitua *`my-cluster`* pelo nome do cluster.
   - Substitua *`v1.11.3-eksbuild.1`* pela versão mais recente listada na tabela das versões mais recentes da versão do seu cluster.
   - A opção `--resolve-conflicts`*`PRESERVE`* preserva os valores de configuração existentes para o complemento. Se você definiu valores personalizados para as configurações do complemento e não usar essa opção, o Amazon EKS sobrescreverá seus valores pelos valores padrão. Se você usar essa opção, recomendamos testar qualquer alteração de campo e valor em um cluster que não seja de produção antes de atualizar o complemento no cluster de produção. Se você alterar esse valor para `OVERWRITE`, todas as configurações serão alteradas para os valores padrão do Amazon EKS. Se você definiu valores personalizados para qualquer configuração, eles poderão ser sobrescritos pelos valores padrão do Amazon EKS. Se você alterar esse valor para `none`, o Amazon EKS não alterará o valor de nenhuma configuração, mas a atualização poderá falhar. Se a atualização falhar, você receberá uma mensagem de erro para ajudar a resolver o conflito.
   - Se você não estiver atualizando uma configuração, remova `--configuration-values '{`*`"replicaCount":3`*`}'` do comando. Se você estiver atualizando uma definição de configuração, substitua *`"replicaCount":3`* pela definição que deseja definir. Neste exemplo, o número de réplicas do CoreDNS é definido como `3`. O valor que você especificar deve ser válido para o esquema da configuração. Se não souber qual é o esquema de configuração, execute `aws eks describe-addon-configuration --addon-name coredns --addon-version` *`v1.11.3-eksbuild.1`* , substituindo *`v1.11.3-eksbuild.1`* pelo número da versão do complemento cuja configuração você deseja ver. O esquema é retornado na saída. Se você tiver alguma configuração personalizada existente que deseja remover e definir os valores de todas as configurações de volta aos padrões do Amazon EKS, remova *`"replicaCount":3`* do comando para que você tenha arquivos vazios `{}`. Para obter mais

informações sobre as configurações do CoreDNS, consulte Personalizando o Serviço DNS na documentação do Kubernetes.

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns --addon-version
 v1.11.3-eksbuild.1 \
    --resolve-conflicts PRESERVE --configuration-values '{"replicaCount":3}'
```

Pode levar alguns segundos para que a atualização seja concluída.

4. Confirme se a versão do complemento foi atualizada. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

Pode levar alguns segundos para que a atualização seja concluída.

Veja um exemplo de saída abaixo.

```
{
    "addon": {
        "addonName": "coredns",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.11.3-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/coredns/
d2c34f06-1111-2222-1eb0-24f64ce37fa4",
        "createdAt": "2023-03-01T16:41:32.442000+00:00",
        "modifiedAt": "2023-03-01T18:16:54.332000+00:00",
        "tags": {},
        "configurationValues": "{\"replicaCount\":3}"
    }
}
```

Atualizar o complemento autogerenciado CoreDNS do Amazon EKS

> **Important**
>
> Recomendamos adicionar o tipo Amazon EKS do complemento ao seu cluster em vez de usar o tipo autogerenciado do complemento. Se você não estiver familiarizado com a diferença entre os tipos, consulte the section called “Complementos do Amazon EKS”. Para obter mais informações sobre como adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criação de um complemento”. Se você não conseguir usar o complemento do Amazon EKS, recomendamos que você envie um problema sobre o motivo pelo qual não pode usar o repositório GitHub para roteiro de contêineres.

Antes de começar, reveja as considerações sobre o upgrade. Para ter mais informações, consulte the section called “Considerações importantes sobre a atualização do CoreDNS”.

1. Confirme que tem o tipo autogerenciado de complemento instalado em seu cluster. Substitua *my-cluster* pelo nome do cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se receber uma mensagem de erro, você tem o tipo autogerenciado do complemento instalado no cluster. Conclua as etapas restantes neste procedimento. Se receber um número de versão, você tem o tipo de complemento do Amazon EKS instalado no cluster. Para atualizar o tipo de Amazon EKS do complemento, use o procedimento em Atualizar o complemento CoreDNS do Amazon EKS em vez de usar este procedimento. Se não estiver familiarizado com a diferença entre os tipos de complemento, consulte the section called “Complementos do Amazon EKS”.
2. Veja qual versão da imagem do contêiner está atualmente instalada em seu cluster.

   ```
   kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
   ```

   Veja um exemplo de saída abaixo.

   ```
   v1.8.7-eksbuild.2
   ```
3. Se a versão atual do CoreDNS for `v1.5.0` ou posterior, mas anterior à versão listada na tabela de versões do CoreDNS, pule esta etapa. Se a sua versão atual for anterior à `1.5.0`, você precisa

modificar o `ConfigMap` para que o CoreDNS use o complemento de encaminhamento, em vez do complemento de proxy.

a. Abra o `ConfigMap` com o comando a seguir.

```
kubectl edit configmap coredns -n kube-system
```

b. Substitua `proxy` na linha a seguir pelo `forward`. Salve o arquivo e saia do editor.

```
proxy . /etc/resolv.conf
```

4. Se você implantou originalmente seu cluster no Kubernetes `1.17` ou anterior, talvez seja necessário remover uma linha descontinuada do manifesto do CoreDNS.

> **Important**
>
> Você deve concluir essa etapa antes de atualizar para a versão `1.7.0` do CoreDNS, mas é recomendável concluir essa etapa mesmo se estiver atualizando para uma versão anterior.

a. Verifique se o manifesto do CoreDNS tem a linha.

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
 grep upstream
```

Se nenhum resultado for retornado, o manifesto não terá a linha e você poderá prosseguir para a próxima etapa para atualizar o CoreDNS. Se um resultado for retornado, você precisará remover a linha.

b. Edite o `ConfigMap` com o comando a seguir, removendo a linha no arquivo que inclui a palavra `upstream` em seu nome. Não altere mais nada no arquivo. Depois que a linha for removida, salve as alterações.

```
kubectl edit configmap coredns -n kube-system -o yaml
```

5. Recupere a versão da imagem atual do CoreDNS:

```
kubectl describe deployment coredns -n kube-system | grep Image
```

Veja um exemplo de saída abaixo.

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. Se você estiver atualizando para o CoreDNS `1.8.3` ou uma versão mais recente, será necessário adicionar a permissão `endpointslices` para o `clusterrole` `system:coredns` do Kubernetes.

```
kubectl edit clusterrole system:coredns -n kube-system
```

Adicione as linhas a seguir abaixo das linhas de permissões existentes na seção `rules` do arquivo.

```
[...]
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
[...]
```

7. Atualize o complemento CoreDNS substituindo *602401143452* e *region-code* pelos valores da saída retornada em uma etapa anterior. Substitua *v1.11.3-eksbuild.1* pela versão do CoreDNS listada na tabela de versões mais recentes para sua versão do Kubernetes.

```
kubectl set image deployment.apps/coredns -n kube-system
 coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.3-
eksbuild.1
```

Veja um exemplo de saída abaixo.

```
deployment.apps/coredns image updated
```

8. Verifique a versão da imagem do contêiner novamente para confirmar que ela foi atualizada para a versão que você especificou na etapa anterior.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

Veja um exemplo de saída abaixo.

```
v1.11.3-eksbuild.1
```

Dimensione pods do CoreDNS para alto tráfego de DNS

Quando você executa um cluster do Amazon EKS com pelo menos um nó, uma implantação de duas réplicas da imagem do CoreDNS é feita por padrão, independentemente do número de nós implantados no cluster. Os pods CoreDNS fornecem resolução de nomes para todos os pods no cluster. As aplicações usam a resolução de nomes para se conectar a pods e serviços no cluster, bem como para se conectar a serviços fora do cluster. À medida que o número de solicitações de resolução de nomes (consultas) dos pods aumenta, os pods do CoreDNS podem ficar sobrecarregados, se tornar lentos e rejeitar solicitações que não conseguem atender.

Para lidar com o aumento da carga nos pods do CoreDNS considere um sistema de ajuste de escala automático para o CoreDNS. O Amazon EKS pode gerenciar o ajuste de escala automático da implantação do CoreDNS na versão do complemento do EKS do CoreDNS. Esse autoscaler do CoreDNS monitora continuamente o estado do cluster, incluindo o número de nós e núcleos de CPU. Com base nessas informações, o controlador adaptará dinamicamente o número de réplicas da implantação do CoreDNS em um cluster do EKS. Esse recurso funciona no CoreDNS `v1.9` e versões posteriores. Para obter mais informações sobre quais versões são compatíveis com o ajuste de escala automático do CoreDNS, consulte a seção a seguir.

O sistema gerencia automaticamente as réplicas do CoreDNS usando uma fórmula dinâmica baseada no número de nós e núcleos de CPU no cluster, calculada como o máximo de (númeroDeNós dividido por 16) e (númeroDeNúcleosDeCPU dividido por 256). Ele avalia a demanda em períodos de pico de 10 minutos, aumentando a escala verticalmente quando necessário para lidar com o aumento da carga de consultas ao DNS, enquanto reduz a escala verticalmente de forma gradual, reduzindo as réplicas em 33% a cada 3 minutos para manter a estabilidade do sistema e evitar interrupções.

Recomendamos usar esse recurso em conjunto com outras práticas recomendadas de autoescalabilidade de clusters do EKS para melhorar a disponibilidade geral da aplicação e a escalabilidade do cluster.

Pré-requisitos

Para o Amazon EKS escalar a implantação do CoreDNS, há três pré-requisitos:

- É necessário usar a versão do complemento do EKS do CoreDNS.
- Seu cluster dove executar pelo menos as versões mínimas do cluster e das versões da plataforma.
- Seu cluster dove executar pelo menos a versão mínima do complemento do EKS do CoreDNS.

Versão mínima do cluster

O ajuste de escala automático do CoreDNS é feito por um novo componente no ambiente de gerenciamento do cluster, gerenciado pelo Amazon EKS. Por isso, você deve atualizar seu cluster para uma versão do EKS que ofereça suporte à versão mínima da plataforma que tem o novo componente.

Um novo cluster do Amazon EKS. Para implantar, consulte Conceitos básicos. O cluster dove executar uma das versões do Kubernetes e as versões da plataforma listadas na tabela a seguir, ou uma versão mais recente. Observe que qualquer versão da plataforma e do Kubernetes posterior às versões listadas também é compatível. Para verificar a versão atual do Kubernetes, substitua *my-cluster* no comando a seguir pelo nome do seu cluster e execute o comando modificado:

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Versão do Kubernetes	Versão da plataforma
Não listado	Todas as versões da plataforma
1.29.3	eks.7
1.28.8	eks.13
1.27.12	eks.17
1.26.15	eks.18

> **Note**
>
> Todas as versões da plataforma ou versões mais recentes do Kubernetes também são compatíveis, por exemplo, a versão 1.30 do Kubernetes, da eks.1 em diante.

Versão mínima do complemento do EKS

Versão do Kubernetes	1.29	1.28
	`v1.11.1-eksbuild.9`	`v1.10.1-eksbuild.11`

Configurar o ajuste de escala automático do CoreDNS no Console de gerenciamento da AWS

1. Certifique-se de que a versão do seu cluster seja igual ou superior à versão mínima do cluster.

   O Amazon EKS atualiza automaticamente os clusters entre versões da plataforma da mesma versão do Kubernetes, e você não pode iniciar esse processo sozinho. Em vez disso, você pode atualizar o cluster para a próxima versão do Kubernetes, e o cluster será atualizado para essa versão K8s e a versão mais recente da plataforma.

   Novas versões do Kubernetes às vezes introduzem alterações significativas. Portanto, recomendamos testar o comportamento das aplicações usando um cluster separado da nova versão do Kubernetes antes de atualizar os clusters de produção.

   Para atualizar um cluster para uma nova versão do Kubernetes, siga o procedimento em Atualizar o cluster existente para uma nova versão do Kubernetes.
2. Certifique-se de ter o complemento do EKS para CoreDNS, e não a implantação autogerenciada do CoreDNS.

   Dependendo da ferramenta com a qual você criou o cluster, talvez você não tenha o tipo de complemento do Amazon EKS instalado em seu cluster atualmente. Para ver qual tipo de complemento está instalado no cluster, execute o comando a seguir. Substitua o `my-cluster` pelo nome do cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se um número de versão for devolvido, você terá o tipo de complemento do Amazon EKS instalado no cluster e poderá prosseguir para a próxima etapa. Se um erro for retornado, você não tem o tipo de complemento do Amazon EKS instalado no cluster. Conclua as etapas restantes do procedimento Criar o complemento CoreDNS do Amazon EKS para substituir a versão autogerenciada pelo complemento do Amazon EKS.

3. Certifique-se de que o complemento do EKS para CoreDNS seja de uma versão igual ou superior à versão mínima do complemento do EKS.

   Veja qual versão do complemento está atualmente instalada no cluster. É possível verificá-la no Console de gerenciamento da AWS ou executando o seguinte comando:

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   Veja abaixo um exemplo de saída.

   ```
   v1.10.1-eksbuild.13
   ```

   Compare esta versão à versão mínima do complemento do EKS na seção anterior. Se necessário, atualize o complemento do EKS para uma versão superior seguindo o procedimento Atualizar o complemento CoreDNS do Amazon EKS.

4. Adicione a configuração de autoescalabilidade às Configurações opcionais do complemento do EKS.

   a. Abra o console do Amazon EKS.

   b. No painel de navegação esquerdo, selecione Clusters e depois o nome do cluster para o qual você deseja configurar o complemento.

   c. Escolha a guia Complementos.

   d. Selecione a caixa no canto superior direito da caixa do complemento CoreDNS e depois escolha Editar.

   e. Na página Configurar CoreDNS:

      i. Selecione a Versão que você deseja usar. Recomendamos que você mantenha a mesma versão da etapa anterior, e atualize a mesma e a sua configuração em ações separadas.

      ii. Expanda Definições de configuração opcionais.

      iii. Insira a chave JSON `"autoscaling":` e o valor de um objeto JSON aninhado com uma chave `"enabled":` e o valor `true` em Valores de configuração. O texto resultante deve ser um objeto JSON válido. Se esse par de chave e valor for o único dado na caixa de texto, coloque-o entre colchetes `{ }`. O seguinte exemplo mostra a autoescalabilidade habilitada:

      ```
      {
        "autoScaling": {
          "enabled": true
      ```

```
  }
}
```

iv. (Opcional) Você pode fornecer valores mínimos e máximos para os quais o ajuste de escala automático pode escalar o número de pods do CoreDNS.

O exemplo a seguir mostra que a autoescalabilidade está habilitada e todas as chaves opcionais têm valores. Recomendamos que o número mínimo de pods do CoreDNS seja sempre maior que dois para fornecer resiliência ao serviço de DNS no cluster.

```
{
  "autoScaling": {
    "enabled": true,
    "minReplicas": 2,
    "maxReplicas": 10
  }
}
```

f. Para aplicar a nova configuração substituindo os pods do CoreDNS, escolha Salvar alterações.

O Amazon EKS aplica alterações nos complementos do EKS usando uma distribuição da implantação do Kubernetes para CoreDNS. É possível acompanhar o status da implantação no Histórico de atualizações do complemento no Console de gerenciamento da AWS e com `kubectl rollout status deployment/coredns --namespace kube-system`.

`kubectl rollout` oferece os seguintes comandos:

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Se a distribuição demorar muito, o Amazon EKS desfará a distribuição e uma mensagem com o tipo Atualização do complemento e o status Falha será adicionada ao Histórico de atualizações do complemento. Para investigar qualquer problema, comece com o histórico da distribuição e execute `kubectl logs` em um pod do CoreDNS para ver os logs do CoreDNS.

5. Se a nova entrada no Histórico de atualizações tiver o status de Êxito, então a distribuição foi concluída e o complemento está usando a nova configuração em todos os pods do CoreDNS. Conforme você altera o número de nós e os núcleos de CPU dos nós no cluster, o Amazon EKS escala o número de réplicas da implantação do CoreDNS.

Configurar o ajuste de escala automático do CoreDNS na AWS Command Line Interface

1. Certifique-se de que a versão do seu cluster seja igual ou superior à versão mínima do cluster.

   O Amazon EKS atualiza automaticamente os clusters entre versões da plataforma da mesma versão do Kubernetes, e você não pode iniciar esse processo sozinho. Em vez disso, você pode atualizar o cluster para a próxima versão do Kubernetes, e o cluster será atualizado para essa versão K8s e a versão mais recente da plataforma.

   Novas versões do Kubernetes às vezes introduzem alterações significativas. Portanto, recomendamos testar o comportamento das aplicações usando um cluster separado da nova versão do Kubernetes antes de atualizar os clusters de produção.

   Para atualizar um cluster para uma nova versão do Kubernetes, siga o procedimento em Atualizar o cluster existente para uma nova versão do Kubernetes.
2. Certifique-se de ter o complemento do EKS para CoreDNS, e não a implantação autogerenciada do CoreDNS.

   Dependendo da ferramenta com a qual você criou o cluster, talvez você não tenha o tipo de complemento do Amazon EKS instalado em seu cluster atualmente. Para ver qual tipo de complemento está instalado no cluster, execute o comando a seguir. Substitua o `my-cluster` pelo nome do cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se receber um número de versão, você tem o tipo de complemento do Amazon EKS instalado no cluster. Se um erro for retornado, você não tem o tipo de complemento do Amazon EKS instalado no cluster. Conclua as etapas restantes do procedimento Criar o complemento CoreDNS do Amazon EKS para substituir a versão autogerenciada pelo complemento do Amazon EKS.
3. Certifique-se de que o complemento do EKS para CoreDNS seja de uma versão igual ou superior à versão mínima do complemento do EKS.

Veja qual versão do complemento está atualmente instalada no cluster. É possível verificá-la no Console de gerenciamento da AWS ou executando o seguinte comando:

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
d : -f 3
```

Veja abaixo um exemplo de saída.

```
v1.10.1-eksbuild.13
```

Compare esta versão à versão mínima do complemento do EKS na seção anterior. Se necessário, atualize o complemento do EKS para uma versão superior seguindo o procedimento Atualizar o complemento CoreDNS do Amazon EKS.

4. Adicione a configuração de autoescalabilidade às Configurações opcionais do complemento do EKS.

   Execute o seguinte comando da AWS CLI. Substitua `my-cluster` pelo nome do cluster e o ARN do perfil do IAM pelo perfil que você está usando.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true}}'
   ```

   O Amazon EKS aplica alterações nos complementos do EKS usando uma distribuição da implantação do Kubernetes para CoreDNS. É possível acompanhar o status da implantação no Histórico de atualizações do complemento no Console de gerenciamento da AWS e com `kubectl rollout status deployment/coredns --namespace kube-system`.

   `kubectl rollout` oferece os seguintes comandos:

   ```
   kubectl rollout

   history  -- View rollout history
   pause    -- Mark the provided resource as paused
   restart  -- Restart a resource
   resume   -- Resume a paused resource
   status   -- Show the status of the rollout
   undo     -- Undo a previous rollout
   ```

Se a distribuição demorar muito, o Amazon EKS desfará a distribuição e uma mensagem com o tipo Atualização do complemento e o status Falha será adicionada ao Histórico de atualizações do complemento. Para investigar qualquer problema, comece com o histórico da distribuição e execute `kubectl logs` em um pod do CoreDNS para ver os logs do CoreDNS.

5. (Opcional) Você pode fornecer valores mínimos e máximos para os quais o ajuste de escala automático pode escalar o número de pods do CoreDNS.

   O exemplo a seguir mostra que a autoescalabilidade está habilitada e todas as chaves opcionais têm valores. Recomendamos que o número mínimo de pods do CoreDNS seja sempre maior que dois para fornecer resiliência ao serviço de DNS no cluster.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true,"minReplicas":2,"maxReplicas":10}}'
   ```

6. Verifique o status da atualização do complemento executando o seguinte comando:

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns
   ```

   Se você vir esta linha: `"status": "ACTIVE"`, a distribuição foi concluída e o complemento está usando a nova configuração em todos os pods do CoreDNS. Conforme você altera o número de nós e os núcleos de CPU dos nós no cluster, o Amazon EKS escala o número de réplicas da implantação do CoreDNS.

Monitorar a resolução de DNS do Kubernetes com métricas do CoreDNS

O CoreDNS como um complemento do EKS expõe as métricas do CoreDNS na porta `9153` no formato do Prometheus no serviço `kube-dns`. É possível usar o Prometheus, o Amazon CloudWatch Agent ou qualquer outro sistema compatível para extrair (coletar) essas métricas.

Para ver um exemplo de configuração de extração compatível com o Prometheus e com o agente do CloudWatch, consulte Configuração do agente do CloudWatch para Prometheus no Guia do usuário do Amazon CloudWatch.

Gerenciar o **kube-proxy** em clusters do Amazon EKS

> **Tip**
>
> Com o Modo automático do Amazon EKS, você não precisa instalar ou atualizar complementos de rede. O Modo automático inclui recursos de rede de pods e balanceamento de carga.
> Para obter mais informações, consulte Modo Automático do EKS.

Recomendamos adicionar o tipo Amazon EKS do complemento ao seu cluster em vez de usar o tipo autogerenciado do complemento. Se você não estiver familiarizado com a diferença entre os tipos, consulte the section called “Complementos do Amazon EKS”. Para obter mais informações sobre como adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criação de um complemento”. Se você não conseguir usar o complemento do Amazon EKS, recomendamos que você envie um problema sobre o motivo pelo qual não pode usar o repositório GitHub para roteiro de contêineres.

O complemento `kube-proxy` é implantado em cada nó do Amazon EC2 em seu cluster do Amazon EKS. Ele mantém as regras de rede em seus nós e permite a comunicação de rede com seus Pods. O complemento não é implantado em nós do Fargate em seu cluster. Para obter mais informações, consulte kubelet na documentação do Kubernetes.

Instalar o complemento do Amazon EKS

kube-proxy Versões do

A tabela a seguir lista a versão mais recente do tipo de complemento do Amazon EKS para cada versão do Kubernetes.

Versão do Kubernetes	**kube-proxy** Versão do
1.34	v1.34.1-eksbuild.2
1.33	v1.33.5-eksbuild.2
1.32	v1.32.9-eksbuild.2
1.31	v1.31.13-eksbuild.2

Versão do Kubernetes	**kube-proxy** Versão do
1.30	v1.30.14-eksbuild.8
1.29	v1.29.15-eksbuild.16
1.28	v1.28.15-eksbuild.31

> **Note**
>
> Uma versão anterior da documentação estava incorreta. As versões `v1.28.5`, `v1.27.9` e `v1.26.12` do `kube-proxy` não estão disponíveis.
> Se você estiver gerenciando esse complemento automaticamente, as versões na tabela poderão não ser as mesmas que as versões autogerenciadas disponíveis.

kube-proxy Imagem do contêiner do

A imagem de contêiner do `kube-proxy` é baseada em uma imagem de base mínima mantida pelo Amazon EKS Distro, que contém pacotes mínimos e não tem shells. Para obter mais informações, consulte Amazon EKS Distro.

A tabela a seguir lista a versão mais recente da imagem de contêiner autogerenciado do `kube-proxy` disponível para cada versão de cluster do Amazon EKS.

Versão	kube-proxy
1.34	v1.34.1-eksbuild.2
1.33	v1.33.5-minimal-eksbuild.2
1.32	v1.32.9-minimal-eksbuild.2
1.31	v1.31.13-minimal-eksbuild.2
1.30	v1.30.14-minimal-eksbuild.8
1.29	v1.29.15-minimal-eksbuild.16

Versão	kube-proxy
1.28	v1.28.15-minimal-eksbuild.31

Ao atualizar um tipo de complemento do Amazon EKS, você especifica uma versão válida do complemento do Amazon EKS, que pode não ser uma versão listada nesta tabela. Isso ocorre porque as versões dos complementos do Amazon EKS nem sempre correspondem às versões de imagem de contêiner especificadas ao atualizar o tipo autogerenciado desse complemento. Ao atualizar o tipo autogerenciado desse complemento, você especifica uma versão de imagem de contêiner válida listada nesta tabela.

Atualizar o complemento autogerenciado **`kube-proxy`** do Kubernetes

> **Important**
>
> Recomendamos adicionar o tipo Amazon EKS do complemento ao seu cluster em vez de usar o tipo autogerenciado do complemento. Se você não estiver familiarizado com a diferença entre os tipos, consulte the section called “Complementos do Amazon EKS”. Para obter mais informações sobre como adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Criação de um complemento”. Se você não conseguir usar o complemento do Amazon EKS, recomendamos que você envie um problema sobre o motivo pelo qual não pode usar o repositório GitHub para roteiro de contêineres.

Pré-requisitos

- Um cluster existente do Amazon EKS. Para implantar, consulte Conceitos básicos.

Considerações

- `Kube-proxy` em um cluster do Amazon EKS tem a mesma política de compatibilidade e inclinação do Kubernetes. Saiba como Verificar a compatibilidade da versão do complemento do Amazon EKS com um cluster.

  1. Confirme que tem o tipo autogerenciado de complemento instalado em seu cluster. Substitua *`my-cluster`* pelo nome do cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
 addon.addonVersion --output text
```

Se receber uma mensagem de erro, você tem o tipo autogerenciado do complemento instalado no cluster. As etapas restantes desse tópico servem para atualizar o tipo autogerenciado do complemento. Se receber um número de versão, você tem o tipo de complemento do Amazon EKS instalado no cluster. Para atualizá-lo, use o procedimento em Atualização de um complemento do Amazon EKS, em vez de usar o procedimento deste tópico. Se não estiver familiarizado com a diferença entre os tipos de complemento, consulte the section called "Complementos do Amazon EKS".

2. Veja qual versão da imagem do contêiner está atualmente instalada em seu cluster.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image
```

Veja um exemplo de saída abaixo.

```
Image:    602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.29.1-
eksbuild.2
```

No resultado de exemplo, *v1.29.1-eksbuild.2* é a versão instalada no cluster.

3. Atualize o complemento `kube-proxy` substituindo *602401143452* e *region-code* pelos valores da sua saída retornada na etapa anterior. Substitua *v1.30.6-eksbuild.3* pela versão do `kube-proxy` listada na tabela Versão de imagem mais recente do contêiner autogerenciado para cada versão de cluster do Amazon EKS.

> **Important**
>
> Os manifestos para cada tipo de imagem são diferentes e não compatíveis entre os tipos de imagem padrão ou mínimo. Você deve usar o mesmo tipo de imagem da imagem anterior, para que o ponto de entrada e os argumentos correspondam.

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-
proxy=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.30.6-
eksbuild.3
```

Veja um exemplo de saída abaixo.

```
daemonset.apps/kube-proxy image updated
```

4. Confirme se agora a nova versão está instalada em seu cluster.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

Veja um exemplo de saída abaixo.

```
v1.30.0-eksbuild.3
```

5. Se você estiver usando os nós `x86` e `Arm` no mesmo cluster e o cluster tiver sido implantado antes de 17 de agosto de 2020. Em seguida, edite seu manifesto `kube-proxy` para incluir um seletor de nó para várias arquiteturas de hardware com o seguinte comando. É uma operação única. Depois que você adicionar o seletor ao manifesto, não precisará fazer isso toda vez que atualizar o complemento. Se o cluster foi implantado em ou após 17 de agosto de 2020, `kube-proxy` já é capaz de multiarquitetura.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Adicione o seletor de nós a seguir ao arquivo no editor e, em seguida, salve o arquivo.
Para obter um exemplo de onde incluir este texto no editor, consulte o arquivo CNI manifest (Manifesto do CNI) no GitHub Isso permite que o Kubernetes extraia a imagem de hardware correta com base na arquitetura de hardware do nó.

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

6. Se o cluster foi originalmente criado com a versão `1.14` ou mais recente do Kubernetes, você pode ignorar esta etapa porque o `kube-proxy` já inclui essa `Affinity Rule`. Se você criou originalmente um cluster do Amazon EKS com a versão `1.13` ou anterior do Kubernetes e quiser usar nós do Fargate no cluster, edite o manifesto do `kube-proxy` para incluir uma regra de `NodeAffinity` para impedir que os pods do `kube-proxy` façam agendamentos em nós do Fargate. Esta é uma edição ocasional. Depois que você adicionar o `Affinity Rule` ao

manifesto, não precisará fazer isso toda vez que atualizar o complemento. Edite o DaemonSet do `kube-proxy`.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Adicione a `Affinity Rule` a seguir à seção `spec` do DaemonSet do arquivo no editor e, em seguida, salve o arquivo. Para obter um exemplo de onde incluir este texto no editor, consulte o arquivo CNI manifest (Manifesto do CNI) no GitHub

```
- key: eks.amazonaws.com/compute-type
  operator: NotIn
  values:
  - fargate
```

Saiba como implantar workloads e complementos no Amazon EKS

As workloads são implantadas em contêineres, que são implantados em pods no Kubernetes. Um pod inclui um ou mais contêineres. Normalmente, um ou mais pods que fornecem o mesmo serviço são implantados em um serviço do Kubernetes. Depois de implantar vários pods que fornecem o mesmo serviço, você pode:

- Visualizar informações sobre as workloads executadas em cada um dos clusters usando o Console de gerenciamento da AWS.
- Aumentar ou reduzir a escala verticalmente dos pods com o Vertical Pod Autoscaler.
- Aumentar ou reduzir a escala horizontalmente do número de pods necessários para atender à demanda com o Horizontal Pod Autoscaler.
- Criar um balanceador de carga da rede externo (para pods acessíveis pela internet) ou um interno (para pods privados) a fim de equilibrar o tráfego de rede entre pods. O balanceador de carga direciona o tráfego na Camada 4 do modelo OSI.
- Crie um Application Load Balancer para equilibrar o tráfego de aplicações entre pods. O balanceador de carga de aplicações direciona o tráfego na Camada 7 do modelo OSI.
- Se você for novo no Kubernetes, este tópico ajudará você a implantar uma aplicação de amostra.
- Você pode restringir endereços IP que podem ser atribuídos a um serviço com `externalIPs`.

Implantar uma aplicação de amostra no Linux

Neste tópico, você implementa uma aplicação de amostra em seu cluster em nós Linux.

Pré-requisitos

- Um cluster existente do Kubernetes com pelo menos um nó. Se você não tiver um cluster do Amazon EKS existente, poderá implantar um usando um dos guias em Conceitos básicos.
- O `Kubectl` instalado em seu computador. Para ter mais informações, consulte the section called “Configuração do kubectl e eksctl”.
- O `Kubectl` configurado para se comunicar com o cluster. Para ter mais informações, consulte the section called “Acessar meu cluster com o kubectl”.

- Se você planeja implantar sua workload de exemplo no Fargate, deverá ter um perfil do Fargate que inclua o mesmo namespace criado neste tutorial, que será `eks-sample-app`, a menos que você altere o nome. Se você criou um cluster com um dos grupos no Conceitos básicos, será necessário criar um novo perfil ou adicionar o namespace ao perfil existente, pois o perfil criado nos guias de introdução não especifica o namespace usado neste tutorial. A VPC também deve conter pelo menos uma sub-rede privada.

Embora muitas variáveis possam ser alteradas nas etapas a seguir, recomendamos alterar apenas os valores das variáveis quando especificado. Depois de entender melhor os pods, as implantações e os serviços do Kubernetes, você pode experimentar alterar outros valores.

Crie um namespace

Um namespace permite agrupar recursos no Kubernetes. Para obter mais informações, consulte Namespaces na documentação do Kubernetes. Se você planeja implantar o exemplo de aplicação para Simplificar o gerenciamento de computação com o AWS Fargate, certifique-se de que o valor de `namespace` em Definir quais pods usam o AWS Fargate quando iniciados seja `eks-sample-app`.

```
kubectl create namespace eks-sample-app
```

Crie uma implantação do Kubernetes

Crie uma implantação do Kubernetes. Essa implantação de exemplo extrai uma imagem de contêiner de um repositório público e implanta três réplicas (pods individuais) dessa imagem em seu cluster. Para saber mais, consulte Implantações na documentação do Kubernetes.

1. Salve o conteúdo a seguir em um arquivo denominado `eks-sample-deployment.yaml`. Os contêineres da aplicação de exemplo não usam armazenamento em rede, mas pode haver aplicações que precisem. Para ter mais informações, consulte Armazenamento de dados de aplicações.
   - Os `values` de `amd64` ou `arm64` na chave `kubernetes.io/arch` significam que a aplicação pode ser implantada em qualquer arquitetura de hardware (se ambos estiverem no cluster). Isso é possível porque essa imagem é uma imagem multiarquitetura, mas nem todas são. É possível determinar a arquitetura de hardware em que há suporte para a imagem visualizando os detalhes da imagem no repositório do qual você está extraindo. Ao implantar imagens que não oferecem suporte a um tipo de arquitetura de hardware ou quando você não deseja que a

imagem seja implantada, remova esse tipo do manifesto. Para obter mais informações, consulte Rótulos, anotações e taints conhecidos na documentação do Kubernetes.

- `kubernetes.io/os: linux` `nodeSelector` significa que se você tivesse nós do Linux e do Windows (por exemplo) no cluster, a imagem só seria implantada nos nós do Linux. Para obter mais informações, consulte Rótulos, anotações e taints conhecidos na documentação do Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-linux-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-linux-app
  template:
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
      nodeSelector:
```

```
        kubernetes.io/os: linux
```

2. Aplique o manifesto de implantação ao seu cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

Criar um serviço

Um serviço permite que você acesse todas as réplicas por meio de um único endereço IP ou nome. Para obter mais informações, consulte Serviço na documentação do Kubernetes. Embora não esteja implementado na aplicação de exemplo, se houver aplicações que precisem interagir com outros serviços da AWS, recomendamos criar contas de serviço do Kubernetes para seus pods e associá-las a contas do AWS IAM. Ao especificar contas de serviço, seus pods podem ter apenas as permissões mínimas especificadas para elas para interagir com outros serviços. Para ter mais informações, consulte the section called “Credenciais com IRSA”.

1. Salve o conteúdo a seguir em um arquivo denominado `eks-sample-service.yaml`. O Kubernetes atribui ao serviço o seu próprio endereço IP acessível somente de dentro do cluster. Para acessar o serviço de fora do seu cluster, implante o AWS Load Balancer Controller para balancear a carga do tráfego de aplicações ou redes para o serviço.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-linux-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  selector:
    app: eks-sample-linux-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. Aplique o manifesto de serviço ao seu cluster.

```
kubectl apply -f eks-sample-service.yaml
```

Revisar os recursos criados

1. Visualize todos os recursos no namespace `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

Veja um exemplo de saída abaixo.

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/eks-sample-linux-deployment-65b7669776-m6qxz   1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-mmxvd   1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-qzn22   1/1     Running   0          27m

NAME                                 TYPE         CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
service/eks-sample-linux-service   ClusterIP    10.100.74.8     <none>        80/TCP
   32m

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-linux-deployment 3/3     3            3           27m

NAME                                                      DESIRED   CURRENT   READY
 AGE
replicaset.apps/eks-sample-linux-deployment-776d8f8fd8    3         3         3
 27m
```

Na saída, você verá o serviço e a implantação especificados nos manifestos de exemplo implantados nas etapas anteriores. Você também vê três pods. Isso ocorre porque 3 `replicas` foram especificadas no manifesto de exemplo. Para obter mais informações sobre pods, consulte Pods na documentação do Kubernetes. O Kubernetes cria automaticamente o recurso `replicaset`, mesmo que não esteja especificado nos manifestos de exemplo. Para obter mais informações sobre `ReplicaSets`, consulte ReplicaSet na documentação do Kubernetes.

> **Note**
>
> O Kubernetes mantém o número de réplicas especificado no manifesto. Se esta fosse uma implantação de produção e você quisesse que o Kubernetes escalasse horizontalmente o número de réplicas ou verticalmente os recursos de computação para

os pods, você usaria as implantações de pods de escala com o Horizontal Pod Autoscaler e os recursos de pods de ajuste com o Vertical Pod Autoscaler para fazer isso.

2. Veja os detalhes do serviço implantado.

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

Veja um exemplo de saída abaixo.

```
Name:              eks-sample-linux-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-linux-app
Annotations:       <none>
Selector:          app=eks-sample-linux-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

Na saída anterior, o valor de `IP:` é um endereço IP exclusivo que pode ser acessado de qualquer nó ou pod dentro do cluster, mas não pode ser acessado de fora do cluster. Os valores de `Endpoints` são endereços IP atribuídos de dentro da VPC aos pods que fazem parte do serviço.

3. Visualize os detalhes de um dos pods listados na saída quando você visualizou o namespace em uma etapa anterior. Substitua *`776d8f8fd8-78w66`* pelo valor retornado para um dos seus pods.

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

Exemplo abreviado de saída

```
Name:         eks-sample-linux-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
```

```
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-linux-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-linux-deployment-65b7669776-m6qxz to
 ip-192-168-45-132.us-west-2.compute.internal
[...]
```

Na saída anterior, o valor de `IP:` é um IP exclusivo atribuído ao pod do bloco CIDR atribuído à sub-rede em que o nó está localizado. Se preferir atribuir a pods endereços IP de blocos CIDR diferentes, você pode alterar o comportamento padrão. Para ter mais informações, consulte the section called “Redes personalizadas”. Você também pode ver que o agendador do Kubernetes agendou o pod no `Node` com o endereço IP *`192.168.45.132`*.

> **Tip**
>
> Em vez de usar a linha de comando, você pode visualizar muitos detalhes dos pods, serviços, implantações e outros recursos do Kubernetes no Console de gerenciamento da AWS. Para ter mais informações, consulte the section called “Acessar recursos do cluster”.

Executar um shell em um pod

1. Execute um shell no pod que você descreveu na etapa anterior, substituindo *`65b7669776-m6qxz`* pelo ID de um de seus pods.

```
kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app -- /
bin/bash
```

2. No shell do pod, visualize a saída do servidor da Web instalado com a sua implantação em uma etapa anterior. Basta especificar o nome do serviço. Ele é resolvido para o endereço IP do serviço pelo CoreDNS, que é implantado com um cluster do Amazon EKS, por padrão.

   ```
   curl eks-sample-linux-service
   ```

   Veja um exemplo de saída abaixo.

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```

3. No shell do pod, exiba o servidor DNS do pod.

   ```
   cat /etc/resolv.conf
   ```

   Veja um exemplo de saída abaixo.

   ```
   nameserver 10.100.0.10
   search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local us-
   west-2.compute.internal
   options ndots:5
   ```

   Na saída anterior, `10.100.0.10` é atribuído automaticamente como `nameserver` para todos os pods implantados no cluster.

4. Desconecte-se do pod digitando `exit`.

5. Ao terminar de usar a aplicação de exemplo, você poderá remover o namespace, o serviço e a implantação de exemplo com o comando a seguir.

   ```
   kubectl delete namespace eks-sample-app
   ```

Próximas etapas

Após implantar a aplicação de exemplo, talvez você queira tentar alguns dos seguintes exercícios:

- Roteamento de aplicações e tráfego HTTP com Application Load Balancers
- Roteamento de tráfego TCP e UDP com Network Load Balancers

Implantar uma aplicação de amostra no Windows

Neste tópico, você implementa uma aplicação de amostra em seu cluster nos nós do Windows.

Pré-requisitos

- Um cluster existente do Kubernetes com pelo menos um nó. Se você não tiver um cluster do Amazon EKS existente, poderá implantar um usando um dos guias em Conceitos básicos. Você deve ter o suporte ao Windows habilitado para o cluster e pelo menos um nó Windows para Amazon EC2.
- O `Kubectl` instalado em seu computador. Para obter mais informações, consulte the section called “Configuração do kubectl e eksctl”.
- O `Kubectl` configurado para se comunicar com o cluster. Para obter mais informações, consulte the section called “Acessar meu cluster com o kubectl”.
- Se você planeja implantar sua workload de exemplo no Fargate, deverá ter um perfil do Fargate que inclua o mesmo namespace criado neste tutorial, que será `eks-sample-app`, a menos que você altere o nome. Se você criou um cluster com um dos grupos no Conceitos básicos, será necessário criar um novo perfil ou adicionar o namespace ao perfil existente, pois o perfil criado nos guias de introdução não especifica o namespace usado neste tutorial. A VPC também deve conter pelo menos uma sub-rede privada.

Embora muitas variáveis possam ser alteradas nas etapas a seguir, recomendamos alterar apenas os valores das variáveis quando especificado. Depois de entender melhor os pods, as implantações e os serviços do Kubernetes, você pode experimentar alterar outros valores.

Crie um namespace

Um namespace permite agrupar recursos no Kubernetes. Para obter mais informações, consulte Namespaces na documentação do Kubernetes. Se você planeja implantar o exemplo de aplicação

para Simplificar o gerenciamento de computação com o AWS Fargate, certifique-se de que o valor de `namespace` em Definir quais pods usam o AWS Fargate quando iniciados seja `eks-sample-app`.

```
kubectl create namespace eks-sample-app
```

Crie uma implantação do Kubernetes

Essa implantação de exemplo extrai uma imagem de contêiner de um repositório público e implanta três réplicas (pods individuais) dessa imagem em seu cluster. Para saber mais, consulte Implantações na documentação do Kubernetes.

1. Salve o conteúdo a seguir em um arquivo denominado `eks-sample-deployment.yaml`. Os contêineres da aplicação de exemplo não usam armazenamento em rede, mas pode haver aplicações que precisem. Para obter mais informações, consulte Armazenamento de dados de aplicações.
   - `kubernetes.io/os: windows` `nodeSelector` significa que se você tivesse nós do Windows e do Linux (por exemplo) no cluster, a imagem só seria implantada nos nós do Windows. Para obter mais informações, consulte Rótulos, anotações e taints conhecidos na documentação do Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-windows-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-windows-app
  template:
    metadata:
      labels:
        app: eks-sample-windows-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
```

```
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
      containers:
      - name: windows-server-iis
        image: mcr.microsoft.com/windows/servercore:ltsc2019
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
        command:
        - powershell.exe
        - -command
        - "Add-WindowsFeature Web-Server; Invoke-WebRequest -UseBasicParsing
 -Uri 'https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.6/
ServiceMonitor.exe' -OutFile 'C:\\ServiceMonitor.exe'; echo '<html><body><br/
><br/><marquee><H1>Hello EKS!!!<H1><marquee></body><html>' > C:\\inetpub\\wwwroot\
\default.html; C:\\ServiceMonitor.exe 'w3svc'; "
      nodeSelector:
        kubernetes.io/os: windows
```

2. Aplique o manifesto de implantação ao seu cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

Criar um serviço

Um serviço permite que você acesse todas as réplicas por meio de um único endereço IP ou nome. Para obter mais informações, consulte Serviço na documentação do Kubernetes. Embora não esteja implementado na aplicação de exemplo, se houver aplicações que precisem interagir com outros serviços da AWS, recomendamos criar contas de serviço do Kubernetes para seus pods e associá-las a contas do AWS IAM. Ao especificar contas de serviço, seus pods podem ter apenas as permissões mínimas especificadas para elas para interagir com outros serviços. Para obter mais informações, consulte the section called “Credenciais com IRSA”.

1. Salve o conteúdo a seguir em um arquivo denominado `eks-sample-service.yaml`. O Kubernetes atribui ao serviço o seu próprio endereço IP acessível somente de dentro do cluster.

Para acessar o serviço de fora do seu cluster, implante o AWS Load Balancer Controller para balancear a carga do tráfego de aplicações ou redes para o serviço.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-windows-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  selector:
    app: eks-sample-windows-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. Aplique o manifesto de serviço ao seu cluster.

```
kubectl apply -f eks-sample-service.yaml
```

Revisar os recursos criados

1. Visualize todos os recursos no namespace `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

Veja um exemplo de saída abaixo.

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/eks-sample-windows-deployment-65b7669776-m6qxz   1/1     Running   0          27m
pod/eks-sample-windows-deployment-65b7669776-mmxvd   1/1     Running   0          27m
pod/eks-sample-windows-deployment-65b7669776-qzn22   1/1     Running   0          27m

NAME                                 TYPE         CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
service/eks-sample-windows-service   ClusterIP    10.100.74.8     <none>        80/
TCP    32m
```

```
NAME                                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-windows-deployment 3/3     3            3           27m

NAME                                                           DESIRED   CURRENT   READY
 AGE
replicaset.apps/eks-sample-windows-deployment-776d8f8fd8    3         3         3
   27m
```

Na saída, você verá o serviço e a implantação especificados nos manifestos de exemplo implantados nas etapas anteriores. Você também vê três pods. Isso ocorre porque 3 `replicas` foram especificadas no manifesto de exemplo. Para obter mais informações sobre pods, consulte Pods na documentação do Kubernetes. O Kubernetes cria automaticamente o recurso `replicaset`, mesmo que não esteja especificado nos manifestos de exemplo. Para obter mais informações sobre `ReplicaSets`, consulte ReplicaSet na documentação do Kubernetes.

> **Note**
>
> O Kubernetes mantém o número de réplicas especificado no manifesto. Se esta fosse uma implantação de produção e você quisesse que o Kubernetes escalasse horizontalmente o número de réplicas ou verticalmente os recursos de computação para os pods, você usaria as implantações de pods de escala com o Horizontal Pod Autoscaler e os recursos de pods de ajuste com o Vertical Pod Autoscaler para fazer isso.

2. Veja os detalhes do serviço implantado.

```
kubectl -n eks-sample-app describe service eks-sample-windows-service
```

Veja um exemplo de saída abaixo.

```
Name:              eks-sample-windows-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-windows-app
Annotations:       <none>
Selector:          app=eks-sample-windows-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
```

```
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

Na saída anterior, o valor de `IP:` é um endereço IP exclusivo que pode ser acessado de qualquer nó ou pod dentro do cluster, mas não pode ser acessado de fora do cluster. Os valores de `Endpoints` são endereços IP atribuídos de dentro da VPC aos pods que fazem parte do serviço.

3. Visualize os detalhes de um dos pods listados na saída quando você visualizou o namespace em uma etapa anterior. Substitua *776d8f8fd8-78w66* pelo valor retornado para um dos seus pods.

```
kubectl -n eks-sample-app describe pod eks-sample-windows-deployment-65b7669776-m6qxz
```

Exemplo abreviado de saída

```
Name:         eks-sample-windows-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-windows-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-windows-deployment-65b7669776-m6qxz
 to ip-192-168-45-132.us-west-2.compute.internal
[...]
```

Na saída anterior, o valor de `IP:` é um IP exclusivo atribuído ao pod do bloco CIDR atribuído à sub-rede em que o nó está localizado. Se preferir atribuir a pods endereços IP de blocos CIDR diferentes, você pode alterar o comportamento padrão. Para obter mais informações, consulte the section called “Redes personalizadas”. Você também pode ver que o agendador do Kubernetes agendou o pod no `Node` com o endereço IP *`192.168.45.132`*.

> **Tip**
>
> Em vez de usar a linha de comando, você pode visualizar muitos detalhes dos pods, serviços, implantações e outros recursos do Kubernetes no Console de gerenciamento da AWS. Para obter mais informações, consulte the section called “Acessar recursos do cluster”.

Executar um shell em um pod

1. Execute um shell no pod que você descreveu na etapa anterior, substituindo *`65b7669776-m6qxz`* pelo ID de um de seus pods.

```
kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-app --
 powershell.exe
```

2. No shell do pod, visualize a saída do servidor da Web instalado com a sua implantação em uma etapa anterior. Basta especificar o nome do serviço. Ele é resolvido para o endereço IP do serviço pelo CoreDNS, que é implantado com um cluster do Amazon EKS, por padrão.

```
Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
```

   Veja um exemplo de saída abaixo.

```
StatusCode        : 200
StatusDescription : OK
Content           : < h t m l > < b o d y > < b r / > < b r / > < m a r q u e e > < H
 1 > H e l l o
                        E K S ! ! ! < H 1 > < m a r q u e e > < / b o d y > < h t m l >
```

3. No shell do pod, exiba o servidor DNS do pod.

```
Get-NetIPConfiguration
```

Resultado abreviado

```
InterfaceAlias       : vEthernet
[...]
IPv4Address          : 192.168.63.14
[...]
DNSServer            : 10.100.0.10
```

Na saída anterior, `10.100.0.10` é atribuído automaticamente como o servidor de DNS para todos os pods implantados no cluster.

4. Desconecte-se do pod digitando `exit`.
5. Ao terminar de usar a aplicação de exemplo, você poderá remover o namespace, o serviço e a implantação de exemplo com o comando a seguir.

```
kubectl delete namespace eks-sample-app
```

Próximas etapas

Após implantar a aplicação de exemplo, talvez você queira tentar alguns dos seguintes exercícios:

- Roteamento de aplicações e tráfego HTTP com Application Load Balancers
- Roteamento de tráfego TCP e UDP com Network Load Balancers

Ajuste os recursos do pod com o Vertical Pod Autoscaler

O Vertical Pod Autoscaler do Kubernetes ajusta automaticamente as reservas de CPU e de memória para os pods, ajudando a "dimensionar corretamente" as aplicações. Esse ajuste pode melhorar a utilização de recursos do cluster e liberar CPU e memória para outros pods. Este tópico ajuda você a implantar o Vertical Pod Autoscaler no cluster e verificar se ele está funcionando.

- Você tem um cluster existente do Amazon EKS. Caso contrário, consulte Conceitos básicos.
- Você tem o servidor de métricas do Kubernetes instalado. Para ter mais informações, consulte the section called “Servidor de métricas”.

- Você estiver usando um cliente do `kubectl` que está configurado para se comunicar com o cluster do Amazon EKS.
- O OpenSSL `1.1.1` ou posterior instalado no seu dispositivo.

Implantar o Vertical Pod Autoscaler

Nesta seção, você implantará o Vertical Pod Autoscaler em seu cluster.

1. Abra uma janela do terminal e navegue até o diretório onde você deseja fazer download do código-fonte do Vertical Pod Autoscaler.
2. Clone o repositório kubernetes/autoscaler do GitHub.

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. Mude para o diretório `vertical-pod-autoscaler`.

```
cd autoscaler/vertical-pod-autoscaler/
```

4. (Opcional) Se você já tiver implantado outra versão do Vertical Pod Autoscaler, remova-a com o comando a seguir.

```
./hack/vpa-down.sh
```

5. Se os nós não tiverem acesso à Internet para o registro do contêiner `registry.k8s.io`, você precisará extrair as seguintes imagens e enviá-las para seu próprio repositório privado. Para obter mais informações sobre como extrair as imagens e enviá-las para seu próprio repositório privado, consulte the section called “Copiar uma imagem para um repositório”.

```
registry.k8s.io/autoscaling/vpa-admission-controller:0.10.0
registry.k8s.io/autoscaling/vpa-recommender:0.10.0
registry.k8s.io/autoscaling/vpa-updater:0.10.0
```

   Se você estiver enviando as imagens para um repositório privado do Amazon ECR, substitua `registry.k8s.io` nos manifestos pelo seu registro. Substitua *`111122223333`* pelo ID da sua conta. Substitua *`region-code`* pela região da AWS em que seu cluster se encontra. Para usar os comandos a seguir, você precisa ter nomeado seu repositório com o mesmo nome do repositório no manifesto. Se você nomeou o seu repositório como algo diferente, será necessário alterá-lo também.

```
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/recommender-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/updater-deployment.yaml
```

6. Implante o Vertical Pod Autoscaler no cluster com o comando a seguir.

```
./hack/vpa-up.sh
```

7. Verifique se os pods do Vertical Pod Autoscaler foram criados com êxito.

```
kubectl get pods -n kube-system
```

   Veja um exemplo de saída abaixo.

```
NAME                                        READY   STATUS    RESTARTS   AGE
[...]
metrics-server-8459fc497-kfj8w              1/1     Running   0          83m
vpa-admission-controller-68c748777d-ppspd   1/1     Running   0          7s
vpa-recommender-6fc8c67d85-gljpl            1/1     Running   0          8s
vpa-updater-786b96955c-bgp9d                1/1     Running   0          8s
```

Testar a instalação do Vertical Pod Autoscaler

Nesta seção, você implantará uma aplicação de exemplo para verificar se o Vertical Pod Autoscaler está funcionando.

1. Implante o exemplo `hamster.yaml` do Vertical Pod Autoscaler com o comando a seguir.

```
kubectl apply -f examples/hamster.yaml
```

2. Obtenha os pods da aplicação de exemplo do `hamster`.

```
kubectl get pods -l app=hamster
```

   Veja um exemplo de saída abaixo.

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. Descreva um dos pods para visualizar sua reserva de `cpu` e `memory`. Substitua *`c7d89d6db-rglf5`* por um dos IDs retornados na saída da etapa anterior.

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

Veja um exemplo de saída abaixo.

```
[...]
Containers:
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:35:16 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     50Mi
[...]
```

Você pode ver que o pod original reserva 100 milicpu de CPU e 50 mebibytes de memória. Para esta aplicação de exemplo, 100 milicpu é menos do que o pod precisa para ser executado, portanto, ele está restrito à CPU. Ele também reserva muito menos memória do que precisa. A implantação `vpa-recommender` do Vertical Pod Autoscaler analisa os pods do hamster para ver se os requisitos de CPU e memória são apropriados. Se ajustes forem necessários, o `vpa-updater` reiniciará os pods com valores atualizados.

4. Aguarde até que o `vpa-updater` inicie um novo pod do hamster. Isso deve levar um ou dois minutos. Você pode monitorar os pods com o comando a seguir.

> **Note**
>
> Se você não tiver certeza de que um novo pod foi iniciado, compare os nomes dos pod com a lista anterior. Quando o novo pod for iniciado, você verá um novo nome de pod.

```
kubectl get --watch Pods -l app=hamster
```

5. Quando um novo pod do hamster for iniciado, descreva-o e visualize as reservas atualizadas de CPU e memória.

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

Veja um exemplo de saída abaixo.

```
[...]
Containers:
  hamster:
    Container ID:
 docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:37:08 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        587m
      memory:     262144k
```

```
[...]
```

Na saída anterior, você pode ver que a reserva de `cpu` e aumentou para 587 milicpu, que é mais de cinco vezes o valor original. A `memory` aumentou para 262.144 Kilobytes, que é cerca de 250 mebibytes, ou cinco vezes o valor original. Esse pod estava com poucos recursos, e o Vertical Pod Autoscaler corrigiu a estimativa com um valor muito mais apropriado.

6. Descreva o recurso `hamster-vpa` para visualizar a nova recomendação.

```
kubectl describe vpa/hamster-vpa
```

Veja um exemplo de saída abaixo.

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/
v1beta2","kind":"VerticalPodAutoscaler","metadata":{"annotations":{},"name":"hamster-
vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:          23
  Resource Version:    14411
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/
verticalpodautoscalers/hamster-vpa
  UID:                 d0d85fb9-e153-11e9-ae53-0205785d75b0
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
  Conditions:
    Last Transition Time:  2019-09-27T18:23:28Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
```

```
      Lower Bound:
        Cpu:     550m
        Memory:  262144k
      Target:
        Cpu:     587m
        Memory:  262144k
      Uncapped Target:
        Cpu:     587m
        Memory:  262144k
      Upper Bound:
        Cpu:     21147m
        Memory:  387863636
Events:          <none>
```

7. Depois de experimentar com a aplicação de exemplo, exclua-a com os comandos a seguir.

```
kubectl delete -f examples/hamster.yaml
```

Escalar implantações de pods com o Horizontal Pod Autoscaler

O Horizontal Pod Autoscaler do Kubernetes escala automaticamente o número de pods em uma implantação, um controlador de replicação ou um conjunto de réplicas com base na utilização da CPU desse recurso. Isso pode ajudar a expandir as aplicações para atender ao aumento da demanda ou a reduzi-las quando os recursos não forem necessários, liberando os nós para outras aplicações. Quando você define uma porcentagem de utilização de CPU, o Horizontal Pod Autoscaler reduz ou expande a aplicação para tentar atingir essa meta.

O Horizontal Pod Autoscaler é um recurso da API padrão no Kubernetes que simplesmente requer que uma origem de métrica (como o servidor de métricas do Kubernetes) esteja instalada no cluster do Amazon EKS para funcionar. Você não precisa implantar ou instalar o Horizontal Pod Autoscaler no cluster para começar a escalar as aplicações. Para obter mais informações, consulte Horizontal Pod Autoscaler na documentação do Kubernetes.

Use este tópico para preparar o Horizontal Pod Autoscaler para o cluster do Amazon EKS e verificar se ele está funcionando com uma aplicação de exemplo.

> **Note**
>
> Este tópico se baseia em Horizontal Pod autoscaler walkthrough na documentação do Kubernetes.

- Você tem um cluster existente do Amazon EKS. Caso contrário, consulte Conceitos básicos.
- Você tem o servidor de métricas do Kubernetes instalado. Para ter mais informações, consulte the section called “Servidor de métricas”.
- Você estiver usando um cliente do `kubectl` que está configurado para se comunicar com o cluster do Amazon EKS.

Execute uma aplicação de teste do Horizontal Pod Autoscaler

Nesta seção, você implanta uma aplicação de exemplo para verificar se o Horizontal Pod Autoscaler está funcionando.

> **Note**
>
> Este exemplo é baseado em Horizontal Pod autoscaler walkthrough na documentação do Kubernetes.

1. Implante uma aplicação simples de servidor Web do Apache, com o comando a seguir.

   ```
   kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
   ```

   Este pod de servidor web do Apache recebe um limite de CPU de 500 milicpu e está distribuindo na porta 80.
2. Crie um recurso Horizontal Pod Autoscaler para a implantação do `php-apache`.

   ```
   kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
   ```

   Esse comando cria um autoscaler que visa 50% de utilização de CPU para a implantação, com um mínimo de um pod e um máximo de dez pods. Quando a carga média da CPU for inferior a 50%, o autoscaler tentará reduzir o número de pods na implantação para um mínimo

de um. Quando a carga for maior que 50%, o autoscaler tentará aumentar o número de pods na implantação, até um máximo de dez. Para obter mais informações, consulte How does the Horizontal Pod Autoscaler work? na documentação do Kubernetes.

3. Descreva o autoscaler com o seguinte comando para visualizar seus detalhes.

```
kubectl get hpa
```

Veja um exemplo de saída abaixo.

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   0%/50%    1         10        1          51s
```

Como você pode ver, a carga atual da CPU é `0%`, pois ainda não há nenhuma carga no servidor. A contagem de pod já está no limite mais baixo (um), portanto, não pode ter a escala reduzida horizontalmente.

4. Crie uma carga para o servidor web executando um contêiner.

```
kubectl run -i \
    --tty load-generator \
    --rm --image=busybox \
    --restart=Never \
    -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
```

5. Para observar a expansão da implantação, execute periodicamente o seguinte comando em um terminal separado do terminal em que você executou a etapa anterior.

```
kubectl get hpa php-apache
```

Veja um exemplo de saída abaixo.

```
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   250%/50%   1         10        5          4m44s
```

A contagem de réplicas pode levar mais de um minuto para aumentar. Enquanto a porcentagem real da CPU for maior do que a porcentagem de meta, a contagem de réplicas aumentará, até 10. Neste caso, é `250%`, então o número de `REPLICAS` continua a aumentar.

> **Note**
>
> Poderá demorar alguns minutos para você ver a contagem de réplicas atingir seu máximo. Se apenas 6 réplicas, por exemplo, forem necessárias para que a carga da CPU permaneça em 50% ou menos, a carga não escalará além de 6 réplicas.

6. Pare a carga. Na janela do terminal em que você está gerando a carga, interrompa a carga mantendo pressionadas as teclas `Ctrl+C`. Você pode observar a escala de réplicas voltar para 1 executando o comando a seguir novamente no terminal em que está observando a redução da escala na horizontal.

```
kubectl get hpa
```

   Veja um exemplo de saída abaixo.

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    0%/50%    1         10        1          25m
```

> **Note**
>
> O período de tempo padrão para reduzir a escala na vertical é de cinco minutos, portanto, levará algum tempo para você ver a contagem de réplicas chegar a 1 novamente, até mesmo quando a percentagem de CPU atual for 0%. O período de tempo é modificável. Para obter mais informações, consulte Horizontal Pod Autoscaler na documentação do Kubernetes.

7. Ao concluir os testes da aplicação de exemplo, exclua os recursos do `php-apache`.

```
kubectl delete deployment.apps/php-apache service/php-apache
 horizontalpodautoscaler.autoscaling/php-apache
```

Roteamento de tráfego TCP e UDP com Network Load Balancers

> **Note**
>
> Novo: o Modo Automático do Amazon EKS automatiza as tarefas de rotina para o balanceamento de carga. Para obter mais informações, consulte:
>
> - the section called “Implantar balanceador de carga”
> - the section called “Create service (Criar serviço)”

O tráfego da rede tem a carga balanceada no L4 do modelo OSI. Para balancear a carga do tráfego da aplicação no L7, você implanta um `ingress` do Kubernetes, que provisiona um AWS Application Load Balancer. Para obter mais informações, consulte the section called “Balanceador de carga da aplicação”. Para saber mais sobre as diferenças entre os dois tipos de balanceamento de carga, consulte Elastic Load Balancing features (Recursos do Elastic Load Balancing) no site da AWS.

Ao criar um Kubernetes `Service` do tipo `LoadBalancer`, o controlador do balanceador de carga do provedor da nuvem AWS cria AWS Classic Load Balancers por padrão, mas também pode criar AWS Network Load Balancers. Este controlador só receberá correções de erros críticos no futuro. Para obter mais informações sobre como usar o balanceador de carga do provedor da nuvem da AWS, consulte AWS cloud provider load balancer controller (Controlador do balanceador de carga do provedor de nuvem da) na documentação do Kubernetes. Seu uso não é abordado neste tópico.

Recomendamos que você use a versão `2.7.2` ou posterior do AWS Load Balancer Controller em vez do controlador do balanceador de carga do provedor de nuvem AWS. O AWS Load Balancer Controller cria AWS Network Load Balancers, mas não cria AWS Classic Load Balancers. O restante deste tópico refere-se ao uso do AWS Load Balancer Controller.

Um AWS Network Load Balancer pode balancear a carga do tráfego de rede para pods implantados em destinos de instância e de IP do Amazon EC2, em destinos de IP do AWS Fargate ou em destinos de IP do Amazon EKS Hybrid Nodes. Para obter mais informações, consulte AWS Load Balancer Controller no GitHub.

Pré-requisitos

Antes que você possa balancear a carga de tráfego da rede usando o AWS Load Balancer Controller, é necessário cumprir os requisitos a seguir.

- Tem um cluster já existente. Se você não tem um cluster, consulte Conceitos básicos. Se precisar atualizar a versão de um cluster existente, consulte the section called “Atualizar a versão do Kubernetes”.
- Tenha o AWS Load Balancer Controller implantado no cluster. Para obter mais informações, consulte the section called “ AWS Load Balancer Controller”. Recomendamos a versão `2.7.2` ou posterior.
- Pelo menos uma sub-rede. Se várias sub-redes marcadas são encontradas em uma zona de disponibilidade, o controlador escolhe a primeira sub-rede por ordem lexicográfica de IDs. A sub-rede deve ter pelo menos oito endereços IP disponíveis.
- Se você estiver usando o AWS Load Balancer Controller, versão `2.1.1` ou anterior, as sub-redes deverão ser marcadas do modo a seguir. Se estiver usando a versão `2.1.2` ou superior, essa etiqueta será opcional. Talvez seja bom marcar uma sub-rede se você tiver vários clusters em execução na mesma VPC ou várias sub-redes de compartilhamento de serviços da AWS em uma VPC e quiser ter mais controle sobre onde os balanceadores de carga são provisionados por cluster. Se você especificar explicitamente os IDs de sub-redes como uma anotação em um objeto de serviço, o Kubernetes e o AWS Load Balancer Controller usarão essas sub-redes diretamente para criar o balanceador de carga. A marcação da sub-rede não é necessária se você optar por usar esse método para provisionar balanceadores de carga, e você pode ignorar os requisitos de marcação de sub-rede privada e pública a seguir. Substitua *`my-cluster`* pelo nome do cluster.
  - Chave – `kubernetes.io/cluster/<my-cluster>`
  - Valor: `shared` ou `owned`
- As suas sub-redes públicas e privadas devem atender aos requisitos a seguir, a menos que você especifique explicitamente os IDs de sub-rede como uma anotação em um objeto de serviço ou de entrada. Se você provisionar balanceadores de carga especificando explicitamente os IDs de sub-redes como uma anotação em um objeto de serviço ou entrada, o Kubernetes e o AWS Load Balancer Controller usarão essas sub-redes diretamente para criar o balanceador de carga, e as tags a seguir não serão necessárias.
  - Sub-redes privadas – Devem ser marcadas no seguinte formato. Isso é para que o Kubernetes e o AWS Load Balancer Controller saibam que as sub-redes podem ser usadas para balanceadores de carga internos. Se você usar `eksctl` ou um modelo do Amazon EKS AWS AWS CloudFormation para criar sua VPC após 26 de março de 2020, as sub-redes serão marcadas adequadamente quando forem criadas. Para obter mais informações sobre os modelos de VPC do Amazon EKS AWS AWS CloudFormation, consulte the section called “Crie uma VPC”.
    - Chave – `kubernetes.io/role/internal-elb`

    - Value (valor – 1
- Sub-redes públicas – Devem ser marcadas no seguinte formato. Isso é para que o Kubernetes saiba que deve usar somente essas sub-redes para balanceadores de carga externos, em vez de escolher uma sub-rede pública em cada zona de disponibilidade (em ordem lexicográfica por ID de sub-rede). Se você usar `eksctl` ou um modelo do Amazon EKS AWS CloudFormation para criar sua VPC após 26 de março de 2020, as sub-redes serão marcadas adequadamente quando forem criadas. Para obter mais informações sobre os modelos de VPC do Amazon EKS AWS CloudFormation, consulte the section called “Crie uma VPC”.
    - Chave – `kubernetes.io/role/elb`
    - Value (valor – 1

  Se as tags de perfil de sub-redes não forem explicitamente adicionadas, o controlador de serviço do Kubernetes analisará a tabela de rotas das sub-redes da VPC do cluster para determinar se a sub-rede é privada ou pública. Recomendamos que você não confie nesse comportamento e adicione explicitamente as etiquetas de função privada ou pública. O AWS Load Balancer Controller não examina as tabelas de rota e requer que as tags privadas e públicas estejam presentes para a detecção automática ter êxito.

Considerações

- A configuração do balanceador de carga é controlada por anotações que são adicionadas ao manifesto para o serviço. As anotações de serviço são diferentes ao usar o AWS Load Balancer Controller e o controlador do balanceador de carga do provedor da nuvem AWS. Certifique-se de revisar as anotações do AWS Load Balancer Controller antes de implantar os serviços.
- Ao usar o plug-in CNI da Amazon VPC para Kubernetes, o AWS Load Balancer Controller pode balancear a carga para destinos de IP ou de instância do Amazon EC2 e destinos de IP do Fargate. Ao usar Alternar plug-ins compatíveis da CNI, o controlador só poderá balancear a carga para destinos de instância, a menos que você esteja balanceando a carga para o Amazon EKS Hybrid Nodes. Para nós híbridos, o controlador pode balancear a carga de destinos de IP. Para obter mais informações sobre os tipos de destino do Network Load Balancer, consulte Target type (Tipos de destino) no Guia do usuário de Network Load Balancers.
- Se quiser adicionar etiquetas ao balanceador de carga quando ou depois que ele for criado, adicione a anotação a seguir na especificação do seu serviço. Para obter mais informações, consulte Etiquetas de recurso da AWS na documentação do AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
```

- Você pode atribuir endereços IP elásticos ao Network Load Balancer adicionando a anotação a seguir. Substitua os valores de exemplo pelo endereço `Allocation IDs` do seus endereços Elastic IP. O número de `Allocation IDs` deve corresponder ao número de sub-redes usadas para o balanceador de carga. Para obter mais informações, consulte a documentação do AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations: eipalloc-
xxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyy
```

- O Amazon EKS adiciona uma regra de entrada ao grupo de segurança do nó para o tráfego de clientes e uma regra para cada sub-rede do balanceador de carga na VPC para as verificações de integridade para cada Network Load Balancer que você criar. A implantação de um serviço do tipo `LoadBalancer` poderá falhar se o Amazon EKS tentar criar regras que excedam a cota para o número máximo de regras permitidas para um grupo de segurança. Para obter mais informações, consulte Security groups (Grupos de segurança) em Amazon VPC quotas (Cotas da Amazon VPC) no Manual do usuário da Amazon VPC. Considere as opções a seguir para minimizar as chances de exceder o número máximo de regras para um grupo de segurança:
  - Solicite um aumento em suas regras por cota de grupo de segurança. Para obter mais informações, consulte Requesting a quota increase (Solicitar um aumento da cota) no Manual do usuário do Service Quotas.
  - Use destinos IP, em vez de destinos instância. Com destinos IP, você pode compartilhar as regras para as mesmas portas de destino. As sub-redes do balanceador de carga podem ser especificadas manualmente com uma anotação. Para obter mais informações, consulte Annotations (Anotações) no GitHub.
  - Use uma entrada em vez de um serviço do tipo `LoadBalancer` para enviar tráfego ao seu serviço. O AWS Application Load Balancer exige menos regras do que os Network Load Balancers. Você pode compartilhar um ALB em várias entradas. Para obter mais informações, consulte the section called “Balanceador de carga da aplicação”. Você não pode compartilhar um Network Load Balancer em vários serviços.
  - Implante seus clusters em várias contas.
- Se os pods forem executados no Windows em um cluster do Amazon EKS, um único serviço com um balanceador de carga poderá ser compatível com até 1.024 pods de backend. Cada pod tem seu próprio endereço IP exclusivo.

- Recomendamos que você só crie novos Network Load Balancers com o AWS Load Balancer Controller. A tentativa de substituir Network Load Balancers existentes criados pelo controlador do balanceador de carga do provedor da nuvem AWS pode resultar em vários Network Load Balancers, que poderão gerar tempo de inatividade na aplicação.

Criar um balanceador de carga da rede

Você pode criar um balanceador de carga da rede com destinos de IP ou de instância.

Criar Network Load Balancer - Destinos de IP

- Você pode usar destinos IP com pods implantados em nós do Amazon EC2, do Fargate ou do Amazon EKS Hybrid Nodes. O serviço do Kubernetes deve ser criado como tipo `LoadBalancer`. Para obter mais informações, consulte Type LoadBalancer na documentação do Kubernetes.

  Para criar um balanceador de carga que use destinos IP, adicione a anotação a seguir a um manifesto de serviço e implante o seu serviço. O valor `external` para `aws-load-balancer-type` faz com que o AWS Load Balancer Controller, em vez do controlador do balanceador de carga do provedor da nuvem AWS, crie o Network Load Balancer. Você pode visualizar um manifesto de serviço de exemplo com as anotações.

  ```
  service.beta.kubernetes.io/aws-load-balancer-type: "external"
  service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
  ```

  > **Note**
  >
  > Se você estiver balanceando a carga para pods do `IPv6`, adicione a anotação a seguir. Você só pode balancear a carga em `IPv6` para destinos IP, e não para destinos de instâncias. Sem essa anotação, o balanceamento de carga ocorrerá em `IPv4`.

  ```
  service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
  ```

  Por padrão, os Network Load Balancers são criados com o `aws-load-balancer-scheme` `internal`. Você pode iniciar balanceadores de carga de rede em qualquer sub-rede na VPC do cluster, incluindo sub-redes que não foram especificadas quando você criou o cluster.

O Kubernetes examina a tabela de rotas para as sub-redes a fim de identificar se elas são públicas ou privadas. As sub-redes públicas têm uma rota direta para a Internet usando um gateway da Internet, mas as sub-redes privadas não têm.

Se você quiser criar um Network Load Balancer em uma sub-rede pública para balancear a carga para nós do Amazon EC2 (o Fargate só pode ser em privadas), especifique `internet-facing` com a seguinte anotação:

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Note**
>
> Para fins de compatibilidade com versões anteriores, o suporte a anotações `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` ainda é oferecido. Porém, recomendamos usar as anotações anteriores para novos balanceadores de carga em vez do `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"`.

> **Important**
>
> Não edite as anotações depois de criar o serviço. Se precisar modificá-lo, exclua o objeto de serviço e crie-o novamente com o valor desejado para essa anotação.

Criar Network Load Balancer - Destinos de instância

- O controlador do balanceador de carga do provedor da nuvem AWS cria Network Load Balancers com apenas destinos de instâncias. A versão `2.2.0` e versões superiores do AWS Load Balancer Controller também criam Network Load Balancers com destinos de instâncias. Recomendamos usá-lo para criar novos Network Load Balancers, em vez do controlador do balanceador de carga do provedor da nuvem AWS. Você pode usar os destinos da instância do Network Load Balancer com pods implantados em nós do Amazon EC2, mas não no Fargate. Para balancear a carga do tráfego de rede em pods implantados no Fargate, você deve usar destinos IP.

  Para implantar um Network Load Balancer em uma sub-rede privada, a sua especificação de serviço deve ter as anotações a seguir. Você pode visualizar um manifesto de serviço de exemplo

com as anotações. O valor `external` para `aws-load-balancer-type` faz com que o AWS Load Balancer Controller, em vez do controlador do balanceador de carga do provedor da nuvem AWS, crie o Network Load Balancer.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
```

Por padrão, os Network Load Balancers são criados com o `aws-load-balancer-scheme` `internal`. Para Network Load Balancers internos, o seu cluster do Amazon EKS deve ser configurado para usar pelo menos uma sub-rede privada na sua VPC. O Kubernetes examina a tabela de rotas para as sub-redes a fim de identificar se elas são públicas ou privadas. As sub-redes públicas têm uma rota direta para a Internet usando um gateway da Internet, mas as sub-redes privadas não têm.

Se você quiser criar um Network Load Balancer em uma sub-rede pública para balancear a carga para nós do Amazon EC2, especifique `internet-facing` com a seguinte anotação:

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Important**
>
> Não edite as anotações depois de criar o serviço. Se precisar modificá-lo, exclua o objeto de serviço e crie-o novamente com o valor desejado para essa anotação.

(Opcional) Implantar uma aplicação de exemplo

- Pelo menos uma sub-rede pública ou privada na VPC do cluster.
- Tenha o AWS Load Balancer Controller implantado no cluster. Para obter mais informações, consulte the section called “ AWS Load Balancer Controller”. Recomendamos a versão `2.7.2` ou posterior.

1. Se estiver implantando no Fargate, certifique-se de ter uma sub-rede privada disponível em sua VPC e crie um perfil do Fargate. Se não estiver implantando no Fargate, ignore esta etapa. Você pode criar o perfil executando o comando a seguir ou no Console de gerenciamento da AWS usando os mesmos valores para `name` e `namespace` que estão no comando. Substitua os valores de exemplo pelos seus próprios.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name nlb-sample-app \
    --namespace nlb-sample-app
```

2. Implante uma aplicação de exemplo.

   a. Crie um namespace para a aplicação.

   ```
   kubectl create namespace nlb-sample-app
   ```

   b. Salve o conteúdo a seguir em seu computador, em um arquivo chamado `sample-deployment.yaml`.

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: nlb-sample-app
     namespace: nlb-sample-app
   spec:
     replicas: 3
     selector:
       matchLabels:
         app: nginx
     template:
       metadata:
         labels:
           app: nginx
       spec:
         containers:
           - name: nginx
             image: public.ecr.aws/nginx/nginx:1.23
             ports:
               - name: tcp
                 containerPort: 80
   ```

   c. Aplique o manifesto ao cluster.

   ```
   kubectl apply -f sample-deployment.yaml
   ```

3. Crie um serviço com um Network Load Balancer voltado para a Internet que faça o balanceamento de carga para destinos IP.

   a. Salve o conteúdo a seguir em seu computador, em um arquivo denominado `sample-service.yaml`. Se estiver implantando nos nós do Fargate, remova a linha `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing`.

   ```
   apiVersion: v1
   kind: Service
   metadata:
     name: nlb-sample-service
     namespace: nlb-sample-app
     annotations:
       service.beta.kubernetes.io/aws-load-balancer-type: external
       service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
       service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
   spec:
     ports:
       - port: 80
         targetPort: 80
         protocol: TCP
     type: LoadBalancer
     selector:
       app: nginx
   ```

   b. Aplique o manifesto ao cluster.

   ```
   kubectl apply -f sample-service.yaml
   ```

4. Verifique se o serviço foi implantado.

   ```
   kubectl get svc nlb-sample-service -n nlb-sample-app
   ```

   Veja abaixo um exemplo de saída.

   ```
   NAME             TYPE           CLUSTER-IP         EXTERNAL-IP
                                                  PORT(S)        AGE
   sample-service   LoadBalancer   10.100.240.137   k8s-nlbsampl-nlbsampl-xxxxxxxxxx-
   xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com  80:32400/TCP   16h
   ```

> **Note**
>
> Os valores para *10.100.240.137* e *xxxxxxxxxx-xxxxxxxxxxxxxxxx* serão diferentes da saída do exemplo (eles serão exclusivos para o seu balanceador de carga) e *us-west-2* pode variar para você, dependendo da região da AWS na qual o seu cluster está localizado.

5. Abra o Console de gerenciamento da AWS do Amazon EC2. Selecione Target Groups (Grupos de destino) em Load Balancing (Balanceamento de carga) no painel de navegação à esquerda. Em Name (Nome), selecione o nome do grupo de destino no qual o valor na coluna Load balancer (Balanceador de carga) corresponda a uma parte do nome na coluna `EXTERNAL-IP` do resultado na etapa anterior. Por exemplo, você selecionaria o grupo de destino chamado `k8s-default-samplese-`*xxxxxxxxxx* se o resultado fosse igual ao da saída acima. O tipo de destino é `IP`, conforme especificado no manifesto de serviço de exemplo.
6. Selecione o Target group (Grupo de destino) e escolha a guia Targets (Destinos). Em Registered targets (Destinos registrados), você verá três endereços IP das três réplicas implantadas em uma etapa anterior. Aguarde até que o status de todos os destinos seja healthy (íntegro) antes de continuar. Pode levar alguns minutos até que todos os destinos fiquem `healthy`. Os destinos podem estar em um estado `unhealthy` antes de serem alterados para um estado `healthy`.
7. Envie tráfego para o serviço substituindo *xxxxxxxxxxxxxx-xxxxxxxxxxxxxxxxxxxxxx* e *us-west-2* pelos valores retornados na saída de uma etapa anterior para `EXTERNAL-IP`. Se você implantou em uma sub-rede privada, precisará visualizar a página em um dispositivo dentro da VPC, como um bastion host. Para obter mais informações, consulte Bastion hosts do Linux na AWS.

   ```
   curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
   ```

   Veja abaixo um exemplo de saída.

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```

8. Ao terminar de usar a implantação, o serviço e o namespace de exemplo, remova-os.

```
kubectl delete namespace nlb-sample-app
```

Roteamento de aplicações e tráfego HTTP com Application Load Balancers

> **Note**
>
> Novo: o Modo Automático do Amazon EKS automatiza as tarefas de rotina para o balanceamento de carga. Para obter mais informações, consulte:
>
> - the section called “Implantar balanceador de carga”
> - the section called “Criar classe de entrada”

Quando você cria um `ingress` do Kubernetes, um AWS Application Load Balancer (ALB) é provisionado e equilibra a carga de tráfego da aplicação. Para saber mais, consulte What is an Application Load Balancer? (O que é um balanceador de carga da aplicação?) no Guia do usuário de balanceadores de carga da aplicação e Ingress (Entrada) na documentação do Kubernetes. ALBs podem ser usados com pods implantados em nós ou no AWS Fargate. Você pode implantar um ALB em sub-redes públicas ou privadas.

O tráfego de aplicações é equilibrado no L7 do modelo OSI. Para balancear a carga do tráfego de rede na L4, você implanta um `service` do Kubernetes do tipo `LoadBalancer`. Este tipo provisiona um AWS Network Load Balancer. Para obter mais informações, consulte the section called “Balanceamento de carga da rede”. Para saber mais sobre as diferenças entre os dois tipos de balanceamento de carga, consulte Elastic Load Balancing features (Recursos do Elastic Load Balancing) no site da AWS.

Pré-requisitos

Antes de balancear a carga de tráfego de uma aplicação específica, você precisa cumprir os requisitos a seguir.

- Tem um cluster já existente. Se você não tem um cluster, consulte Conceitos básicos. Se precisar atualizar a versão de um cluster existente, consulte the section called “Atualizar a versão do Kubernetes”.

- Tenha o AWS Load Balancer Controller implantado no cluster. Para obter mais informações, consulte the section called “ AWS Load Balancer Controller”. Recomendamos a versão `2.7.2` ou posterior.
- As duas sub-redes devem estar em diferentes zonas de disponibilidade. O AWS Load Balancer Controller escolhe uma sub-rede de cada zona de disponibilidade. Quando várias sub-redes marcadas são encontradas em uma zona de disponibilidade, o controlador escolhe a primeira sub-rede, por ordem lexicográfica de IDs. Cada sub-rede deve ter pelo menos oito endereços IP disponíveis.

  Se você estiver usando vários grupos de segurança anexados ao nó de processamento, exatamente um grupo de segurança deve ser marcado como se segue. Substitua *my-cluster* pelo nome do cluster.

  - Chave – `kubernetes.io/cluster/<my-cluster>`
  - Valor: `shared` ou `owned`
- Se você estiver usando o AWS Load Balancer Controller, versão `2.1.1` ou anterior, as sub-redes deverão ser marcadas como a seguir. Se estiver usando a versão `2.1.2` ou superior, essa marcação é opcional. Porém, recomendamos que você marque uma sub-rede em qualquer um dos seguintes casos. Você tiver vários clusters que estão sendo executados na mesma VPC ou tiver vários serviços da AWS que compartilham sub-redes em uma VPC. Ou, você quiser ter mais controle sobre onde os balanceadores de carga são provisionados para cada cluster. Substitua *my-cluster* pelo nome do cluster.
  - Chave – `kubernetes.io/cluster/<my-cluster>`
  - Valor: `shared` ou `owned`
- As sub-redes públicas e privadas devem atender aos seguintes requisitos, Isso só não ocorre caso você especifique explicitamente IDs de sub-redes como uma anotação em um objeto de serviço ou entrada. Pressuponha que você provisione balanceadores de carga especificando explicitamente os IDs de sub-redes como uma anotação em um objeto de serviço ou entrada. Nessa situação, o Kubernetes e o AWS Load Balancer Controller usam as sub-redes diretamente para criar o balanceador de carga, e as tags a seguir não são necessárias.
  - Sub-redes privadas – Devem ser marcadas no seguinte formato. Isso é para que o Kubernetes e o AWS Load balancer Controller saibam que as sub-redes podem ser usadas para balanceadores de carga internos. Se você usar o `eksctl` ou um modelo do AWS CloudFormation para Amazon EKS a fim de criar sua VPC após 26 de março de 2020, as sub-redes serão marcadas adequadamente quando criadas. Para obter mais informações sobre

modelos de VPC do AWS CloudFormation para Amazon EKS, consulte the section called “Crie uma VPC”.

- Chave – `kubernetes.io/role/internal-elb`
- Value (valor – 1

- Sub-redes públicas – Devem ser marcadas no seguinte formato. Isso ocorre para que o Kubernetes saiba que deve usar somente as sub-redes especificadas para balanceadores de carga externos. Dessa forma, o Kubernetes não escolhe uma sub-rede pública em cada zona de disponibilidade (por ordem lexicográfica dos IDs de sub-rede). Se você usar o `eksctl` ou um modelo do AWS CloudFormation para Amazon EKS a fim de criar sua VPC após 26 de março de 2020, as sub-redes serão marcadas adequadamente quando criadas. Para obter mais informações sobre modelos de VPC do AWS CloudFormation para Amazon EKS, consulte the section called “Crie uma VPC”.
  - Chave – `kubernetes.io/role/elb`
  - Value (valor – 1

  Se as tags de perfil de sub-rede não forem explicitamente adicionadas, o controlador de serviço do Kubernetes analisará a tabela de rotas das sub-redes da VPC do cluster. Isso ocorre para determinar se a sub-rede é privada ou pública. Recomendamos que você não confie nesse comportamento. Em vez disso, adicione explicitamente as etiquetas de função privada ou pública. O AWS Load Balancer Controller não examina tabelas de rotas. Também requer que as etiquetas privadas e públicas estejam presentes para a detecção automática ser bem-sucedida.

- O AWS Load Balancer Controller criará ALBs e os recursos compatíveis da AWS necessários sempre que um recurso de entrada do Kubernetes for criado em um cluster com a anotação `kubernetes.io/ingress.class: alb`. O recurso de entrada configura o ALB para rotear o tráfego de HTTP ou HTTPS para diferentes pods no cluster. Para garantir que os seus objetos de entrada usem o AWS Load Balancer Controller, adicione a anotação a seguir à sua especificação de entrada do Kubernetes. Para obter mais informações, consulte Ingress specification (Especificação de entrada) na documentação do GitHub.

```
annotations:
    kubernetes.io/ingress.class: alb
```

> **Note**
>
> Se você estiver balanceando a carga para pods `IPv6`, adicione a anotação a seguir à sua especificação de entrada. Você só pode balancear a carga em `IPv6` para destinos IP, e

não para destinos de instâncias. Sem essa anotação, o balanceamento de carga ocorrerá em `IPv4`.

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

- O AWS Load Balancer Controller oferece suporte aos seguintes modos de tráfego:
  - Instance: registra nós dentro do cluster como destinos para o ALB. O tráfego que chega ao ALB é roteado para o `NodePort` para o seu serviço e, depois, enviado por proxy para os pods. Esse é o modo de tráfego padrão. Também é possível especificá-lo explicitamente com a anotação `alb.ingress.kubernetes.io/target-type: instance`.

    Note

    O serviço do Kubernetes deve especificar o tipo de `NodePort` ou `LoadBalancer` para usar esse modo de tráfego.

  - IP – Registra pods como destinos para o ALB. O tráfego que chega ao ALB é roteado diretamente para os pods do seu serviço. É necessário especificar a anotação `alb.ingress.kubernetes.io/target-type: ip` para usar esse modo de tráfego. O tipo de IP de destino é necessário quando os pods de destino estão em execução no Fargate ou no Amazon EKS Hybrid Nodes.
- Para marcar ALBs criados pelo controlador, adicione a seguinte anotação ao controlador: `alb.ingress.kubernetes.io/tags`. Para obter uma lista de todas as anotações disponíveis compatíveis com o AWS Load Balancer Controller, consulte Ingress annotations (Anotações de entrada) no GitHub.
- Atualizar ou retroceder a versão do controlador de ALB pode introduzir alterações importantes para recursos que dependem dele. Para obter mais informações sobre as alterações que são apresentadas em cada versão, consulte as Notas de release do controlador de ALB no GitHub.

Reutilizar ALBs com grupos de entrada

Você pode compartilhar um Application Load Balancer em vários recursos de serviço usando o `IngressGroups`.

Para integrar uma entrada em um grupo, adicione a anotação a seguir a uma especificação do recurso de entrada do Kubernetes.

```
alb.ingress.kubernetes.io/group.name: my-group
```

O nome do grupo deve:

- Ter no máximo 63 caracteres de comprimento.
- Consistir em letras minúsculas, números, `-` e `.`
- Iniciar e terminar com um número ou letra.

O controlador mescla automaticamente as regras de entrada para todas as entradas no mesmo grupo de entrada. Ele os suporta com um único ALB. A maioria das anotações definidas em uma entrada só se aplica aos caminhos definidos por essa entrada. Por padrão, os recursos de entrada não pertencem a nenhum grupo de entrada.

> **Warning**
>
> Risco potencial de segurança
> Especifique um grupo de entrada para uma entrada somente quando todos os usuários do Kubernetes que têm permissão RBAC para criar ou modificar recursos de entrada estiverem dentro do mesmo limite de confiança. Se você adicionar a anotação com um nome de grupo, talvez outros usuários do Kubernetes possam criar ou modificar suas entradas para pertencerem ao mesmo grupo de entrada. Fazer isso pode causar comportamento indesejável, como substituir regras existentes por regras de prioridade mais alta.

Você pode adicionar um número de ordem para o seu recurso de entrada.

```
alb.ingress.kubernetes.io/group.order: '10'
```

O número pode ser de 1 a 1000. O número mais baixo para todas as entradas no mesmo grupo de entrada será avaliado primeiro. Todas as entradas sem essa anotação são avaliadas com um valor zero. As regras duplicadas com um número mais alto podem substituir as regras com um número mais baixo. Por padrão, a ordem de regra entre as entradas no mesmo grupo de entrada é determinada por ordem léxica com base no namespace e no nome.

> **Important**
>
> Certifique-se de que cada entrada no mesmo grupo de entrada tenha um número de prioridade exclusivo. Não é possível ter números de ordens duplicados nas entradas.

(Opcional) Implantar uma aplicação de exemplo

- Pelo menos uma sub-rede pública ou privada na VPC do cluster.
- Tenha o AWS Load Balancer Controller implantado no cluster. Para obter mais informações, consulte the section called “ AWS Load Balancer Controller”. Recomendamos a versão `2.7.2` ou posterior.

É possível executar a aplicação de exemplo em um cluster que tenha nós do Amazon EC2, pods do Fargate ou ambos.

1. Se não estiver implantando no Fargate, ignore esta etapa. Se estiver implantando no Fargate, crie um perfil do Fargate. Você pode criar o perfil executando o comando a seguir ou no Console de gerenciamento da AWS usando os mesmos valores para `name` e `namespace` que estão no comando. Substitua os valores de exemplo pelos seus próprios.

   ```
   eksctl create fargateprofile \
       --cluster my-cluster \
       --region region-code \
       --name alb-sample-app \
       --namespace game-2048
   ```

2. Implante o jogo 2048 como uma aplicação de exemplo para verificar se o AWS Load Balancer Controller cria um AWS ALB como resultado do objeto de entrada. Realize as etapas para o tipo de sub-rede em que está implantando.

   a. Se você estiver implantando em pods em um cluster criado com a família do `IPv6`, pule para a próxima etapa.

      - Público:

      ```
      kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
      ```

- Privado:

  A. Faça download do manifesto.

  ```
  curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
  ```

  B. Edite o arquivo e encontre a linha que diz `alb.ingress.kubernetes.io/scheme: internet-facing`.

  C. Altere o endereço *voltado à Internet* para `internal` e salve o arquivo.

  D. Aplique o manifesto ao cluster.

  ```
  kubectl apply -f 2048_full.yaml
  ```

b. Se você estiver implantando em pods em um cluster criado com a família do IPv6, conclua as etapas a seguir.

   i. Faça download do manifesto.

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
   ```

   ii. Abra o arquivo em um editor e adicione a linha a seguir às anotações nas especificações de entrada.

   ```
   alb.ingress.kubernetes.io/ip-address-type: dualstack
   ```

   iii. Se você estiver balanceando a carga para pods internos, em vez de para pods voltados para a internet, altere a linha que apresenta `alb.ingress.kubernetes.io/scheme: internet-facing` para `alb.ingress.kubernetes.io/scheme: internal`.

   iv. Salve o arquivo.

   v. Aplique o manifesto ao cluster.

   ```
   kubectl apply -f 2048_full.yaml
   ```

3. Após alguns minutos, verifique se o recurso de entrada foi criado com o comando a seguir.

```
kubectl get ingress/ingress-2048 -n game-2048
```

Veja um exemplo de saída abaixo.

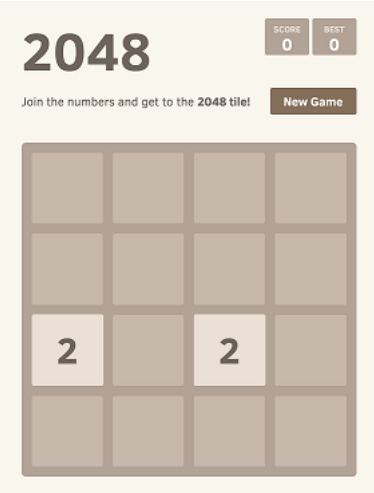
```
NAME           CLASS    HOSTS   ADDRESS
                      PORTS   AGE
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-
code.elb.amazonaws.com   80      2m32s
```

> **Note**
>
> Se você criou o balanceador de carga em uma sub-rede privada, o valor de `ADDRESS` na saída anterior é prefaciado com `internal-`.

Se a sua entrada não for criada com êxito após alguns minutos, execute o comando a seguir para visualizar os logs do AWS Load Balancer Controller. Os logs podem conter mensagens de erro que podem ajudar a diagnosticar problemas na sua implantação.

```
kubectl logs -f -n kube-system -l app.kubernetes.io/instance=aws-load-balancer-
controller
```

1. Se implantou em uma sub-rede pública, abra um navegador e acesse o URL `ADDRESS` do resultado do comando anterior para ver a aplicação de exemplo. Se não vir nada, atualize o navegador e tente novamente. Se você implantou em uma sub-rede privada, precisará visualizar a página em um dispositivo dentro da VPC, como um bastion host. Para obter mais informações, consulte Bastion hosts do Linux na AWS.



2. Ao terminar de fazer os testes com a aplicação de exemplo, exclua-a executando um dos comandos a seguir.

- Se você aplicou o manifesto, em vez de aplicar uma cópia que baixou, use o comando a seguir.

```
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
```

- Se você baixou e editou o manifesto, use o comando a seguir.

```
kubectl delete -f 2048_full.yaml
```

Restringir endereços IP externos que podem ser atribuídos aos serviços

Os serviços do Kubernetes podem ser acessados de um cluster por meio de:

- Um endereço IP de cluster atribuído automaticamente pelo Kubernetes
- Qualquer endereço IP que você especificar para a propriedade `externalIPs` em uma especificação de serviço. Os endereços IP externos não são gerenciados pelo Kubernetes e são da responsabilidade do administrador do cluster. Endereços IP externos especificados com `externalIPs` são diferentes do endereço IP externo atribuído a um serviço do tipo `LoadBalancer` por um provedor de nuvem.

Para saber mais sobre os serviços do Kubernetes, consulte Services (Serviços) na documentação do Kubernetes. Você pode restringir os endereços IP que podem ser especificados para `externalIPs`em uma especificação de serviço.

1. Implante `cert-manager` para gerenciar certificados do webhook. Para obter mais informações, consulte a documentação cert-manager.

```
kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert-manager.yaml
```

2. Verifique se os pods `cert-manager` estão em execução.

```
kubectl get pods -n cert-manager
```

Veja um exemplo de saída abaixo.

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-58c8844bb8-nlx7q              1/1     Running   0          15s
cert-manager-cainjector-745768f6ff-696h5   1/1     Running   0          15s
cert-manager-webhook-67cc76975b-4v4nk      1/1     Running   0          14s
```

3. Analise os serviços existentes para garantir que nenhum deles tenha endereços IP externos atribuídos a eles que não estejam contidos no bloco CIDR, ao qual você deseja limitar os endereços.

   ```
   kubectl get services -A
   ```

   Veja um exemplo de saída abaixo.

   ```
   NAMESPACE                   NAME                                      TYPE
    CLUSTER-IP       EXTERNAL-IP     PORT(S)         AGE
   cert-manager                cert-manager                              ClusterIP
    10.100.102.137   <none>          9402/TCP        20m
   cert-manager                cert-manager-webhook                      ClusterIP
    10.100.6.136     <none>          443/TCP         20m
   default                     kubernetes                                ClusterIP
    10.100.0.1       <none>          443/TCP         2d1h
   externalip-validation-system   externalip-validation-webhook-service   ClusterIP
    10.100.234.179   <none>          443/TCP         16s
   kube-system                 kube-dns                                  ClusterIP
    10.100.0.10      <none>          53/UDP,53/TCP   2d1h
   my-namespace                my-service                                ClusterIP
    10.100.128.10    192.168.1.1     80/TCP          149m
   ```

   Se qualquer um dos valores for endereços IP que não estiverem dentro do bloco ao qual você deseja restringir o acesso, você precisará alterar os endereços para que fiquem dentro do bloco e reimplantar os serviços. Por exemplo, o serviço `my-service` na saída anterior tem um endereço IP externo atribuído a ele que não está dentro do exemplo de bloco CIDR na etapa 5.

4. Faça download do manifesto de webhook IP externo. Você também pode visualizar o source code for the webhook (código-fonte para webhook) no GitHub.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
   ```

5. Especifique blocos CIDR. Abra o arquivo baixado em seu editor e remova o \# no início das linhas a seguir.

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

Substitua `10.0.0.0/8` pelo seu próprio bloco CIDR. Especifique quantos blocos quiser. Se especificar vários blocos, adicione uma vírgula entre eles.

6. Se o seu cluster não estiver na região `us-west-2` AWS, substitua `us-west-2`, `602401143452` e `amazonaws.com` no arquivo pelos seguintes comandos. Antes de executar os comandos, substitua *`region-code`* e *`111122223333`* pelo valor da sua região da AWS na lista em Exibir registros de imagens de contêineres da Amazon para complementos do Amazon EKS.

```
sed -i.bak -e 's|602401143452|111122223333|' externalip-webhook.yaml
sed -i.bak -e 's|us-west-2|region-code|' externalip-webhook.yaml
sed -i.bak -e 's|amazonaws.com||' externalip-webhook.yaml
```

7. Aplique o manifesto ao cluster.

```
kubectl apply -f externalip-webhook.yaml
```

Ocorrerá uma falha na tentativa de implantar um serviço no cluster com um endereço IP especificado para `externalIPs` que não estiver contido nos blocos especificados na etapa Especificar blocos CIDR.

Copiar uma imagem de contêiner de um repositório para outro

Este tópico descreve como extrair uma imagem de contêiner de um repositório ao qual seus nós não têm acesso e enviar a imagem para um repositório ao qual seus nós têm acesso. Você pode enviar a imagem para o Amazon ECR ou para um repositório alternativo ao qual seus nós têm acesso.

- O mecanismo do Docker instalado e configurado em seu computador. Para obter mais informações, consulte Install Docker Engine (Instalar mecanismo do Docker) na documentação do Docker.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing

e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.

- Um endpoint da VPC de interface para o Amazon ECR se você quiser que seus nós extraiam imagens de contêiner ou enviem imagens de contêiner para um repositório privado do Amazon ECR pela rede da Amazon. Para obter mais informações, consulte Criação dos endpoints da VPC para o Amazon ECR no Guia do usuário do Amazon Elastic Container Registry.

Conclua as etapas a seguir para extrair uma imagem de contêiner de um repositório e enviá-la para seu próprio repositório. Nos exemplos a seguir, fornecidos neste tópico, é extraída a imagem do plug-in CNI do Amazon VPC para o auxiliar de métricas do Kubernetes. Ao seguir estas etapas, certifique-se de substituir os valores de exemplo por seus próprios valores.

1. Se você ainda não tem um repositório do Amazon ECR ou outro repositório, crie um ao qual os nós tenham acesso. O comando a seguir cria um repositório privado do Amazon ECR. Um nome de repositório privado do Amazon ECR deve começar com uma letra. Ele pode conter apenas letras minúsculas, números, hifens (-), sublinhados (_) e barras (/). Para obter mais informações, consulte Criar um repositório privado no Guia do usuário do Amazon Elastic Container Registry.

   É possível substituir *`cni-metrics-helper`* pelo que você escolher. Como prática recomendada, crie um repositório separado para cada imagem. Recomendamos isso porque as etiquetas de imagem devem ser exclusivas em um repositório. Substitua *`region-code`* por uma região da AWS compatível com o Amazon ECR.

   ```
   aws ecr create-repository --region region-code --repository-name cni-metrics-helper
   ```

2. Determine o registro, o repositório e a etiqueta (opcional) da imagem que seus nós precisam extrair. Essas informações estão no formato `registry/repository[:tag]`.

   Muitos dos tópicos do Amazon EKS sobre a instalação de imagens exigem que você aplique um arquivo manifesto ou instale a imagem usando um chart do Helm. No entanto, antes de aplicar um arquivo manifesto ou instalar um chart do Helm, primeiro visualize o conteúdo do manifesto ou arquivo `values.yaml` do chart. Dessa forma, é possível determinar o registro, o repositório e a etiqueta a serem extraídos.

   Por exemplo, você pode encontrar a seguinte linha no arquivo de manifesto do plug-in Amazon VPC CNI para o auxiliar de métricas do Kubernetes. O registro é `602401143452.dkr.ecr.us-`

`west-2.amazonaws.com`, que é um registro privado do Amazon ECR. O repositório é `cni-metrics-helper`.

```
image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.12.6"
```

Você poderá ver as seguintes variações para o local de uma imagem:

- Somente `repository-name:tag`. Nesse caso, `docker.io` geralmente é o registro, mas não especificado, pois o Kubernetes o prepõe a um nome de repositório por padrão se nenhum registro for especificado.
- `repository-name/repository-namespace/repository:tag`. Um namespace de repositório é opcional, mas às vezes é especificado pelo proprietário do repositório para categorizar imagens. Por exemplo, todas as imagens do Amazon EC2 na Galeria Pública do Amazon ECR usam o namespace `aws-ec2`.

  Antes de instalar uma imagem com Helm, veja o arquivo `values.yaml` do Helm para determinar a localização da imagem. Por exemplo, o arquivo values.yaml do plug-in Amazon VPC CNI para o auxiliar de métricas do Kubernetes inclui as seguintes linhas.

  ```
  image:
    region: us-west-2
    tag: v1.12.6
    account: "602401143452"
    domain: "amazonaws.com"
  ```

3. Extraia a imagem do contêiner especificada no arquivo manifesto.

   a. Se você estiver extraindo de um registro público, como a Galeria pública do Amazon ECR, é possível pular para a próxima subetapa, porque a autenticação não é necessária. Nesse exemplo, você se autentica em um registro privado do Amazon ECR que contém o repositório da imagem do auxiliar de métricas CNI. O Amazon EKS mantém a imagem em cada registro listado em Exibir registros de imagens de contêineres da Amazon para complementos do Amazon EKS. Você pode se autenticar em qualquer um dos registros substituindo *602401143452* e *region-code* pelas informações de um registro diferente. Existe um registro separado para cada região da AWS em que o Amazon EKS é compatível.

   ```
   aws ecr get-login-password --region region-code | docker login --username AWS --
   password-stdin 602401143452.dkr.region-code.amazonaws.com
   ```

b. Extraia a imagem. Nesse exemplo, você extrai do registro no qual se autenticou na subetapa anterior. Substitua *602401143452* e *region-code* pelas informações que você forneceu na subetapa anterior.

```
docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-
helper:v1.12.6
```

4. Etiquete a imagem que você extraiu com seu registro, repositório e etiqueta. O exemplo a seguir pressupõe que você extraiu a imagem do arquivo manifesto e a enviará para o repositório privado do Amazon ECR criado na primeira etapa. Substitua *111122223333* pelo ID da sua conta. Substitua *region-code* pela região AWS na qual você criou o repositório privado do Amazon ECR.

```
docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-code.amazonaws.com/
cni-metrics-helper:v1.12.6
```

5. Faça a autenticação em seu registro. Neste exemplo, você autentica no registro privado do Amazon ECR criado na primeira etapa. Para obter mais informações, consulte Registry authentication (Autenticação de registro) no Guia do usuário do Amazon Elastic Container Registry.

```
aws ecr get-login-password --region region-code | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com
```

6. Envie a imagem ao seu repositório. Neste exemplo, você envia a imagem ao registro privado do Amazon ECR criado na primeira etapa. Para obter mais informações, consulte Envio de uma imagem do Docker no Guia do usuário do Amazon Elastic Container Registry.

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
```

7. Atualize o arquivo manifesto usado para determinar a imagem em uma etapa anterior com o `registry/repository:tag` da imagem que você enviou. Se você estiver instalando com um chart do Helm, muitas vezes há uma opção para especificar o `registry/repository:tag`. Ao instalar o chart, especifique o `registry/repository:tag` da imagem que você enviou ao seu repositório.

Visualizar registros de imagens de contêineres da Amazon para complementos do Amazon EKS

Quando você implanta complementos do AWS Amazon EKS no cluster, os nós extraem as imagens de contêiner necessárias do registro especificado no mecanismo de instalação do complemento, como um manifesto de instalação ou um arquivo `values.yaml` do Helm. As imagens são extraídas de um repositório privado do Amazon ECR no Amazon EKS. O Amazon EKS replica as imagens em um repositório em cada região da AWS compatível com o Amazon EKS. Seus nós podem extrair a imagem do contêiner pela Internet de qualquer um dos registros a seguir. Como alternativa, seus nós podem extrair a imagem pela rede da Amazon se você tiver criado um endpoint de VPC de interface para o Amazon ECR (AWS PrivateLink) na sua VPC. Os registros exigem autenticação com uma conta do AWS IAM. Seus nós são autenticados usando a função do IAM de nó do Amazon EKS, que possui as permissões na política do IAM gerenciada AmazonEC2ContainerRegistryReadOnly associada a ela.

Região da AWS	Registro
af-south-1	877085696533.dkr.ecr.af-south-1.amazonaws.com
ap-east-1	800184023465.dkr.ecr.ap-east-1.amazonaws.com
ap-east-2	533267051163.dkr.ecr.ap-east-1.amazonaws.com
ap-southeast-3	296578399912.dkr.ecr.ap-southeast-3.amazonaws.com
ap-south-2	900889452093.dkr.ecr.ap-south-2.amazonaws.com
ap-southeast-4	491585149902.dkr.ecr.ap-southeast-4.amazonaws.com
ap-southeast-6	333609536671.dkr.ecr.ap-southeast-6.amazonaws.com

Região da AWS	Registro
ap-south-1	602401143452.dkr.ecr.ap-south-1.amazonaws.com/
ap-northeast-3	602401143452.dkr.ecr.ap-northeast-3.amazonaws.com
ap-northeast-2	602401143452.dkr.ecr.ap-northeast-2.amazonaws.com
ap-southeast-1	602401143452.dkr.ecr.ap-southeast-1.amazonaws.com
ap-southeast-2	602401143452.dkr.ecr.ap-southeast-2.amazonaws.com
ap-southeast-7	121268973566.dkr.ecr.ap-southeast-7.amazonaws.com
ap-northeast-1	602401143452.dkr.ecr.ap-northeast-1.amazonaws.com
cn-north-1	918309763551.dkr.ecr.cn-north-1.amazonaws.com.cn
cn-northwest-1	961992271922.dkr.ecr.cn-northwest-1.amazonaws.com.cn
eu-central-1	602401143452.dkr.ecr.eu-central-1.amazonaws.com
eu-west-1	602401143452.dkr.ecr.eu-west-1.amazonaws.com
eu-west-2	602401143452.dkr.ecr.eu-west-2.amazonaws.com/
eu-south-1	590381155156.dkr.ecr.eu-south-1.amazonaws.com

Região da AWS	Registro
eu-west-3	602401143452.dkr.ecr.eu-west-3.amazonaws.com/
eu-south-2	455263428931.dkr.ecr.eu-south-2.amazonaws.com
eu-north-1	602401143452.dkr.ecr.eu-north-1.amazonaws.com
eu-central-2	900612956339.dkr.ecr.eu-central-2.amazonaws.com
il-central-1	066635153087.dkr.ecr.il-central-1.amazonaws.com
mx-central-1	730335286997.dkr.ecr.mx-central-1.amazonaws.com
me-south-1	558608220178.dkr.ecr.me-south-1.amazonaws.com/
me-central-1	759879836304.dkr.ecr.me-central-1.amazonaws.com
us-east-1	602401143452.dkr.ecr.us-east-1.amazonaws.com/
us-east-2	602401143452.dkr.ecr.us-east-2.amazonaws.com/
us-west-1	602401143452.dkr.ecr.us-west-1.amazonaws.com
us-west-2	602401143452.dkr.ecr.us-west-2.amazonaws.com/
ca-central-1	602401143452.dkr.ecr.ca-central-1.amazonaws.com

Região da AWS	Registro
ca-west-1	761377655185.dkr.ecr.ca-west-1.amazonaws.com
sa-east-1	602401143452.dkr.ecr.sa-east-1.amazonaws.com/
us-gov-east-1	151742754352.dkr.ecr.us-gov-east-1.amazonaws.com/
us-gov-west-1	013241004608.dkr.ecr.us-gov-west-1.amazonaws.com/

Complementos do Amazon EKS

Um complemento é um software que fornece recursos operacionais de suporte para aplicações Kubernetes, mas que não é específico da aplicação. Isso inclui software como atendentes de observação ou drivers do Kubernetes que permitem que o cluster interaja com recursos da AWS para rede, computação e armazenamento. O software complementar é normalmente criado e mantido pela comunidade Kubernetes, pelos provedores de nuvem como a AWS ou por fornecedores terceirizados. O Amazon EKS instala automaticamente complementos autogerenciados, como CoreDNS, `kube-proxy` e plug-in CNI da Amazon VPC para Kubernetes em todos os clusters. Observe que o complemento da CNI da VPC não é compatível com o Amazon EKS Hybrid Nodes e não é implantado em nós híbridos. Você pode alterar a configuração padrão dos complementos e atualizá-los quando desejar.

Os complementos do Amazon EKS fornecem instalação e gerenciamento de um conjunto de complementos selecionados especialmente para os clusters do Amazon EKS. Todos os complementos do Amazon EKS incluem os patches de segurança mais recentes e as correções de bugs, e são validados pela AWS para funcionar com o Amazon EKS. Os complementos do Amazon EKS permitem que você garanta consistentemente que os clusters do Amazon EKS estejam seguros e estáveis, além de reduzirem a quantidade de trabalho que você precisa fazer para instalar, configurar e atualizar complementos. Se um complemento autogerenciado, como o `kube-proxy`, já estiver em execução em seu cluster e estiver disponível como um complemento do Amazon EKS, você pode instalar o complemento `kube-proxy` do Amazon EKS para começar a se beneficiar dos recursos dos complementos do Amazon EKS.

Você pode atualizar campos específicos de configuração gerenciados pelo Amazon EKS por meio da API do Amazon EKS. Você também pode modificar os campos de configuração não gerenciados pelo Amazon EKS diretamente no cluster do Kubernetes, assim que o complemento for iniciado. Isso inclui a definição dos campos de configuração específicos para um complemento, quando aplicável. Essas alterações não serão substituídas pelo Amazon EKS depois de feitas. Isso é possível usando o recurso de aplicação do lado do servidor do Kubernetes. Para obter mais informações, consulte the section called “Campos que você pode personalizar”.

É possível usar os complementos do Amazon EKS com qualquer tipo de nó do Amazon EKS. Para obter mais informações, consulte Gerenciar computação.

Você pode adicionar, atualizar ou excluir os complementos do Amazon EKS usando a API do Amazon EKS, Console de gerenciamento da AWS, AWS CLI e `eksctl`. Você também pode criar complementos do Amazon EKS usando o AWS CloudFormation.

Considerações

Considere o seguinte ao usar complementos do Amazon EKS:

- Para configurar complementos para o cluster, a entidade principal do IAM deve ter permissões do IAM para trabalhar com complementos. Para obter mais informações, consulte as ações com `Addon` no nome, em Ações definidas pelo Amazon Elastic Kubernetes Service.
- Os complementos do Amazon EKS são executados nos nós que você provisionar ou configurar para o cluster. Os tipos de nós incluem instâncias do Amazon EC2, do Fargate e de nós híbridos.
- Você pode modificar campos que não são gerenciados pelo Amazon EKS para personalizar a instalação de um complemento do Amazon EKS. Para obter mais informações, consulte the section called “Campos que você pode personalizar”.
- Se você criar um cluster com o Console de gerenciamento da AWS, os complementos CoreDNS para Amazon EKS, `kube-proxy` para Amazon EKS e plug-in CNI da Amazon VPC para Kubernetes serão adicionados automaticamente ao cluster. Se você usar `eksctl` para criar o cluster com um arquivo `config`, o `eksctl` também poderá criar o cluster com complementos do Amazon EKS. Se você criar o cluster usando o `eksctl` sem o arquivo `config`, ou com qualquer outra ferramenta, os complementos autogerenciados CoreDNS, `kube-proxy` e plug-in CNI da Amazon VPC para Kubernetes serão instalados, em vez dos complementos do Amazon EKS. Você pode gerenciá-los você mesmo ou pode adicionar os complementos do Amazon EKS manualmente após a criação do cluster. Independentemente do método usado para criar o cluster, o complemento da CNI da VPC não será instalado em nós híbridos.

- O `ClusterRoleBinding` `eks:addon-cluster-admin` vincula o `ClusterRole` `cluster-admin` à identidade do `eks:addon-manager` do Kubernetes. O perfil tem as permissões necessárias para que a identidade do `eks:addon-manager` crie namespaces do Kubernetes e instale complementos nos namespaces. Se o `ClusterRoleBinding` `eks:addon-cluster-admin` for removido, o cluster do Amazon EKS continuará funcionando, mas o Amazon EKS não poderá mais gerenciar os complementos. Todos os clusters que começam com as seguintes versões da plataforma usam o novo `ClusterRoleBinding`.
- Um subconjunto de complementos do EKS da AWS foi validado para compatibilidade com o Amazon EKS Hybrid Nodes. Para obter mais informações, consulte a tabela de compatibilidade em the section called “ Complementos da AWS”.

Namespace personalizado para complementos

Para comunidade e complementos da AWS, você pode especificar opcionalmente um namespace personalizado durante a criação do complemento. Depois de instalar um complemento em um namespace específico, você deve removê-lo e recriá-lo para alterar seu namespace.

Se você não especificar um namespace, ele usará o namespace predefinido para o complemento.

Use namespaces personalizados para melhor organização e isolamento de objetos de complemento em seu cluster do EKS. Essa flexibilidade ajuda você a alinhar os complementos às suas necessidades operacionais e à estratégia de namespace existente.

Você pode definir um namespace personalizado ao criar um complemento. Para obter mais informações, consulte the section called “Criação de um complemento”.

Obtenha um namespace predefinido para complemento

O namespace predefinido para um complemento é o namespace no qual ele será instalado se você não especificar um.

Para obter o namespace predefinido para um complemento, use o seguinte comando:

```
aws eks describe-addon-versions --addon-name <addon-name> --query
 "addons[].defaultNamespace"
```

Resultado do exemplo:

```
[
```

```
    "kube-system"
]
```

Considerações sobre o Modo automático do Amazon EKS

O Modo automático do Amazon EKS inclui recursos que oferecem funcionalidades essenciais de cluster, incluindo:

- Rede de pods
- Rede de serviços
- DNS do cluster
- Ajuste de escala automático
- Armazenamento em bloco
- Controlador do balanceador de carga
- Agente da Identidade de Pods
- Agente de monitoramento de nós

Com a computação no Modo automático, muitos complementos do EKS comumente usados tornam-se redundantes, como:

- CNI da Amazon VPC
- kube-proxy
- CoreDNS
- Driver da CSI do Amazon EBS
- Atendente de Identidade de Pods do EKS

No entanto, se o cluster combinar o Modo automático com outras opções de computação, como instâncias autogerenciadas do EC2, grupos de nós gerenciados ou o AWS Fargate, esses complementos continuarão sendo necessários. A AWS aprimorou os complementos do EKS com regras de antiafinidade que garantem automaticamente que os pods de complementos sejam programados somente em tipos de computação compatíveis. Além disso, os usuários agora podem aproveitar a API `DescribeAddonVersions` de complementos do EKS para verificar os computeTypes compatíveis para cada complemento e suas versões específicas. Com o Modo automático do EKS, os controladores listados acima também são executados em uma infraestrutura

de propriedade da AWS. Portanto, você pode nem mesmo vê-los nas contas, a menos que esteja usando o Modo automático do EKS com outros tipos de computação. Nesse caso, você verá os controladores que instalou no cluster.

Caso esteja planejando habilitar o Modo automático do EKS em um cluster existente, talvez seja necessário atualizar a versão de determinados complementos. Para obter mais informações, consulte the section called “Versões de complementos necessárias” para o Modo automático do EKS.

Suporte

A AWS disponibiliza diferentes tipos de complementos, cada um com níveis de suporte variados.

- Complementos da AWS: esses complementos são desenvolvidos e têm suporte completo por parte da AWS.
  - Use um complemento da AWS para interagir com outros serviços da AWS, como o Amazon EFS.
  - Para obter mais informações, consulte the section called “ Complementos da AWS”.
- Complementos do AWS Marketplace: esses complementos são analisados pela AWS e têm suporte por parte de um parceiro independente da AWS.
  - Use um complemento do Marketplace para adicionar recursos valiosos e sofisticados ao seu cluster, como o monitoramento com o Splunk.
  - Para obter mais informações, consulte the section called “Complementos do Marketplace”.
- Complementos da comunidade: esses complementos são analisados pela AWS, mas têm suporte por parte da comunidade de código aberto.
  - Use um complemento da comunidade para reduzir a complexidade de instalação de softwares de código aberto amplamente utilizados, como o Kubernetes Metrics Server.
  - Os complementos da comunidade têm seus pacotes gerados a partir da fonte pela AWS. A AWS realiza apenas a validação de compatibilidade de versão para esses complementos da comunidade.
  - Para obter mais informações, consulte the section called “Complementos da comunidade”.

A tabela apresentada a seguir detalha o escopo de suporte para cada tipo de complemento:

Categoria	Recurso	Complementos da AWS	Complementos do AWS Marketplace	Complementos da comunidade
Desenvolvimento	Desenvolvido pela AWS	Sim	Não	Sim
Desenvolvimento	Validado pela AWS	Sim	Não	Sim*
Desenvolvimento	Validado por um parceiro da AWS	Não	Sim	Não
Manutenção	Analisado pela AWS	Sim	Sim	Sim
Manutenção	Corrigido pela AWS	Sim	Não	Sim
Manutenção	Corrigido por um parceiro da AWS	Não	Sim	Não
Distribuição	Publicado pela AWS	Sim	Não	Sim
Distribuição	Publicado por um parceiro da AWS	Não	Sim	Não
Suporte	Suporte básico para a instalação por parte da AWS	Sim	Sim	Sim
Suporte	Suporte completo por parte da AWS	Sim	Não	Não

Categoria	Recurso	Complementos da AWS	Complementos do AWS Marketplace	Complementos da comunidade
Suporte	Suporte completo por parte de um parceiro da AWS	Não	Sim	Não

*: a validação dos complementos da comunidade inclui apenas a compatibilidade da versão do Kubernetes. Por exemplo, se você instalar um complemento da comunidade em um cluster, a AWS verifica se ele é compatível com a versão do Kubernetes do seu cluster.

Os complementos do AWS Marketplace podem fazer o download de dependências adicionais de software de fontes externas fora da AWS. Essas dependências externas não são analisadas ou validadas pela AWS. Considere os requisitos de segurança ao implantar complementos do AWS Marketplace que busquem dependências externas.

Complementos da AWS

Os complementos do Amazon EKS a seguir estão disponíveis para serem criados no seu cluster. Você pode visualizar a lista mais atualizada de complementos disponíveis usando `eksctl`, Console de gerenciamento da AWS ou AWS CLI. Para ver todos os complementos disponíveis ou instalar um complemento, consulte the section called “Criação de um complemento”. Se um complemento precisar de permissões do IAM, você deve ter um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte the section called “Provedor OIDC do IAM”. Você pode criar ou excluir um complemento após tê-lo instalado. Para ter mais informações, consulte the section called “Atualização de um complemento” ou the section called “Remover um complemento”. Para obter mais informações sobre considerações específicas para a execução de complementos do EKS com o Amazon EKS Hybrid Nodes, consulte the section called “Configurar complementos”.

É possível usar qualquer um dos complementos do Amazon EKS a seguir.

Descrição	Saiba mais	Tipos de computação compatíveis
Fornecer uma rede VPC nativa para seu cluster	the section called “Plug-in CNI da Amazon VPC para Kubernetes”	EC2
Um servidor DNS flexível e extensível que pode servir como DNS de cluster do Kubernetes	the section called “CoreDNS”	EC2, Fargate, Modo automático do EKS, EKS Hybrid Nodes
Manter as regras de rede em cada nó do Amazon EC2	the section called “`Kube-proxy`”	EC2, EKS Hybrid Nodes
Fornecer armazenamento do Amazon EBS para seu cluster	the section called “Driver da CSI do Amazon EBS”	EC2
Fornecer armazenamento do Amazon EFS para seu cluster	the section called “Driver da CSI do Amazon EFS”	EC2, Modo automático do EKS
Fornecer armazenamento do Amazon FSx para Lustre para o cluster	the section called “Driver CSI do Amazon FSx”	EC2, Modo automático do EKS
Fornecer armazenamento do Amazon S3 para seu cluster	the section called “Driver da CSI do Mountpoint para Amazon S3”	EC2, Modo automático do EKS
Detectar problemas adicionais de integridade de nós	the section called “Agente de monitoramento de nós”	EC2, EKS Hybrid Nodes
Habilite o uso da funcional idade de captura de snapshots em drivers da CSI compatíve is, como o driver da CSI do Amazon EBS	the section called “Controlador de snapshots da CSI”	EC2, Fargate, Modo automático do EKS, EKS Hybrid Nodes

Descrição	Saiba mais	Tipos de computação compatíveis
A governança de tarefas do SageMaker HyperPod otimiza a alocação e o uso de recursos computacionais entre equipes em clusters do Amazon EKS, resolvendo ineficiências na priorização de tarefas e no compartilhamento de recursos.	the section called “Governança de tarefas do Amazon SageMaker HyperPod”	EC2 e modo automático do EKS
O complemento de observabilidade do Amazon SageMaker HyperPod fornece recursos abrangentes de monitoramento e observabilidade para clusters do HyperPod.	the section called “Complemento de observabilidade do Amazon SageMaker HyperPod”	EC2 e modo automático do EKS
O operador de treinamento do Amazon SageMaker HyperPod permite um treinamento distribuído eficiente em clusters do Amazon EKS com recursos avançados de agendamento e gerenciamento de recursos.	the section called “Operador de treinamento do Amazon SageMaker HyperPod”	EC2, Modo automático do EKS
Um atendente do Kubernetes responsável por coletar e enviar dados de fluxo de rede para o Amazon CloudWatch, possibilitando um monitoramento detalhado das conexões de TCP nos nós do cluster.	the section called “Agente do AWS Network Flow Monitor”	EC2, Modo automático do EKS

Descrição	Saiba mais	Tipos de computação compatíveis
Distribuição do projeto OpenTelemetry segura, pronta para produção e com suporte da AWS	the section called “AWS Distro para OpenTelemetry”	EC2, Fargate, Modo automático do EKS, EKS Hybrid Nodes
Serviço de monitoramento de segurança que analisa e processa fontes de dados fundamentais, incluindo eventos de gerenciamento do AWS CloudTrail e logs de fluxo do Amazon VPC. O Amazon GuardDuty também processa recursos, como logs de auditoria do Kubernetes e monitoramento de runtime	the section called “Agente do Amazon GuardDuty”	EC2, Modo automático do EKS
Serviço de monitoramento e observabilidade fornecido pela AWS. Esse complemento instala o Agente do CloudWatch e habilita o CloudWatch Application Signals e o CloudWatch Container Insights com observabilidade aprimorada para o Amazon EKS	the section called “Agente do Amazon CloudWatch Observability”	EC2, Modo automático do EKS, EKS Hybrid Nodes

Descrição	Saiba mais	Tipos de computação compatíveis
Capacidade de gerenciar credenciais para suas aplicações de forma semelhante a como os perfis de instância do EC2 fornecem credenciais para instâncias do EC2	the section called “Atendente de Identidade de Pods do EKS”	EC2, EKS Hybrid Nodes
Habilite o cert-manager para emitir certificados X.509 da AWS Private CA. Precisa do cert-manager.	the section called “Conector de CA Privado da AWS para Kubernetes”	EC2, Fargate, Modo automático do EKS, EKS Hybrid Nodes
Geração de métricas do Prometheus sobre a performance do dispositivo de rede SR-IOV	the section called “Exportador de métricas de rede SR-IOV”	EC2
Recupere segredos do AWS Secrets Manager e parâmetros do AWS Systems Manager Parameter Store e monte-os como arquivos em pods do Kubernetes.	the section called “Provedor de drivers CSI da AWS Secrets Store”	EC2, Modo automático do EKS, EKS Hybrid Nodes
Com os Spaces, é possível criar e gerenciar aplicações do JupyterLab e do Editor de Código para executar workloads de ML interativas.	the section called “Amazon SageMaker Spaces”	HyperPod

Plug-in CNI da Amazon VPC para Kubernetes

O complemento do plug-in CNI da Amazon VPC para Kubernetes para Amazon EKS é um plug-in de interface de rede de contêineres (CNI) do Kubernetes que fornece rede VPC nativa para o cluster. O tipo autogerenciado ou gerenciado desse complemento é instalado por padrão em cada nó do Amazon EC2. Para obter mais informações, consulte Plug-in de interface de rede de contêineres (CNI) do Kubernetes.

> **Note**
>
> Você não precisa instalar esse complemento nos clusters do Modo automático do Amazon EKS. Para obter mais informações, consulte the section called “Considerações sobre o Modo automático do Amazon EKS”.

O nome do complemento do Amazon EKS é `vpc-cni`.

Permissões obrigatórias do IAM

Este complemento usa o recurso de perfis do IAM para contas de serviço do Amazon EKS. Para obter mais informações, consulte the section called “Credenciais com IRSA”.

Se seu cluster usar a família `IPv4`, é necessário ter as permissões na AmazonEKS_CNI_Policy. Se o seu cluster utilizar a família `IPv6`, é necessário criar uma política do IAM com as permissões no modo IPv6. Você pode criar um perfil do IAM, anexar uma das políticas a ele e anotar a conta de serviço do Kubernetes usada pelo complemento com o comando a seguir.

Substitua *`my-cluster`* pelo nome do cluster e *`AmazonEKSVPCCNIRole`* pelo nome do seu perfil. Se seu cluster usar a família `IPv6`, substitua *`AmazonEKS_CNI_Policy`* pelo nome da política que você criou. Esse comando requer que você tenha o eksctl instalado no dispositivo. Caso precise usar uma ferramenta diferente para criar o perfil, anexe a política a ele e anote na conta de serviço do Kubernetes. Consulte the section called “Atribuir perfil do IAM”.

```
eksctl create iamserviceaccount --name aws-node --namespace kube-system --cluster my-
cluster --role-name AmazonEKSVPCCNIRole \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy --
approve
```

Informações sobre a atualização

Você só pode atualizar uma versão secundária de cada vez. Por exemplo, se a versão atual for `1.28.x-eksbuild.y` e você quiser atualizar para `1.30.x-eksbuild.y`, será necessário atualizar sua versão atual para `1.29.x-eksbuild.y` e depois atualizá-la novamente para a versão `1.30.x-eksbuild.y`. Para obter mais informações sobre como atualizar o complemento, consulte the section called “Atualizar (complemento do EKS)”.

CoreDNS

O complemento CoreDNS do Amazon EKS é um servidor DNS flexível e extensível que pode servir como DNS de cluster do Kubernetes. Por padrão, o tipo autogerenciado ou gerenciado desse complemento foi instalado quando você criou seu cluster. Quando você executa um cluster do Amazon EKS com pelo menos um nó, duas réplicas da imagem do CoreDNS são implantadas por padrão, independentemente do número de nós implantados em seu cluster. Os Pods CoreDNS fornecem resolução de nomes para todos os Pods no cluster. Você poderá implementar os pods do CoreDNS em nós do Fargate se o seu cluster incluir um perfil do Fargate com um namespace que corresponda ao namespace da implementação do CoreDNS. Para obter mais informações, consulte . the section called “Definir perfis”

> **Note**
>
> Você não precisa instalar esse complemento nos clusters do Modo automático do Amazon EKS. Para obter mais informações, consulte the section called “Considerações sobre o Modo automático do Amazon EKS”.

O nome do complemento do Amazon EKS é `coredns`.

Permissões obrigatórias do IAM

Este complemento não exige nenhuma permissão.

Mais informações

Para saber mais sobre o CoreDNS, consulte Usando o CoreDNS para Descoberta de Serviços e Personalizando o Serviço DNS na documentação do Kubernetes.

`Kube-proxy`

O complemento `Kube-proxy` do Amazon EKS mantém as regras de rede em cada nó do Amazon EC2. Esse componente viabiliza a comunicação de rede com os pods. O tipo autogerenciado ou gerenciado desse complemento é instalado por padrão em cada nó do Amazon EC2 em seu cluster.

> **Note**
>
> Você não precisa instalar esse complemento nos clusters do Modo automático do Amazon EKS. Para obter mais informações, consulte the section called “Considerações sobre o Modo automático do Amazon EKS”.

O nome do complemento do Amazon EKS é `kube-proxy`.

Permissões obrigatórias do IAM

Este complemento não exige nenhuma permissão.

Informações sobre a atualização

Antes de atualizar sua versão atual, considere os seguintes requisitos:

- `Kube-proxy` em um cluster do Amazon EKS tem a mesma política de compatibilidade e inclinação do Kubernetes.

Mais informações

Para saber mais sobre `kube-proxy`, consulte kube-proxy na documentação do Kubernetes.

Driver da CSI do Amazon EBS

O complemento do driver CSI do Amazon EBS para Amazon EKS é um plug-in Container Storage Interface (CSI) do Kubernetes que fornece armazenamento do Amazon EBS para o cluster.

> **Note**
>
> Você não precisa instalar esse complemento nos clusters do Modo automático do Amazon EKS. O Modo automático inclui um recurso de armazenamento em blocos. Para obter mais informações, consulte the section called “Implantar workload com estado”.

O nome do complemento do Amazon EKS é `aws-ebs-csi-driver`.

Permissões obrigatórias do IAM

Este complemento utiliza o recurso de perfis do IAM para contas de serviço do Amazon EKS. Para obter mais informações, consulte the section called “Credenciais com IRSA”. São necessárias as permissões na política gerenciada do AmazonEBSCSIDriverPolicy AWS. Você pode criar um perfil do IAM e anexar a política gerenciada a ela com o seguinte comando. Substitua *`my-cluster`* pelo nome do seu cluster e *`AmazonEKS_EBS_CSI_DriverRole`* pelo nome do seu perfil. Esse comando requer que você tenha o eksctl instalado no dispositivo. Se você precisar usar uma ferramenta diferente ou precisar usar uma chave KMS personalizada para criptografia, consulte the section called “Etapa 1: criar um perfil do IAM”.

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
    --approve
```

Mais informações

Para saber mais a respeito do complemento, consulte the section called “Amazon EBS”.

Driver da CSI do Amazon EFS

O complemento do driver CSI do Amazon EFS para Amazon EKS é um plug-in Container Storage Interface (CSI) do Kubernetes que fornece armazenamento do Amazon EFS para o cluster.

O nome do complemento do Amazon EKS é `aws-efs-csi-driver`.

Permissões obrigatórias do IAM

Permissões necessárias do IAM: este complemento utiliza perfis do IAM para recursos de contas de serviço do Amazon EKS. Para obter mais informações, consulte the section called “Credenciais com IRSA”. São necessárias as permissões na política gerenciada do AmazonEFSCSIDriverPolicy AWS. Você pode criar um perfil do IAM e anexar a política gerenciada a ele com o seguinte comando. Substitua *`my-cluster`* pelo nome do seu cluster e *`AmazonEKS_EFS_CSI_DriverRole`* pelo nome

do seu perfil. Esses comandos exigem que você tenha o eksctl instalado em seu dispositivo. Se precisar usar outra ferramenta, consulte the section called “Etapa 1: criar um perfil do IAM”.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
    --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

Mais informações

Para saber mais a respeito do complemento, consulte the section called “Amazon EFS”.

Driver CSI do Amazon FSx

O complemento do driver CSI do Amazon FSx para o Amazon EKS é um plug-in do Container Storage Interface (CSI) para Kubernetes que fornece armazenamento do Amazon FSx para Lustre para o cluster.

O nome do complemento do Amazon EKS é `aws-fsx-csi-driver`.

> **Note**
>
> - Instalações preexistentes do driver CSI do Amazon FSx no cluster podem causar falhas na instalação do complemento. Quando você tentar instalar a versão do complemento do Amazon EKS quando houver um driver CSI do FSx que não seja para o EKS, a instalação falhará devido a conflitos de recursos. Use o sinalizador `OVERWRITE` durante a instalação para resolver esse problema:
>
>   ```
>   aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
>    --resolve-conflicts OVERWRITE
>   ```

- O complemento do driver CSI do Amazon FSx para o EKS requer o atendente da Identidade de Pods do EKS para autenticação. Sem esse componente, o complemento falhará com o erro `Amazon EKS Pod Identity agent is not installed in the cluster`, impedindo operações de volume. Instale o atendente da Identidade de Pods antes ou depois de implantar o complemento do driver CSI do FSx. Para obter mais informações, consulte the section called “Configurar o atendente”.

Permissões obrigatórias do IAM

Este complemento utiliza o recurso de perfis do IAM para contas de serviço do Amazon EKS. Para obter mais informações, consulte the section called “Credenciais com IRSA”. As permissões na política AmazonFSxFullAccess gerenciada pela AWS são necessárias. Você pode criar um perfil do IAM e anexar a política gerenciada a ela com o seguinte comando. Substitua *`my-cluster`* pelo nome do seu cluster e *`AmazonEKS_FSx_CSI_DriverRole`* pelo nome do seu perfil. Esse comando exige a instalação do eksctl no seu dispositivo.

```
eksctl create iamserviceaccount \
    --name fsx-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_FSx_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/AmazonFSxFullAccess \
    --approve
```

Mais informações

Para saber mais a respeito do complemento, consulte the section called “Amazon FSx para Lustre”.

Driver da CSI do Mountpoint para Amazon S3

O complemento do driver CSI do Mountpoint para Amazon S3 para Amazon EKS é um plug-in Container Storage Interface (CSI) do Kubernetes que fornece armazenamento do Amazon S3 para o cluster.

O nome do complemento do Amazon EKS é `aws-mountpoint-s3-csi-driver`.

Permissões obrigatórias do IAM

Este complemento usa o recurso de perfis do IAM para contas de serviço do Amazon EKS. Para obter mais informações, consulte the section called “Credenciais com IRSA”.

O perfil do IAM criado exigirá uma política que conceda acesso ao S3. Siga as recomendações de permissões de IAM do Mountpoint ao criar a política. Como alternativa, você pode usar a política AmazonS3FullAccess gerenciada pela AWS, mas ela concede mais permissões do que as necessárias para o Mountpoint.

Você pode criar um perfil do IAM e anexar a política gerenciada a ele com os comandos a seguir. Substitua *my-cluster* pelo nome do seu cluster, *region-code* pelo código da região da AWS correto, *AmazonEKS_S3_CSI_DriverRole* pelo nome do seu perfil e *AmazonEKS_S3_CSI_DriverRole_ARN* pelo ARN do perfil. Esses comandos exigem que você tenha o eksctl instalado em seu dispositivo. Para obter instruções sobre como usar o console do IAM ou a AWS CLI, consulte the section called “Etapa 2: Criar uma função do IAM”.

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

Mais informações

Para saber mais a respeito do complemento, consulte the section called “Mountpoint para Amazon S3”.

Controlador de snapshots da CSI

O controlador de snapshots da Container Storage Interface (CSI) permite utilizar a funcionalidade de captura de snapshots em drivers da CSI compatíveis, como o driver da CSI do Amazon EBS.

O nome do complemento do Amazon EKS é `snapshot-controller`.

Permissões obrigatórias do IAM

Este complemento não exige nenhuma permissão.

Mais informações

Para saber mais a respeito do complemento, consulte the section called “Controlador de snapshots da CSI”.

Governança de tarefas do Amazon SageMaker HyperPod

A governança de tarefas do SageMaker HyperPod é um sistema de gerenciamento robusto projetado para simplificar a alocação de recursos e garantir a utilização eficiente dos recursos computacionais entre as equipes e os projetos para seus clusters do Amazon EKS. Isso permite que os administradores definam:

- Níveis de prioridade para diferentes tarefas
- Alocação de recursos computacionais para cada equipe
- Como as equipes compartilham e utilizam recursos ociosos
- Se uma equipe pode interromper suas próprias tarefas

A governança de tarefas do HyperPod também oferece observabilidade para clusters do Amazon EKS, proporcionando visibilidade em tempo real sobre a capacidade do cluster. Isso inclui a disponibilidade e o uso de computação, a alocação e a utilização por equipe, além de informações sobre o tempo de execução e de espera das tarefas, permitindo uma tomada de decisão informada e um gerenciamento proativo dos recursos.

O nome do complemento do Amazon EKS é `amazon-sagemaker-hyperpod-taskgovernance`.

Permissões obrigatórias do IAM

Este complemento não exige nenhuma permissão.

Mais informações

Para saber mais sobre o complemento, consulte Governança de tarefas do SageMaker HyperPod

Complemento de observabilidade do Amazon SageMaker HyperPod

O complemento de observabilidade do Amazon SageMaker HyperPod fornece recursos abrangentes de monitoramento e observabilidade para clusters do HyperPod. Esse complemento implanta e gerencia automaticamente componentes essenciais de monitoramento, incluindo exportador de nós, exportador DCGM, kube-state-metrics e exportador EFA. Ele coleta e encaminha métricas para uma instância do Amazon Managed Prometheus (AMP) designada pelo cliente e expõe um endpoint do OTLP para métricas personalizadas e ingestão de eventos de tarefas de treinamento de clientes.

O complemento se integra ao ecossistema HyperPod mais amplo, extraindo métricas de vários componentes, incluindo o complemento de governança de tarefas do HyperPod, operador de treinamento do HyperPod, Kubeflow e KEDA. Todas as métricas coletadas são centralizadas no Amazon Managed Prometheus, permitindo que os clientes obtenham uma visão unificada de observabilidade por meio dos painéis do Amazon Managed Grafana. Isso fornece visibilidade de ponta a ponta da integridade do cluster, da utilização de recursos e da performance da tarefa de treinamento em todo o ambiente do HyperPod.

O nome do complemento do Amazon EKS é `amazon-sagemaker-hyperpod-observability`.

Permissões obrigatórias do IAM

Este complemento usa o recurso de perfis do IAM para contas de serviço do Amazon EKS. Para obter mais informações, consulte the section called “Credenciais com IRSA”. As seguintes políticas gerenciadas são necessárias:

- `AmazonPrometheusRemoteWriteAccess`: para métricas de gravação remota do cluster para o AMP
- `CloudWatchAgentServerPolicy`: para gravação remota dos logs do cluster para o CloudWatch

Mais informações

Para saber mais sobre o complemento e seus recursos, consulte SageMaker HyperPod Observability.

Operador de treinamento do Amazon SageMaker HyperPod

O operador de treinamento do Amazon SageMaker HyperPod ajuda você a acelerar o desenvolvimento de modelos de IA generativa gerenciando o treinamento distribuído com eficiência em grandes clusters de GPU. Ele apresenta recursos inteligentes de recuperação de

falhas, detecção de tarefas suspensas e gerenciamento em nível de processo que minimizam as interrupções no treinamento e reduzem os custos. Diferentemente da infraestrutura de treinamento tradicional, que exige a reinicialização completa do trabalho quando ocorrem falhas, esse operador implementa a recuperação cirúrgica do processo para manter suas tarefas de treinamento funcionando sem problemas.

O operador também funciona com as funções de monitoramento e observabilidade da saúde do HyperPod, fornecendo visibilidade em tempo real da execução do treinamento e monitoramento automático de métricas críticas, como picos de perdas e degradação da throughput. Você pode definir políticas de recuperação por meio de configurações simples de YAML sem alterações no código, permitindo que você responda e se recupere rapidamente de estados de treinamento irrecuperáveis. Esses recursos de monitoramento e recuperação trabalham juntos para manter o desempenho ideal do treinamento e, ao mesmo tempo, minimizar a sobrecarga operacional.

O nome do complemento do Amazon EKS é `amazon-sagemaker-hyperpod-training-operator`.

Para obter mais informações, consulte Como usar o operador de treinamento do HyperPod no Guia do desenvolvedor do Amazon SageMaker.

Permissões obrigatórias do IAM

Esse complemento requer permissões do IAM e usa a Identidade de Pods do EKS.

A AWS sugere a política gerenciada `AmazonSageMakerHyperPodTrainingOperatorAccess`.

Para obter mais informações, consulte Instalar o operador de treinamento no Guia do desenvolvedor do Amazon SageMaker.

Mais informações

Para saber mais sobre o complemento, consulte o Operador de treinamento do SageMaker HyperPod.

Agente do AWS Network Flow Monitor

O atendente do Amazon CloudWatch Network Flow Monitor é uma aplicação Kubernetes que coleta estatísticas de conexões de TCP de todos os nós em um cluster e publica relatórios de fluxo de rede nas APIs de ingestão do Amazon CloudWatch Network Flow Monitor.

O nome do complemento do Amazon EKS é `aws-network-flow-monitoring-agent`.

Permissões obrigatórias do IAM

Este complemento requer permissões do IAM.

Você precisa anexar a política gerenciada `CloudWatchNetworkFlowMonitorAgentPublishPolicy` ao complemento.

Para obter mais informações sobre a configuração do IAM necessária, consulte a IAM Policy no repositório do atendente do Amazon CloudWatch Network Flow Monitor no GitHub.

Para obter mais informações sobre a política gerenciada, consulte CloudWatchNetworkFlowMonitorAgentPublishPolicy no Guia do usuário do Amazon CloudWatch.

Mais informações

Para saber mais sobre o complemento, consulte o repositório do atendente do Amazon CloudWatch Network Flow Monitor no GitHub.

Agente de monitoramento de nós

O complemento do atendente de monitoramento de nós do Amazon EKS pode detectar problemas adicionais de integridade dos nós. Esses sinais extras de integridade também podem ser aproveitados pelo recurso opcional de reparo automático de nós para substituir automaticamente os nós, conforme necessário.

> **Note**
>
> Você não precisa instalar esse complemento nos clusters do Modo automático do Amazon EKS. Para obter mais informações, consulte the section called “Considerações sobre o Modo automático do Amazon EKS”.

O nome do complemento do Amazon EKS é `eks-node-monitoring-agent`.

Permissões obrigatórias do IAM

Esse complemento não precisa de nenhuma permissão adicional.

Mais informações

Para obter mais informações, consulte the section called “Integridade de nós”.

AWS Distro para OpenTelemetry

O complemento AWS Distro for OpenTelemetry do Amazon EKS uma distribuição segura, pronta para produção e com suporte da AWS do projeto OpenTelemetry. Para obter mais informações, consulte AWS Distro for OpenTelemetry no GitHub.

O nome do complemento do Amazon EKS é `adot`.

Permissões obrigatórias do IAM

Este complemento só exigirá permissões do IAM se você estiver usando um dos recursos personalizados pré-configurados que podem ser escolhidos via configuração avançada.

Mais informações

Para obter mais informações, consulte Getting Started with AWS Distro for OpenTelemetry using EKS Add-Ons na documentação do AWS Distro para OpenTelemetry.

O ADOT exige que o complemento do `cert-manager` seja implantado no cluster como um pré-requisito. Caso contrário, ele não funcionará se for implantado diretamente usando a propriedade `cluster_addons` https://registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest. Para obter mais requisitos, consulte Requirements for Getting Started with AWS Distro for OpenTelemetry using EKS Add-Ons na documentação do AWS Distro para OpenTelemetry.

Agente do Amazon GuardDuty

O complemento do atendente Amazon GuardDuty para o Amazon EKS coleta eventos de runtime (acesso a arquivos, execução de processos, conexões de rede) dos nós do cluster do EKS para análise pelo Monitoramento de Runtime do GuardDuty. O GuardDuty em si (não o atendente) é o serviço de monitoramento de segurança que analisa e processa fontes de dados básicas, incluindo eventos de gerenciamento do AWS CloudTrail e logs de fluxo da Amazon VPC, bem como recursos, como logs de auditoria do Kubernetes e monitoramento de runtime.

O nome do complemento do Amazon EKS é `aws-guardduty-agent`.

Permissões obrigatórias do IAM

Este complemento não exige nenhuma permissão.

Mais informações

Para obter mais informações, consulte Monitoramento do runtime para clusters do Amazon EKS no Amazon GuardDuty.

- Para detectar possíveis ameaças de segurança em seus clusters do Amazon EKS, habilite o monitoramento de runtime do Amazon GuardDuty e implante o atendente de segurança do GuardDuty em seus clusters do Amazon EKS.

Agente do Amazon CloudWatch Observability

O complemento de atendente do Amazon CloudWatch Observability do Amazon EKS aprimora o serviço de monitoramento e observabilidade fornecido pela AWS. Esse complemento instala o Agente do CloudWatch e habilita o CloudWatch Application Signals e o CloudWatch Container Insights com observabilidade aprimorada para o Amazon EKS. Para obter mais informações, consulte Agente do Amazon CloudWatch.

O nome do complemento do Amazon EKS é `amazon-cloudwatch-observability`.

Permissões obrigatórias do IAM

Este complemento usa o recurso de perfis do IAM para contas de serviço do Amazon EKS. Para obter mais informações, consulte the section called “Credenciais com IRSA”. As permissões nas políticas gerenciadas pela AWS AWSXrayWriteOnlyAccess e CloudWatchAgentServerPolicy são necessárias. Você pode criar um perfil do IAM, anexar as políticas gerenciadas a ele e anotar a conta de serviço do Kubernetes usada pelo complemento com o comando a seguir. Substitua *my-cluster* pelo nome do seu cluster e *AmazonEKS_Observability_role* pelo nome do seu perfil. Esse comando exige a instalação do eksctl no seu dispositivo. Caso precise usar uma ferramenta diferente para criar o perfil, anexe a política a ele e anote na conta de serviço do Kubernetes. Consulte the section called “Atribuir perfil do IAM”.

```
eksctl create iamserviceaccount \
    --name cloudwatch-agent \
    --namespace amazon-cloudwatch \
    --cluster my-cluster \
    --role-name AmazonEKS_Observability_Role \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
    --approve
```

Mais informações

Para saber mais, consulte Instalar o Agente do CloudWatch.

Conector de CA Privado da AWS para Kubernetes

O Conector de CA Privado da AWS para Kubernetes é um complemento do cert-manager que permite que os usuários obtenham certificados da AWS Private Certificate Authority (AWS Private CA).

- O nome do complemento do Amazon EKS é `aws-privateca-connector-for-kubernetes`.
- O namespace do complemento é `aws-privateca-issuer`.

Esse complemento requer o `cert-manager`. O `cert-manager` está disponível no Amazon EKS como um complemento da comunidade. Para obter mais informações sobre esse complemento, consulte the section called “Cert Manager”. Para obter mais informações sobre como instalar complementos, consulte the section called “Criação de um complemento”.

Permissões obrigatórias do IAM

Esse complemento requer permissões do IAM.

Use a Identidade de Pods do EKS para anexar a política do IAM `AWSPrivateCAConnectorForKubernetesPolicy` à conta de serviço `aws-privateca-issuer` do Kubernetes. Para obter mais informações, consulte the section called “Usar Identidade de Pods”.

Para obter informações sobre as permissões necessárias, consulte AWSPrivateCAConnectorForKubernetesPolicy na Referência de políticas gerenciadas pela AWS.

Mais informações

Para obter mais informações, consulte o repositório do GitHub do Emissor de CA Privado da AWS para Kubernetes.

Para obter mais informações sobre a configuração do complemento, consulte values.yaml no repositório `aws-privateca-issuer` do GitHub. Confirme se a versão de values.yaml corresponde à versão do complemento instalado no cluster.

Esse complemento tolera o taint `CriticalAddonsOnly` usado pelo NodePool `system` do Modo automático do EKS. Para obter mais informações, consulte the section called “Executar complementos críticos”.

Atendente de Identidade de Pods do EKS

O complemento do atentende de Identidade de Pods do Amazon EKS permite gerenciar credenciais para suas aplicações de forma semelhante a como os perfis de instância do EC2 fornecem credenciais para instâncias do EC2.

> **Note**
>
> Você não precisa instalar esse complemento nos clusters do Modo automático do Amazon EKS. O Modo automático do Amazon EKS automático se integra à Identidade de Pods do EKS. Para obter mais informações, consulte the section called “Considerações sobre o Modo automático do Amazon EKS”.

O nome do complemento do Amazon EKS é `eks-pod-identity-agent`.

Permissões obrigatórias do IAM

O complemento do atendente da Identidade de Pods em si não exige um perfil do IAM. Ele usa as permissões do perfil do IAM do nó para o Amazon EKS para funcionar, mas não precisa de um perfil do IAM dedicado para o complemento.

Informações sobre a atualização

Você só pode atualizar uma versão secundária de cada vez. Por exemplo, se a versão atual for `1.28.x-eksbuild.y` e você quiser atualizar para `1.30.x-eksbuild.y`, será necessário atualizar sua versão atual para `1.29.x-eksbuild.y` e depois atualizá-la novamente para a versão `1.30.x-eksbuild.y`. Para obter mais informações sobre como atualizar o complemento, consulte the section called “Atualização de um complemento”.

Exportador de métricas de rede SR-IOV

O complemento Exportador de métricas de rede SR-IOV do Amazon EKS coleta e expõe métricas sobre dispositivos de rede SR-IOV no formato Prometheus. Ele permite o monitoramento da performance da rede SR-IOV nos nós bare metal do EKS. O exportador é executado como um DaemonSet em nós com interfaces de rede compatíveis com SR-IOV e exporta métricas que podem ser coletadas pelo Prometheus.

> **Note**
>
> Esse complemento requer nós com interfaces de rede compatíveis com SR-IOV.

Propriedade	Valor
Nome do complemento	`sriov-network-metrics-exporter`
Namespace	`monitoring`
Documentação	Repositório GitHub do Exportador de métricas de rede SR-IOV
Nome da conta de serviço	Nenhum
política do IAM gerenciada	Nenhum
Permissões do IAM personalizadas	Nenhum

Provedor de drivers CSI da AWS Secrets Store

O provedor da AWS para drivers CSI da Secrets Store é um complemento que possibilita recuperar segredos no AWS Secrets Manager e parâmetros no AWS Systems Manager Parameter Store, realizando a montagem desses itens como arquivos dentro de pods do Kubernetes.

Permissões obrigatórias do IAM

O complemento não exige permissões do IAM. No entanto, os pods da aplicação exigirão permissões do IAM para buscar segredos do AWS Secrets Manager e parâmetros do AWS Systems Manager Parameter Store. Após a instalação do complemento, o acesso deve ser configurado por meio de perfis do IAM para contas de serviço (IRSA, na sigla em inglês) ou Identidade de Pods do EKS. Para usar IRSA, consulte a documentação de configuração de IRSA do Secrets Manager. Para usar a Identidade de Pods do EKS, consulte a documentação de configuração da Identidade de Pods do Secrets Manager.

A AWS sugere a política gerenciada `AWSSecretsManagerClientReadOnlyAccess`.

Para obter mais informações sobre as permissões necessárias, consulte `AWSSecretsManagerClientReadOnlyAccess` na Referência de políticas gerenciadas pela AWS.

Mais informações

Para obter mais informações, consulte o repositório do GitHub secrets-store-csi-driver-provider-aws.

Para saber mais sobre o complemento, consulte a documentação do AWS Secrets Manager referente ao complemento.

Amazon SageMaker Spaces

O complemento Amazon SageMaker Spaces oferece a capacidade de executar IDEs e cadernos em clusters do EKS ou do HyperPod-EKS. Os administradores podem usar o console do EKS para instalar o complemento no cluster e definir configurações padrão de espaço, como imagens, recursos de computação, armazenamento local para configurações de cadernos (armazenamento adicional a ser anexado aos espaços), sistemas de arquivos e scripts de inicialização.

Os desenvolvedores de IA podem usar o kubectl para criar, atualizar e excluir espaços. Eles têm a flexibilidade de usar as configurações padrão fornecidas pelos administradores ou personalizar os ajustes. Os desenvolvedores de IA podem acessar os espaços no EKS ou no HyperPod-EKS usando os IDEs do VS Code locais e/ou os navegadores da web, que hospedam os IDEs do JupyterLab ou do CodeEditor em domínios de DNS personalizados configurados pelos administradores. Eles também podem usar o recurso de encaminhamento de portes do Kubernetes para acessar os espaços nos navegadores da web.

O nome do complemento do Amazon EKS é `amazon-sagemaker-spaces`.

Permissões obrigatórias do IAM

Esse complemento requer permissões do IAM. Para obter mais informações sobre a configuração do IAM necessária, consulte Configuração de permissões do IAM no Guia do desenvolvedor do Amazon SageMaker.

Mais informações

Para saber mais sobre o complemento e suas funcionalidades, consulte Cadernos do SageMaker AI no HyperPod no Guia do desenvolvedor do Amazon SageMaker.

Complementos da comunidade

É possível usar as APIs da AWS para instalar complementos da comunidade, como o Kubernetes Metrics Server. Você pode optar por instalar complementos da comunidade como complementos do Amazon EKS para reduzir a complexidade da manutenção do software em múltiplos clusters.

Por exemplo, você pode usar a API da AWS, a CLI ou o Console de Gerenciamento para instalar os complementos da comunidade. É possível instalar um complemento da comunidade durante a criação do cluster.

Você gerencia os complementos da comunidade de forma semelhante aos complementos do Amazon EKS existentes. Os complementos da comunidade são diferentes dos complementos existentes, pois contam com um escopo de suporte exclusivo.

> **Note**
>
> Usar complementos da comunidade fica a seu critério. Como parte do modelo de responsabilidade compartilhada entre você e a AWS, espera-se que você entenda o que está instalando no cluster quanto a esses plug-ins de terceiros. Você também é responsável pelos complementos da comunidade que atendam às necessidades de segurança do cluster. Para obter mais informações, consulte the section called “Suporte para software implantado no EKS”.

Os complementos da comunidade não são desenvolvidos pela AWS. A AWS apenas valida os complementos da comunidade quanto à compatibilidade de versões. Por exemplo, se você instalar um complemento da comunidade em um cluster, a AWS verifica se ele é compatível com a versão do Kubernetes do seu cluster.

No entanto, é importante ressaltar que a AWS não fornece suporte completo para os complementos da comunidade. A AWS oferece suporte somente para as operações de ciclo de vida realizadas usando as APIs da AWS, como a instalação e a exclusão de complementos.

Se você precisar de suporte para um complemento da comunidade, utilize os recursos do projeto existente. Por exemplo, é possível pode criar um problema no GitHub no repositório dedicado ao projeto.

Determinação do tipo de complemento

É possível usar a AWS CLI para determinar o tipo de um complemento do Amazon EKS.

Use o comando da CLI apresentado a seguir para recuperar informações sobre um complemento. Você pode substituir `metrics-server` pelo nome de qualquer outro complemento.

```
aws eks describe-addon-versions --addon-name metrics-server
```

Analise a saída da CLI para o campo `owner`.

```
{
    "addons": [
        {
            "addonName": "metrics-server",
            "type": "observability",
            "owner": "community",
            "addonVersions": [
```

Se o valor de `owner` for `community`, o complemento é classificado como complemento da comunidade. A AWS oferece suporte somente para instalar, atualizar e remover o complemento. Se você tiver dúvidas sobre a funcionalidade e sobre a operação do complemento, use recursos da comunidade, como problemas do GitHub.

Instalação ou atualização do complemento da comunidade

Você instala ou atualiza complementos da comunidade da mesma forma que os outros complementos do Amazon EKS.

- the section called “Criação de um complemento”
- the section called “Atualização de um complemento”
- the section called “Remover um complemento”

Complementos da comunidade disponíveis

Os complementos da comunidade apresentados a seguir estão disponíveis no Amazon EKS.

- the section called “Servidor de métricas do Kubernetes”
- the section called “kube-state-metrics”
- the section called “Exportador de nós do Prometheus”
- the section called “Cert Manager”
- the section called “DNS externo”

- the section called “Fluent Bit”

Servidor de métricas do Kubernetes

O Kubernetes Metrics Server fornece métricas de recursos de contêineres de forma escalável e eficiente para os pipelines internos de escalabilidade automática do Kubernetes. Ele coleta métricas de recursos dos Kubelets e as disponibiliza no servidor de API do Kubernetes por meio da API Metrics, permitindo seu uso pelo Horizontal Pod Autoscaler e pelo Vertical Pod Autoscaler.

Propriedade	Valor
Nome do complemento	`metrics-server`
Namespace	`kube-system`
Documentação	GitHub Readme
Nome da conta de serviço	Nenhum
política do IAM gerenciada	Nenhum
Permissões do IAM personalizadas	Nenhum

kube-state-metrics

Agente complementar para gerar e expor métricas em nível de cluster.

O estado dos objetos do Kubernetes na API do Kubernetes pode ser exposto como métricas. Um agente complementar denominado kube-state-metrics pode se conectar ao servidor da API do Kubernetes e expor um endpoint HTTP com métricas geradas do estado de objetos individuais no cluster. Ele expõe várias informações sobre o estado dos objetos, como rótulos e anotações, horários de inicialização e término, status ou a fase em que o objeto está atualmente.

Propriedade	Valor
Nome do complemento	`kube-state-metrics`
Namespace	`kube-state-metrics`

Propriedade	Valor
Documentação	Metrics for Kubernetes Object States na documentação do Kubernetes
Nome da conta de serviço	Nenhum
política do IAM gerenciada	Nenhum
Permissões do IAM personalizadas	Nenhum

Exportador de nós do Prometheus

Exportador Prometheus para métricas de hardware e sistema operacional expostas por kernels *NIX, escrito em Go com coletores de métricas plugáveis. O Prometheus Node Exporter expõe uma grande variedade de métricas relacionadas ao hardware e ao kernel.

Propriedade	Valor
Nome do complemento	`prometheus-node-exporter`
Namespace	`prometheus-node-exporter`
Documentação	Monitoring Linux host metrics with the Node Exporter na documentação do Prometheus
Nome da conta de serviço	Nenhum
política do IAM gerenciada	Nenhum
Permissões do IAM personalizadas	Nenhum

Cert Manager

O Cert Manager pode ser usado para gerenciar a criação e renovação de certificados.

Propriedade	Valor
Nome do complemento	`cert-manager`
Namespace	`cert-manager`
Documentação	Documentos do Cert Manager
Nome da conta de serviço	Nenhum
política do IAM gerenciada	Nenhum
Permissões do IAM personalizadas	Nenhum

DNS externo

O complemento DNS externo do EKS pode ser usado para gerenciar registros de DNS do Route 53 por meio de recursos do Kubernetes.

As permissões de DNS externo podem ser reduzidas para `route53:ChangeResourceRecordSets`, `route53:ListHostedZones` e `route53:ListResourceRecordSets` nas zonas hospedadas que você deseja gerenciar.

Propriedade	Valor
Nome do complemento	`external-dns`
Namespace	`external-dns`
Documentação	GitHub Readme
Nome da conta de serviço	`external-dns`
política do IAM gerenciada	`arn:aws:iam::aws:policy/AmazonRoute53FullAccess`
Permissões do IAM personalizadas	Nenhum

Fluent Bit

O Fluent Bit é um processador e encaminhador de logs leve e de alta performance. Ele permite coletar dados/logs de diferentes fontes, unificá-los e enviá-los para vários destinos, incluindo o Amazon CloudWatch Logs, o Amazon S3 e o Amazon Kinesis Data Firehose. O Fluent Bit foi projetado com a eficiência de recursos e performance em mente, tornando-o ideal para ambientes do Kubernetes.

Esse complemento não requer permissões do IAM na configuração padrão. No entanto, talvez seja necessário conceder permissões do IAM a esse complemento se você configurar um local de saída da AWS. Para obter mais informações, consulte the section called “Usar Identidade de Pods”.

Propriedade	Valor
Nome do complemento	`fluent-bit`
Namespace	`fluent-bit`
Documentação	Documentação do Fluent Bit
Nome da conta de serviço	`fluent-bit`
política do IAM gerenciada	Nenhum
Permissões do IAM personalizadas	Nenhum

Visualizar atribuições

Você pode baixar as atribuições de código aberto e as informações de licença para os complementos da comunidade.

1. Determine o nome e a versão do complemento para o qual você deseja baixar as atribuições.
2. Atualize o seguinte comando com o nome e a versão:

```
curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/<add-on-name>/<add-on-version>/attributions.zip
```

Por exemplo:

```
curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/kube-state-metrics/
v2.14.0-eksbuild.1/attributions.zip
```

3. Use o comando para baixar o arquivo.

Use este arquivo zip para visualizar informações sobre as atribuições de licença.

Complementos do AWS Marketplace

Além da lista anterior de complementos do Amazon EKS, você também pode adicionar uma ampla seleção de complementos de software operacional do Amazon EKS de fornecedores de software independentes. Escolha um complemento para saber mais sobre ele e sobre seus requisitos de instalação.

Accuknox

O nome do complemento é `accuknox_kubearmor` e o namespace é `kubearmor`. A Accuknox publica o complemento.

Para obter informações sobre o complemento, consulte Introdução ao KubeArmor na documentação da KubeArmor.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Akuity

O nome do complemento é `akuity_agent` e o namespace é `akuity`. A Akuity publica o complemento.

Para obter informações sobre como usar o complemento, consulte Instalar o Akuity Agent no Amazon EKS com o complemento Akuity do EKS na documentação do Akuity Platform.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Calyptia

O nome do complemento é `calyptia_fluent-bit` e o namespace é `calytia-fluentbit`. A Calyptia publica o complemento.

Para obter informações sobre o complemento, consulte Conceitos básicos do Calyptia Core Agent no site de documentação do Calyptia.

Nome da conta de serviço

O nome da conta de serviço é `clyptia-fluentbit`.

Política do IAM gerenciada da AWS

Este complemento usa a política gerenciada `AWSMarketplaceMeteringRegisterUsage`. Para obter mais informações sobre essa política, consulte AWSMarketplaceMeteringRegisterUsage no Guia de referência de políticas gerenciadas pela AWS.

Comando para criar o perfil do IAM necessário

O comando a seguir requer que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte the section called "Provedor OIDC do IAM". Substitua *`my-cluster`* pelo nome do seu cluster e *`my-calyptia-role`* pelo nome do seu perfil. Esse comando exige a instalação do eksctl no seu dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil e anotar na conta de serviço do Kubernetes, consulte the section called "Atribuir perfil do IAM".

```
eksctl create iamserviceaccount --name service-account-name  --namespace calyptia-
fluentbit --cluster my-cluster --role-name my-calyptia-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/
AWSMarketplaceMeteringRegisterUsage --approve
```

Cisco Observability Collector

O nome do complemento é `cisco_cisco-cloud-observability-collectors` e o namespace é `appdynamics`. A Cisco publica o complemento.

Para obter informações sobre o complemento, consulte Use the Cisco Cloud Observability AWS Marketplace Add-Ons na documentação do Cisco AppDynamics.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Cisco Observability Operator

O nome do complemento é `cisco_cisco-cloud-observability-operators` e o namespace é `appdynamics`. A Cisco publica o complemento.

Para obter informações sobre o complemento, consulte Use the Cisco Cloud Observability AWS Marketplace Add-Ons na documentação do Cisco AppDynamics.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

CLOUDSOFT

O nome do complemento é `cloudsoft_cloudsoft-amp` e o namespace é `cloudsoft-amp`. A CLOUDSOFT publica o complemento.

Para obter informações sobre o complemento, consulte Complemento do Amazon EKS na documentação da CLOUDSOFT.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Cribl

O nome do complemento é `cribl_cribledge` e o namespace é `cribledge`. A Cribl publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento Cribl do Amazon EKS na documentação da Cribl

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Dynatrace

O nome do complemento é `dynatrace_dynatrace-operator` e o namespace é `dynatrace`. A Dynatrace publica o complemento.

Para obter informações sobre o complemento, consulte Kubernetes monitoring na documentação do Dynatrace.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Datree

O nome do complemento é `datree_engine-pro` e o namespace é `datree`. A Datree publica o complemento.

Para obter informações sobre o complemento, consulte Integração do Amazon EKS na documentação da Datree.

Nome da conta de serviço

O nome da conta de serviço é datree-webhook-server-awsmp.

Política do IAM gerenciada da AWS

A política gerenciada é AWSLicenseManagerConsumptionPolicy. Para obter mais informações sobre essa política, consulte AWSLicenseManagerConsumptionPolicy no Guia de referência de políticas gerenciadas pela AWS.

Comando para criar o perfil do IAM necessário

O comando a seguir requer que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte the section called "Provedor OIDC do IAM". Substitua *my-cluster* pelo nome do seu cluster e *my-datree-role* pelo nome do seu perfil. Esse comando exige a instalação do eksctl no seu dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil e anotar na conta de serviço do Kubernetes, consulte the section called "Atribuir perfil do IAM".

```
eksctl create iamserviceaccount --name datree-webhook-server-awsmp --namespace datree
 --cluster my-cluster --role-name my-datree-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Permissões personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Datadog

O nome do complemento é `datadog_operator` e o namespace é `datadog-agent`. A Datadog publica o complemento.

Para obter informações sobre o complemento, consulte Instalação do Datadog Agent no Amazon EKS com o complemento Datadog Operator na documentação da Datadog.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Groundcover

O nome do complemento é `groundcover_agent` e o namespace é `groundcover`. A Groundcover publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento Groundcover do Amazon EKS na documentação da Groundcover.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

IBM Instana

O nome do complemento é `instana-agent` e o namespace é `instana-agent`. A IBM publica o complemento.

Para obter informações sobre o complemento, consulte Implementar a observabilidade das workloads do Amazon EKS usando o complemento Instana Amazon EKS e Monitorar e otimizar os custos do Amazon EKS com o IBM Instana e o Kubecost no blog da AWS.

O Instana Observability (Instana) oferece um complemento do Amazon EKS que implanta atendentes Instana nos clusters do Amazon EKS. Os clientes podem usar esse complemento para coletar e analisar dados de desempenho em tempo real para obter informações sobre suas aplicações em contêineres. O complemento Instana Amazon EKS fornece visibilidade em seus ambientes Kubernetes. Depois de implantado, o atendente Instana descobre automaticamente componentes em seus clusters do Amazon EKS, incluindo nós, namespaces, implantações, serviços e pods.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Grafana Labs

O nome do complemento é `grafana-labs_kubernetes-monitoring` e o namespace é `monitoring`. A Grafana Labs publica o complemento.

Para obter informações sobre o complemento, consulte Configurar o monitoramento do Kubernetes como um complemento com o Amazon EKS na documentação da Grafana Labs.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Guance

- Publicador: GUANCE
- Nome: `guance_datakit`
- Namespace: `datakit`
- Nome da conta de serviço: nenhuma conta de serviço é usada com esse complemento.
- Política gerenciada do IAM da AWS: nenhuma política gerenciada é usada com esse complemento.
- Permissões personalizadas do IAM: nenhuma permissão personalizada é usada com esse complemento.
- Instruções de configuração e uso: consulte Usar o complemento do Amazon EKS na documentação da Guance.

HA Proxy

O nome é `haproxy-technologies_kubernetes-ingress-ee` e o namespace é `haproxy-controller`. A HA Proxy publica o complemento.

Para obter informações sobre o complemento, consulte Integração do Amazon EKS na documentação da Datree.

Nome da conta de serviço

O nome da conta de serviço é `customer defined`.

Política do IAM gerenciada da AWS

A política gerenciada é AWSLicenseManagerConsumptionPolicy. Para obter mais informações sobre essa política, consulte AWSLicenseManagerConsumptionPolicy no Guia de referência de políticas gerenciadas pela AWS.

Comando para criar o perfil do IAM necessário

O comando a seguir requer que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte the section called "Provedor OIDC do IAM". Substitua *`my-cluster`* pelo nome do seu cluster e *`my-haproxy-role`*

pelo nome do seu perfil. Esse comando exige a instalação do eksctl no seu dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil e anotar na conta de serviço do Kubernetes, consulte the section called “Atribuir perfil do IAM”.

```
eksctl create iamserviceaccount --name service-account-name  --namespace haproxy-
controller --cluster my-cluster --role-name my-haproxy-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Permissões personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Kpow

O nome do complemento é `factorhouse_kpow` e o namespace é `factorhouse`. A Factorhouse publica o complemento.

Para obter informações sobre o complemento, consulte AWS Marketplace LM na documentação do Kpow.

Nome da conta de serviço

O nome da conta de serviço é `kpow`.

Política do IAM gerenciada da AWS

A política gerenciada é AWSLicenseManagerConsumptionPolicy. Para obter mais informações sobre essa política, consulte AWSLicenseManagerConsumptionPolicy no Guia de referência de políticas gerenciadas pela AWS.

Comando para criar o perfil do IAM necessário

O comando a seguir requer que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte the section called “Provedor OIDC do IAM”. Substitua *`my-cluster`* pelo nome do cluster e *`my-role-name`* pelo nome desejado para o perfil. Esse comando exige a instalação do eksctl no seu dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil e anotar na conta de serviço do Kubernetes, consulte the section called “Atribuir perfil do IAM”.

```
eksctl create iamserviceaccount --name kpow --namespace factorhouse --cluster my-
cluster --role-name my-kpow-role \
```

```
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Permissões personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Kubecost

O nome do complemento é `kubecost_kubecost` e o namespace é `kubecost`. A Kubecost publica o complemento.

Para obter mais informações sobre o complemento, consulte AWS Cloud Billing Integration na documentação do Kubecost.

Você deve ter os volumes de armazenamento do Kubernetes com o Amazon EBS instalados no cluster. Caso contrário, receberá um erro.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Kasten

O nome do complemento é `kasten_k10` e o namespace é `kasten-io`. A Kasten by Veeam publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o K10 na AWS usando o complemento do Amazon EKS na documentação da Kasten.

Você deve ter o driver CSI do Amazon EBS instalado no cluster com um `StorageClass` padrão.

Nome da conta de serviço

O nome da conta de serviço é `k10-k10`.

Política do IAM gerenciada da AWS

A política gerenciada é AWSLicenseManagerConsumptionPolicy. Para obter mais informações sobre essa política, consulte AWSLicenseManagerConsumptionPolicy no Guia de referência de políticas gerenciadas pela AWS.

Comando para criar o perfil do IAM necessário

O comando a seguir requer que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um, consulte the section called "Provedor OIDC do IAM". Substitua *my-cluster* pelo nome do seu cluster e *my-kasten-role* pelo nome do seu perfil. Esse comando exige a instalação do eksctl no seu dispositivo. Se você precisar usar uma ferramenta diferente para criar o perfil e anotar na conta de serviço do Kubernetes, consulte the section called "Atribuir perfil do IAM".

```
eksctl create iamserviceaccount --name k10-k10 --namespace kasten-io --cluster my-
cluster --role-name my-kasten-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Permissões personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Kong

O nome do complemento é `kong_konnect-ri` e o namespace é `kong`. A Kong publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento Kong Gateway do EKS na documentação da Kong.

Você deve ter os volumes de armazenamento do Kubernetes com o Amazon EBS instalados no cluster. Caso contrário, receberá um erro.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

LeakSignal

O nome do complemento é `leaksignal_leakagent` e o namespace é `leakagent`. A LeakSignal publica o complemento.

Para obter informações sobre o complemento, consulte https://www.leaksignal.com/docs/LeakAgent/Deployment/ AWS%20EKS%20Addon/[Instalar o complemento LeakAgent] na documentação do LeakSignal.

Você deve ter os volumes de armazenamento do Kubernetes com o Amazon EBS instalados no cluster. Caso contrário, receberá um erro.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

NetApp

O nome do complemento é `netapp_trident-operator` e o namespace é `trident`. A NetApp publica o complemento.

Para obter informações sobre o complemento, consulte Configure the Trident EKS add-on na documentação da NetApp.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

New Relic

O nome do complemento é `new-relic_kubernetes-operator` e o namespace é `newrelic`. A New Relic publica o complemento.

Para obter informações sobre o complemento, consulte Instalando o complemento do New Relic para EKS na documentação do New Relic.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Rafay

O nome do complemento é `rafay-systems_rafay-operator` e o namespace é `rafay-system`. A Rafay publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento Rafay do Amazon EKS na documentação da Rafay.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Rad Security

- Publicador: RAD SECURITY
- Nome: `rad-security_rad-security`
- Namespace: `ksoc`
- Nome da conta de serviço: nenhuma conta de serviço é usada com esse complemento.
- Política gerenciada do IAM da AWS: nenhuma política gerenciada é usada com esse complemento.
- Permissões personalizadas do IAM: nenhuma permissão personalizada é usada com esse complemento.
- Instruções de configuração e uso: consulte Instalação do Rad por meio do AWS Marketplace na documentação do Rad Security.

SolarWinds

- Publicador: SOLARWINDS
- Nome: `solarwinds_swo-k8s-collector-addon`
- Namespace: `solarwinds`
- Nome da conta de serviço: nenhuma conta de serviço é usada com esse complemento.
- Política gerenciada do IAM da AWS: nenhuma política gerenciada é usada com esse complemento.
- Permissões personalizadas do IAM: nenhuma permissão personalizada é usada com esse complemento.
- Instruções de configuração e uso: consulte Monitorar um cluster do Amazon EKS na documentação da SolarWinds.

Solo

O nome do complemento é `solo-io_istio-distro` e o namespace é `istio-system`. A Solo publica o complemento.

Para obter mais informações sobre o complemento, consulte Instalar o Istio na documentação da Solo.io.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Snyk

- Publicador: SNYK
- Nome: `snyk_runtime-sensor`
- Namespace: `snyk_runtime-sensor`
- Nome da conta de serviço: nenhuma conta de serviço é usada com esse complemento.
- Política gerenciada do IAM da AWS: nenhuma política gerenciada é usada com esse complemento.
- Permissões personalizadas do IAM: nenhuma permissão personalizada é usada com esse complemento.
- Instruções de configuração e uso: consulte o Sensor de runtime do Snyk nos documentos do usuário da Snyk.

Stormforge

O nome do complemento é `stormforge_optimize-Live` e o namespace é `stormforge-system`. A Stormforge publica o complemento.

Para obter informações sobre o complemento, consulte Instalação do StormForge Agent na documentação da StormForge.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

SUSE

- Publicador: SUSE
- Nome: `suse_observability-agent`
- Namespace: `suse-observability`
- Nome da conta de serviço: nenhuma conta de serviço é usada com esse complemento.
- Política gerenciada do IAM da AWS: nenhuma política gerenciada é usada com esse complemento.
- Permissões personalizadas do IAM: nenhuma permissão personalizada é usada com esse complemento.
- Instruções de configuração e uso: consulte Início rápido na documentação do SUSE.

Splunk

O nome do complemento é `splunk_splunk-otel-collector-chart` e o namespace é `splunk-monitoring`. A Splunk publica o complemento.

Para obter informações sobre o complemento, consulte Instalar o complemento Splunk do Amazon EKS na documentação da Splunk.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Teleport

O nome do complemento é `teleport_teleport` e o namespace é `teleport`. A Teleport publica o complemento.

Para obter mais informações sobre o complemento, consulte How Teleport Works na documentação do Teleport.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Tetrate

O nome do complemento é `tetrate-io_istio-distro` e o namespace é `istio-system`. A Tetrate Io publica o complemento.

Para obter informações sobre o complemento, consulte o site do Tetrate Istio Distro.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Upbound Universal Crossplane

O nome do complemento é `upbound_universal-crossplane` e o namespace é `upbound-system`. A Upbound publica o complemento.

Para obter informações sobre o complemento, consulte Upbound Universal Crossplane (UXP) na documentação da Upbound.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Upwind

O nome do complemento é `upwind` e o namespace é `upwind`. A Upwind publica o complemento.

Para obter mais informações sobre o complemento, consulte a Documentação da UpWind.

Nome da conta de serviço

Nenhuma conta de serviço é usada com esse complemento.

Política do IAM gerenciada da AWS

Nenhuma política gerenciada é usada com esse complemento.

Permissões do IAM personalizadas

Nenhuma permissão personalizada é usada com esse complemento.

Criar um complemento do Amazon EKS

Os complementos do Amazon EKS são um software complementar para clusters do Amazon EKS. Todos os complementos do Amazon EKS:

- Incluem os patches de segurança e as correções de erros mais recentes.
- São validados pela AWS para funcionar com o Amazon EKS.
- Reduzem o trabalho necessário para gerenciar software complementar.

Você pode criar um complemento do Amazon EKS usando o `eksctl`, o Console de gerenciamento da AWS ou a AWS CLI. Caso o complemento necessite de um perfil do IAM, consulte os detalhes

para o complemento específico em Complementos do Amazon EKS para obter informações sobre como criar o perfil.

Pré-requisitos

Antes de criar um complemento, faça o seguinte:

- O cluster deverá existir para que você possa criar um complemento para ele. Para obter mais informações, consulte the section called “Criar um cluster”.
- Verifique se o complemento exige um perfil do IAM. Para obter mais informações, consulte the section called “Verificação da compatibilidade”.
- Verifique se a versão do complemento do Amazon EKS é compatível com o cluster. Para obter mais informações, consulte the section called “Verificação da compatibilidade”.
- Verifique se a versão 0.190.0 ou posterior da ferramenta de linha de comando `eksctl` está instalada em seu computador ou no AWS CloudShell. Para obter mais informações, consulte Instalação no site do `eksctl`.

Procedimento

Você pode criar um complemento do Amazon EKS usando o `eksctl`, o Console de gerenciamento da AWS ou a AWS CLI. Se o complemento exigir um perfil do IAM, consulte os detalhes para esse complemento em específico em Complementos do Amazon EKS disponíveis da AWS para obter informações sobre como criar o perfil.

Criar complemento (eksctl)

1. Visualize os nomes dos complementos disponíveis para uma versão de cluster. Substitua *1.33* pela versão do cluster.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 | grep AddonName
```

   Veja abaixo um exemplo de saída.

```
"AddonName": "aws-ebs-csi-driver",
                        "AddonName": "coredns",
                        "AddonName": "kube-proxy",
                        "AddonName": "vpc-cni",
                        "AddonName": "adot",
                        "AddonName": "dynatrace_dynatrace-operator",
```

```
                        "AddonName": "upbound_universal-crossplane",
                        "AddonName": "teleport_teleport",
                        "AddonName": "factorhouse_kpow",
                        [...]
```

2. Visualize as versões disponíveis para o complemento que você deseja criar. Substitua *1.33* pela versão do cluster. Substitua *name-of-complemento* pelo nome do complemento para o qual você deseja visualizar as versões. O nome deve ser um dos nomes retornados na etapa anterior.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-addon |
 grep AddonVersion
```

   A saída a seguir é um exemplo do que é retornado para o complemento denominado `vpc-cni`. Você pode ver que o complemento tem várias versões disponíveis.

```
"AddonVersions": [
    "AddonVersion": "v1.12.0-eksbuild.1",
    "AddonVersion": "v1.11.4-eksbuild.1",
    "AddonVersion": "v1.10.4-eksbuild.1",
    "AddonVersion": "v1.9.3-eksbuild.1",
```

   a. Determine se o complemento que você deseja criar é um complemento do Amazon EKS ou do AWS Marketplace. O AWS Marketplace tem complementos de terceiros que exigem que você conclua etapas adicionais para criar o complemento.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-
addon | grep ProductUrl
```

      Se nenhuma saída for retornada, o complemento será um complemento do Amazon EKS. Se a saída for retornada, então o complemento é um complemento do AWS Marketplace. A saída a seguir é para um complemento denominado `teleport_teleport`.

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?sku=3bda70bb-566f-4976-806c-
f96faef18b26"
```

      Você pode obter mais informações sobre o complemento no AWS Marketplace com o URL retornado. Se o complemento exigir uma assinatura, você poderá assinar o complemento por meio do AWS Marketplace. Se você for criar um complemento no AWS Marketplace, a entidade principal do IAM que estiver usando para criar o complemento deverá ter permissão

para criar o perfil vinculada ao serviço AWSServiceRoleForAWSLicenseManagerRole. Para obter informações sobre como atribuir permissões a uma entidade do IAM, consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do IAM.

3. Crie um complemento do Amazon EKS. Copie o comando e substitua *user-data* da seguinte forma:
   - Substitua *my-cluster* pelo nome do cluster.
   - Substitua *name-of-complemento* pelo nome do complemento que você deseja criar.
   - Se você quiser uma versão do complemento anterior à mais recente, substitua *latest* pelo número da versão retornada na saída de uma etapa anterior que você deseja usar.
   - Se o complemento usar um perfil de conta de serviço, substitua *111122223333* pelo ID da conta e substitua *role-name* pelo nome do perfil. Para obter instruções para criar um perfil para a conta de serviço, consulte a documentação do complemento que você está criando. Para obter uma lista de complementos, consulte the section called " Complementos da AWS". A especificação de um perfil da conta de serviço exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte the section called "Provedor OIDC do IAM".

     Se o complemento não usar um perfil da conta de serviço, exclua `--service-account-role-arnarn:aws:iam::111122223333:role/role-name`.
   - Esse exemplo de comando sobrescreve a configuração de qualquer versão autogerenciada do complemento existente, se houver alguma. Se você não quiser substituir a configuração de um complemento autogerenciado existente, remova a opção *--force*. Se você remover a opção e o complemento do Amazon EKS precisar sobrescrever a configuração de um complemento autogerenciado existente, a criação do complemento do Amazon EKS falhará com uma mensagem de erro para ajudar a resolver o conflito. Antes de especificar essa opção, certifique-se de que o complemento Amazon EKS não gerencie as configurações que você precisa gerenciar, pois essas configurações são substituídas por essa opção.

```
eksctl create addon --cluster my-cluster --name name-of-addon --version latest \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --force
```

   Você pode ver uma lista de todas as opções disponíveis para o comando.

```
eksctl create addon --help
```

Para obter mais informações sobre as opções disponíveis, consulte Complementos na documentação do `eksctl`.

Criar complemento (console do AWS)

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome do cluster para o qual deseja criar o complemento.
4. Escolha a guia Add-ons (Complementos).
5. Escolha Obter mais complementos.
6. Na página Selecionar complementos, escolha os complementos que você deseja adicionar ao cluster. Você pode adicionar quantos complementos do Amazon EKS e do AWS Marketplace desejar.

   Para os complementos do AWS Marketplace, a entidade principal do IAM que você está usando para criar o complemento deve ter permissões para ler os direitos do complemento no AWS LicenseManager. AWS O LicenseManager requer o perfil vinculado ao serviço (SLR), AWSServiceRoleForAWSLicenseManagerRole que permite que os recursos da AWS gerenciem licenças em seu nome. O SLR é exigido uma única vez para cada conta, e você não precisará criar SLRs separados para cada complemento nem para cada cluster. Para obter informações sobre como atribuir permissões a uma entidade principal do IAM consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do IAM.

   Se os complementos do AWS Marketplace que você deseja instalar não estiverem listados, será possível clicar na numeração da página para visualizar resultados de páginas adicionais ou pesquisar na caixa de pesquisa. Nas Opções de filtragem, também é possível filtrar por categoria, fornecedor ou modelo de preço e, em seguida, escolher os complementos nos resultados da pesquisa. Após selecionar os complementos que você deseja instalar, escolha Next (Avançar).
7. Na página Definir as configurações dos complementos selecionados, faça o seguinte:
   a. Escolha Exibir opções de assinatura para abrir o formulário Opções de assinatura. Revise as seções Detalhes de preços e Aspectos jurídicos e escolha o botão Inscrever-se para continuar.
   b. Em Versão, escolha a versão que você deseja usar. Recomendamos a versão marcada como latest (a mais recente), a menos que o complemento individual que você está criando recomende outra versão. Para determinar se um complemento tem uma versão recomendada,

consulte a documentação do complemento que você está criando. Para obter uma lista de complementos, consulte the section called “ Complementos da AWS”.

c. Há duas opções para configurar perfis para complementos: perfil do IAM das Identidade de Pods do EKS e perfis do IAM para contas de serviço (IRSA). Siga a etapa apropriada para sua opção preferida abaixo. Se todos os complementos que você selecionou mostrarem Exige assinatura em Status, selecione Próximo. Não é possível configurar esses complementos até que você se torne assinante deles após a criação do cluster. Para os complementos que não mostram Exige assinatura em Status, faça o seguinte:

   i. Para Perfil do IAM de Identidade de Pods para a conta de serviço, é possível usar um perfil do IAM de Identidade de Pods do EKS existente ou criar um usando o botão Criar perfil recomendado. Esse campo só fornecerá opções com a política de confiança apropriada. Quando não há nenhum perfil para seleção, é porque você não tem nenhum perfil existente com uma política de confiança correspondente. Para configurar um perfil do IAM de Identidade de Pods do EKS para contas de serviço do complemento selecionado, escolha Criar perfil recomendado. O assistente de criação de perfis se abrirá em uma janela separada. O assistente preencherá automaticamente as informações do perfil da maneira a seguir. Para cada complemento em que você deseja criar o perfil do IAM de Identidade de Pods do EKS, conclua as etapas no assistente do IAM da forma a seguir.

      - Na etapa Selecionar entidade confiável, a opção de serviço da AWS para o EKS e o caso de uso para EKS: Identidade de Pods já vêm selecionados previamente, e a política de confiança apropriada será preenchida automaticamente para o complemento. Por exemplo, o perfil será criado com a política de confiança apropriada contendo a entidade principal do IAM pods.eks.amazonaws.com, conforme detalhado em the section called “Benefícios dos EKS Pod Identities”. Escolha Próximo.
      - Na etapa Adicionar permissões, a política gerenciada apropriada para a política de perfil é pré-selecionada para o complemento. Por exemplo, para o complemento CNI da Amazon VPC, o perfil será criado com a política gerenciada `AmazonEKS_CNI_Policy`, conforme detalhado em the section called “Plug-in CNI da Amazon VPC para Kubernetes”. Escolha Próximo.
      - Na etapa Nomear, revisar e criar, em Nome do perfil, o nome do perfil padrão é preenchido automaticamente para o complemento. Por exemplo, para o complemento CNI da Amazon VPC, o perfil será criado com a o nome AmazonEKSPodIdentityAmazonVPCCNIRole. Em Descrição, a descrição padrão é preenchida automaticamente com a descrição apropriada para o complemento. Por exemplo, para o complemento CNI do Amazon VPC, o perfil será criado com a descrição Permite que pods em execução no cluster Amazon EKS acessem

recursos da AWS. Em Política de confiança, visualize a política de confiança preenchida para o complemento. Selecione Criar perfil.

OBSERVAÇÃO: manter o nome do perfil padrão permite que o EKS pré-selecione o perfil para complementos em novos clusters ou ao adicionar complementos a clusters existentes. Você ainda pode substituir esse nome, e o perfil estará disponível para o complemento em seus clusters, mas o perfil precisará ser selecionado manualmente no menu suspenso.

ii. Para complementos que não tenham Requer assinatura em seu Status e para os quais você deseja configurar perfis usando o IRSA, consulte a documentação do complemento que você está criando para criar uma política do IAM e anexá-la a um perfil. Para obter uma lista de complementos, consulte the section called " Complementos da AWS". Para selecionar um perfil do IAM, você precisa ter um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte the section called "Provedor OIDC do IAM".

iii. Escolha Optional configuration settings (Configurações opcionais).

iv. Se o complemento exigir configuração, insira-a na caixa Valores de configuração. Para determinar se o complemento exige informações de configuração, consulte a documentação do complemento que você está criando. Para obter uma lista de complementos, consulte the section called " Complementos da AWS".

v. Selecione uma das opções disponíveis em Método de resolução de conflitos. Se você escolher Substituir como Método de resolução de conflitos, uma ou mais configurações do complemento existente poderão ser substituídas pelas configurações do complemento do Amazon EKS. Se você não habilitar esta opção e houver um conflito com suas configurações existentes, a operação falhará. É possível usar a mensagem de erro resultante para solucionar o conflito. Antes de escolher essa opção, certifique-se de que o complemento do Amazon EKS não gerencie as configurações que você precisa autogerenciar.

vi. Se você quiser instalar o complemento em um namespace específico, insira-o no campo Namespace. Para complementos da AWS e da comunidade, você pode definir um namespace Kubernetes personalizado para instalar o complemento. Para obter mais informações, consulte the section called "Namespace personalizado para complementos".

vii. Escolha Próximo.

8. Na página Adicionar tags, escolha Criar. Depois que a instalação dos complementos for concluída, você verá os complementos instalados.

9. Se algum dos complementos que você instalou exigir uma assinatura, conclua as seguintes etapas:
   a. Escolha o botão Assinar no canto inferior direito do complemento. Você será direcionado para a página do complemento no AWS Marketplace. Leia as informações sobre o complemento, como a visão geral do produto e as informações sobre preços.
   b. Selecione o botão Continue to Subscribe (Continuar a assinar) no canto superior direito da página do complemento.
   c. Leia todos os Terms and Conditions (Termos e condições). Se você concordar com eles, escolha Accept Terms (Aceitar termos). O processamento da assinatura pode levar alguns minutos. Enquanto a assinatura estiver sendo processada, o botão Return to Amazon EKS Console (Retornar ao console do Amazon EKS) estará acinzentado.
   d. Depois que o processamento da assinatura for concluído, o botão Return to Amazon EKS Console (Retornar ao console do Amazon EKS) não estará mais acinzentado. Escolha o botão para voltar à guia Add-ons (Complementos) do console Amazon EKS para o cluster.
   e. Para o complemento que você assinou, escolha Remove and reinstall (Remover e reinstalar) e depois escolha Reinstall add-on(Reinstalar complemento). A instalação do complemento pode levar vários minutos. Quando a instalação estiver concluída, você poderá configurar o complemento.

Criar complemento (AWS CLI)

1. Você precisa da versão `2.12.3` ou posterior ou da versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
2. Determine quais complementos estão disponíveis. Você pode ver todos os complementos disponíveis, seu tipo e seu editor. Você também pode ver o URL dos complementos disponíveis no AWS Marketplace. Substitua *`1.33`* pela versão do cluster.

```
aws eks describe-addon-versions --kubernetes-version 1.33 \
```

```
    --query 'addons[].{MarketplaceProductUrl: marketplaceInformation.productUrl,
 Name: addonName, Owner: owner Publisher: publisher, Type: type}' --output table
```

Veja abaixo um exemplo de saída.

```
-----------------------------------------------------------------------------------------------
|
 DescribeAddonVersions
  |
+-------------------------------------------------------------
+------------------------------+------------------+-------------
+---------------------+
|                     MarketplaceProductUrl                     |             Name
          |      Owner       |  Publisher   |        Type         |
+-------------------------------------------------------------
+------------------------------+------------------+-------------
+---------------------+
|  None                                                         |  aws-ebs-csi-driver
          |  aws             |  eks         |  storage            |
|  None                                                         |  coredns
          |  aws             |  eks         |  networking         |
|  None                                                         |  kube-proxy
          |  aws             |  eks         |  networking         |
|  None                                                         |  vpc-cni
          |  aws             |  eks         |  networking         |
|  None                                                         |  adot
          |  aws             |  eks         |  observability      |
| https://aws.amazon.com/marketplace/pp/prodview-brb73nceicv7u |
 dynatrace_dynatrace-operator |  aws-marketplace |  dynatrace   |  monitoring
 |
| https://aws.amazon.com/marketplace/pp/prodview-uhc2iwi5xysoc |  upbound_universal-
crossplane |  aws-marketplace |  upbound     |  infra-management   |
| https://aws.amazon.com/marketplace/pp/prodview-hd2ydsrgqy4li |  teleport_teleport
         |  aws-marketplace |  teleport    |  policy-management  |
| https://aws.amazon.com/marketplace/pp/prodview-vgghgqdsplhvc |  factorhouse_kpow
         |  aws-marketplace |  factorhouse |  monitoring         |
|  [...]                                                        |  [...]
          |  [...]           |  [...]       |  [...]              |
+-------------------------------------------------------------
+------------------------------+------------------+-------------
+---------------------+
```

A sua saída pode ser diferente. Neste exemplo de saída, há três complementos diferentes disponíveis do tipo `networking` e cinco complementos com um editor do tipo `eks`. Os complementos com `aws-marketplace` na coluna `Owner` podem requerer uma assinatura antes que você possa instalá-los. Você pode visitar a URL para obter mais informações e fazer uma assinatura do complemento.

3. Você pode ver quais versões estão disponíveis para cada complemento. Substitua *`1.33`* pela versão do cluster, e substitua *`vpc-cni`* pelo nome de um complemento retornado na etapa anterior.

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

Veja abaixo um exemplo de saída.

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

A versão com `True` na coluna `Defaultversion` é a versão com a qual o complemento foi criado, por padrão.

4. (Opcional) Encontre as opções de configuração para o complemento escolhido executando o seguinte comando:

```
aws eks describe-addon-configuration --addon-name vpc-cni --addon-version v1.12.0-
eksbuild.1
```

```
{
    "addonName": "vpc-cni",
    "addonVersion": "v1.12.0-eksbuild.1",
```

```
    "configurationSchema": "{\"$ref\":\"#/definitions/VpcCni\",\"$schema\":\"http://
json-schema.org/draft-06/schema#\",\"definitions\":{\"Cri\":{\"additionalProperties
\":false,\"properties\":{\"hostPath\":{\"$ref\":\"#/definitions/HostPath\"}},\"title
\":\"Cri\",\"type\":\"object\"},\"Env\":{\"additionalProperties\":false,\"properties
\":{\"ADDITIONAL_ENI_TAGS\":{\"type\":\"string\"},\"AWS_VPC_CNI_NODE_PORT_SUPPORT
\":{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_ENI_MTU\":{\"format\":
\"integer\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CONFIGURE_RPFILTER\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":{\"format\":
\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOGLEVEL\":{\"type\":\"string\"},
\"AWS_VPC_K8S_CNI_LOG_FILE\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_RANDOMIZESNAT
\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_VETHPREFIX\":{\"type\":\"string\"},
\"AWS_VPC_K8S_PLUGIN_LOG_FILE\":{\"type\":\"string\"},\"AWS_VPC_K8S_PLUGIN_LOG_LEVEL
\":{\"type\":\"string\"},\"DISABLE_INTROSPECTION\":{\"format\":\"boolean\",\"type
\":\"string\"},\"DISABLE_METRICS\":{\"format\":\"boolean\",\"type\":\"string
\"},\"DISABLE_NETWORK_RESOURCE_PROVISIONING\":{\"format\":\"boolean\",\"type
\":\"string\"},\"ENABLE_POD_ENI\":{\"format\":\"boolean\",\"type\":\"string
\"},\"ENABLE_PREFIX_DELEGATION\":{\"format\":\"boolean\",\"type\":\"string\"},
\"WARM_ENI_TARGET\":{\"format\":\"integer\",\"type\":\"string\"},\"WARM_PREFIX_TARGET
\":{\"format\":\"integer\",\"type\":\"string\"}},\"title\":\"Env\",\"type\":
\"object\"},\"HostPath\":{\"additionalProperties\":false,\"properties\":{\"path
\":{\"type\":\"string\"}},\"title\":\"HostPath\",\"type\":\"object\"},\"Limits
\":{\"additionalProperties\":false,\"properties\":{\"cpu\":{\"type\":\"string
\"},\"memory\":{\"type\":\"string\"}},\"title\":\"Limits\",\"type\":\"object\"},
\"Resources\":{\"additionalProperties\":false,\"properties\":{\"limits\":{\"$ref\":
\"#/definitions/Limits\"},\"requests\":{\"$ref\":\"#/definitions/Limits\"}},\"title
\":\"Resources\",\"type\":\"object\"},\"VpcCni\":{\"additionalProperties\":false,
\"properties\":{\"cri\":{\"$ref\":\"#/definitions/Cri\"},\"env\":{\"$ref\":\"#/
definitions/Env\"},\"resources\":{\"$ref\":\"#/definitions/Resources\"}},\"title\":
\"VpcCni\",\"type\":\"object\"}}}"
}
```

A saída é um esquema JSON padrão.

Aqui está um exemplo de valores de configuração válidos, no formato JSON, que funcionam com o esquema acima.

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
```

```
}
```

Aqui está um exemplo de valores de configuração válidos, no formato YAML, que funcionam com o esquema acima.

```
resources:
  limits:
    cpu: 100m
```

5. Determine se o complemento requer permissões do IAM. Nesse caso, você precisa (1) determinar se deseja usar identidades de pods do EKS ou perfis do IAM para contas de serviço (IRSA), (2) determinar o ARN do perfil do IAM a ser usado com o complemento e (3) determinar o nome da conta do serviço Kubernetes usada pelo complemento. Para obter mais informações, consulte the section called “Recuperar informações do IAM”.
   - O Amazon EKS sugerirá o uso de Identidades de Pods do EKS se o complemento oferecer suporte a esse recurso. Isso requer que o Agente de Identidade de Pods esteja instalado no seu cluster. Para obter mais informações sobre como usar Identidades de Pods com complementos, consulte the section called “Perfis do IAM”.
   - Se o complemento ou seu cluster não estiver configurado para Identidades de Pods do EKS, use o IRSA. Confirme se o IRSA está configurado no seu cluster.
   - Analise a documentação dos complementos do Amazon EKS para determinar se o complemento exige permissões do IAM e o nome da conta do serviço Kubernetes associada.
     a. Crie um complemento do Amazon EKS. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado:
   - Substitua *my-cluster* pelo nome do cluster.
   - Substitua *vpc-cni* por um nome de complemento retornado na saída da etapa anterior que você deseja criar.
   - Substitua *o version-number* pela versão retornada na saída da etapa anterior que você deseja usar.
   - Se você quiser instalar o complemento em um namespace personalizado do Kubernetes, adicione a opção `--namespace-config 'namespace=<my-namespace>`. Essa opção só está disponível para a AWS e complementos da comunidade. Para obter mais informações, consulte . the section called “Namespace personalizado para complementos”

- Se o complemento não exigir permissões do IAM, exclua *<service-account-configuration>*.
- Execute um destes procedimentos:
  - Se o complemento (1) exigir permissões de IAM e (2) seu cluster usar identidades de pod EKS, substitua *<service-account-configuration>* pela seguinte associação de identidade de pod. Substitua *<service-account-name>* pelo nome da conta de serviço usada pelo complemento. Substitua *<role-arn>* pelo ARN de um perfil do IAM. O perfil deve ter a política de confiança exigida pelas Identidades de Pods do EKS.

    ```
    --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
    ```

  - Se o complemento (1) exigir permissões de IAM e (2) seu cluster usar IRSA, substitua *<service-account-configuration>* pela seguinte configuração de IRSA. Substitua *111122223333* pelo ID da sua conta e *role-name* pelo nome de um perfil IAM existente que você criou. Para obter instruções para criar o perfil, consulte a documentação do complemento que você está criando. Para obter uma lista de complementos, consulte the section called “ Complementos da AWS”. A especificação de um perfil da conta de serviço exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte the section called “Provedor OIDC do IAM”.

    ```
    --service-account-role-arn arn:aws::iam::111122223333:role/role-name
    ```

- Esses comandos de exemplo substituem a opção `--configuration-values` de qualquer versão autogerenciada existente do complemento, se houver. Substitua isso pelos valores de configuração desejados, como uma string ou uma entrada de arquivo. Se você não deseja fornecer valores de configuração, exclua a opção `--configuration-values`. Se você não quiser que a AWS CLI substitua a configuração de um complemento autogerenciado existente, remova a opção *--resolve-conflicts OVERWRITE*. Se você remover a opção e o complemento do Amazon EKS precisar sobrescrever a configuração de um complemento autogerenciado existente, a criação do complemento do Amazon EKS falhará com uma mensagem de erro para ajudar a resolver o conflito. Antes de especificar essa opção, certifique-se de que o complemento Amazon EKS não gerencie as configurações que você precisa gerenciar, pois essas configurações são substituídas por essa opção.

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
    <service-account-configuration> --configuration-values '{"resources":
{"limits":{"cpu":"100m"}}}' --resolve-conflicts OVERWRITE
```

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
   <service-account-configuration> --configuration-values 'file://example.yaml' --
resolve-conflicts OVERWRITE
```

Para obter uma lista completa das opções disponíveis, consulte `create-addon` na Referência da linha de comando do Amazon EKS. Se o complemento que você criou tiver `aws-marketplace` listado na coluna `Owner` de uma etapa anterior, a criação poderá falhar e você poderá receber uma mensagem de erro semelhante a que se segue.

```
{
    "addon": {
        "addonName": "addon-name",
        "clusterName": "my-cluster",
        "status": "CREATE_FAILED",
        "addonVersion": "version",
        "health": {
            "issues": [
                {
                    "code": "AddonSubscriptionNeeded",
                    "message": "You are currently not subscribed to this add-on. To
 subscribe, visit the AWS Marketplace console, agree to the seller EULA, select the
 pricing type if required, then re-install the add-on"
                }
            ]
        }
    }
}
```

Se você receber um erro semelhante ao erro da saída anterior, visite a URL na saída de uma etapa anterior para assinar o complemento. Depois de fazer a assinatura, execute o comando `create-addon` novamente.

Atualizar um complemento do Amazon EKS

O Amazon EKS não atualiza automaticamente o complemento quando novas versões são lançadas nem depois que você atualiza o cluster para uma nova versão secundária do Kubernetes. Para atualizar um complemento para um cluster existente, você deve iniciar a atualização. Após a atualização ser iniciada, o Amazon EKS atualizará o complemento do Amazon EKS para você. Antes de atualizar um complemento, revise sua documentação atual. Para obter uma lista dos complementos disponíveis, consulte the section called " Complementos da AWS". Se o complemento exigir um perfil do IAM, consulte os detalhes para esse complemento em específico em Complementos do Amazon EKS disponíveis da AWS para obter informações sobre como criar o perfil.

Pré-requisitos

Antes de criar um complemento, faça o seguinte:

- Verifique se o complemento exige um perfil do IAM. Para obter mais informações, consulte the section called "Complementos do Amazon EKS".
- Verifique se a versão do complemento do Amazon EKS é compatível com o cluster. Para obter mais informações, consulte the section called "Verificação da compatibilidade".

Procedimento

Você pode atualizar um complemento do Amazon EKS usando `eksctl`, Console de gerenciamento da AWS ou AWS CLI.

Complemento de atualização (eksctl)

1. Determine os complementos e as versões dos complementos instalados atualmente no cluster. Substitua *`my-cluster`* pelo nome do cluster.

   ```
   eksctl get addon --cluster my-cluster
   ```

   Veja abaixo um exemplo de saída.

   ```
   NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
   coredns     v1.8.7-eksbuild.2    ACTIVE  0
   kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0                v1.23.8-eksbuild.2
   ```

```
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0                 v1.12.0-eksbuild.1,v1.11.4-
eksbuild.1,v1.11.3-eksbuild.1,v1.11.2-eksbuild.1,v1.11.0-eksbuild.1
```

Sua saída pode ser diferente, dependendo dos complementos e versões que você tiver no cluster. Você pode ver que, na saída do exemplo anterior, dois complementos existentes no cluster têm versões mais recentes disponíveis na coluna `UPDATE AVAILABLE`.

2. Atualize o complemento.
   a. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário:
      - Substitua *`my-cluster`* pelo nome do cluster.
      - Substitua *`region-code`* pela região da AWS em que seu cluster se encontra.
      - Substitua *`vpc-cni`* pelo nome de um complemento retornado na saída da etapa anterior que você deseja atualizar.
      - Se quiser atualizar para uma versão anterior à última versão disponível, substitua *`latest`* pelo número de versão retornado na saída da etapa anterior que você deseja usar. Alguns complementos têm versões recomendadas. Para obter mais informações, consulte a documentação do complemento que você está atualizando. Para obter uma lista de complementos, consulte a seção chamada the section called “ Complementos da AWS”.* Se o complemento usar uma conta de serviço do Kubernetes e um perfil do IAM, substitua *`111122223333`* pelo ID da sua conta e *`role-name`* pelo nome de um perfil do IAM existente que você tenha criado. Para obter instruções para criar o perfil, consulte a documentação do complemento que você está criando. Para obter uma lista de complementos, consulte the section called “ Complementos da AWS”. A especificação de um perfil da conta de serviço exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte the section called “Provedor OIDC do IAM”.

        Se o complemento não usar uma conta de serviço do Kubernetes e um perfil do IAM, exclua a linha `serviceAccountRoleARN: arn:aws:iam::`*`111122223333`*`:role/`*`role-name`* .
      - A opção *`preserve`* (manter) mantém os valores existentes para o complemento. Se você definiu valores personalizados para as configurações do complemento e não usar essa opção, o Amazon EKS sobrescreverá seus valores pelos valores padrão. Se você usar essa opção, recomendamos testar qualquer alteração de campo e valor em um cluster que não seja de produção antes de atualizar o complemento no cluster de produção. Se você alterar esse valor para `overwrite`, todas as configurações serão alteradas para os valores padrão

do Amazon EKS. Se você definiu valores personalizados para qualquer configuração, eles poderão ser sobrescritos pelos valores padrão do Amazon EKS. Se você alterar esse valor para none, o Amazon EKS não alterará o valor de nenhuma configuração, mas a atualização poderá falhar. Se a atualização falhar, você receberá uma mensagem de erro para ajudar a resolver o conflito.

```
cat >update-addon.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code

addons:
- name: vpc-cni
  version: latest
  serviceAccountRoleARN: arn:aws:iam::111122223333:role/role-name
  resolveConflicts: preserve
EOF
```

b. Execute o comando modificado para criar o arquivo `update-addon.yaml`.

c. Aplique o arquivo de configuração ao cluster.

```
eksctl update addon -f update-addon.yaml
```

Para obter mais informações sobre atualização de complementos, consulte Atualizar complementos na documentação do `eksctl`.

Atualizar complemento (console do AWS)

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome do cluster para o qual você deseja atualizar o complemento..
4. Escolha a guia Add-ons (Complementos).
5. Escolha o complemento que deseja atualizar.
6. Escolha Editar.
7. Na página Configurar ***nome do complemento*** , faça o seguinte:

a. Selecione a Versão que você deseja usar. O complemento pode ter uma versão recomendada. Para obter mais informações, consulte a documentação do complemento que você está atualizando. Para obter uma lista de complementos, consulte the section called “ Complementos da AWS”.

b. Há duas opções para configurar perfis para complementos: perfil do IAM das Identidade de Pods do EKS e perfis do IAM para contas de serviço (IRSA). Siga a etapa apropriada para sua opção preferida abaixo. Se todos os complementos que você selecionou mostrarem Exige assinatura em Status, selecione Próximo. Para os complementos que não mostram Exige assinatura em Status, faça o seguinte:

   i. Para Perfil do IAM de Identidade de Pods para a conta de serviço, é possível usar um perfil do IAM de Identidade de Pods do EKS existente ou criar um usando o botão Criar perfil recomendado. Esse campo só fornecerá opções com a política de confiança apropriada. Quando não há nenhum perfil para seleção, é porque você não tem nenhum perfil existente com uma política de confiança correspondente. Para configurar um perfil do IAM de Identidade de Pods do EKS para contas de serviço do complemento selecionado, escolha Criar perfil recomendado. O assistente de criação de perfis se abrirá em uma janela separada. O assistente preencherá automaticamente as informações do perfil da maneira a seguir. Para cada complemento em que você deseja criar o perfil do IAM de Identidade de Pods do EKS, conclua as etapas no assistente do IAM da forma a seguir.

      - Na etapa Selecionar entidade confiável, a opção de serviço da AWS para o EKS e o caso de uso para EKS: Identidade de Pods já vêm selecionados previamente, e a política de confiança apropriada será preenchida automaticamente para o complemento. Por exemplo, o perfil será criado com a política de confiança apropriada contendo a entidade principal do IAM pods.eks.amazonaws.com, conforme detalhado em the section called “Benefícios dos EKS Pod Identities”. Escolha Próximo.
      - Na etapa Adicionar permissões, a política gerenciada apropriada para a política de perfil é pré-selecionada para o complemento. Por exemplo, para o complemento CNI da Amazon VPC, o perfil será criado com a política gerenciada `AmazonEKS_CNI_Policy`, conforme detalhado em the section called “Plug-in CNI da Amazon VPC para Kubernetes”. Escolha Próximo.
      - Na etapa Nomear, revisar e criar, em Nome do perfil, o nome do perfil padrão é preenchido automaticamente para o complemento. Por exemplo, para o complemento CNI da Amazon VPC, o perfil será criado com a o nome AmazonEKSPodIdentityAmazonVPCCNIRole. Em Descrição, a descrição padrão é preenchida automaticamente com a descrição apropriada para o complemento. Por exemplo, para o complemento CNI do Amazon VPC, o perfil será

criado com a descrição Permite que pods em execução no cluster Amazon EKS acessem recursos da AWS. Em Política de confiança, visualize a política de confiança preenchida para o complemento. Selecione Criar perfil.

> **Note**
>
> Manter o nome do perfil padrão permite que o EKS pré-selecione o perfil para complementos em novos clusters ou ao adicionar complementos a clusters existentes. Você ainda pode substituir esse nome, e o perfil estará disponível para o complemento em seus clusters, mas o perfil precisará ser selecionado manualmente no menu suspenso.

ii. Para complementos que não tenham Requer assinatura em seu Status e para os quais você deseja configurar perfis usando o IRSA, consulte a documentação do complemento que você está criando para criar uma política do IAM e anexá-la a um perfil. Para obter uma lista de complementos, consulte the section called “ Complementos da AWS”. Para selecionar um perfil do IAM, você precisa ter um provedor OpenID Connect (OIDC) do IAM para o cluster. Para determinar se você já tem um ou se precisa criar um para o seu cluster, consulte the section called “Provedor OIDC do IAM”.

c. Expanda Definições de configuração opcionais.

d. Em Valores de configuração, insira informações de configuração específica do complemento. Para obter mais informações, consulte a documentação do complemento que você está atualizando. Para obter uma lista de complementos, consulte the section called “ Complementos da AWS”. .. Para Método de resolução de conflitos, selecione uma das opções. Se você definiu valores personalizados para as configurações do complemento, recomendamos escolher a opção Preserve (Manter). Se você não escolher essa opção, o Amazon EKS sobrescreverá seus valores com os valores padrão. Se você usar essa opção, recomendamos testar qualquer alteração de campo e valor em um cluster que não seja de produção antes de atualizar o complemento no cluster de produção. Se você alterar esse valor para overwrite, todas as configurações serão alteradas para os valores padrão do Amazon EKS. Se você definiu valores personalizados para qualquer configuração, eles poderão ser sobrescritos pelos valores padrão do Amazon EKS. Se você alterar esse valor para none, o Amazon EKS não vai alterar o valor de nenhuma configuração, mas a atualização poderá falhar. Se a atualização falhar, você receberá uma mensagem de erro para ajudar a resolver o conflito.

8. Escolha Salvar alterações.

Atualizar complemento (AWS CLI)

1. Você precisa da versão `2.12.3` ou posterior ou da versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
2. Veja uma lista de complementos instalados. Substitua *`my-cluster`* pelo nome do cluster.

   ```
   aws eks list-addons --cluster-name my-cluster
   ```

   Veja abaixo um exemplo de saída.

   ```
   {
       "addons": [
           "coredns",
           "kube-proxy",
           "vpc-cni"
       ]
   }
   ```

3. Visualize a versão atual do complemento que você deseja atualizar. Substitua *`my-cluster`* pelo nome do cluster e *`vpc-cni`* pelo nome do complemento que você deseja atualizar.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    "addon.addonVersion" --output text
   ```

   Veja abaixo um exemplo de saída.

   ```
   v1.10.4-eksbuild.1
   ```

4. Determine quais versões do complemento estão disponíveis para a versão do cluster. Substitua *`1.33`* pela versão do seu cluster e *`vpc-cni`* pelo nome do complemento que você deseja atualizar.

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

Veja abaixo um exemplo de saída.

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

A versão com `True` na coluna `Defaultversion` é a versão com a qual o complemento foi criado, por padrão.

5. Atualize seu complemento. Copie o conteúdo a seguir no seu dispositivo. Faça as seguintes modificações no comando, conforme necessário, e execute o comando modificado. Para obter mais informações sobre esse comando, consulte update-complemento na Referência de linha de comando do Amazon EKS.
   - Substitua *my-cluster* pelo nome do cluster.
   - Substitua *vpc-cni* pelo nome do complemento que você deseja atualizar e que foi retornado na saída de uma etapa anterior.
   - Substitua *o número da versão* pela versão retornada na saída da etapa anterior para a qual você deseja atualizar. Alguns complementos têm versões recomendadas. Para obter mais informações, consulte a documentação do complemento que você está atualizando. Para obter uma lista de complementos, consulte a seção chamada the section called “ Complementos da AWS”.* Se o complemento usar uma conta de serviço do Kubernetes e um perfil do IAM, substitua *111122223333* pelo ID da sua conta e *role-name* pelo nome de um perfil do IAM existente que você tenha criado. Para obter instruções para criar o perfil, consulte a documentação do complemento que você está criando. Para obter uma lista de complementos, consulte the section called “ Complementos da AWS”. A especificação de um perfil da conta de serviço exige que você tenha um provedor OpenID Connect (OIDC) do IAM para o cluster. Para

determinar se você já tem um ou se precisa criar um para o seu cluster, consulte the section called “Provedor OIDC do IAM”.

Se o complemento não usar uma conta de serviço do Kubernetes e um perfil do IAM, exclua a linha `serviceAccountRoleARN: arn:aws:iam::111122223333:role/role-name` .

- A opção `--resolve-conflicts PRESERVE` mantém os valores existentes para o complemento. Se você definiu valores personalizados para as configurações do complemento e não usar essa opção, o Amazon EKS sobrescreverá seus valores pelos valores padrão. Se você usar essa opção, recomendamos testar qualquer alteração de campo e valor em um cluster que não seja de produção antes de atualizar o complemento no cluster de produção. Se você alterar esse valor para `OVERWRITE`, todas as configurações serão alteradas para os valores padrão do Amazon EKS. Se você definiu valores personalizados para qualquer configuração, eles poderão ser sobrescritos pelos valores padrão do Amazon EKS. Se você alterar esse valor para `NONE`, o Amazon EKS não alterará o valor de nenhuma configuração, mas a atualização poderá falhar. Se a atualização falhar, você receberá uma mensagem de erro para ajudar a resolver o conflito.
- Se quiser remover todas as configurações personalizadas, execute a atualização usando a opção *`--configuration-values '{}'`*. Isso definirá todas as configurações personalizadas aos valores padrão. Se você não quiser alterar sua configuração personalizada, não forneça o sinalizador *`--configuration-values`*. Se você quiser ajustar uma configuração personalizada, substitua *`{}`* pelos novos parâmetros.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
configuration-values '{}' --resolve-conflicts PRESERVE
```

6. Verifique o status da atualização. Substitua *`my-cluster`* pelo nome do seu cluster e *`vpc-cni`* pelo nome do complemento que está atualizando.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

Veja abaixo um exemplo de saída.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
```

```
        "status": "UPDATING",
    }
}
```

Quando a atualização estiver concluída, o status será `ACTIVE`.

Verificar a compatibilidade da versão do complemento do Amazon EKS com um cluster

Antes de criar um complemento do Amazon EKS, é necessário verificar se a versão do complemento do Amazon EKS é compatível com seu cluster.

Use a API describe-addon-versions para listar as versões disponíveis dos complementos do EKS e quais versões do Kubernetes são compatíveis com cada versão do complemento.

1. Verifique se a AWS CLI está instalada e funcionando com o `aws sts get-caller-identity`. Se esse comando não funcionar, saiba como começar a usar a AWS CLI.
2. Determine o nome do complemento para o qual você deseja recuperar as informações de compatibilidade de versão, por exemplo, `amazon-cloudwatch-observability`.
3. Determine a versão do Kubernetes do seu cluster, por exemplo, `1.33`.
4. Use a AWS CLI para recuperar as versões de complemento compatíveis com a versão do Kubernetes do seu cluster.

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --
kubernetes-version 1.33
```

Veja um exemplo de saída abaixo.

```
{
    "addons": [
        {
            "addonName": "amazon-cloudwatch-observability",
            "type": "observability",
            "addonVersions": [
                {
                    "addonVersion": "vX.X.X-eksbuild.X",
                    "architecture": [
                        "amd64",
```

```
                        "arm64"
                    ],
                    "computeTypes": [
                        "ec2",
                        "auto",
                        "hybrid"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.33",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                }
            ]
        }
    ]
}
```

Essa saída mostra que a versão do complemento `vX.X.X-eksbuild.X` é compatível com o cluster do Kubernetes versão `1.33`.

Compatibilidade de complementos com tipos de computação

O campo `computeTypes` na saída `describe-addon-versions` indica a compatibilidade de um complemento com os nós híbridos ou gerenciados pelo Modo Automático do EKS. Os complementos marcados como `auto` funcionam com a infraestrutura baseada em nuvem gerenciada pela AWS do Modo Automático do EKS, enquanto os marcados como `hybrid` podem ser executados em nós on-premises conectados ao ambiente de gerenciamento em nuvem do EKS.

Para obter mais informações, consulte the section called “Considerações sobre o Modo automático do Amazon EKS”.

Remover um complemento do Amazon EKS de um cluster

Você pode remover um complemento do Amazon EKS do seu cluster usando `eksctl`, Console de gerenciamento da AWS ou AWS CLI.

Quando um complemento do Amazon EKS é removido de um cluster:

- Não há tempo de inatividade para a funcionalidade que o complemento fornece.
- Se estiver usando Perfis do IAM para contas de serviço (IRSA) e não houver um perfil do IAM associado ao complemento, o perfil do IAM não será removido.
- Se você estiver usando identidades de pods, todas as associações de identidades de pods pertencentes ao complemento serão removidas. Se você especificar a opção `--preserve` na AWS CLI, as associações serão preservadas.
- O Amazon EKS deixará de gerenciar as configurações do complemento.
- O console deixará de notificar você quando novas versões estão disponíveis.
- Você não poderá atualizar o complemento usando nenhuma das ferramentas ou APIs da AWS.
- Você pode optar por deixar o software complementar no cluster para que ele seja autogerenciado ou pode remover o software complementar do cluster. Você só deverá remover o software complementar do cluster se não houver nenhum recurso no cluster que dependa da funcionalidade fornecida pelo complemento.

Pré-requisitos

Antes de criar um complemento, faça o seguinte:

- Um cluster existente do Amazon EKS. Para implantar, consulte Conceitos básicos.
- Verifique se o complemento exige um perfil do IAM. Para obter mais informações, consulte
- Versão `0.215.0` ou posterior da ferramenta de linha de comando da `eksctl` instalada no seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.

Procedimento

Você tem duas opções ao remover um complemento do Amazon EKS.

- Preserve add-on software on your cluster (Preservar software de complemento no cluster): esta opção remove o gerenciamento do Amazon EKS de qualquer configuração. Também remove a capacidade do Amazon EKS de notificar você sobre atualizações e de atualizar automaticamente o complemento do Amazon EKS depois de iniciar uma atualização. No entanto, ele preserva o software de complemento em seu cluster. Essa opção torna o complemento em uma instalação

autogerenciada, em vez de um complemento do Amazon EKS. Com essa opção, não há tempo de inatividade para o complemento.

- Remover completamente o software do complemento do cluster: recomendamos remover o complemento do Amazon EKS do cluster se não houver recursos no cluster que dependam dele.

Você pode remover um complemento do Amazon EKS usando `eksctl`, Console de gerenciamento da AWS ou AWS CLI.

Remover o complemento (eksctl)

1. Determine quais são os complementos instalados atualmente no cluster. Substitua *`my-cluster`* pelo nome do cluster.

   ```
   eksctl get addon --cluster my-cluster
   ```

   Veja abaixo um exemplo de saída.

   ```
   NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
   coredns     v1.8.7-eksbuild.2    ACTIVE  0
   kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0
   vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0
   [...]
   ```

   Sua saída pode ser diferente, dependendo dos complementos e versões que você tiver no cluster.

2. Remova o complemento. Substitua *`my-cluster`* pelo nome do seu cluster e *`name-of-complemento`* pelo nome do complemento retornado na saída da etapa anterior que você deseja remover. Se você remover a opção *`--preserve`*, além de o Amazon EKS não gerenciar mais o complemento, o software do complemento será excluído do cluster.

   ```
   eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
   ```

   Para obter mais informações sobre a remoção de complementos, consulte Excluir complementos na documentação do `eksctl`.

Remover o complemento (Console do AWS)

1. Abra o console do Amazon EKS.

2. No painel de navegação à esquerda, escolha Clusters.
3. Escolha o nome do cluster para o qual você deseja remover o complemento do Amazon EKS.
4. Escolha a guia Add-ons (Complementos).
5. Escolha o complemento que deseja remover.
6. Escolha Remover.
7. Na caixa de diálogo Remover: ***nome do complemento*** de confirmação , faça o seguinte:
   a. Se quiser que o Amazon EKS pare de gerenciar as configurações do complemento, selecione Preservar no cluster. Faça isso se quiser manter o software de complemento em seu cluster. Isso serve para que você possa gerenciar todas as configurações do complemento por conta própria.
   b. Insira o nome do complemento.
   c. Escolha Remover.

Remover o complemento (AWS CLI)

1. Você precisa da versão `0.215.0` ou posterior da ferramenta de linha de comando da `eksctl` instalada no seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
2. Veja uma lista de complementos instalados. Substitua *`my-cluster`* pelo nome do cluster.

```
aws eks list-addons --cluster-name my-cluster
```

   Veja abaixo um exemplo de saída.

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni",
        "name-of-addon"
    ]
}
```

3. Remova o complemento instalado. Substitua *`my-cluster`* pelo nome do seu cluster e *`name-of-complemento`* pelo nome do complemento que você deseja remover. A remoção de *`--preserve`* exclui o software complementar do seu cluster.

```
aws eks delete-addon --cluster-name my-cluster --addon-name name-of-addon --preserve
```

Veja o exemplo de saída abreviado abaixo.

```
{
    "addon": {
        "addonName": "name-of-add-on",
        "clusterName": "my-cluster",
        "status": "DELETING",
    }
}
```

4. Verifique o status da remoção. Substitua *my-cluster* pelo nome do seu cluster e *name-of-complemento* pelo nome do complemento que você está removendo.

```
aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
```

Após a exclusão do complemento, a saída de exemplo será semelhante a esta.

```
An error occurred (ResourceNotFoundException) when calling the DescribeAddon
 operation: No addon: name-of-addon found in cluster: my-cluster
```

Perfis do IAM para complementos do Amazon EKS

Alguns complementos do Amazon EKS precisam de permissões e perfis do IAM para chamar APIs da AWS. Por exemplo, o complemento CNI da Amazon VPC chama determinadas APIs da AWS para configurar recursos de rede na sua conta. Esses complementos precisam receber permissão usando o IAM. Mais especificamente, a conta de serviço do pod que executa o complemento precisa estar associada a um perfil do IAM com uma política do IAM específica.

A forma recomendada de conceder permissões da AWS para workloads de cluster é usando o recurso Identidades de Pods do Amazon EKS. Você pode usar uma Associação de identidade de pods para mapear a conta de serviço de um complemento para um perfil do IAM. Se um pod usa uma conta de serviço que tem uma associação, o Amazon EKS define variáveis de ambiente nos contêineres do pod. As variáveis de ambiente configuram os SDKs da AWS, incluindo a AWS CLI, para usar as credenciais de Identidade de pods do EKS. Para ter mais informações, consulte the section called “Identidade de Pods”.

Os complementos do Amazon EKS podem ajudar a gerenciar o ciclo de vida das associações de identidades de pods correspondentes ao complemento. Por exemplo, você pode criar ou atualizar um complemento do Amazon EKS e a associação de identidade de pods necessária em uma única chamada de API. O Amazon EKS também fornece uma API para recuperar políticas do IAM sugeridas.

1. Confirme se o agente de identidade de pods do Amazon EKS está configurado no cluster.
2. Determine se o complemento que você deseja instalar requer permissões do IAM usando a operação `describe-addon-versions` da AWS CLI. Se o sinalizador `requiresIamPermissions` for `true`, você deverá usar a operação `describe-addon-configurations` para determinar as permissões necessárias para o complemento. A resposta inclui uma lista sugerida de políticas do IAM gerenciadas.
3. Recupere o nome da conta do serviço Kubernetes e a política do IAM sugerida usando a operação `describe-addon-configuration` da CLI. Avalie o escopo da política sugerida em relação aos seus requisitos de segurança.
4. Crie um perfil do IAM usando a política de permissões sugerida e a política de confiança exigida pela Identidade de Pod. Para ter mais informações, consulte the section called “Criar uma associação de identidade de pod (console do AWS)”.
5. Crie ou atualize um complemento do Amazon EKS usando a CLI. Especifique pelo menos uma associação de identidade de pods. Uma associação de identidade de pods é o nome de uma conta de serviço do Kubernetes e o ARN do perfil do IAM.
   - Associações de identidades de pods criadas usando as APIs do complemento pertencem ao respectivo complemento. Se você excluir o complemento, a associação de identidade de pods também será excluída. Você pode evitar essa exclusão em cascata usando a opção `preserve` ao excluir um complemento usando a AWS CLI ou a API. Também é possível atualizar ou excluir diretamente a associação de identidade de pods, se necessário. Os complementos não podem assumir a propriedade de associações de identidades de pods existentes. Você deve excluir a associação existente e recriá-la usando uma operação de criação ou atualização de complemento.
   - O Amazon EKS recomenda o uso de associações de identidades de pods para gerenciar permissões do IAM para complementos. O método anterior, perfis do IAM para contas de serviço (IRSA), ainda é compatível. Você pode especificar um `serviceAccountRoleArn` do IRSA e uma associação de identidade de pods para um complemento. Se o agente de identidade de pods do EKS estiver instalado no cluster, `serviceAccountRoleArn` será ignorado, e o EKS usará a associação de identidade de pods fornecida. Se a Identidade de Pods não estiver habilitada, `serviceAccountRoleArn` será usado.

- Se você atualizar as associações de identidades de pods para um complemento existente, o Amazon EKS iniciará uma reinicialização contínua dos pods do complemento.

Recuperar informações do IAM sobre um complemento do Amazon EKS

Antes de criar um complemento, use a AWS CLI para determinar:

- Se o complemento requer permissões do IAM
- A política de IAM sugerida para uso

Procedimento

1. Determine o nome do complemento que você deseja instalar e a versão Kubernetes do seu cluster. Para obter mais informações sobre complementos, consulte the section called "Complementos do Amazon EKS".
2. Use a AWS CLI para determinar se o complemento requer permissões do IAM.

```
aws eks describe-addon-versions \
--addon-name <addon-name> \
--kubernetes-version <kubernetes-version>
```

Por exemplo:

```
aws eks describe-addon-versions \
--addon-name aws-ebs-csi-driver \
--kubernetes-version 1.30
```

Analise o seguinte exemplo de saída. Observe que `requiresIamPermissions` é `true` e a versão padrão do complemento. Você precisa especificar a versão do complemento ao recuperar a política do IAM recomendada.

```
{
    "addons": [
        {
            "addonName": "aws-ebs-csi-driver",
            "type": "storage",
            "addonVersions": [
                {
```

```
                    "addonVersion": "v1.31.0-eksbuild.1",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.30",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                    "requiresConfiguration": false,
                    "requiresIamPermissions": true
                },
[...]
```

3. Se o complemento exigir permissões do IAM, use a AWS CLI para obter uma política do IAM recomendada.

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name <addon-name> \
--addon-version <addon-version>
```

Por exemplo:

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name aws-ebs-csi-driver \
--addon-version v1.31.0-eksbuild.1
```

Analise a seguinte saída. Anote o `recommendedManagedPolicies`.

```
[
    {
        "serviceAccount": "ebs-csi-controller-sa",
        "recommendedManagedPolicies": [
            "arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"
        ]
```

```
    }
]
```

4. Crie um perfil do IAM e anexe a Política gerenciada recomendada. Como alternativa, revise a política gerenciada e defina o escopo das permissões conforme apropriado. Para obter mais informações, consulte the section called “Criar uma associação de identidade de pod (console do AWS)”.

Referência de suporte de Identidade de Pods

A tabela a seguir indica se determinados complementos do Amazon EKS oferecem suporte à Identidade de Pods do EKS.

Nome do complemento	Suporte para Identidade de Pods	Versão mínima exigida
Driver da CSI do Amazon EBS	Sim	v1.26.0-eksbuild.1
CNI da Amazon VPC	Sim	v1.15.5-eksbuild.1
Amazon EFS CSI Driver	Sim	v2.0.5-eksbuild.1
AWS Distro para OpenTelemetry	Sim	v0.94.1-eksbuild.1
Driver da CSI do Mountpoint para Amazon S3	Não	N/D
Agente de observabilidade do Amazon CloudWatch	Sim	v3.1.0-eksbuild.1

Essa tabela foi atualizada pela última vez em 28 de outubro de 2024.

Usar identidades de Pods para atribuir um perfil do IAM a um complemento do Amazon EKS

Alguns complementos do Amazon EKS precisam de permissões e perfis do IAM. Antes de adicionar e atualizar um complemento do Amazon EKS para usar uma associação de identidade de pods,

verifique o perfil e a política a serem usados. Para ter mais informações, consulte the section called “Recuperar informações do IAM”.

1. Determine:
   - `cluster-name`: o nome do cluster no qual o complemento será instalado.
   - `addon-name`: o nome do complemento a ser instalado.
   - `service-account-name`: o nome da conta do serviço Kubernetes usada pelo complemento.
   - `iam-role-arn`: o ARN de um perfil do IAM com permissões suficientes para o complemento. O perfil deve ter a política de confiança necessária para a identidade de pods do EKS. Para ter mais informações, consulte the section called “Criar uma associação de identidade de pod (console do AWS)”.
2. Atualize o complemento usando a AWS CLI. Também é possível especificar associações de identidades de pods ao criar um complemento usando a mesma sintaxe `--pod-identity-assocations`. Observe que, quando você especifica associações de identidades de pods ao atualizar um complemento, todas as associações anteriores de identidades de pods são substituídas.

   ```
   aws eks update-addon --cluster-name <cluster-name> \
   --addon-name <addon-name> \
   --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
   ```

   Por exemplo:

   ```
   aws eks update-addon --cluster-name mycluster \
   --addon-name aws-ebs-csi-driver \
   --pod-identity-associations 'serviceAccount=ebs-csi-controller-sa,roleArn=arn:aws:iam::123456789012:role/StorageDriver'
   ```
3. Valide se a associação de identidade de pods foi criada:

   ```
   aws eks list-pod-identity-associations --cluster-name <cluster-name>
   ```

   Se for bem-sucedido, você deve ver uma saída semelhante à seguinte. Observe o OwnerARN do complemento do EKS.

   ```
   {
       "associations": [
   ```

```
        {
            "clusterName": "mycluster",
            "namespace": "kube-system",
            "serviceAccount": "ebs-csi-controller-sa",
            "associationArn": "arn:aws:eks:us-
west-2:123456789012:podidentityassociation/mycluster/a-4wvljrezsukshq1bv",
            "associationId": "a-4wvljrezsukshq1bv",
            "ownerArn": "arn:aws:eks:us-west-2:123456789012:addon/mycluster/aws-ebs-
csi-driver/9cc7ce8c-2e15-b0a7-f311-426691cd8546"
        }
    ]
}
```

Remover associações de identidades de pods de um complemento do Amazon EKS

Remova as associações de identidades de pods de um complemento do Amazon EKS.

1. Determine:
   - `cluster-name`: o nome do cluster EKS no qual instalar o complemento.
   - `addon-name`: o nome do complemento do Amazon EKS a ser instalado.
2. Atualize o complemento para especificar uma matriz vazia de associações de identidades de pods.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations "[]"
```

Solucionar problemas de identidades de pods para complementos do EKS

Se os seus complementos estiverem encontrando erros ao tentar operações da API, do SDK ou da AWS CLI, verifique se:

- O Agente de identidade de pods está instalado no seu cluster.
  - Para obter informações sobre como instalar o Pod Identity Agent, consulte the section called “Configurar o atendente”.
- O complemento tem uma associação de identidade de pods válida.

- Use a AWS CLI para recuperar as associações do nome da conta de serviço usada pelo complemento.

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

- O perfil do IAM tem a política de confiança necessária para as identidades de pods.
  - Use a AWS CLI para recuperar a política de confiança de um complemento.

```
aws iam get-role --role-name <role-name> --query Role.AssumeRolePolicyDocument
```

- O perfil do IAM tem as permissões necessárias para o complemento.
  - Use o AWS CloudTrail para rever eventos do `AccessDenied` ou `UnauthorizedOperation`.
- O nome da conta de serviço na associação de identidade de pods corresponde ao nome da conta de serviço usado pelo complemento.
  - Para obter informações sobre os complementos disponíveis, consulte the section called “ Complementos da AWS”.
- Verifique a configuração de MutatingWebhookConfiguration denominada `pod-identity-webhook`
  - As `admissionReviewVersions` do webhook precisam ser `v1beta1` e não funcionam com a `v1`.

Determinar os campos que podem ser personalizados para os complementos do Amazon EKS

Os complementos do Amazon EKS são instalados no cluster usando as configurações padrão de prática recomendada. Para obter mais informações sobre como adicionar um complemento do Amazon EKS ao cluster, consulte the section called “Complementos do Amazon EKS”.

Talvez você queira personalizar a configuração de um complemento do Amazon EKS para habilitar recursos avançados. O Amazon EKS usa o recurso de aplicação do lado do servidor do Kubernetes para habilitar o gerenciamento de um complemento pelo Amazon EKS sem substituir a configuração por configurações que não são gerenciadas pelo Amazon EKS. Para obter mais informações, consulte Server Side Apply na documentação do Kubernetes. Para isso, o Amazon EKS gerencia um conjunto mínimo de campos para cada complemento instalado. Você poderá modificar todos os campos que não forem gerenciados pelo Amazon EKS ou outro processo do ambiente de gerenciamento do Kubernetes, como o `kube-controller-manager`, sem problemas.

> **Important**
>
> A modificação de um campo gerenciado pelo Amazon EKS impede que o Amazon EKS gerencie o complemento e pode resultar na substituição das alterações quando um complemento for atualizado.

Sintaxe do gerenciamento de campos

Quando você visualiza os detalhes de um objeto do Kubernetes, tanto os campos gerenciados quanto os não gerenciados são retornados na saída. Os campos gerenciados podem ser de um dos seguintes tipos:

- Fully managed (Totalmente gerenciado) – Todas as chaves para o campo são gerenciadas pelo Amazon EKS. Modificações em qualquer valor neste campo geram conflito.
- Partially managed (Parcialmente gerenciado) – Algumas chaves para o campo são gerenciadas pelo Amazon EKS. Somente modificações nas chaves gerenciadas explicitamente pelo Amazon EKS geram conflito.

Ambos os tipos de campos são marcados com `manager: eks`.

Cada chave é um `.` que representa o próprio campo, que sempre mapeia para um conjunto vazio, ou uma string que representa um subcampo ou item. A saída para o gerenciamento de campo consiste nos seguintes tipos de declarações:

- `f:`*`name`* onde *`name`* é o nome de um campo em uma lista.
- `k:`*`keys`* em que *`keys`* é um mapa dos campos de um item de lista.
- `v:`*`value`* em que *`value`* é o valor exato formatado em JSON de um item da lista.
- `i:`*`index`* onde *`o índice`* é a posição de um item na lista.

As seguintes partes do resultado para o complemento CoreDNS ilustram as declarações anteriores:

- Fully managed fields (Campos totalmente gerenciados) – Se um campo gerenciado tiver um (campo) `f:` especificado, mas não `k:` (chave), todo o campo é gerenciado. Modificações em quaisquer valores neste campo causam um conflito.

No resultado a seguir, é possível ver que o contêiner chamado `coredns` é gerenciado pelo `eks`. Os subcampos `args`, `image` e `imagePullPolicy` também são gerenciados pelo `eks`. Modificações em qualquer valor nesse campo geram conflito.

```
[...]
f:containers:
  k:{"name":"coredns"}:
  .: {}
  f:args: {}
  f:image: {}
  f:imagePullPolicy: {}
[...]
manager: eks
[...]
```

- Partially managed fields (campos parcialmente gerenciados) – Se uma chave gerenciada tiver um valor especificado, as chaves declaradas são gerenciadas para esse campo. A modificação das chaves especificadas gera conflito.

  No resultado a seguir, é possível ver que o `eks` gerencia os volumes `config-volume` e `tmp`, definidos com a chave `name`.

```
[...]
f:volumes:
  k:{"name":"config-volume"}:
    .: {}
    f:configMap:
      f:items: {}
      f:name: {}
    f:name: {}
  k:{"name":"tmp"}:
    .: {}
    f:name: {}
[...]
manager: eks
[...]
```

- Adição de chaves em campos gerenciados parcialmente – Se apenas um valor de chave específico for gerenciado, você pode adicionar, com segurança, mais chaves, como, por exemplo, argumentos, em um campo sem gerar conflito. Se você adicionar outras chaves, certifique-se

de que o campo não seja gerenciado primeiro. Adicionar ou modificar qualquer valor gerenciado causa um conflito.

No resultado a seguir, é possível ver que tanto a chave do `name`, quando o `name` são gerenciados. Adicionar ou modificar qualquer nome de contêiner causa um conflito com essa chave gerenciada.

```
[...]
f:containers:
  k:{"name":"coredns"}:
[...]
    f:name: {}
[...]
manager: eks
[...]
```

Procedimento

Você pode usar o `kubectl` para ver quais campos são gerenciados pelo Amazon EKS para qualquer complemento do Amazon EKS.

Você poderá modificar todos os campos que não forem gerenciados pelo Amazon EKS ou outro processo do ambiente de gerenciamento do Kubernetes, como o `kube-controller-manager`, sem problemas.

1. Determine qual complemento você deseja examinar. Para ver todas as `deployments` e os DaemonSets implantados no cluster, consulte the section called “Acessar recursos do cluster”.
2. Exiba os campos gerenciados para um complemento executando o seguinte comando:

   ```
   kubectl get type/add-on-name -n add-on-namespace -o yaml
   ```

   Por exemplo, você pode ver os campos gerenciados do complemento CoreDNS com o comando a seguir.

   ```
   kubectl get deployment/coredns -n kube-system -o yaml
   ```

   O gerenciamento de campo é listado na seção a seguir no resultado retornado.

   ```
   [...]
   managedFields:
   ```

```
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
[...]
```

> **Note**
>
> Se a saída não exibir `managedFields`, adicione `--show-managed-fields` ao comando e execute-o novamente. A versão do `kubectl` que você está usando determina se os campos gerenciados serão retornados por padrão.

Próximas etapas

Personalize os campos não pertencentes à AWS para seu complemento.

Validar assinaturas de imagem de contêiner durante a implantação

Se você usar o AWS Signer e quiser verificar as imagens de contêineres assinadas no momento da implementação, poderá usar uma das seguintes soluções:

- Gatekeeper e Ratify: use o Gatekeeper como controlador de admissão e o Ratify configurado com um plug-in AWS Signer como um hook da Web para validar assinaturas.
- Kyverno: um mecanismo de política do Kubernetes configurado com um plug-in do AWS Signer para validar assinaturas.

> **Note**
>
> Antes de verificar as assinaturas das imagens do contêiner, configure o repositório confiável e a política de confiança do Notation, conforme exigido pelo controlador de admissão selecionado.

Funcionalidades do EKS

> **Tip**
>
> Conceitos básicos: Criação de funcionalidades do ACK | Criação de funcionalidades do Argo CD | Criação de funcionalidades do kro

As funcionalidades do Amazon EKS são um conjunto, em camadas. de recursos de cluster totalmente gerenciados que auxiliam a aumentar a produtividade dos desenvolvedores e a reduzir a complexidade de desenvolvimento e de escalabilidade com o Kubernetes. As funcionalidades do EKS são recursos nativos do Kubernetes para implantação contínua declarativa, gerenciamento de recursos da AWS e criação e orquestração de recursos do Kubernetes, tudo totalmente gerenciado pela AWS. Com as funcionalidades do EKS, é possível concentrar-se mais na criação e na escalabilidade das workloads, transferindo a carga operacional desses serviços fundamentais de plataforma para a AWS. Essas funcionalidades são executadas no próprio EKS, e não em seus clusters, dispensando a instalação, a manutenção e a escalabilidade de componentes essenciais de plataforma nos nós de processamento.

Para começar a usar, você pode criar uma ou mais funcionalidades do EKS em um cluster do EKS novo ou já existente. Para realizar essa ação, é possível usar a AWS CLI, o Console de gerenciamento da AWS, as APIs do EKS, o eksctl ou suas ferramentas preferidas de infraestrutura como código. Apesar das funcionalidades do EKS serem desenvolvidas para operar em conjunto, elas são recursos de nuvem independentes, permitindo que você realize a escolha de acordo com as necessidades e o caso de uso da sua aplicação.

Todas as versões do Kubernetes com suporte por parte do EKS são compatíveis com as funcionalidades do EKS.

> **Note**
>
> As funcionalidades do EKS estão disponíveis para uso em todas as regiões comerciais da AWS nas quais o Amazon EKS está disponível. Para obter uma lista das regiões com suporte, consulte Endpoints e cotas do Amazon EKS na Referência geral da AWS.

Capacidades disponíveis

AWS Controlllers for Kubernetes (ACK)

O ACK viabiliza o gerenciamento de recursos da AWS por meio de APIs do Kubernetes, permitindo a criação e o gerenciamento de buckets do S3, bancos de dados do RDS, perfis do IAM e outros recursos da AWS utilizando recursos personalizados do Kubernetes. O ACK faz a reconciliação contínua entre o estado desejado e o estado atual na AWS, corrigindo qualquer desvio ao longo do tempo para manter a integridade do sistema e dos recursos configurados conforme especificado. É possível gerenciar recursos da AWS juntamente com as workloads do Kubernetes usando as mesmas ferramentas e fluxos de trabalho, com suporte para mais de 50 serviços da AWS, incluindo o S3, o RDS, o DynamoDB e o Lambda. O ACK fornece suporte para o gerenciamento de recursos entre contas e entre regiões, permitindo arquiteturas complexas de gerenciamento de sistemas com várias contas e vários clusters. O ACK é compatível com recursos somente leitura e a adoção somente leitura, facilitando a migração de outras ferramentas de infraestrutura como código para os sistemas baseados em Kubernetes.

Saiba mais sobre o ACK →

Argo CD

O Argo CD aplica o modelo de implantação contínua baseada em GitOps para as aplicações, utilizando repositórios do Git como a fonte da verdade para as workloads e para o estado do sistema. O Argo CD faz a sincronização automática de recursos de aplicações para os clusters com base em repositórios do Git, identificando e corrigindo desvios para assegurar que as aplicações implantadas estejam alinhadas ao estado desejado. É possível implantar e gerenciar aplicações em vários clusters de uma única instância do Argo CD, com a implantação automatizada de repositórios do Git sempre que alterações forem confirmadas. O uso conjunto do Argo CD e do ACK fornece um sistema de GitOps fundamental, simplificando o gerenciamento de dependências das workloads, além de fornecer suporte a projetos de sistemas completos, incluindo o gerenciamento de clusters e de infraestrutura em grande escala. O Argo CD se integra ao Centro de Identidade da AWS para autenticação e autorização, e fornece uma interface do usuário do Argo hospedada para a visualização da integridade das aplicações e do status de implantação.

Saiba mais sobre o Argo CD →

kro (Kube Resource Orchestrator)

O kro possibilita a criação de APIs personalizadas do Kubernetes que compõem vários recursos em abstrações de nível superior, permitindo que as equipes responsáveis pela plataforma definam padrões reutilizáveis para combinações comuns de recursos, os blocos de criação da nuvem. Com o kro, você pode compor recursos do Kubernetes e da AWS em abstrações unificadas, usando uma sintaxe simplificada que viabiliza configurações dinâmicas e o uso de lógica condicional. O kro permite que as equipes responsáveis pela plataforma disponibilizem funcionalidades de autoatendimento com as devidas barreiras de proteção, possibilitando que os desenvolvedores provisionem infraestruturas complexas por meio de APIs simples e especializadas, enquanto preservam os padrões e as práticas recomendadas da organização. Os recursos do kro nada mais são do que recursos do Kubernetes, definidos em manifestos do Kubernetes, que podem ser armazenados no Git ou enviados para registros compatíveis com OCI, como o Amazon ECR, para facilitar a distribuição por toda a organização.

Saiba mais sobre o kro →

Benefícios das funcionalidades do EKS

As funcionalidades do EKS são totalmente gerenciadas pela AWS, o que dispensa a instalação, a manutenção e a escalabilidade de serviços essenciais do cluster. A AWS lida com a aplicação de patches de segurança, atualizações e gerenciamento operacional, permitindo que as equipes concentrem esforços no desenvolvimento na AWS e não nas operações de cluster. Ao contrário dos complementos convencionais do Kubernetes que consomem recursos do cluster, as funcionalidades são executadas no EKS, e não em nós de processamento. Essa abordagem libera capacidade e recursos do cluster para as workloads, enquanto minimiza a carga operacional necessária para o gerenciamento de controladores internos do cluster e de outros componentes de plataforma.

Com as funcionalidades do EKS, é possível gerenciar implantações, recursos da AWS, recursos personalizados do Kubernetes e composições usando APIs nativas do Kubernetes e ferramentas como o `kubectl`. Todas as funcionalidades funcionam no contexto dos clusters, identificando e corrigindo automaticamente desvios de configuração em recursos de aplicações e de infraestrutura de nuvem. Você pode implantar e gerenciar recursos em vários clusters, contas da AWS e regiões usando um único ponto de controle, simplificando as operações em ambientes distribuídos de alta complexidade.

As funcionalidades do EKS são projetadas para fluxos de trabalho de GitOps, o que fornece um gerenciamento declarativo e com versionamento de aplicações e de infraestrutura. As alterações

seguem do Git para todo o sistema, fornecendo trilhas de auditoria, funcionalidades de reversão e fluxos de trabalho colaborativos que se integram às práticas de desenvolvimento existentes. Esta abordagem nativa do Kubernetes dispensa o uso de diversas ferramentas ou o gerenciamento de sistemas de infraestrutura como código externos aos clusters, garantindo a existência de uma única fonte de verdade como referência. O estado desejado, definido em configurações declarativas do Kubernetes com versionamento, é continuamente aplicado em todo o seu ambiente.

Preços

Com as funcionalidades do EKS, não há compromissos antecipados nem taxas mínimas. A cobrança é feita por recurso da funcionalidade e se baseia em cada hora de atividade do recurso no cluster do Amazon EKS. Os recursos específicos do Kubernetes, gerenciados pelas funcionalidades do EKS, também são faturados a uma taxa horária.

Para obter informações atualizadas sobre preços, consulte a página de preços do Amazon EKS.

> **Tip**
>
> É possível empregar o Explorador de Custos da AWS e os Relatórios de Custos e Uso para monitorar os custos das funcionalidades de forma isolada de outros custos do EKS. Você pode marcar as funcionalidades com o nome do cluster, tipo de funcionalidade e outros detalhes para fins de alocação de custos.

Funcionamento das funcionalidades do EKS

Cada funcionalidade constitui um recurso da AWS criado no cluster do EKS. Após a criação, a funcionalidade é executada no EKS e é totalmente gerenciada pela AWS.

> **Note**
>
> É possível criar apenas um recurso da funcionalidade de cada tipo, nomeadamente, Argo CD, ACK e kro, por cluster. Não é possível criar mais de um recurso da funcionalidade do mesmo tipo em um único cluster.

A interação com as funcionalidades do cluster é feita por meio de APIs e ferramentas nativas do Kubernetes:

- Use o `kubectl` para aplicar recursos personalizados do Kubernetes
- Use os repositórios do Git como a fonte da verdade para fluxos de trabalho de GitOps

Algumas funcionalidades oferecem suporte a ferramentas adicionais. Por exemplo:

- Use a CLI do Argo CD para configurar e gerenciar repositórios e clusters na funcionalidade do Argo CD
- Use a interface do usuário do Argo CD para visualizar e gerenciar aplicações controladas pela funcionalidade do Argo CD

Embora concebidas para operar de forma integrada, as funcionalidades são independentes, e você escolhe quais deseja ativar. Você pode habilitar uma, duas ou todas as três funcionalidades, de acordo com a sua demanda, ajustando a configuração à medida que suas necessidades mudarem.

Todos os tipos de computação do EKS são compatíveis com o uso das funcionalidades do EKS. Para obter mais informações, consulte Gerenciar computação.

Para obter configurações de segurança e detalhes sobre os perfis do IAM, consulte the section called “Considerações para funcionalidades do EKS”. Para obter padrões de arquitetura de vários clusters, consulte the section called “Considerações”.

Casos de uso comuns

GitOps aplicado a aplicações e infraestrutura

Use o Argo CD na implantação de aplicações e de componentes operacionais, e o ACK para o gerenciamento de configurações de cluster e para o provisionamento de infraestrutura, empregando repositórios do Git como fonte para ambos. Toda a sua pilha, que é composta por aplicações, bancos de dados, armazenamento e rede, é definida como código e implantada automaticamente.

Exemplo: uma equipe de desenvolvimento envia alterações para o Git. O Argo CD realiza a implantação da aplicação atualizada, enquanto o ACK provisiona um novo banco de dados do RDS com a configuração adequada. Todas as alterações são passíveis de auditoria e de reversão, e mantêm a consistência em diferentes ambientes.

Engenharia de plataforma com autoatendimento

Use o kro para criar APIs personalizadas que compõem recursos do ACK e do Kubernetes. As equipes responsáveis pela plataforma definem padrões aprovados com barreiras de proteção.

As equipes responsáveis pela aplicação usam APIs simplificadas de alto nível para realizar o provisionamento de pilhas inteiras.

Exemplo: uma equipe responsável pela plataforma cria uma API de “WebApplication” para provisionar automaticamente os recursos Deployment, Service e Ingress, e um bucket do S3. Os desenvolvedores usam essa API sem a necessidade de compreender a complexidade subjacente ou as permissões da AWS.

Gerenciamento de aplicações em vários clusters

Use o Argo CD para implantar aplicações em vários clusters do EKS distribuídos em diferentes regiões ou contas. Gerencie todas as implantações com uma única instância do Argo CD, empregando políticas e fluxos de trabalho consistentes.

Exemplo: implante a mesma aplicação em clusters de desenvolvimento, preparação e produção distribuídos por diversas regiões. O Argo CD assegura a sincronização de cada ambiente com a ramificação correspondente no Git.

Gerenciamento de vários clusters

Use o ACK para definir e provisionar clusters do EKS, o kro para personalizar as configurações dos clusters de acordo com os padrões organizacionais, e o Argo CD para gerenciar o ciclo de vida e a configuração do cluster. Essa abordagem garante o gerenciamento integral do cluster, abrangendo desde o provisionamento inicial até a operação cotidiana.

Exemplo: defina clusters do EKS usando o ACK e o kro para provisionar e gerenciar a infraestrutura de clusters, definindo padrões organizacionais para rede, políticas de segurança, complementos e outras configurações. Empregue o Argo CD para a criação e para o gerenciamento contínuo de clusters, configurações e atualizações de versões do Kubernetes em toda a frota, fazendo uso de padrões consistentes e do gerenciamento de ciclo de vida automatizado.

Migrações e modernização

Simplifique a migração para o EKS com o provisionamento de recursos de nuvem nativos e fluxos de trabalho de GitOps. Use o ACK para adotar recursos existentes da AWS sem a necessidade de recriá-los, e o Argo CD para operacionalizar as implantações de workloads a partir do Git.

Exemplo: uma equipe em processo de migração do EC2 para o EKS integra bancos de dados do RDS e buckets do S3 utilizando o ACK, e usa o Argo CD para realizar a implantação de aplicações

conteinerizadas do Git. O caminho de migração é bem definido e os processos operacionais são padronizados desde o início.

Inicialização de contas e de regiões

Automatize a implementação da infraestrutura entre contas e regiões usando o Argo CD e o ACK de forma integrada. Defina a infraestrutura como código no Git e permita que as funcionalidades gerenciem a implantação e o gerenciamento.

Exemplo: uma equipe responsável pela plataforma mantém repositórios do Git que definem configurações padrão de conta, incluindo VPCs, perfis do IAM, instâncias do RDS e pilhas de monitoramento. O Argo CD implanta essas configurações em novas contas e regiões automaticamente, garantindo a consistência e reduzindo o tempo de configuração manual de dias para minutos.

Como trabalhar com recursos de funcionalidade

Este tópico descreve operações comuns para o gerenciamento de recursos de funcionalidade em todos os tipos de funcionalidades.

Recursos de funcionalidade do EKS

As funcionalidades do EKS são recursos da AWS que permitem funcionalidades gerenciadas no cluster do Amazon EKS. As funcionalidades são executadas no EKS, eliminando a necessidade de instalar e manter controladores e outros componentes operacionais em seus nós de processamento. As funcionalidades são criadas para um cluster específico do EKS e permanecem vinculadas a esse cluster durante todo o ciclo de vida.

Cada recurso de funcionalidade conta com:

- Um nome exclusivo em seu cluster
- Um tipo de funcionalidade (ACK, ARGOCD ou KRO)
- Um nome do recurso da Amazon (ARN), especificando o nome e o tipo
- Um perfil do IAM da funcionalidade
- Um status que indica seu estado atual
- Configuração, tanto genérica quanto específica ao tipo de funcionalidade

Noções básicas sobre o status da funcionalidade

Os recursos de funcionalidade contam com um status que indica o estado atual. Você pode visualizar o status e a integridade da funcionalidade no console do EKS ou usando a AWS CLI.

Console do:

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do seu cluster.
3. Escolha a guia Funcionalidades para visualizar o status de todas as funcionalidades.
4. Para obter informações detalhadas sobre a integridade, escolha a guia Observabilidade, depois Monitorar cluster e, em seguida, a guia Funcionalidades.

AWS CLI:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability-name
```

Status da funcionalidade

CREATING: a funcionalidade está sendo configurada. Você pode sair do console e a funcionalidade continuará sendo criada em segundo plano.

ACTIVE: a funcionalidade está em execução e pronta para uso. Se os recursos não estiverem funcionando conforme esperado, verifique o status do recurso e as permissões de IAM. Para obter orientações, consulte the section called “Solução de problemas das funcionalidades”.

UPDATING: as alterações de configuração estão sendo aplicadas. Aguarde até que o status retorne para `ACTIVE`.

DELETING: a funcionalidade está sendo removida do cluster.

CREATE_FAILED: a configuração encontrou um erro. As causas comuns incluem:

- Política de confiança do perfil do IAM incorreta ou ausente
- O perfil do IAM não existe ou não está acessível

- Problemas de acesso ao cluster
- Parâmetros de configuração inválidos

Verifique a seção de integridade da funcionalidade para obter detalhes específicos do erro.

UPDATE_FAILED: a atualização da configuração falhou. Verifique a seção de integridade da funcionalidade para obter detalhes e valide as permissões do IAM.

> **Tip**
>
> Para obter orientações detalhadas sobre solução de problemas, consulte:
>
> - the section called “Solução de problemas das funcionalidades”: soluções de problemas gerais para problemas de funcionalidades
> - the section called “Solução de problemas”: problemas específicos do ACK
> - the section called “Solução de problemas”: problemas específicos do Argo CD
> - the section called “Solução de problemas”: problemas específicos do kro

Criação de funcionalidades

Para criar uma funcionalidade em seu cluster, consulte os seguintes tópicos:

- the section called “Criação da funcionalidade do ACK”: crie uma funcionalidade do ACK para gerenciar recursos da AWS usando APIs do Kubernetes
- the section called “Criar o recurso Argo CD”: crie uma funcionalidade do Argo CD para entrega contínua de GitOps
- the section called “Criação da funcionalidade do kro”: crie uma funcionalidade do kro para composição e orquestração de recursos

Listagem de funcionalidades

É possível listar todos os recursos da funcionalidade de um cluster.

Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.

2. Selecione o nome do cluster para abrir a página de detalhes do cluster.
3. Escolha a guia Funcionalidades.
4. Visualize os recursos de funcionalidade em Funcionalidades gerenciadas.

AWS CLI

Use o comando `list-capabilities` para visualizar todas as funcionalidades em seu cluster. Substitua *`region-code`* pela região AWS em que seu cluster se encontra e substitua *`my-cluster`* pelo nome do seu cluster.

```
aws eks list-capabilities \
  --region region-code \
  --cluster-name my-cluster
```


```
{
    "capabilities": [
        {
            "capabilityName": "my-ack",
            "arn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/ack/my-
ack/abc123",
            "type": "ACK",
            "status": "ACTIVE",
            "createdAt": "2025-11-02T10:30:00.000000-07:00",
            "modifiedAt": "2025-11-02T10:32:15.000000-07:00",
        },
        {
            "capabilityName": "my-kro",
            "arn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/kro/my-
kro/abc123",
            "type": "KRO",
            "status": "ACTIVE",
            "version": "v0.6.3",
            "createdAt": "2025-11-02T10:30:00.000000-07:00",
            "modifiedAt": "2025-11-02T10:32:15.000000-07:00",
        },
        {
            "capabilityName": "my-argocd",
            "arn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/argocd/my-
argocd/abc123",
            "type": "ARGOCD",
```

```
            "status": "ACTIVE",
            "version": "3.1.8-eks-1",
            "createdAt": "2025-11-21T08:22:28.486000-05:00",
            "modifiedAt": "2025-11-21T08:22:28.486000-05:00"
        }
    ]
}
```

Descrição de uma funcionalidade

Obtenha informações detalhadas sobre uma funcionalidade específica, incluindo a configuração e o status.

Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do cluster para abrir a página de detalhes do cluster.
3. Escolha a guia Funcionalidades.
4. Em Funcionalidades gerenciadas, escolha a funcionalidade que deseja visualizar.
5. Visualize os detalhes da funcionalidade, incluindo o status, a configuração e o horário de criação.

AWS CLI

Use o comando `describe-capability` para visualizar informações detalhadas. Substitua *`region-code`* pela região da AWS em que o seu cluster está, substitua *`my-cluster`* pelo nome do seu cluster e substitua *`capability-name`* pelo nome da funcionalidade (ack, argocd ou kro).

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name
```

Resultado do exemplo:

```
{
  "capability": {
    "capabilityName": "my-ack",
```

```
    "capabilityArn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/ack/my-
ack/abc123",
    "clusterName": "my-cluster",
    "type": "ACK",
    "roleArn": "arn:aws:iam::111122223333:role/AmazonEKSCapabilityACKRole",
    "status": "ACTIVE",
    "configuration": {},
    "tags": {},
    "health": {
      "issues": []
    },
    "createdAt": "2025-11-19T17:11:30.242000-05:00",
    "modifiedAt": "2025-11-19T17:11:30.242000-05:00",
    "deletePropagationPolicy": "RETAIN"
  }
}
```

Atualização da configuração de uma funcionalidade

É possível atualizar alguns aspectos da configuração de uma funcionalidade após ela ter sido criada. As opções de configuração específicas variam conforme o tipo de funcionalidade.

> **Note**
>
> Os recursos da funcionalidade do EKS são totalmente gerenciados, incluindo a aplicação de patches e as atualizações de versão. A atualização de uma funcionalidade atualizará a configuração do recurso e não resultará em atualizações de versão dos componentes da funcionalidade gerenciada.

AWS CLI

Use o comando `update-capability` para modificar uma funcionalidade:

```
aws eks update-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name \
  --role-arn arn:aws:iam::[.replaceable]111122223333:role/NewCapabilityRole
```

> **Note**
>
> Algumas propriedades de uma funcionalidade não permitem atualização após a sua criação. Consulte a documentação específica da funcionalidade para obter detalhes sobre o que pode ser modificado.

Exclusão de uma funcionalidade

Se uma funcionalidade não for mais necessária no seu cluster, é possível realizar a exclusão do recurso da funcionalidade.

> **Important**
>
> Exclua os recursos do cluster antes de excluir a funcionalidade.
> A exclusão de um recurso da funcionalidade não exclui automaticamente os recursos criados por meio dessa funcionalidade:
>
> - Todas as definições de recursos personalizados (CRDs, na sigla em inglês) do Kubernetes permanecem instaladas em seu cluster.
> - Os recursos do ACK permanecem em seu cluster, e os recursos correspondentes da AWS permanecem em sua conta.
> - As Applications do Argo CD e seus recursos do Kubernetes permanecem em seu cluster.
> - As ResourceGraphDefinitions e instâncias do kro permanecem em seu cluster.
>
> Você deve excluir esses recursos antes de excluir a funcionalidade para evitar recursos órfãos.
> Opcionalmente, você pode optar por reter os recursos da AWS associados aos recursos do Kubernetes e do ACK. Consulte Considerações sobre o ACK.

Console

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do cluster para abrir a página de detalhes do cluster.
3. Escolha a guia Funcionalidades.

4. Selecione a funcionalidade que deseja excluir na lista de Funcionalidades gerenciadas.
5. Escolha Excluir funcionalidade.
6. No diálogo de confirmação, digite o nome da funcionalidade para confirmar a exclusão.
7. Escolha Excluir.

AWS CLI

Use o comando `delete-capability` para excluir um recurso de funcionalidade:

Substitua *`region-code`* pela região da AWS em que o seu cluster está, substitua *`my-cluster`* pelo nome do seu cluster e substitua *`capability-name`* pelo nome da funcionalidade que deseja excluir.

```
aws eks delete-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name
```

Próximas etapas

- the section called “Recursos Kubernetes”: saiba mais sobre os recursos do Kubernetes fornecidos por cada tipo de funcionalidade
- the section called “Conceitos do ACK”: compreenda os conceitos do ACK e o ciclo de vida dos recursos
- the section called “Como trabalhar com o Argo CD”: saiba como trabalhar com funcionalidades do Argo CD para fluxos de trabalho de GitOps
- the section called “Conceitos do kro”: compreenda os conceitos do kro e a composição de recursos

Recursos da funcionalidade do Kubernetes

Após habilitar uma funcionalidade no cluster, você passará a interagir com ela principalmente por meio da criação e do gerenciamento de recursos personalizados do Kubernetes no cluster. Cada funcionalidade fornece seu próprio conjunto de definições de recursos personalizados (CRDs, na sigla em inglês) que ampliam a API do Kubernetes com funcionalidades específicas.

Recursos do Argo CD

Quando você habilita a funcionalidade do Argo CD, torna-se possível criar e gerenciar os seguintes recursos do Kubernetes:

Aplicação

Define a implantação de um repositório do Git para um cluster de destino. Os recursos do tipo `Application` especificam o repositório de origem, o namespace de destino e a política de sincronização. É possível criar até mil recursos do tipo `Application` para cada instância da funcionalidade do Argo CD.

ApplicationSet

Gera diversos recursos do tipo `Application` baseados em modelos, permitindo implantações em vários clusters e diversos ambientes. Os recursos do tipo `ApplicationSet` utilizam geradores para criar recursos do tipo `Application` dinamicamente com base em listas de clusters, diretórios do Git ou outras fontes.

AppProject

Fornece agrupamento lógico e controle de acesso para recursos do tipo `Application`. Os recursos do tipo `AppProject` definem quais repositórios, clusters e namespaces os recursos do tipo `Application` podem usar, permitindo multilocação e limites de segurança.

Exemplo de recurso do tipo `Application`:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/org/repo
    targetRevision: main
    path: manifests
  destination:
    server: https://kubernetes.default.svc
    namespace: production
```

Para obter mais informações sobre recursos e conceitos do Argo CD, consulte the section called “Conceitos do Argo CD”.

Recursos do kro

Quando você habilita a funcionalidade do kro, torna-se possível criar e gerenciar os seguintes recursos do Kubernetes:

ResourceGraphDefinition (RGD)

Define uma API personalizada que compõe diversos recursos do Kubernetes e da AWS em uma abstração de nível superior. As equipes responsáveis pela plataforma criam recursos do tipo `ResourceGraphDefinition` para disponibilizar padrões reutilizáveis com barreiras de proteção.

Instâncias de recursos personalizados

Após criar um recurso do tipo `ResourceGraphDefinition`, você pode criar instâncias da API personalizada definida por `ResourceGraphDefinition`. O kro cria e gerencia automaticamente os recursos especificados na `ResourceGraphDefinition`.

Exemplo de recurso do tipo `ResourceGraphDefinition`:

```
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: web-application
spec:
  schema:
    apiVersion: v1alpha1
    kind: WebApplication
    spec:
      name: string
      replicas: integer
  resources:
    - id: deployment
      template:
        apiVersion: apps/v1
        kind: Deployment
        # ... deployment spec
    - id: service
```

```
      template:
        apiVersion: v1
        kind: Service
        # ... service spec
```

Exemplo de instância do tipo `WebApplication`:

```
apiVersion: v1alpha1
kind: WebApplication
metadata:
  name: my-web-app
  namespace: default
spec:
  name: my-web-app
  replicas: 3
```

Ao aplicar esta instância, o kro cria automaticamente os recursos `Deployment` e `Service` definidos na `ResourceGraphDefinition`.

Para obter mais informações sobre recursos e conceitos do kro, consulte the section called "Conceitos do kro".

Recursos do ACK

Quando você habilita a funcionalidade do ACK, torna-se possível criar e gerenciar recursos da AWS por meio de recursos personalizados do Kubernetes. O ACK fornece mais de 200 CRDs para mais de 50 serviços da AWS, permitindo definir recursos da AWS em conjunto com workloads do Kubernetes e gerenciar recursos dedicados de infraestrutura da AWS com o Kubernetes.

Exemplos de recursos do ACK:

Bucket do S3

: Os recursos do tipo `Bucket` realizam a criação e o gerenciamento de buckets do Amazon S3 com políticas de ciclo de vida, criptografia e versionamento.

DBInstance do RDS

: Os recursos do tipo `DBInstance` realizam o provisionamento e o gerenciamento de instâncias de bancos de dados do Amazon RDS com janelas de manutenção e backups automáticos.

Tabela do DynamoDB

Os recursos do tipo `Table` realizam a criação e o gerenciamento de tabelas do DynamoDB com capacidade sob demanda ou provisionada.

Perfil do IAM

Os recursos do tipo `Role` definem os perfis do IAM com as políticas de confiança e de permissão para acesso aos serviços da AWS.

Função do Lambda

Os recursos do tipo `Function` realizam a criação e o gerenciamento de funções do Lambda, incluindo a configuração de código, runtime e perfil de execução.

Exemplo de especificação para um recurso do tipo `Bucket`:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-app-bucket
spec:
  name: my-unique-bucket-name-12345
  versioning:
    status: Enabled
  encryption:
    rules:
      - applyServerSideEncryptionByDefault:
          sseAlgorithm: AES256
```

Para obter mais informações sobre recursos e conceitos do ACK, consulte the section called “Conceitos do ACK”.

Limites de recurso

As funcionalidades EKS têm os seguintes limites de recursos:

Limites de uso do Argo CD:

- Máximo de mil recursos do tipo `Application` por instância da funcionalidade do Argo CD
- Máximo de cem clusters remotos configurados por instância da funcionalidade do Argo CD

Limites de configuração de recursos:

- Máximo de 64 recursos do Kubernetes por recurso do tipo `Application` no Argo CD
- Máximo de 64 recursos do Kubernetes por `ResourceGraphDefinition` no kro

> **Note**
>
> Estes limites são válidos para a quantidade de recursos gerenciados por cada instância de funcionalidade. Caso necessite de limites superiores, é possível fazer a implantação das funcionalidades em diversos clusters.

Próximas etapas

Para realizar tarefas específicas de cada funcionalidade e acessar configurações avançadas, consulte os seguintes tópicos:

- the section called “Conceitos do ACK”: compreenda os conceitos do ACK e o ciclo de vida dos recursos
- the section called “Como trabalhar com o Argo CD”: saiba como trabalhar com funcionalidades do Argo CD para fluxos de trabalho de GitOps
- the section called “Conceitos do kro”: compreenda os conceitos do kro e a composição de recursos

Considerações sobre as funcionalidades do EKS

Este tópico aborda considerações importantes para o uso das funcionalidades do EKS, incluindo o planejamento do controle de acesso, a escolha entre as funcionalidades do EKS e as soluções autogerenciadas, os padrões de arquitetura para implantações em vários clusters e as práticas recomendadas de operação.

Perfis do IAM para a funcionalidade e controle de acesso RBAC do Kubernetes

Cada recurso de funcionalidade do EKS tem um perfil do IAM para a funcionalidade configurado. O perfil da funcionalidade é usado para conceder permissões de serviços da AWS às funcionalidades do EKS para agirem em seu nome. Por exemplo, para usar a funcionalidade do EKS para o ACK

gerenciar buckets do Amazon S3, você concederá permissões administrativas de buckets do S3 à funcionalidade, permitindo que ela crie e gerencie buckets.

Após a configuração da funcionalidade, é possível criar e gerenciar recursos do S3 na AWS usando recursos personalizados do Kubernetes no cluster. O RBAC do Kubernetes atua como um mecanismo de controle de acesso no cluster, definindo quais usuários e grupos têm permissão para criar e gerenciar esses recursos personalizados. Por exemplo, conceda a usuários e grupos específicos do RBAC do Kubernetes permissões para criar e gerenciar recursos de `Bucket` nos namespaces que você escolher.

Assim, o IAM e o RBAC do Kubernetes atuam como duas metades de um sistema de controle de acesso completo, que gerencia as permissões vinculadas às funcionalidades do EKS e aos seus recursos. É fundamental definir a combinação adequada de permissões do IAM e de políticas de acesso do RBAC para atender ao seu caso de uso.

Para obter informações adicionais sobre perfis do IAM de funcionalidades e permissões do Kubernetes, consulte the section called “Considerações para funcionalidades do EKS”.

Padrões de arquitetura de vários clusters

Ao implantar funcionalidades em vários clusters, considere estes padrões arquiteturais comuns:

Modelo hub-and-spoke com gerenciamento centralizado

Execute todas as três funcionalidades em um cluster com gerenciamento centralizado para orquestrar workloads e gerenciar a infraestrutura de nuvem em vários clusters de workload.

- O Argo CD, no cluster de gerenciamento, implanta aplicações nos clusters de workload em diferentes regiões ou contas.
- O ACK, no cluster de gerenciamento, provisiona recursos da AWS (como RDS, S3 e IAM) para todos os clusters.
- O kro, no cluster de gerenciamento, cria abstrações de plataforma portáveis que funcionam em todos os clusters.

Esse padrão centraliza o gerenciamento de workloads e da infraestrutura de nuvem, podendo simplificar as operações para organizações que gerenciam muitos clusters.

GitOps descentralizadas

As workloads e a infraestrutura de nuvem são gerenciadas pelas funcionalidades no mesmo cluster em que as workloads estão sendo executadas.

- O Argo CD gerencia os recursos de aplicações no cluster local
- Os recursos do ACK são usados para atender às necessidades do cluster e das workloads.
- As abstrações de plataforma do kro são instaladas e orquestram recursos locais.

Esse padrão descentraliza as operações, com equipes responsáveis pelo gerenciamento dos próprios serviços de plataforma dedicados a um ou mais clusters.

Modelo hub-and-spoke com implantação híbrida do ACK

Combine modelos centralizados e descentralizados, com implantações de aplicações centralizadas e gerenciamento de recursos baseado em escopo e em propriedade.

- Cluster de hub:
  - Argo CD gerencia implantações de GitOps no cluster local e em todos os clusters de workload remotos.
  - O ACK é usado no cluster de gerenciamento para recursos de escopo administrativo (como bancos de dados de produção, perfis do IAM e VPCs).
  - O kro é usado no cluster de gerenciamento para abstrações de plataforma reutilizáveis.
- Clusters de spoke:
  - As workloads são gerenciadas por meio do Argo CD no cluster centralizado do hub.
  - O ACK é usado localmente para recursos de escopo da workload (como buckets do S3, instâncias do ElastiCache e filas do SQS).
  - O kro é usado localmente para composições de recursos e padrões de blocos de criação.

Esse padrão separa responsabilidades: equipes responsáveis pela plataforma gerenciam a infraestrutura crítica de forma centralizada nos clusters de gerenciamento (opcionalmente incluindo clusters de workload), enquanto equipes responsáveis pela aplicação definem e administram recursos de nuvem juntamente com as workloads.

Escolha de um padrão

Considere estes fatores ao selecionar uma arquitetura:

- Estrutura organizacional: equipes responsáveis pela plataforma centralizadas geralmente optam por padrões de hub, enquanto equipes descentralizadas podem preferir funcionalidades por cluster
- Escopo dos recursos: recursos de escopo administrativo, como bancos de dados e perfis do IAM, costumam se beneficiar de um gerenciamento centralizado; recursos de workload, por sua vez, como buckets e filas, podem ser administrados localmente
- Autoatendimento: equipes responsáveis pela plataforma centralizadas podem desenvolver e distribuir recursos personalizados prescritivos, possibilitando o autoatendimento seguro de recursos de nuvem para demandas comuns de workloads
- Gerenciamento da frota de clusters: clusters de gerenciamento centralizados fornecem um ambiente de gerenciamento de propriedade do cliente para o gerenciamento da frota de clusters do EKS, juntamente com outros recursos de escopo administrativo
- Requisitos de conformidade: algumas organizações requerem controle centralizado para fins de auditoria e governança
- Complexidade operacional: menos instâncias de funcionalidades simplificam as operações, mas podem gerar gargalos

> **Note**
>
> É possível começar a usar um padrão e migrar para outro à medida que a plataforma se desenvolve. As funcionalidades são independentes, permitindo que você as implante de formas diferentes em clusters distintos, de acordo com suas necessidades.

Comparação entre funcionalidades do EKS e soluções autogerenciadas

As funcionalidades do EKS oferecem experiências totalmente gerenciadas para ferramentas e controladores populares do Kubernetes que são executadas no EKS. Isso difere das soluções autogerenciadas, que precisam ser instaladas e gerenciadas diretamente no cluster.

Principais diferenças

Implantação e gerenciamento

As funcionalidades do EKS são totalmente gerenciadas pela AWS, sem necessidade de instalação, configuração ou manutenção do software dos componentes. A AWS instala e gerencia automaticamente todas as definições de recursos personalizados (CRDs, na sigla em inglês) do Kubernetes necessárias no cluster.

Com soluções autogerenciadas, você instala e configura o software do cluster usando charts do Helm, kubectl ou outros operadores. Nas soluções autogerenciadas, você mantém controle total sobre o ciclo de vida do software e a configuração do runtime, possibilitando personalizações em todas as camadas da solução.

Operações e manutenção

A AWS gerencia a aplicação de patches e outras operações do ciclo de vida do software para as funcionalidades do EKS, com atualizações automáticas e patches de segurança. As funcionalidades do EKS são integradas aos recursos da AWS para configurações simplificadas, oferecem alta disponibilidade e tolerância a falhas incorporadas e eliminam a necessidade de solução de problemas de workloads de controladores dentro do cluster.

As soluções autogerenciadas requerem que você monitore a integridade dos componentes e dos logs, aplique patches de segurança e atualizações de versão, configure alta disponibilidade com várias réplicas e budgets de interrupção de pods, realize a solução e a correção de problemas em workloads de controladores, e gerencie lançamentos e versões. Você tem controle total sobre as implantações, mas isso geralmente exige soluções sob medida para acesso a clusters privados e outras integrações, que devem estar alinhadas aos padrões organizacionais e aos requisitos de conformidade da segurança.

Consumo de recursos

As funcionalidades do EKS são executadas no EKS e de forma externa aos clusters, liberando tanto recursos de nós quanto recursos de cluster. As funcionalidades não usam recursos de workloads do cluster, não consomem CPU ou memória nos nós de processamento, escalam automaticamente e têm impacto mínimo no planejamento de capacidade do cluster.

As soluções autogerenciadas executam controladores e outros componentes nos nós de processamento, consumindo diretamente recursos desses nós, IPs do cluster e outros recursos do cluster. O gerenciamento de serviços do cluster exige planejamento de capacidade para as workloads, além de planejamento e configuração de solicitações e limites de recursos para atender aos requisitos de escalabilidade e de alta disponibilidade.

Suporte a recursos

Por serem recursos de serviço totalmente gerenciados, as funcionalidades do EKS são naturalmente opinativas quando comparadas às soluções autogerenciadas. Apesar de as funcionalidades oferecerem suporte à maior parte dos recursos e casos de uso, a abrangência será diferente quando comparadas às soluções autogerenciadas.

Com as soluções autogerenciadas, você tem controle total sobre a configuração, os recursos opcionais e outros aspectos da funcionalidade do software. É possível executar imagens personalizadas próprias, ajustar todos os aspectos da configuração e manter controle completo sobre a funcionalidade da solução autogerenciada.

Considerações sobre custos

Cada recurso da funcionalidade do EKS tem um custo horário relacionado, que varia conforme o tipo de funcionalidade. Os recursos do cluster gerenciados por essa funcionalidade também têm custo horário próprio, com precificação separada. Para obter mais informações, consulte Preços do Amazon EKS.

As soluções autogerenciadas não têm custos diretos relacionados a cobranças da AWS, mas é necessário pagar pelos recursos de computação do cluster consumidos por controladores e workloads relacionados. Além do consumo de recursos de nós e de cluster, o custo total de propriedade das soluções autogerenciadas inclui despesas operacionais e com manutenção, solução de problemas e suporte.

Como escolher entre funcionalidades do EKS e soluções autogerenciadas

Funcionalidades do EKS Considere esta opção quando desejar reduzir a sobrecarga operacional e se concentrar em valor diferenciado no software e nos sistemas, em vez de nas operações de plataforma do cluster para requisitos fundamentais. Use as funcionalidades do EKS quando desejar minimizar a carga operacional de patches de segurança e o gerenciamento do ciclo de vida do software, liberar recursos de nós e do cluster para workloads de aplicação, simplificar a configuração e o gerenciamento de segurança, e se beneficiar da abrangência de suporte da AWS. As funcionalidades do EKS são adequadas para a maioria dos casos de uso em ambientes de produção e constituem a abordagem recomendada para novas implantações.

Soluções autogerenciadas Considere esta opção quando precisar de versões específicas da API de recursos do Kubernetes, desenvolvimentos personalizados de controladores, tiver automação e ferramentas existentes voltadas a implantações autogerenciadas, ou necessitar de um nível avançado de personalização das configurações de runtime dos controladores. Nas soluções autogerenciadas, é possível obter flexibilidade para casos de uso específicos, mantendo controle total sobre a implantação e a configuração de runtime.

> **Note**
>
> As funcionalidades do EKS podem coexistir com soluções autogerenciadas em seu cluster, permitindo que migrações sejam feitas de forma gradual.

Comparações específicas por funcionalidade

Para comparações detalhadas, incluindo funcionalidades específicas de cada funcionalidade, diferenças na versão original e caminhos de migração, consulte:

- the section called “Comparação em relação à ferramenta autogerenciada”
- the section called “Comparação em relação à ferramenta autogerenciada”
- the section called “Comparação em relação à ferramenta autogerenciada”

Implantação de recursos da AWS usando o Kubernetes com o AWS Controllers for Kubernetes (ACK)

O AWS Controllers for Kubernetes (ACK) permite definir e gerenciar recursos de serviços da AWS diretamente do Kubernetes. Com o AWS Controllers for Kubernetes (ACK), é possível gerenciar recursos de workloads e de infraestrutura de nuvem por meio de recursos personalizados do Kubernetes, juntamente com as workloads das aplicações, usando APIs e ferramentas conhecidas do Kubernetes.

Com as funcionalidades do EKS, o ACK é totalmente gerenciado pela AWS, o que remove a necessidade de instalar, fazer a manutenção e escalar os controladores do ACK em seus clusters.

Funcionamento do ACK

O ACK converte especificações de recursos personalizados do Kubernetes em chamadas de API da AWS. Quando você cria, atualiza ou exclui um recurso personalizado do Kubernetes que representa um recurso de serviço da AWS, o ACK realiza as chamadas de API da AWS necessárias para criar, atualizar ou excluir o recurso da AWS.

Cada recurso da AWS compatível com o ACK conta com sua própria definição de recurso personalizado (CRD, na sigla em inglês), que define o esquema da API do Kubernetes para

especificar a configuração. Por exemplo, o ACK fornece CRDs para o S3, incluindo buckets, políticas de bucket e outros recursos do S3.

O ACK realiza a reconciliação contínua entre o estado atual dos seus recursos da AWS e o estado desejado especificado nos recursos personalizados do Kubernetes Se um recurso se desviar do estado desejado, o ACK detecta essa divergência e executa ações corretivas para restabelecer o alinhamento. As alterações nos recursos do Kubernetes são refletidas imediatamente no estado dos recursos da AWS, enquanto a detecção passiva de desvios e a correção de alterações diretamente nos recursos da AWS podem demorar até dez horas (o período de ressincronização), embora normalmente ocorram muito antes.

Exemplo de manifesto de recurso do bucket do S3

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-ack-bucket
spec:
  name: my-unique-bucket-name
```

Ao aplicar esse recurso personalizado ao cluster, o ACK cria um bucket do Amazon S3 na sua conta, se ele ainda não existir. As alterações posteriores nesse recurso, por exemplo, a definição de uma camada de armazenamento diferente da camada padrão ou a adição de uma política, serão aplicadas ao recurso do S3 na AWS. Quando esse recurso é excluído do cluster, o bucket do S3 na AWS é excluído por padrão.

Benefícios do ACK

O ACK disponibiliza o gerenciamento de recursos da AWS de forma nativa no Kubernetes, permitindo que você gerencie recursos da AWS com as mesmas APIs e ferramentas do Kubernetes usadas pelas suas aplicações. Essa abordagem unificada simplifica o fluxo de trabalho de gerenciamento da infraestrutura, eliminando a necessidade de alternar entre diferentes ferramentas ou de aprender sistemas distintos de infraestrutura como código. Seus recursos da AWS são definidos declarativamente em manifestos do Kubernetes, permitindo fluxos de trabalho de GitOps e práticas de infraestrutura como código que se integram perfeitamente aos processos de desenvolvimento existentes.

O ACK realiza a reconciliação contínua entre o estado desejado e o estado atual dos recursos da AWS, corrigindo desvios e garantindo a consistência em toda a infraestrutura. Essa reconciliação

contínua garante que alterações imperativas realizadas de forma externa aos recursos da AWS sejam revertidas automaticamente para o estado definido em sua configuração, preservando a integridade da infraestrutura como código. É possível configurar o ACK para gerenciar recursos em diversas contas e regiões da AWS, possibilitando arquiteturas complexas em várias contas sem a necessidade de ferramentas adicionais.

Para organizações que estão no processo de migração de outras ferramentas de gerenciamento de infraestrutura, o ACK oferece suporte à adoção de recursos, permitindo que você traga recursos da AWS existentes para o gerenciamento do ACK sem precisar recriá-los. O ACK ainda disponibiliza recursos somente para leitura, permitindo a observação de recursos da AWS sem permissão para modificá-los, além de anotações para, opcionalmente, reter recursos da AWS mesmo quando o recurso do Kubernetes é excluído do cluster.

Para saber mais informações e começar a usar a funcionalidade do EKS para o ACK, consulte the section called “Conceitos do ACK” e the section called “Considerações”.

Serviços da AWS compatíveis

O ACK oferece suporte a uma ampla variedade de serviços da AWS, incluindo, mas não se limitando a:

- Amazon EC2
- Amazon S3
- Amazon RDS
- Amazon DynamoDB
- Amazon ElastiCache
- Amazon EKS
- Amazon SQS
- Amazon SNS
- AWS Lambda
- IAM da AWS

Todos os serviços da AWS listados como disponíveis para o público geral na origem são compatíveis com a funcionalidade do EKS para o ACK. Consulte a lista completa de serviços da AWS compatíveis para obter mais detalhes.

Integração com outras funcionalidades gerenciadas do EKS

O ACK pode ser integrado a outras funcionalidades gerenciadas do EKS.

- Argo CD: use o Argo CD para gerenciar a implantação de recursos do ACK em diversos clusters, possibilitando fluxos de trabalho de GitOps para a infraestrutura da AWS.
  - Embora o ACK aproveite os benefícios de GitOps quando combinado com o Argo CD, ele não depende de integração com o Git.
- kro (Kube Resource Orchestrator): use o kro para combinar recursos complexos a partir de recursos do ACK, criando abstrações de nível superior que simplificam o gerenciamento de recursos.
  - Com o kro, é possível criar recursos personalizados compostos que definem tanto recursos do Kubernetes quanto recursos da AWS. Esses recursos personalizados podem ser usados pelos membros da equipe para implantar aplicações complexas de forma rápida.

Conceitos básicos do ACK

Para começar a usar a funcionalidade do EKS para o ACK:

1. Crie e configure um perfil do IAM da funcionalidade com as permissões necessárias para que o ACK gerencie recursos da AWS em seu nome.
2. Crie um recurso da funcionalidade do ACK em seu cluster do EKS por meio do Console da AWS, da AWS CLI ou da ferramenta de infraestrutura como código de sua preferência.
3. Aplique os recursos personalizados do Kubernetes no cluster para começar a gerenciar os recursos da AWS diretamente no Kubernetes.

Criação de uma funcionalidade do ACK

Este capítulo explica como criar uma funcionalidade do ACK no cluster do Amazon EKS.

Pré-requisitos

Antes de criar uma funcionalidade do ACK, certifique-se de ter:

- Um cluster do Amazon EKS
- Um perfil do IAM para a funcionalidade com permissões para o ACK gerenciar recursos da AWS

- Permissões do IAM suficientes para criar recursos de funcionalidades em clusters do EKS
- A ferramenta de CLI apropriada instalada e configurada, ou o acesso ao console do EKS

Para obter instruções sobre como criar o perfil do IAM para a funcionalidade, consulte the section called “Perfil do IAM para a funcionalidade”.

> **Important**
>
> O ACK é uma funcionalidade de gerenciamento de infraestrutura que concede a capacidade de criar, modificar e excluir recursos da AWS. Trata-se de uma funcionalidade de escopo de administrador que deve ser controlada cuidadosamente. Qualquer pessoa com permissão para criar recursos do Kubernetes no cluster pode, efetivamente, criar recursos da AWS por meio do ACK, sujeito às permissões do perfil do IAM para a funcionalidade. O perfil do IAM para a funcionalidade que você fornece determina quais recursos da AWS o ACK pode criar e gerenciar. Para obter orientações sobre como criar um perfil apropriado com permissões de privilégio mínimo, consulte the section called “Perfil do IAM para a funcionalidade” e the section called “Considerações para funcionalidades do EKS”.

Escolha da ferramenta

É possível criar uma funcionalidade do ACK por meio do Console de gerenciamento da AWS, da AWS CLI ou do eksctl:

- the section called “Console”: use o Console para obter uma experiência guiada
- the section called “ AWS CLI”: use a AWS CLI para obter scripts e automação
- the section called “eksctl”: use o eksctl para obter uma experiência nativa do Kubernetes

O que ocorre durante a criação de uma funcionalidade do ACK

Ao criar uma funcionalidade do ACK:

1. O EKS cria o serviço da funcionalidade do ACK e o configura para monitorar e gerenciar recursos no cluster
2. As definições de recursos personalizados (CRDs, na sigla em inglês) são instaladas no cluster
3. A funcionalidade assume o perfil do IAM para a funcionalidade, que foi fornecido por você

4. O ACK começa a monitorar os recursos personalizados no cluster
5. O status da funcionalidade é alterado de `CREATING` para `ACTIVE`

Assim que estiver ativa, você poderá criar recursos personalizados do ACK no cluster para gerenciar recursos da AWS

Próximas etapas

Após criar a funcionalidade do ACK:

- the section called “Conceitos do ACK”: entenda os conceitos do ACK e comece a usar o serviço com os recursos da AWS
- the section called “Conceitos do ACK”: saiba mais sobre reconciliação, exportações de campos e padrões de adoção de recursos
- the section called “Configurar permissões”: configure permissões do IAM e padrões para várias contas

Criação de uma funcionalidade do ACK por meio do Console

Este tópico descreve como criar uma funcionalidade do AWS Controllers for Kubernetes (ACK) usando o Console de gerenciamento da AWS.

Criação da funcionalidade do ACK

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do cluster para abrir a página de detalhes do cluster.
3. Escolha a guia Funcionalidades.
4. No painel de navegação à esquerda, escolha AWS Controllers for Kubernetes (ACK).
5. Escolha Criar uma funcionalidade do AWS Controllers for Kubernetes.
6. No perfil da funcionalidade do IAM:
   - Se você já tiver um perfil da funcionalidade do IAM, selecione-o no menu suspenso
   - Se a criação de um perfil for necessária, escolha Criar perfil de administrador

     Isso abre o console do IAM em uma nova guia com a política de confiança preenchida previamente e a política gerenciada `AdministratorAccess`. É possível desmarcar esta política e adicionar outras permissões, se preferir.

Após criar o perfil, retorne ao console do EKS e o perfil será selecionado automaticamente.

> **Important**
>
> A política `AdministratorAccess` sugerida concede permissões abrangentes e destina-se a simplificar as etapas iniciais do uso do serviço. Para ambientes de produção, substitua essa política por uma personalizada que forneça somente as permissões exigidas pelos serviços específicos da AWS que você pretende gerenciar usando o ACK. Para obter orientações sobre como criar políticas de privilégio mínimo, consulte the section called “Configurar permissões” e the section called “Considerações para funcionalidades do EKS”.

7. Escolha Criar.

O processo de criação da funcionalidade é iniciado.

Verificação da ativação da funcionalidade

1. Na guia Funcionalidades, visualize o status da funcionalidade do ACK.
2. Aguarde até que o status mude de `CREATING` para `ACTIVE`.
3. Assim que estiver com o status ativo, a funcionalidade estará pronta para uso.

Para obter informações sobre os status das funcionalidades e sobre a solução de problemas, consulte the section called “Como trabalhar com funcionalidades”.

Verificação da disponibilidade de recursos personalizados

Após a funcionalidade estar ativa, verifique se os recursos personalizados do ACK estão disponíveis no cluster.

Utilizar o console

1. Navegue até o cluster no console do Amazon EKS
2. Escolha a guia Recursos
3. Escolha Extensões
4. Escolha CustomResourceDefinitions

Você deverá visualizar várias CRDs listadas para os recursos da AWS.

Uso do kubectl

```
kubectl api-resources | grep services.k8s.aws
```

Você deverá visualizar várias APIs listadas para os recursos da AWS.

> **Note**
>
> A funcionalidade do AWS Controllers for Kubernetes instalará diversas CRDs para vários tipos de recursos da AWS.

Próximas etapas

- the section called “Conceitos do ACK”: entenda os conceitos do ACK e comece a usar o serviço
- the section called “Configurar permissões”: configure as permissões do IAM para outros serviços da AWS
- the section called “Como trabalhar com funcionalidades”: gerencie os recursos da funcionalidade do ACK

Criação de uma funcionalidade do ACK por meio da AWS CLI

Este tópico descreve como criar uma funcionalidade do AWS Controllers for Kubernetes (ACK) usando a AWS CLI.

Pré-requisitos

- AWS CLI: versão `2.12.3` ou em versões posteriores. Para verificar a versão, execute `aws --version`. Para obter mais informações, consulte Instalação no Guia do Usuário da Interface de Linha de Comando AWS.
- **`kubectl`** – uma ferramenta de linha de comando para trabalhar com clusters do Kubernetes. Para obter mais informações, consulte the section called “Configuração do kubectl e eksctl”.

Etapa 1: criação de um perfil da funcionalidade do IAM

Crie um arquivo de política de confiança:

```
cat > ack-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Crie o perfil do IAM:

```
aws iam create-role \
  --role-name ACKCapabilityRole \
  --assume-role-policy-document file://ack-trust-policy.json
```

Anexe a política gerenciada `AdministratorAccess` ao perfil:

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

Important

A política `AdministratorAccess` sugerida concede permissões abrangentes e destina-se a simplificar as etapas iniciais do uso do serviço. Para ambientes de produção, substitua essa política por uma personalizada que forneça somente as permissões exigidas pelos serviços específicos da AWS que você pretende gerenciar usando o ACK. Para obter orientações sobre como criar políticas de privilégio mínimo, consulte the section called “Configurar permissões” e the section called “Considerações para funcionalidades do EKS”.

Etapa 2: criação da funcionalidade do ACK

Crie o recurso da funcionalidade do ACK no cluster. Substitua *region-code* pela região AWS em que seu cluster se encontra e substitua *my-cluster* pelo nome do seu cluster.

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack \
  --type ACK \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/ACKCapabilityRole \
  --delete-propagation-policy RETAIN
```

O comando é concluído de imediato, mas a funcionalidade demora algum tempo para se tornar ativa, conforme o EKS cria a infraestrutura e os componentes necessários para a funcionalidade. O EKS instalará as definições de recursos personalizados do Kubernetes relacionadas a essa funcionalidade no cluster durante o processo de criação.

> **Note**
>
> Caso ocorra um erro indicando a inexistência do cluster ou falta de permissões, verifique o seguinte:
>
> - Se o nome do cluster está correto
> - Se a AWS CLI está configurada para a região correta
> - Se você tem as permissões do IAM obrigatórias

Etapa 3: verificação da ativação da funcionalidade

Aguarde até que a funcionalidade se torne ativa. Substitua *region-code* pela região AWS em que seu cluster se encontra e substitua *my-cluster* pelo nome do seu cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack \
  --query 'capability.status' \
  --output text
```

A funcionalidade estará pronta assim que o status mostrar `ACTIVE`. Não prossiga para a próxima etapa até que o status seja `ACTIVE`.

Também é possível visualizar os detalhes completos da funcionalidade:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack
```

Etapa 4: verificação da disponibilidade de recursos personalizados

Após a funcionalidade estar ativa, verifique se os recursos personalizados do ACK estão disponíveis no cluster:

```
kubectl api-resources | grep services.k8s.aws
```

Você deverá visualizar várias APIs listadas para os recursos da AWS.

> **Note**
>
> A funcionalidade do AWS Controllers for Kubernetes instalará diversas CRDs para vários tipos de recursos da AWS.

Próximas etapas

- the section called “Conceitos do ACK”: entenda os conceitos do ACK e comece a usar o serviço
- the section called “Configurar permissões”: configure as permissões do IAM para outros serviços da AWS
- the section called “Como trabalhar com funcionalidades”: gerencie os recursos da funcionalidade do ACK

Criação de uma funcionalidade do ACK por meio do eksctl

Este tópico descreve como criar uma funcionalidade do AWS Controllers for Kubernetes (ACK) usando o eksctl.

> **Note**
>
> Para realizar as etapas seguintes, você deve estar utilizando o eksctl na versão `0.220.0` ou em versões posteriores. Para verificar a versão, execute `eksctl version`.

Etapa 1: criação de um perfil da funcionalidade do IAM

Crie um arquivo de política de confiança:

```
cat > ack-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Crie o perfil do IAM:

```
aws iam create-role \
  --role-name ACKCapabilityRole \
  --assume-role-policy-document file://ack-trust-policy.json
```

Anexe a política gerenciada `AdministratorAccess` ao perfil:

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

> **Important**
>
> A política `AdministratorAccess` sugerida concede permissões abrangentes e destina-se a simplificar as etapas iniciais do uso do serviço. Para ambientes de produção, substitua essa política por uma personalizada que forneça somente as permissões exigidas pelos serviços específicos da AWS que você pretende gerenciar usando o ACK. Para obter orientações sobre como criar políticas de privilégio mínimo, consulte the section called "Configurar permissões" e the section called "Considerações para funcionalidades do EKS".

> **Important**
>
> Esta política concede permissões para o gerenciamento de buckets do S3 com `"Resource": "*"`, o que permite operações em todos os buckets do S3.
> Para uso em ambientes de produção: * Restrinja o campo `Resource` a ARNs de buckets específicos ou padrões de nomes * Use chaves de condição do IAM para limitar o acesso por etiquetas de recurso * Conceda apenas as permissões mínimas necessárias para o seu caso de uso
> Para uso com outros serviços da AWS, consulte the section called "Configurar permissões".

Anexe a política à função:

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):policy/ACKS3Policy
```

Etapa 2: criação da funcionalidade do ACK

Crie a funcionalidade do ACK por meio do eksctl. Substitua *`region-code`* pela região AWS em que seu cluster se encontra e substitua *`my-cluster`* pelo nome do seu cluster.

```
eksctl create capability \
  --cluster [.replaceable]`my-cluster` \
  --region [.replaceable]`region-code` \
  --name ack \
  --type ACK \
```

```
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
text):role/ACKCapabilityRole \
  --ack-service-controllers s3
```

> **Note**
>
> O sinalizador `--ack-service-controllers` é opcional. Se omitido, o ACK habilita todos os controladores disponíveis. Para otimizar a performance e a segurança, recomendamos habilitar somente os controladores necessários. É possível definir vários controladores ao mesmo tempo: `--ack-service-controllers s3,rds,dynamodb`

O comando é executado de forma imediata, mas a funcionalidade demora algum tempo para se tornar ativa.

Etapa 3: verificação da ativação da funcionalidade

Verifique o status da funcionalidade:

```
eksctl get capability \
  --cluster [.replaceable]`my-cluster` \
  --region [.replaceable]`region-code` \
  --name ack
```

A funcionalidade estará pronta assim que o status mostrar `ACTIVE`.

Etapa 4: verificação da disponibilidade de recursos personalizados

Após a funcionalidade estar ativa, verifique se os recursos personalizados do ACK estão disponíveis no cluster:

```
kubectl api-resources | grep services.k8s.aws
```

Você deverá visualizar várias APIs listadas para os recursos da AWS.

> **Note**
>
> A funcionalidade do AWS Controllers for Kubernetes instalará diversas CRDs para vários tipos de recursos da AWS.

Próximas etapas

- the section called “Conceitos do ACK”: entenda os conceitos do ACK e comece a usar o serviço
- the section called “Configurar permissões”: configure as permissões do IAM para outros serviços da AWS
- the section called “Como trabalhar com funcionalidades”: gerencie os recursos da funcionalidade do ACK

Conceitos do ACK

O ACK gerencia recursos da AWS usando APIs do Kubernetes, por meio da reconciliação contínua entre o estado desejado definido em seus manifestos e o estado atual na AWS. Ao criar ou atualizar um recurso personalizado do Kubernetes, o ACK faz as chamadas de API da AWS necessárias para criar ou modificar o recurso correspondente na AWS, monitorando-o em seguida para detectar desvios e atualizando o status no Kubernetes para refletir o estado atual. Esta abordagem possibilita o gerenciamento da infraestrutura por meio de ferramentas e fluxos de trabalho conhecidos do Kubernetes, ao mesmo tempo em que garante a conformidade entre o cluster e a AWS.

Este tópico explica os conceitos fundamentais sobre como o ACK realiza a gestão de recursos da AWS por meio de APIs do Kubernetes.

Conceitos básicos do ACK

Após criar a funcionalidade do ACK (consulte the section called “Criação da funcionalidade do ACK”), você pode começar a gerenciar recursos da AWS no seu cluster usando manifestos do Kubernetes.

Como exemplo, crie este manifesto de bucket do S3 em `bucket.yaml`, definindo um nome exclusivo para o seu bucket.

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-test-bucket
  namespace: default
spec:
  name: my-unique-bucket-name-12345
```

Aplique o manifesto:

```
kubectl apply -f bucket.yaml
```

Verifique o status:

```
kubectl get bucket my-test-bucket
kubectl describe bucket my-test-bucket
```

Verifique se o bucket foi criado na AWS:

```
aws s3 ls | grep my-unique-bucket-name-12345
```

Exclua o recurso do Kubernetes:

```
kubectl delete bucket my-test-bucket
```

Verifique se o bucket foi excluído da AWS:

```
aws s3 ls | grep my-unique-bucket-name-12345
```

O bucket não deve mais aparecer na lista, demonstrando que o ACK gerencia todo o ciclo de vida dos recursos da AWS.

Para obter mais informações sobre os conceitos básicos do ACK, consulte Getting Started with ACK.

Ciclo de vida e reconciliação de recursos

O ACK usa um ciclo contínuo de reconciliação para assegurar que os recursos da AWS estejam em conformidade com o estado desejado estabelecido nos manifestos do Kubernetes.

Funcionamento da reconciliação:

1. Você cria ou atualiza um recurso personalizado do Kubernetes (por exemplo, um bucket do S3)
2. O ACK detecta a alteração e compara o estado desejado com o estado real na AWS
3. Se houver divergência, o ACK faz chamadas de API da AWS para reconciliar a diferença
4. O status do recurso no Kubernetes é atualizado pelo ACK para refletir o estado atual

5. Esse ciclo se repete de forma contínua, geralmente em intervalos de algumas horas

O processo de reconciliação ocorre ao criar um novo recurso no Kubernetes, ao atualizar o campo `spec` de um recurso já existente ou caso o ACK detecte divergências na AWS causadas por alterações manuais realizadas fora do controle do ACK. Além disso, o ACK executa a reconciliação periódica com um intervalo de sincronização de dez horas. As alterações nos recursos do Kubernetes acionam a reconciliação imediata, enquanto a detecção passiva de desvios em alterações de recursos da AWS em suas versões originais ocorre durante a sincronização periódica.

Ao seguir o exemplo de introdução acima, o ACK executa estas etapas:

1. Verifica se o bucket existe na AWS
2. Se não existir, faz a chamada `s3:CreateBucket`
3. Atualiza o status no Kubernetes com o ARN e o estado do bucket
4. Continua o monitoramento para detectar desvios

Para saber mais sobre como o ACK funciona, consulte ACK Reconciliation.

Condições de status

Os recursos do ACK usam condições de status para informar seus estados. Compreender essas condições ajuda você a solucionar problemas e entender a integridade do recurso.

- Ready: indica que o recurso está pronto para uso (condição padrão do Kubernetes).
- ACK.ResourceSynced: indica que a especificação do recurso corresponde ao estado do recurso na AWS.
- ACK.Terminal: indica a ocorrência de um erro irrecuperável.
- ACK.Adopted: indica que o recurso foi importado de um recurso da AWS já existente, não sendo uma nova criação.
- ACK.Recoverable: indica um erro passível de recuperação que pode ser solucionado sem alterar a especificação.
- ACK.Advisory: fornece informações consultivas sobre o recurso.
- ACK.LateInitialized: indica se a inicialização tardia dos campos foi concluída.
- ACK.ReferencesResolved: Indica se todos os campos de `AWSResourceReference` foram resolvidos.

- ACK.IAMRoleSelected: Indica se um IAMRoleSelector foi selecionado para gerenciar este recurso.

Verifique o status do recurso:

```
# Check if resource is ready
kubectl get bucket my-bucket -o jsonpath='{.status.conditions[?
(@.type=="Ready")].status}'

# Check for terminal errors
kubectl get bucket my-bucket -o jsonpath='{.status.conditions[?
(@.type=="ACK.Terminal")]}'
```

Status de exemplo:

```
status:
  conditions:
  - type: Ready
    status: "True"
    lastTransitionTime: "2024-01-15T10:30:00Z"
  - type: ACK.ResourceSynced
    status: "True"
    lastTransitionTime: "2024-01-15T10:30:00Z"
  - type: ACK.Terminal
    status: "True"
  ackResourceMetadata:
    arn: arn:aws:s3:::my-unique-bucket-name
    ownerAccountID: "111122223333"
    region: us-west-2
```

Para saber mais sobre o status e as condições do ACK, consulte ACK Conditions.

Políticas de exclusão

A política de exclusão do ACK controla o que acontece com os recursos da AWS quando você exclui o recurso do Kubernetes.

Exclusão (padrão)

O recurso da AWS é excluído quando você exclui o recurso do Kubernetes. Este é o comportamento padrão.

```
# No annotation needed - this is the default
```

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: temp-bucket
spec:
  name: temporary-bucket
```

A exclusão deste recurso exclui o bucket do S3 na AWS.

Manter

O recurso da AWS é mantido quando você exclui o recurso do Kubernetes:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: important-bucket
  annotations:
    services.k8s.aws/deletion-policy: "retain"
spec:
  name: production-data-bucket
```

A exclusão deste recurso o remove do Kubernetes, mas mantém o bucket do S3 na AWS.

A política `retain` é útil para bancos de dados de produção que devem existir além do ciclo de vida do recurso do Kubernetes, recursos compartilhados que são usados por diversas aplicações, recursos com dados importantes que não devem ser excluídos acidentalmente ou casos de uso em que o ACK é usado apenas para configurar o recurso e, depois, o libera de volta para o gerenciamento manual.

Para saber mais sobre a política de exclusão do ACK, consulte ACK Deletion Policy.

Adoção de recursos

A adoção permite que você traga recursos existentes da AWS para o gerenciamento do ACK, sem a necessidade de recriá-los.

Situações para o uso da adoção:

- Ao migrar a infraestrutura atual para ser gerenciada pelo ACK
- Ao recuperar recursos da AWS que ficaram “órfãos” em caso de exclusão acidental do recurso no Kubernetes

- Ao importar recursos que foram gerados por outras ferramentas, como Terraform ou CloudFormation

Funcionamento da adoção:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: existing-bucket
  annotations:
    services.k8s.aws/adoption-policy: "adopt-or-create"
spec:
  name: my-existing-bucket-name
```

Ao criar este recurso:

1. O ACK verifica se um bucket com esse nome existe na AWS
2. Se encontrado, o ACK o adota (sem realizar chamadas de API para criação)
3. O ACK importa as configurações atuais da AWS
4. O ACK atualiza o status no Kubernetes para refletir o estado real
5. A partir daí, novas alterações seguem o fluxo de reconciliação comum

Após a adoção, o gerenciamento de recursos funciona como em qualquer outro recurso do ACK. Isso significa que remover o recurso do Kubernetes apagará o recurso na AWS, exceto se a política de deleção estiver configurada como `retain`.

Ao adotar recursos, o recurso da AWS já deve existir e o ACK precisa de permissões de leitura para localizá-lo. A política `adopt-or-create` faz a adoção se o recurso for localizado ou a criação automática se ele não existir. Isso é útil quando você deseja um fluxo de trabalho declarativo que funcione independentemente da existência prévia do recurso.

Para saber mais sobre a adoção de recursos no ACK, consulte ACK Resource Adoption.

Recursos entre contas e entre regiões

O ACK pode gerenciar recursos em diferentes contas e regiões da AWS usando um único cluster.

Anotações para recursos entre regiões

É possível especificar a região de um recurso da AWS utilizando uma anotação:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: eu-bucket
  annotations:
    services.k8s.aws/region: eu-west-1
spec:
  name: my-eu-bucket
```

Você também pode especificar a região de todos os recursos da AWS criados em um determinado namespace:

Anotações para namespace

Define uma região padrão para todos os recursos dentro de um namespace:

```
apiVersion: v1
kind: Namespace
metadata:
  name: production
  annotations:
    services.k8s.aws/default-region: us-west-2
```

Os recursos criados sob este namespace seguirão essa região, exceto se houver uma anotação específica no próprio recurso que a sobreponha.

Entre contas

Use os seletores de perfil do IAM para vincular perfis específicos do IAM a determinados namespaces:

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
  name: target-account-config
spec:
  arn: arn:aws:iam::444455556666:role/ACKTargetAccountRole
  namespaceSelector:
    names:
```

```
      - production
```

Os recursos criados no namespace mapeado usam automaticamente o perfil especificado.

Para saber mais sobre os seletores de perfil do IAM, consulte ACK Cross-Account Resource Management. Para obter detalhes sobre a configuração entre contas, consulte the section called "Configurar permissões".

Tratamento de erros e comportamento de novas tentativas

O ACK trata automaticamente os erros transitórios, repetindo operações com falhas.

Estratégia de novas tentativas:

- Erros transitórios (por exemplo, limite de taxa, problemas temporários no serviço e permissões insuficientes) acionam automaticamente novas tentativas
- O recuo exponencial é usado para não sobrecarregar as APIs da AWS
- O número máximo de tentativas varia de acordo com o tipo de erro
- Erros permanentes (por exemplo parâmetros inválidos e conflitos de nome de recurso) não geram novas tentativas

Verifique os detalhes do erro no status do recurso utilizando `kubectl describe`:

```
kubectl describe bucket my-bucket
```

Verifique as condições de status em busca de mensagens de erro, eventos que apresentam tentativas de reconciliação recentes e o campo `message`, que detalha o motivo das falhas nas condições de status. Erros comuns incluem permissões do IAM insuficientes, conflitos de nomes de recursos na AWS, valores de configuração inválidos na `spec` e cotas de serviço da AWS excedidas.

Para solucionar erros comuns, consulte the section called "Solução de problemas".

Composição de recursos com o kro

Para compor e conectar diversos recursos do ACK entre si, empregue a funcionalidade do EKS para o kro (Kube Resource Orchestrator). Com o kro, você define grupos de recursos de forma declarativa, permitindo a troca de configurações entre os recursos para simplificar o gerenciamento de arquiteturas complexas.

Para obter exemplos detalhados de como criar composições personalizadas de recursos com recursos do ACK, consulte the section called “Conceitos do kro”.

Próximas etapas

- the section called “Considerações”: acesse padrões específicos do EKS e estratégias de integração

Configuração das permissões do ACK

O ACK precisa de permissões do IAM para realizar a criação e o gerenciamento de recursos da AWS em seu nome. Este tópico explica como o IAM funciona com o ACK e fornece orientações sobre como configurar permissões para diferentes casos de uso.

Funcionamento do IAM com o ACK

O ACK usa perfis do IAM para se autenticar na AWS e executar ações em seus recursos. Existem duas maneiras de fornecer permissões ao ACK:

Perfil da funcionalidade: consiste no perfil do IAM que você fornece ao criar a funcionalidade do ACK. Este perfil é usado por padrão para todas as operações do ACK.

Seletores de perfil do IAM: consistem em perfis adicionais do IAM que podem ser mapeados para namespaces ou recursos específicos. Esses perfis substituem o perfil da funcionalidade para recursos dentro de seu escopo.

Quando o ACK precisa criar ou gerenciar um recurso, ele determina qual perfil do IAM deve usar:

1. Verifica se um “IAMRoleSelector” corresponde ao namespace do recurso
2. Se uma correspondência for encontrada, assume esse perfil do IAM
3. Caso contrário, usa o perfil da funcionalidade

Esta abordagem possibilita um gerenciamento de permissões flexível, desde configurações simples de perfil único até configurações complexas de diversas contas e várias equipes.

Conceitos básicos: configuração simples de permissões

Para ambientes de desenvolvimento, testes ou casos de uso simples, você pode adicionar todas as permissões de serviço necessárias diretamente ao perfil da funcionalidade.

Essa abordagem é ideal quando:

- Você está começando a usar o ACK
- Todos os recursos estão na mesma conta da AWS
- Uma única equipe gerencia todos os recursos do ACK
- Você confia que todos os usuários do ACK tenham as mesmas permissões

Prática recomendada para ambientes de produção: seletores de perfil do IAM

Para ambientes de produção, use os seletores de perfil do IAM para implementar o acesso de privilégio mínimo e o isolamento em nível de namespace.

Ao usar seletores de perfil do IAM, o perfil da funcionalidade precisa apenas da permissão `sts:AssumeRole` para assumir os perfis específicos de cada serviço. Não é necessário adicionar nenhuma permissão de serviço da AWS (como S3 ou RDS) ao perfil da funcionalidade em si, pois essas permissões são concedidas aos perfis individuais do IAM que o perfil da funcionalidade assume.

Escolha do modelo de permissões:

Use permissões diretas, com a adição de permissões de serviço ao perfil da funcionalidade, quando:

- Você está começando a usar o serviço e deseja a configuração mais simples
- Todos os recursos estão na mesma conta que o cluster
- Você tem requisitos de permissão administrativos para todo o cluster
- Todas as equipes podem compartilhar as mesmas permissões

Use seletores de perfil do IAM quando:

- Gerenciar recursos em várias contas da AWS
- Diferentes equipes ou namespaces precisarem de permissões distintas
- Você precisar de controle de acesso granular por namespace
- Você desejar seguir as práticas de segurança de privilégio mínimo

É possível começar com permissões diretas e migrar para os seletores de perfil do IAM posteriormente, conforme suas necessidades evoluem.

Vantagens de usar seletores de perfil do IAM em ambientes produção:

- Privilégio mínimo: cada namespace recebe apenas as permissões de que necessita
- Isolamento de equipes: a Equipe A não pode usar as permissões da Equipe B por engano
- Auditoria facilitada: mapeamento claro de qual namespace usa cada perfil
- Suporte entre contas: necessário para gerenciar recursos em várias contas
- Separação de responsabilidades: diferentes serviços ou ambientes usam perfis distintos

Configuração básica do seletor de perfil do IAM

Etapa 1: criação de um perfil do IAM específico para o serviço

Crie um perfil do IAM com permissões para serviços específicos da AWS:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Configure a política de confiança para permitir que o perfil da funcionalidade o assuma:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ACKCapabilityRole"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

Etapa 2: concessão da permissão AssumeRole ao perfil da funcionalidade

Adicione permissão ao perfil da funcionalidade para assumir o perfil específico do serviço:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/ACK-S3-Role"
    }
  ]
}
```

Etapa 3: criação do IAMRoleSelector

Mapeie o perfil do IAM para um namespace:

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
  name: s3-namespace-config
spec:
  arn: arn:aws:iam::111122223333:role/ACK-S3-Role
  namespaceSelector:
    names:
      - s3-resources
```

Etapa 4: criação de recursos no namespace mapeado

Os recursos no namespace `s3-resources` usam automaticamente o perfil especificado:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-bucket
  namespace: s3-resources
spec:
```

```
  name: my-production-bucket
```

Gerenciamento de várias contas

Use seletores de perfil do IAM para gerenciar recursos em várias contas da AWS.

Etapa 1: criação do perfil do IAM entre contas

Na conta de destino (444455556666), crie um perfil que confie no perfil da funcionalidade da conta de origem:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ACKCapabilityRole"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Anexe permissões específicas de serviço a este perfil.

Etapa 2: concessão da permissão AssumeRole

Na conta de origem (111122223333), permita que o perfil da funcionalidade assuma o perfil da conta de destino:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::444455556666:role/ACKTargetAccountRole"
    }
  ]
}
```

Etapa 3: criação do IAMRoleSelector

Mapeie o perfil entre contas para um namespace:

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
  name: production-account-config
spec:
  arn: arn:aws:iam::444455556666:role/ACKTargetAccountRole
  namespaceSelector:
    names:
      - production
```

Etapa 4: criação de recursos

Os recursos no namespace `production` são criados na conta de destino:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-bucket
  namespace: production
spec:
  name: my-cross-account-bucket
```

Padrões avançados de seletores de perfil do IAM

Para obter configurações avançadas, incluindo selecionadores de rótulos, mapeamento de perfis específicos para recursos e exemplos adicionais, consulte a documentação relacionada ao IRSA para o ACK.

Próximas etapas

- the section called “Conceitos do ACK”: compreenda os conceitos do ACK e o ciclo de vida dos recursos
- the section called “Conceitos do ACK”: saiba mais sobre as políticas de adoção e exclusão de recursos
- the section called “Considerações para funcionalidades do EKS”: compreenda as práticas recomendadas de segurança para as funcionalidades

Considerações sobre o ACK para o EKS

Este tópico aborda considerações importantes para o uso da funcionalidade do EKS para o ACK, incluindo a configuração do IAM, os padrões para várias contas e a integração com outras funcionalidades do EKS.

Padrões de configuração do IAM

A funcionalidade do ACK usa um perfil de funcionalidade do IAM para se autenticar na AWS. Escolha o padrão do IAM mais adequado às suas necessidades.

Simples: um único perfil de funcionalidade

Para ambientes de desenvolvimento, testes ou casos de uso simples, atribua todas as permissões necessárias diretamente ao perfil de funcionalidade.

Quando utilizar:

- Conceitos básicos do ACK
- Implantações em uma única conta
- Todos os recursos são gerenciados por uma única equipe
- Ambientes de desenvolvimento e de teste

Exemplo: adicione permissões do S3 e do RDS ao seu perfil de funcionalidade com condições de marcação de recursos:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:*"],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-west-2", "us-east-1"]
        }
      }
    },
```

```
    {
      "Effect": "Allow",
      "Action": ["rds:*"],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-west-2", "us-east-1"]
        },
      }
    }
  ]
}
```

Este exemplo restringe as operações do S3 e do RDS a regiões específicas e requer que os recursos do RDS tenham uma etiqueta `ManagedBy: ACK`.

Produção: seletores de perfil do IAM

Para ambientes de produção, use os seletores de perfil do IAM para implementar o acesso de privilégio mínimo e o isolamento em nível de namespace.

Quando utilizar:

- Ambientes de produção
- Clusters gerenciados por várias equipes
- Gerenciamento de recursos em várias contas
- Requisitos de segurança de privilégio mínimo
- Diferentes serviços que necessitam de permissões distintas

Benefícios:

- Cada namespace recebe apenas as permissões de que necessita
- Isolamento de equipes, portanto, a Equipe A não pode usar as permissões da Equipe B
- Auditoria e conformidade facilitadas
- Obrigatório para o gerenciamento de recursos entre contas

Para obter a configuração detalhada do seletor de perfil do IAM, consulte the section called “Configurar permissões”.

Integração com outras funcionalidades do EKS

GitOps com Argo CD

Empregue a funcionalidade do EKS para o Argo CD a fim de implantar recursos do ACK de repositórios do Git, permitindo fluxos de trabalho do GitOps para o gerenciamento de infraestrutura.

Considerações:

- Armazene os recursos do ACK junto aos manifestos da aplicação para um GitOps de ponta a ponta
- Organize por ambiente, serviço ou tipo de recurso com base na estrutura da sua equipe
- Use a sincronização automática do Argo CD para reconciliação contínua
- Habilite a supressão para remover automaticamente recursos excluídos
- Considere o uso de padrões hub-and-spoke para o gerenciamento da infraestrutura de vários clusters

O GitOps oferece trilhas de auditoria, funcionalidades de reversão e gerenciamento declarativo da infraestrutura. Para saber mais sobre o Argo CD, consulte the section called “Como trabalhar com o Argo CD”.

Composição de recursos com o kro

Use a funcionalidade do EKS para o kro (Kube Resource Orchestrator) para compor vários recursos do ACK em abstrações de nível superior e APIs personalizadas.

Situações para o uso do kro com o ACK:

- Criação de padrões reutilizáveis para pilhas de infraestrutura comuns (banco de dados + backup + monitoramento)
- Desenvolvimento de plataformas de autoatendimento com APIs simplificadas para as equipes destinadas ao trabalho com APIs aplicação
- Gerenciamento de dependências de recursos e transferência de valores entre recursos (como o ARN de um bucket do S3 para uma função do Lambda)
- Padronização das configurações de infraestrutura entre as equipes
- Redução da complexidade ao ocultar detalhes de implementação atrás de recursos personalizados

Padrões de exemplo:

- Pilha da aplicação: bucket do S3 + fila do SQS + configuração de notificação
- Configuração do banco de dados: instância do RDS + grupo de parâmetros + grupo de segurança + segredos
- Rede: VPC + sub-redes + tabelas de rotas + grupos de segurança

O kro gerencia a ordenação de dependências, a propagação de status e o gerenciamento do ciclo de vida dos recursos compostos. Para saber mais sobre o kro, consulte the section called “Conceitos do kro”.

Organização dos recursos

Organize os recursos do ACK usando namespaces do Kubernetes e etiquetas de recursos da AWS para obter melhor gerenciamento, controle de acesso e rastreamento de custos.

Organização de namespaces

Use namespaces do Kubernetes para a separação lógica de recursos do ACK por ambiente (por exemplo, produção, preparação e desenvolvimento), equipe (como plataforma, dados e ML) ou aplicação.

Benefícios:

- RBAC com escopo de namespace para controle de acesso
- Definição de regiões padrão por namespace usando anotações
- Gerenciamento e limpeza de recursos facilitados
- Separação lógica alinhada à estrutura organizacional

Marcação de recursos

O EKS aplica etiquetas automaticamente aos recursos da AWS gerenciados pelo ACK, incluindo o ARN do recurso de funcionalidade. Adicione etiquetas adicionais para alocação de custos, rastreamento de propriedade e fins organizacionais.

Etiquetas recomendadas:

- Ambiente (por exemplo, produção, preparação e desenvolvimento)

- Propriedade da equipe ou do departamento
- Centro de custo para alocação de faturamento
- Nome da aplicação ou do serviço
- ManagedBy: ACK (para identificar recursos gerenciados pelo ACK)

Migração de outras ferramentas de infraestrutura como código

Muitas organizações estão encontrando valor na padronização do Kubernetes além da orquestração de suas workloads. Migrar a infraestrutura e o gerenciamento de recursos da AWS para o ACK permite que você padronize o gerenciamento da infraestrutura utilizando as APIs do Kubernetes em conjunto com as workloads da aplicação.

Benefícios da padronização no Kubernetes para infraestrutura:

- Fonte única de verdade: gerencie tanto aplicações quanto infraestrutura no Kubernetes, permitindo uma prática de GitOps de ponta a ponta
- Ferramentas unificadas: as equipes usam recursos e ferramentas do Kubernetes em vez de aprender diversas ferramentas e frameworks
- Reconciliação consistente: o ACK reconcilia continuamente os recursos da AWS assim como o Kubernetes faz com as workloads, detectando e corrigindo desvios em comparação com ferramentas imperativas
- Composições nativas: com o kro e o ACK juntos, referencie recursos da AWS diretamente nos manifestos da aplicação e de recursos, transferindo strings de conexão e ARNs entre os recursos
- Operações simplificadas: um único ambiente de gerenciamento para implantações, reversões e observabilidade em todo o sistema

O ACK permite a adoção de recursos da AWS já existentes sem recriá-los, possibilitando migrações sem tempo de inatividade provenientes do CloudFormation, do Terraform ou de recursos externos ao cluster.

Adote um recurso existente:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: existing-bucket
```

```
  annotations:
    services.k8s.aws/adoption-policy: "adopt-or-create"
spec:
  name: my-existing-bucket-name
```

Após a adoção, o recurso passa a ser gerenciado pelo ACK e pode ser atualizado por meio de manifestos do Kubernetes. Você pode realizar a migração de forma incremental, adotando recursos conforme necessário, enquanto mantém as ferramentas de IaC existentes para outros recursos.

O ACK também oferece suporte a recursos somente de leitura. Para recursos gerenciados por outras equipes ou ferramentas que você deseja referenciar, mas não modificar, combine a adoção com a política de exclusão `retain` e conceda permissões somente de leitura no IAM. Isso permite que as aplicações localizem infraestruturas compartilhadas (por exemplo, VPCs, perfis de funcionalidade do IAM e chaves do KMS) por meio das APIs do Kubernetes, sem o risco de modificações acidentais.

Para obter mais informações sobre a adoção de recursos, consulte the section called “Conceitos do ACK”.

Políticas de exclusão

As políticas de exclusão controlam o que acontece com os recursos da AWS quando você exclui o recurso correspondente no Kubernetes. Escolha a política adequada com base no ciclo de vida do recurso e em seus requisitos operacionais.

Exclusão (padrão)

O recurso da AWS é excluído quando você exclui o recurso correspondente no Kubernetes. Isso mantém a consistência entre o cluster e a AWS, garantindo que não ocorra o acúmulo de recursos.

Situações para o uso da exclusão:

- Ambientes de desenvolvimento e de teste nos quais a limpeza é importante
- Recursos efêmeros vinculados ao ciclo de vida da aplicação (por exemplo, bancos de dados de teste e buckets temporários)
- Recursos que não devem existir sem a aplicação (por exemplo filas do SQS e clusters do ElastiCache)
- Otimização de custos com a limpeza automática de recursos não utilizados
- Ambientes gerenciados com GitOps, nos quais a remoção do recurso no Git deve excluir a infraestrutura

A política de exclusão padrão alinha-se ao modelo declarativo do Kubernetes, ou seja, o que está no cluster corresponde ao que existe na AWS.

Reter

O recurso da AWS é mantido quando você exclui o recurso correspondente no Kubernetes. Esta opção protege dados críticos e permite que os recursos permaneçam ativos mesmo após a remoção da representação no Kubernetes.

Situações para o uso da retenção:

- Bancos de dados de produção com dados críticos que devem ser mantidos apesar de mudanças no cluster
- Buckets de armazenamento de longo prazo com requisitos de conformidade ou auditoria
- Recursos compartilhados usados por diversas aplicações ou equipes
- Recursos em processo de migração para diferentes ferramentas de gerenciamento
- Cenários de recuperação de desastres em que você deseja preservar a infraestrutura
- Recursos com dependências complexas que exigem uma desativação cuidadosa

```
apiVersion: rds.services.k8s.aws/v1alpha1
kind: DBInstance
metadata:
  name: production-db
  annotations:
    services.k8s.aws/deletion-policy: "retain"
spec:
  dbInstanceIdentifier: prod-db
  # ... configuration
```

Important

Recursos retidos continuam a gerar custos na AWS e devem ser excluídos manualmente da AWS quando não forem mais necessários. Use a marcação de recursos para rastrear recursos retidos para limpeza posterior.

Para saber mais sobre as políticas de exclusão, consulte the section called “Conceitos do ACK”.

Documentação original

Para obter informações detalhadas sobre o uso do ACK:

- Guia de uso do ACK: criação e gerenciamento de recursos
- Referência da API do ACK: documentação completa da API para todos os serviços
- Documentação do ACK: documentação abrangente para o usuário

Próximas etapas

- the section called “Configurar permissões”: configure permissões do IAM e padrões para várias contas
- the section called “Conceitos do ACK”: compreenda os conceitos do ACK e o ciclo de vida dos recursos
- the section called “Solução de problemas”: solucione problemas do ACK
- the section called “Como trabalhar com o Argo CD”: implante recursos do ACK com GitOps
- the section called “Conceitos do kro”: componha recursos do ACK em abstrações de nível superior

Solução de problemas em funcionalidades do ACK

Este tópico fornece orientações para a solução de problemas à funcionalidade do EKS para o ACK, incluindo verificações de integridade da funcionalidade, verificação do status dos recursos e problemas relacionados à permissão do IAM.

> **Note**
>
> As funcionalidades do EKS são totalmente gerenciadas e executadas de forma externa ao cluster. Você não tem acesso aos logs do controlador nem aos namespaces do controlador. A solução de problemas se concentra na integridade da funcionalidade, no status dos recursos e na configuração do IAM.

Funcionalidade está no status ACTIVE, mas os recursos não estão sendo criados

Se a sua funcionalidade do ACK apresentar o status `ACTIVE`, mas os recursos não estiverem sendo criados na AWS, verifique a integridade da funcionalidade, o status dos recursos e as permissões do IAM.

Verifique a integridade da funcionalidade:

Você pode visualizar problemas de integridade e de status da funcionalidade no console do EKS ou usando a AWS CLI.

Console do:

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do seu cluster.
3. Escolha a guia Observabilidade.
4. Escolha Monitorar cluster.
5. Escolha a guia Funcionalidades para visualizar a integridade e o status de todas as funcionalidades.

AWS CLI:

```
# View capability status and health
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack

# Look for issues in the health section
```

Causas comuns:

- Ausência de permissões do IAM: o perfil da funcionalidade não conta com permissões para o serviço da AWS
- Namespace incorreto: recursos criados em um namespace sem o IAMRoleSelector adequado
- Especificação de recurso inválida: é necessário verificar as condições de status do recurso para encontrar erros de validação
- Controle de utilização da API: os limites de taxa da API da AWS foram atingidos

- Webhooks de admissão: os webhooks de admissão estão impedindo o controlador de atualizar o status do recurso

Verifique o status do recurso:

```
# Describe the resource to see conditions and events
kubectl describe bucket my-bucket -n default

# Look for status conditions
kubectl get bucket my-bucket -n default -o jsonpath='{.status.conditions}'

# View resource events
kubectl get events --field-selector involvedObject.name=my-bucket -n default
```

Verifique as permissões do IAM:

```
# View the Capability Role's policies
aws iam list-attached-role-policies --role-name my-ack-capability-role
aws iam list-role-policies --role-name my-ack-capability-role

# Get specific policy details
aws iam get-role-policy --role-name my-ack-capability-role --policy-name policy-name
```

Recursos são criados na AWS, mas que não aparecem no Kubernetes

O ACK rastreia apenas os recursos criados por meio de manifestos do Kubernetes. Para gerenciar recursos existentes da AWS com o ACK, empregue o recurso de adoção.

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: existing-bucket
  annotations:
    services.k8s.aws/adoption-policy: "adopt-or-create"
spec:
  name: my-existing-bucket-name
```

Para obter mais informações sobre a adoção de recursos, consulte the section called “Conceitos do ACK”.

Recursos entre contas não estão sendo criados

Se os recursos não estiverem sendo criados em uma conta de destino da AWS ao usar os seletores de perfil do IAM, verifique a relação de confiança e a configuração do IAMRoleSelector.

Verifique a relação de confiança:

```
# Check the trust policy in the target account role
aws iam get-role --role-name cross-account-ack-role --query
 'Role.AssumeRolePolicyDocument'
```

A política de confiança deve permitir que o perfil da funcionalidade da conta de origem a assuma.

Confirme a configuração do IAMRoleSelector:

```
# List IAMRoleSelectors (cluster-scoped)
kubectl get iamroleselector

# Describe specific selector
kubectl describe iamroleselector my-selector
```

Verifique o alinhamento do namespace:

Os IAMRoleSelectors são recursos de escopo do cluster, mas operam em namespaces específicos. Certifique-se de que os recursos do ACK estejam em um namespace que corresponda ao selecionador de namespace do IAMRoleSelector:

```
# Check resource namespace
kubectl get bucket my-cross-account-bucket -n production

# List all IAMRoleSelectors (cluster-scoped)
kubectl get iamroleselector

# Check which namespace the selector targets
kubectl get iamroleselector my-selector -o jsonpath='{.spec.namespaceSelector}'
```

Verifique a condição IAMRoleSelected:

Verifique se o IAMRoleSelector foi correspondido com sucesso ao seu recurso, consultando a condição `ACK.IAMRoleSelected`:

```
# Check if IAMRoleSelector was matched
kubectl get bucket my-cross-account-bucket -n production -o
 jsonpath='{.status.conditions[?(@.type=="ACK.IAMRoleSelected")]}'
```

Se a condição for `False` ou estiver ausente, o selecionador de namespace do IAMRoleSelector não corresponde ao namespace do recurso. Verifique se o `namespaceSelector` do seletor corresponde aos rótulos de namespace do seu recurso.

Verifique as permissões do perfil da funcionalidade:

O perfil da funcionalidade precisa da permissão `sts:AssumeRole` para o perfil da conta de destino:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::[.replaceable]`444455556666`:role/[.replaceable]`cross-
account-ack-role`"
    }
  ]
}
```

Para obter informações detalhadas sobre a configuração entre contas, consulte the section called “Configurar permissões”.

Próximas etapas

- the section called “Considerações”: acesse considerações e práticas recomendadas do ACK
- the section called “Configurar permissões”: configure permissões do IAM e padrões para várias contas
- the section called “Conceitos do ACK”: compreenda os conceitos do ACK e o ciclo de vida dos recursos
- the section called “Solução de problemas das funcionalidades”: acesse orientações gerais para solução de problemas de funcionalidades

Comparação da funcionalidade do EKS para o ACK em relação ao ACK autogerenciado

A funcionalidade do EKS para o ACK é equivalente à dos controladores do ACK autogerenciados, porém apresenta vantagens operacionais significativas. Para obter uma comparação geral das funcionalidades do EKS em relação às soluções autogerenciadas, consulte the section called “Considerações”. Este tópico se concentra nas diferenças específicas do ACK.

Diferenças em relação à versão original do ACK

A funcionalidade do EKS para o ACK se baseia na versão original dos controladores do ACK, porém apresenta diferenças na integração com o IAM.

Perfil da funcionalidade do IAM: a funcionalidade usa um perfil do IAM dedicado com uma política de confiança que permite o serviço `capabilities.eks.amazonaws.com` da entidade principal, e não o IRSA (perfis do IAM para contas de serviços). É possível anexar políticas do IAM diretamente ao perfil da funcionalidade, sem a necessidade de criar ou anotar contas de serviços do Kubernetes ou configurar provedores OIDC. Uma prática recomendada para casos de uso em ambientes de produção é configurar permissões de serviços usando o `IAMRoleSelector`. Consulte the section called “Configurar permissões” para obter mais detalhes.

Compatibilidade de recursos: os recursos personalizados do ACK funcionam de forma idêntica à versão original do ACK, sem a necessidade de alterações nos arquivos YAML de recursos do ACK. A funcionalidade usa as mesmas APIs do Kubernetes e CRDs, portanto, ferramentas como o `kubectl` funcionam de maneira semelhante. Todos os controladores e recursos em disponibilidade geral (GA, na sigla em inglês) da versão original do ACK são compatíveis.

Para obter a documentação completa do ACK e os guias específicos dos serviços, consulte a documentação do ACK.

Caminho de migração

Você pode migrar de um ACK autogerenciado para a funcionalidade gerenciada sem tempo de inatividade:

1. Atualize o controlador do ACK autogerenciado para usar `kube-system` para concessões de eleição de líder, por exemplo:

```
helm upgrade --install ack-s3-controller \
```

```
  oci://public.ecr.aws/aws-controllers-k8s/s3-chart \
  --namespace ack-system \
  --set leaderElection.namespace=kube-system
```

Isso transfere a concessão do controlador para o `kube-system`, permitindo que a funcionalidade gerenciada se coordene com ele.

2. Crie a funcionalidade do ACK no cluster (consulte the section called “Criação da funcionalidade do ACK”)
3. A funcionalidade gerenciada reconhece os recursos da AWS gerenciados pelo ACK existentes e assume a reconciliação
4. Reduza gradualmente a escala verticalmente ou remova as implantações dos controladores autogerenciados:

```
helm uninstall ack-s3-controller --namespace ack-system
```

Com essa abordagem, ambos os controladores podem coexistir com segurança durante a migração. A funcionalidade gerenciada assume automaticamente os recursos que antes eram gerenciados pelos controladores autogerenciados, assegurando reconciliação contínua sem gerar conflitos.

Próximas etapas

- the section called “Criação da funcionalidade do ACK”: crie um recurso de funcionalidade do ACK
- the section called “Conceitos do ACK”: compreenda os conceitos do ACK e o ciclo de vida dos recursos
- the section called “Configurar permissões”: configure o IAM e as permissões

Implantação contínua com o Argo CD

O Argo CD é uma ferramenta de entrega contínua declarativa e baseada em GitOps para Kubernetes. Com o Argo CD, você pode automatizar a implantação e o gerenciamento do ciclo de vida de suas aplicações em vários clusters e ambientes. O Argo CD oferece suporte a vários tipos de origem, incluindo repositórios do Git, registros do Helm (HTTP e OCI) e imagens OCI, proporcionando flexibilidade para organizações com requisitos de segurança e conformidade distintos.

Com as funcionalidades do EKS, o Argo CD é totalmente gerenciado pela AWS, o que elimina a necessidade de instalação, manutenção e escalabilidade dos controladores do Argo CD e de suas dependências nos clusters.

Funcionamento do Argo CD

O Argo CD segue o padrão de GitOps, no qual a origem da sua aplicação (por exemplo, repositório do Git, registro do Helm ou imagem OCI) é a fonte da verdade para definir o estado desejado da aplicação. Ao criar um recurso `Application` do Argo CD, você especifica a origem que contém os manifestos da aplicação e o cluster e o namespace do Kubernetes de destino. O Argo CD monitora continuamente tanto a origem quanto o estado em execução no cluster, sincronizando automaticamente quaisquer alterações para garantir que o estado do cluster corresponda ao estado desejado.

> **Note**
>
> Com a funcionalidade do EKS para o Argo CD, o software do Argo CD é executado no ambiente de gerenciamento da AWS, e não em seus nós de processamento. Isso significa que os nós de processamento não precisam de acesso direto a repositórios do Git ou a registros do Helm, pois a funcionalidade gerencia o acesso à origem a partir da conta da AWS.

O Argo CD fornece três tipos de recursos primários:

- Application: define uma implantação proveniente de um repositório do Git para um cluster de destino
- ApplicationSet: gera diversas Applications usando modelos para implantações em vários clusters
- AppProject: fornece agrupamento lógico e controle de acesso para as Applications

Exemplo: criação de uma Application do Argo CD

O exemplo a seguir mostra como criar um recurso `Application` do Argo CD:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
```

```
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    name: my-cluster
    namespace: guestbook
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

Benefícios do Argo CD

O Argo CD implementa um fluxo de trabalho de GitOps no qual você define as configurações da aplicação em repositórios do Git e o Argo CD sincroniza automaticamente as aplicações para corresponderem ao estado desejado. Essa abordagem centrada no Git fornece uma trilha de auditoria completa de todas as alterações, permite reversões fáceis e integra-se naturalmente aos seus processos existentes de revisão e aprovação de código. O Argo CD detecta e reconcilia automaticamente o desvio entre o estado desejado no Git e o estado real nos clusters, garantindo que as implantações permaneçam consistentes com a configuração declarada.

Com o Argo CD, é possível implantar e gerenciar aplicações em vários clusters usando uma única instância do Argo CD, simplificando as operações em ambientes com vários clusters e diversas regiões. A interface do usuário do Argo CD fornece funcionalidades de visualização e monitoramento, permitindo que você visualize o status da implantação, a integridade e o histórico das aplicações. A interface do usuário se integra ao Centro de Identidade da AWS (anteriormente AWS SSO) para autenticação e autorização simplificadas, permitindo que você controle o acesso com a infraestrutura de gerenciamento de identidades existente.

Como parte das funcionalidades gerenciadas do EKS, o Argo CD é totalmente gerenciado pela AWS, eliminando a necessidade de instalar, configurar e manter a infraestrutura do Argo CD. A AWS cuida da escalabilidade, da aplicação de patches e do gerenciamento operacional, permitindo que as equipes se concentrem na entrega de aplicações em vez da manutenção de ferramentas.

Integração com o Centro de Identidade da AWS

As funcionalidades gerenciadas do EKS fornecem integração direta entre o Argo CD e o Centro de Identidade da AWS, permitindo autenticação e autorização simplificadas para os usuários. Ao habilitar a funcionalidade do Argo CD, é possível configurar a integração com o Centro de Identidade da AWS para mapear grupos e usuários do Centro de Identidade para perfis de RBAC do Argo CD, permitindo controlar quem pode acessar e gerenciar aplicações no Argo CD.

Integração com outras funcionalidades gerenciadas do EKS

O Argo CD pode ser integrado a outras funcionalidades gerenciadas do EKS.

- AWS Controllers for Kubernetes (ACK): use o Argo CD para gerenciar a implantação de recursos do ACK em diversos clusters, possibilitando fluxos de trabalho de GitOps para a infraestrutura da AWS.
- kro (Kube Resource Orchestrator): use o Argo CD para implantar composições do kro em vários clusters, possibilitando uma composição de recursos consistente em todo o seu ambiente do Kubernetes.

Conceitos básicos do Argo CD

Para começar a usar a funcionalidade do EKS para o Argo CD:

1. Crie e configure um perfil da funcionalidade do IAM com as permissões necessárias para que o Argo CD acesse os repositórios do Git e gerencie aplicações
2. Crie um recurso da funcionalidade do Argo CD em seu cluster do EKS por meio do Console da AWS, da AWS CLI ou da ferramenta de infraestrutura como código de sua preferência.
3. Configure o acesso ao repositório e registre os clusters para a implantação de aplicações.
4. Crie recursos Application para implantar as aplicações usando repositórios do Git.

Criar um recurso Argo CD

Este tópico explica como criar uma funcionalidade do Argo CD no cluster do Amazon EKS.

Pré-requisitos

Antes de criar uma funcionalidade do Argo CD, certifique-se de ter:

- Um cluster do Amazon EKS existente executando uma versão compatível do Kubernetes (há suporte para todas as versões nos períodos de suporte padrão e estendido)
- Centro de Identidade da AWS configurado: necessário para a autenticação do Argo CD (não há suporte para usuários locais)
- Um perfil do IAM da funcionalidade com permissões para o Argo CD
- Permissões do IAM suficientes para criar recursos de funcionalidades em clusters do EKS
- `kubectl` configurado para o estabelecimento de comunicação com o cluster
- (Opcional) A CLI do Argo CD instalada para facilitar o gerenciamento de clusters e repositórios
- (Para a CLI ou o eksctl) A ferramenta de CLI apropriada instalada e configurada

Para obter instruções sobre como criar o perfil do IAM para a funcionalidade, consulte the section called “Perfil do IAM para a funcionalidade”. Para obter a configuração do Centro de Identidade, consulte Conceitos básicos do Centro de Identidade da AWS.

> **Important**
>
> O perfil do IAM para a funcionalidade que você fornece determina quais recursos da AWS o Argo CD pode acessar. Isso inclui o acesso a repositórios do Git por meio do CodeConnections e a segredos no Secrets Manager. Para obter orientações sobre como criar um perfil apropriado com permissões de privilégio mínimo, consulte the section called “Perfil do IAM para a funcionalidade” e the section called “Considerações para funcionalidades do EKS”.

Escolha da ferramenta

É possível criar uma funcionalidade do Argo CD por meio do Console de gerenciamento da AWS, da AWS CLI ou do eksctl:

- the section called “Console”: use o Console para obter uma experiência guiada
- the section called “ AWS CLI”: use a AWS CLI para obter scripts e automação
- the section called “eksctl”: use o eksctl para obter uma experiência nativa do Kubernetes

O que ocorre durante a criação de uma funcionalidade do Argo CD

Ao criar uma funcionalidade do Argo CD:

1. O EKS cria o serviço da funcionalidade do Argo CD e o configura para monitorar e gerenciar recursos no cluster
2. As definições de recursos personalizados (CRDs, na sigla em inglês) são instaladas no cluster
3. A funcionalidade assume o perfil do IAM para a funcionalidade, que foi fornecido por você
4. O Argo CD começa a monitorar seus recursos personalizados
5. O status da funcionalidade é alterado de `CREATING` para `ACTIVE`
6. A interface do usuário do Argo CD torna-se acessível por meio do endpoint

Uma vez ativa, você poderá criar Applications do Argo CD no cluster para realizar implantações usando os repositórios do Git.

Próximas etapas

Após criar a funcionalidade do Argo CD:

- the section called “Como trabalhar com o Argo CD”: configure o acesso ao repositório, registre os clusters de destino e crie Applications
- the section called “Conceitos do Argo CD”: aprenda sobre os princípios de GitOps, as políticas de sincronização e os padrões para vários clusters
- the section called “Considerações sobre o Argo CD”: conheça os padrões de arquitetura de vários clusters e a configuração avançada

Criação de uma funcionalidade do Argo CD por meio do Console

Este tópico descreve como criar uma funcionalidade do Argo CD usando o Console de gerenciamento da AWS.

Pré-requisitos

- Centro de Identidade da AWS configurado: o Argo CD requer o Centro de Identidade da AWS para autenticação. Não há suporte para usuários locais. Se você não tiver o Centro de Identidade da AWS configurado, consulte Conceitos básicos do Centro de Identidade da AWS para criar uma instância do Centro de Identidade, e Adicionar usuários e Adicionar grupos para criar usuários e grupos para acesso ao Argo CD.

Criação da funcionalidade do Argo CD

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do cluster para abrir a página de detalhes do cluster.
3. Escolha a guia Funcionalidades.
4. Na barra de navegação à esquerda, selecione Argo CD.
5. Escolha Criar funcionalidade do Argo CD.
6. No perfil da funcionalidade do IAM:
   - Se você já tiver um perfil da funcionalidade do IAM, selecione-o no menu suspenso
   - Se a criação de um perfil for necessária, escolha Criar perfil do Argo CD

     Isso abre o console do IAM em uma nova guia com a política de confiança preenchida previamente e fornece acesso total de leitura ao Secrets Manager. Nenhuma outra permissão é adicionada por padrão, mas você pode adicioná-las se necessário. Se você planeja usar repositórios do CodeCommit ou outros serviços da AWS, adicione as permissões adequadas antes de criar o perfil.

     Após criar o perfil, retorne ao console do EKS e o perfil será selecionado automaticamente.

     > **Note**
     >
     > Se você planeja usar as integrações opcionais com o AWS Secrets Manager ou o AWS CodeConnections, precisará adicionar permissões ao perfil. Para obter exemplos da política do IAM e de orientações de configuração, consulte the section called “ AWS Secrets Manager” e the section called “ AWS CodeConnections”.
7. Configure a integração com o Centro de Identidade da AWS:
   a. Selecione Habilitar integração com o Centro de Identidade da AWS.
   b. Escolha sua instância do Centro de Identidade no menu suspenso.
   c. Configure os mapeamentos de perfil para RBAC atribuindo usuários ou grupos aos perfis do Argo CD (ADMIN, EDITOR ou VIEWER).
8. Escolha Criar.

O processo de criação da funcionalidade é iniciado.

Verificação da ativação da funcionalidade

1. Na guia Funcionalidades, visualize o status da funcionalidade do Argo CD.
2. Aguarde até que o status mude de `CREATING` para `ACTIVE`.
3. Assim que estiver com o status ativo, a funcionalidade estará pronta para uso.

Para obter informações sobre os status das funcionalidades e sobre a solução de problemas, consulte the section called “Como trabalhar com funcionalidades”.

Acesso à interface do usuário do Argo CD

Depois que a funcionalidade estiver ativa, você poderá acessar a interface do usuário do Argo CD:

1. Na página da funcionalidade do Argo CD, escolha Abrir interface do usuário do Argo CD.
2. A interface do usuário do Argo CD será aberta em uma nova guia do navegador.
3. Em seguida, você pode criar Applications e gerenciar implantações por meio da interface do usuário.

Próximas etapas

- the section called “Como trabalhar com o Argo CD”: configure repositórios, registre clusters e crie Applications
- the section called “Considerações sobre o Argo CD”: acesse a arquitetura de vários clusters e a configuração avançada
- the section called “Como trabalhar com funcionalidades”: gerencie os recursos da funcionalidade do Argo CD

Criação de uma funcionalidade do Argo CD por meio da AWS CLI

Este tópico descreve como criar uma funcionalidade do Argo CD usando a AWS CLI.

Pré-requisitos

- AWS CLI: versão `2.12.3` ou em versões posteriores. Para verificar a versão, execute `aws --version`. Para obter mais informações, consulte Instalação no Guia do Usuário da Interface de Linha de Comando AWS.

- **`kubectl`** – uma ferramenta de linha de comando para trabalhar com clusters do Kubernetes. Para obter mais informações, consulte the section called “Configuração do kubectl e eksctl”.
- Centro de Identidade da AWS configurado: o Argo CD requer o Centro de Identidade da AWS para autenticação. Não há suporte para usuários locais. Se você não tiver o Centro de Identidade da AWS configurado, consulte Conceitos básicos do Centro de Identidade da AWS para criar uma instância do Centro de Identidade, e Adicionar usuários e Adicionar grupos para criar usuários e grupos para acesso ao Argo CD.

Etapa 1: criação de um perfil da funcionalidade do IAM

Crie um arquivo de política de confiança:

```
cat > argocd-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Crie o perfil do IAM:

```
aws iam create-role \
  --role-name ArgoCDCapabilityRole \
  --assume-role-policy-document file://argocd-trust-policy.json
```

> **Note**
>
> Se você planeja usar as integrações opcionais com o AWS Secrets Manager ou o AWS CodeConnections, precisará adicionar permissões ao perfil. Para obter exemplos da

política do IAM e de orientações de configuração, consulte the section called “ AWS Secrets Manager” e the section called “ AWS CodeConnections”.

Etapa 2: criação da funcionalidade do Argo CD

Crie o recurso da funcionalidade do Argo CD no cluster.

Primeiro, defina as variáveis de ambiente para a sua configuração do Centro de Identidade:

```
# Get your Identity Center instance ARN (replace region if your IDC instance is in a
 different region)
export IDC_INSTANCE_ARN=$(aws sso-admin list-instances --region [.replaceable]`region`
 --query 'Instances[0].InstanceArn' --output text)

# Get a user ID for RBAC mapping (replace with your username and region if needed)
export IDC_USER_ID=$(aws identitystore list-users \
  --region [.replaceable]`region` \
  --identity-store-id $(aws sso-admin list-instances --region [.replaceable]`region` --
query 'Instances[0].IdentityStoreId' --output text) \
  --query 'Users[?UserName==`your-username`].UserId' --output text)

echo "IDC_INSTANCE_ARN=$IDC_INSTANCE_ARN"
echo "IDC_USER_ID=$IDC_USER_ID"
```

Crie a funcionalidade com integração ao Centro de Identidade. Substitua *region-code* pela região da AWS em que seu cluster está localizado e *my-cluster* pelo nome do seu cluster:

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --type ARGOCD \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/ArgoCDCapabilityRole \
  --delete-propagation-policy RETAIN \
  --configuration '{
    "argoCd": {
      "awsIdc": {
        "idcInstanceArn": "'$IDC_INSTANCE_ARN'",
        "idcRegion": "'[.replaceable]`region-code`'"
      },
```

```
      "rbacRoleMappings": [{
        "role": "ADMIN",
        "identities": [{
          "id": "'$IDC_USER_ID'",
          "type": "SSO_USER"
        }]
      }]
    }
  }'
```

O comando é concluído de imediato, mas a funcionalidade demora algum tempo para se tornar ativa, conforme o EKS cria a infraestrutura e os componentes necessários para a funcionalidade. O EKS instalará as definições de recursos personalizados do Kubernetes relacionadas a essa funcionalidade no cluster durante o processo de criação.

> **Note**
>
> Caso ocorra um erro indicando a inexistência do cluster ou falta de permissões, verifique o seguinte:
>
> - Se o nome do cluster está correto
> - Se a AWS CLI está configurada para a região correta
> - Se você tem as permissões do IAM obrigatórias

Etapa 3: verificação da ativação da funcionalidade

Aguarde até que a funcionalidade se torne ativa. Substitua *region-code* pela região da AWS em que seu cluster está localizado e *my-cluster* pelo nome do seu cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --query 'capability.status' \
  --output text
```

A funcionalidade estará pronta assim que o status mostrar `ACTIVE`. Não prossiga para a próxima etapa até que o status seja `ACTIVE`.

Também é possível visualizar os detalhes completos da funcionalidade:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd
```

Etapa 4: verificação da disponibilidade de recursos personalizados

Após a funcionalidade estar ativa, verifique se os recursos personalizados do Argo CD estão disponíveis no cluster:

```
kubectl api-resources | grep argoproj.io
```

Os tipos de recursos `Application` e `ApplicationSet` devem aparecer na lista apresentada.

Próximas etapas

- the section called “Como trabalhar com o Argo CD”: configure repositórios, registre clusters e crie Applications
- the section called “Considerações sobre o Argo CD”: acesse a arquitetura de vários clusters e a configuração avançada
- the section called “Como trabalhar com funcionalidades”: gerencie os recursos da funcionalidade do Argo CD

Criação de uma funcionalidade do Argo CD por meio do eksctl

Este tópico descreve como criar uma funcionalidade do Argo CD usando o eksctl.

> **Note**
>
> Para realizar as etapas seguintes, você deve estar utilizando o eksctl na versão `0.220.0` ou em versões posteriores. Para verificar a versão, execute `eksctl version`.

Etapa 1: criação de um perfil da funcionalidade do IAM

Crie um arquivo de política de confiança:

```
cat > argocd-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Crie o perfil do IAM:

```
aws iam create-role \
  --role-name ArgoCDCapabilityRole \
  --assume-role-policy-document file://argocd-trust-policy.json
```

> **Note**
>
> Para esta configuração básica, não são necessárias políticas do IAM adicionais. Se você planeja usar o Secrets Manager para credenciais de repositório ou para o CodeConnections, precisará adicionar permissões ao perfil. Para obter exemplos da política do IAM e de orientações de configuração, consulte the section called “ AWS Secrets Manager” e the section called “ AWS CodeConnections”.

Etapa 2: obtenção da configuração do Centro de Identidade da AWS

Obtenha o ARN da instância do Centro de Identidade e o ID do usuário para a configuração de RBAC:

```
# Get your Identity Center instance ARN
aws sso-admin list-instances --query 'Instances[0].InstanceArn' --output text
```

```
# Get your Identity Center region
aws sso-admin list-instances --query 'Instances[0].IdentityStoreId' --output text | cut
 -d'/' -f1

# Get a user ID for admin access (replace 'your-username' with your Identity Center
 username)
aws identitystore list-users \
  --identity-store-id $(aws sso-admin list-instances --query
 'Instances[0].IdentityStoreId' --output text) \
  --query 'Users[?UserName==`your-username`].UserId' --output text
```

Anote esses valores. Você precisará deles na próxima etapa.

Etapa 3: criação de um arquivo de configuração do eksctl

Crie um arquivo chamado `argocd-capability.yaml` com o conteúdo a seguir. Substitua os valores com espaços reservados pelo nome do cluster, região, ARN do perfil do IAM, ARN da instância do Centro de Identidade, região do Centro de Identidade e ID do usuário:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

capabilities:
  - name: my-argocd
    type: ARGOCD
    roleArn: arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole
    configuration:
      argocd:
        awsIdc:
          idcInstanceArn: arn:aws:sso:::instance/ssoins-123abc
          idcRegion: idc-region-code
        rbacRoleMappings:
          - role: ADMIN
            identities:
              - id: 38414300-1041-708a-01af-5422d6091e34
                type: SSO_USER
```

> **Note**
>
> É possível adicionar vários usuários ou grupos aos mapeamentos de RBAC. Para grupos, use `type: SSO_GROUP` e forneça o ID do grupo. Os perfis disponíveis são `ADMIN`, `EDITOR` e `VIEWER`.

Etapa 4: criação da funcionalidade do Argo CD

Aplique o arquivo de configuração:

```
eksctl create capability -f argocd-capability.yaml
```

O comando é executado de forma imediata, mas a funcionalidade demora algum tempo para se tornar ativa.

Etapa 5: verificação da ativação da funcionalidade

Verifique o status da funcionalidade. Substitua *`region-code`* pela região AWS em que seu cluster se encontra e substitua *`my-cluster`* pelo nome do seu cluster.

```
eksctl get capability \
  --region region-code \
  --cluster my-cluster \
  --name my-argocd
```

A funcionalidade estará pronta assim que o status mostrar `ACTIVE`.

Etapa 6: verificação da disponibilidade de recursos personalizados

Após a funcionalidade estar ativa, verifique se os recursos personalizados do Argo CD estão disponíveis no cluster:

```
kubectl api-resources | grep argoproj.io
```

Os tipos de recursos `Application` e `ApplicationSet` devem aparecer na lista apresentada.

Próximas etapas

- the section called “Como trabalhar com o Argo CD”: aprenda a criar e gerenciar Applications do Argo CD

- the section called “Considerações sobre o Argo CD”: configure o SSO e o acesso a vários clusters
- the section called “Como trabalhar com funcionalidades”: gerencie os recursos da funcionalidade do Argo CD

Conceitos do Argo CD

O Argo CD implementa o GitOps ao tratar o Git como a única fonte de verdade para as implantações da aplicação. Este tópico apresenta um exemplo prático e, em seguida, explica os conceitos fundamentais que você precisa entender ao trabalhar com a funcionalidade do EKS para o Argo CD.

Conceitos básicos do Argo CD

Após criar a funcionalidade do Argo CD (consulte the section called “Criar o recurso Argo CD”), é possível começar a implantar as aplicações. Este exemplo demonstra o registro de um cluster e a criação de uma Application.

Etapa 1: Configurar

Registrar o cluster (obrigatório)

Registre o cluster no local em que você deseja implantar aplicações. Para este exemplo, registraremos o mesmo cluster em que o Argo CD está sendo executado (você pode usar o nome `in-cluster` para obter compatibilidade com a maioria dos exemplos do Argo CD):

```
# Get your cluster ARN
CLUSTER_ARN=$(aws eks describe-cluster \
  --name my-cluster \
  --query 'cluster.arn' \
  --output text)

# Register the cluster using Argo CD CLI
argocd cluster add $CLUSTER_ARN \
  --aws-cluster-name $CLUSTER_ARN \
  --name in-cluster \
  --project default
```

> **Note**
>
> Para obter informações sobre a configuração da CLI do Argo CD para trabalho com a funcionalidade do Argo CD no EKS, consulte the section called “Uso da CLI do Argo CD com a funcionalidade gerenciada”.

Como alternativa, registre o cluster usando um Kubernetes Secret (consulte the section called “Registro de clusters” para obter mais detalhes).

Configurar o acesso ao repositório (opcional)

Este exemplo utiliza um repositório público do GitHub, portanto, nenhuma configuração de repositório é necessária. Para repositórios privados, configure o acesso usando o AWS Secrets Manager, o CodeConnections ou o Kubernetes Secrets (consulte the section called “Configuração dos repositórios” para obter mais detalhes).

No caso de serviços da AWS (como ECR para charts do Helm, CodeConnections e CodeCommit), é possível referenciá-los diretamente nos recursos da Application sem precisar criar um Repository. O perfil da funcionalidade deve ter as permissões do IAM necessárias. Para mais detalhes, consulte the section called “Configuração dos repositórios”.

Etapa 2: criar um aplicativo

Crie este manifesto da Application em `my-app.yaml`:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    name: in-cluster
    namespace: guestbook
  syncPolicy:
```

```
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - CreateNamespace=true
```

Aplique a Application:

```
kubectl apply -f my-app.yaml
```

Após aplicar esta Application, o Argo CD: 1. Sincroniza a aplicação do Git com o cluster (implantação inicial) 2. Monitora o repositório do Git em busca de alterações 3. Sincroniza automaticamente as alterações posteriores com o seu cluster 4. Detecta e corrige qualquer desvio em relação ao estado desejado 5. Fornece o status da integridade e o histórico de sincronização na interface do usuário

Visualize o status da aplicação:

```
kubectl get application guestbook -n argocd
```

Você também pode visualizar a aplicação usando a CLI do Argo CD (`argocd app get guestbook`) ou a interface do usuário do Argo CD (que é acessível do console do EKS, na guia Funcionalidades do seu cluster).

> **Note**
>
> Use o nome do cluster em `destination.name` (o nome que você utilizou ao registrar o cluster). A funcionalidade gerenciada não oferece suporte ao padrão local “in-cluster” (`kubernetes.default.svc`).

Conceitos principais

Princípios de GitOps e tipos de origem

O Argo CD implementa o GitOps, no qual a origem da sua aplicação é a única fonte de verdade para as implantações:

- Declarativo: o estado desejado é declarado usando manifestos no formato YAML, charts do Helm ou sobreposições do Kustomize
- Versionado: cada alteração é rastreada com uma trilha de auditoria completa

- Automatizado o Argo CD monitora continuamente as origens e sincroniza as alterações automaticamente
- Autorreparação: detecta e corrige desvios entre o estado desejado e o estado atual do cluster

Tipos de origem com suporte:

- Repositórios do Git: GitHub, GitLab, Bitbucket, CodeCommit (HTTPS, SSH ou CodeConnections)
- Registros do Helm: registros HTTP (como `https://aws.github.io/eks-charts`) e registros OCI (como `public.ecr.aws`)
- Imagens OCI: imagens de contêiner contendo manifestos ou charts do Helm (como `oci://registry-1.docker.io/user/my-app`)

Essa flexibilidade permite que as organizações escolham origens que atendam aos seus requisitos de segurança e conformidade. Por exemplo, organizações que restringem o acesso ao Git proveniente de clusters podem usar o ECR para charts do Helm ou imagens OCI.

Para obter mais informações, consulte Application Sources na documentação do Argo CD.

Sincronização e reconciliação

O Argo CD monitora continuamente as origens e os clusters para detectar e corrigir diferenças:

1. Realiza a verificação de alterações nas origens (padrão: a cada seis minutos)
2. Compara o estado desejado com o estado do cluster
3. Marca as aplicações como `Synced` ou `OutOfSync`
4. Sincroniza as alterações automaticamente (se configurado) ou aguarda aprovação manual
5. Monitora a integridade do recurso após a sincronização

Os ciclos de sincronização controlam a ordem de criação de recursos usando anotações:

```
metadata:
  annotations:
    argocd.argoproj.io/sync-wave: "0"  # Default if not specified
```

Os recursos são aplicados na ordem do ciclo (com números menores primeiro, incluindo números negativos como `-1`). Isso permite a criação de dependências, como namespaces (ciclo `-1`) antes das implantações (ciclo `0`).

A autorreparação reverte automaticamente as alterações manuais:

```
spec:
  syncPolicy:
    automated:
      selfHeal: true
```

> **Note**
>
> A funcionalidade gerenciada usa o rastreamento de recursos baseado em anotações (em vez de empregar o rastreamento baseado em rótulos) para obter melhor compatibilidade com as convenções do Kubernetes e com as outras ferramentas.

Para obter informações detalhadas sobre as fases de sincronização, os hooks e os padrões avançados, consulte a documentação de sincronização do Argo CD.

Integridade da aplicação

O Argo CD monitora a integridade de todos os recursos da aplicação:

Status de integridade: * Íntegro: todos os recursos estão sendo executados conforme o esperado * Em progresso: os recursos estão sendo criados ou atualizados * Degradado: alguns recursos não estão íntegros (por exemplo, pods falhando e trabalhos com erros) * Suspenso: a aplicação está pausada intencionalmente * Ausente: os recursos definidos no Git não estão presentes no cluster

O Argo CD conta com verificações de integridade integradas para recursos comuns do Kubernetes (por exemplo, Deployments, StatefulSets, Jobs e outros) e oferece suporte a verificações de integridade personalizadas para CRDs.

A saúde da aplicação é determinada por todos os seus recursos. Dessa forma, se qualquer recurso estiver no estado `Degraded`, a aplicação será considerada `Degraded`.

Para obter mais informações, consulte Resource Health na documentação do Argo CD.

Padrões de diversos clusters

O Argo CD é compatível com dois padrões principais de implantação:

Hub-and-spoke: execute o Argo CD em um cluster de gerenciamento dedicado que realiza a implantação para diversos clusters de workloads: * Controle e visibilidade centralizados * Políticas

consistentes em todos os clusters * Apenas uma instância do Argo CD para gerenciamento * Separação clara entre o ambiente de gerenciamento e as workloads

Por cluster: execute o Argo CD em cada cluster, gerenciando apenas as aplicações daquele cluster específico: * Isolamento de clusters (portanto, uma falha não afeta os demais) * Rede simplificada (sem necessidade de comunicação entre clusters) * Configuração inicial simplificada (sem necessidade de registro de clusters)

Escolha o modelo hub-and-spoke para equipes de plataformas que gerenciam muitos clusters ou o modelo por cluster para equipes independentes ou quando os clusters precisarem de isolamento total.

Para obter configurações detalhadas de diversos clusters, consulte the section called “Considerações sobre o Argo CD”.

Projetos

Os Projects fornecem agrupamento lógico e controle de acesso para as Applications:

- Restrições de origem: limitam quais repositórios do Git podem ser utilizados
- Restrições de destino: limitam quais clusters e namespaces podem ser destinos de implantação
- Restrições de recursos: limitam quais tipos de recursos do Kubernetes podem ser implantados
- Integração com o RBAC: mapeia projetos para IDs de usuários e de grupos do Centro de Identidade da AWS

As Applications pertencem a um único projeto. Se não for especificado, elas utilizarão o projeto `default`, que não conta com restrições por padrão. Em ambientes de produção, estabeleça projetos com as restrições adequadas.

Para obter as configurações de projetos e os padrões de RBAC, consulte the section called “Configurar permissões”.

Organização de repositórios

A maioria das equipes usa a organização baseada em diretórios com sobreposições do Kustomize ou arquivos de valores do Helm para diferentes ambientes:

```
my-app/
```

```
### base/
#   ### deployment.yaml
#   ### service.yaml
### overlays/
    ### dev/
    #   ### kustomization.yaml
    ### staging/
    #   ### kustomization.yaml
    ### prod/
        ### kustomization.yaml
```

Essa abordagem oferece flexibilidade e clareza, mantendo todas as configurações de ambiente em um único repositório.

Para obter padrões detalhados da estrutura de repositórios e das práticas recomendadas, consulte a documentação de práticas recomendadas do Argo CD.

Opções de sincronização

Realize um ajuste fino do comportamento de sincronização com estas opções conhecidas:

- `CreateNamespace=true`: cria automaticamente o namespace de destino
- `ServerSideApply=true`: usa o Server-Side Apply para obter uma melhor resolução de conflitos
- `SkipDryRunOnMissingResource=true`: ignora a simulação quando as CRDs ainda não existem (sendo útil para instâncias do kro)

```
spec:
  syncPolicy:
    syncOptions:
    - CreateNamespace=true
    - ServerSideApply=true
```

Para obter uma lista completa de opções de sincronização, consulte a documentação de opções de sincronização do Argo CD.

Próximas etapas

- the section called “Configuração dos repositórios”: configure o acesso ao repositório do Git
- the section called “Registro de clusters”: registre os clusters de destino para a implantação

- the section called “Criação de Applications”: crie sua primeira Application
- the section called “Considerações sobre o Argo CD”: acesse os padrões específicos para EKS, a integração com o Centro de Identidade e a configuração de diversos clusters
- Documentação do Argo CD: acesse a documentação abrangente do Argo CD, incluindo hooks de sincronização, verificações de integridade e padrões avançados

Configuração das permissões do Argo CD

A funcionalidade gerenciada do Argo CD se integra ao Centro de Identidade da AWS para autenticação e usa perfis de RBAC integrados para autorização. Este tópico explica como configurar permissões para usuários e equipes.

Funcionamento de permissões com o Argo CD

A funcionalidade do Argo CD usa o Centro de Identidade da AWS para autenticação e fornece três perfis de RBAC integrados para autorização.

Quando um usuário acessa o Argo CD:

1. A autenticação é realizada por meio do Centro de Identidade da AWS (que pode ser federado ao seu provedor de identidades corporativo)
2. O Centro de Identidade da AWS fornece informações de usuários e grupos ao Argo CD
3. O Argo CD mapeia usuários e grupos para perfis de RBAC com base na sua configuração
4. Os usuários visualizam somente as aplicações e os recursos aos quais têm permissão de acessar

Perfis de RBAC integrados

A funcionalidade do Argo CD fornece três perfis integrados que você mapeia para usuários e grupos do Centro de Identidade da AWS.

ADMIN

Acesso total a todas as aplicações e configurações:

- Criação, atualização e exclusão de Applications e ApplicationSets
- Gerenciamento da configuração do Argo CD
- Registro e gerenciamento de clusters de destino para implantação

- Configuração do acesso ao repositório
- Gerenciamento de projetos
- Visualização do status e do histórico de todas as aplicações

EDITOR

Capacidade de criação e modificação de aplicações, sem permissão para alteração das configurações do Argo CD:

- Criação e atualização de Applications e ApplicationSets
- Sincronização e atualização de aplicações
- Visualização do status e do histórico da aplicação
- Sem permissão para a exclusão de aplicações
- Sem permissão para a alteração da configuração do Argo CD
- Sem permissão para o gerenciamento de clusters ou de repositórios

VIEWER

Acesso somente leitura às aplicações:

- Visualização do status e do histórico da aplicação
- Visualização de manifestos e de recursos da aplicação
- Sem permissão para a realização de quaisquer alterações
- Sem permissão para a sincronização e atualização de aplicações

Configuração de mapeamentos de perfil

Mapeie usuários e grupos do Centro de Identidade da AWS para os perfis do Argo CD durante a criação ou a atualização da funcionalidade.

Exemplo de mapeamento de perfil:

```
{
  "rbacRoleMapping": {
    "ADMIN": ["AdminGroup", "alice@example.com"],
    "EDITOR": ["DeveloperGroup", "DevOpsTeam"],
```

```
    "VIEWER": ["ReadOnlyGroup", "bob@example.com"]
  }
}
```

> **Note**
>
> Os nomes dos perfis diferenciam maiúsculas de minúsculas e devem estar em letras maiúsculas (ADMIN, EDITOR e VIEWER).

> **Important**
>
> A integração das funcionalidades do EKS com o Centro de Identidade da AWS fornece suporte para até mil identidades por funcionalidade do Argo CD. Uma identidade pode ser um usuário ou um grupo.

Atualize os mapeamentos de perfil:

```
aws eksfe update-capability \
  --region us-east-1 \
  --cluster-name cluster \
  --capability-name capname \
  --endpoint "https://eks.ap-northeast-2.amazonaws.com" \
  --role-arn "arn:aws:iam::[.replaceable]111122223333:role/
[.replaceable]`EKSCapabilityRole`" \
  --configuration '{
    "argoCd": {
      "rbacRoleMappings": {
        "addOrUpdateRoleMappings": [
          {
            "role": "ADMIN",
            "identities": [
              { "id": "686103e0-f051-7068-b225-e6392b959d9e", "type": "SSO_USER" }
            ]
          }
        ]
      }
    }
  }'
```

Uso da conta de administrador

A conta de administrador é destinada à configuração inicial e às tarefas administrativas, como o registro de clusters e a configuração de repositórios.

Situações apropriadas para o uso da conta de administrador:

- Configuração inicial da funcionalidade
- Desenvolvimento individual ou demonstrações rápidas
- Tarefas administrativas (como registro de cluster, configuração de repositórios e criação de projetos)

Práticas recomendadas para a conta de administrador:

- Sem permissão para a confirmação de tokens da conta para o controle de versão
- Rotação imediata de tokens em caso de exposição
- Limitação do uso de tokens da conta para tarefas de configuração e de administração
- Definição de prazos curtos de expiração (máximo de 12 horas)
- Limite de criação de apenas cinco tokens simultâneos na conta

Situações para uso do acesso baseado em projetos::

- Ambientes de desenvolvimento compartilhados com vários usuários
- Qualquer ambiente com características do ambiente de produção
- Quando há necessidade de trilhas de auditoria para identificação de ações por parte dos usuários
- Quando há necessidade da aplicação de restrições de recursos ou limites de acesso

Para ambientes de produção e cenários com vários usuários, use o controle de acesso baseado em projetos com perfis de RBAC dedicados e mapeados para grupos do Centro de Identidade da AWS.

Controle de acesso baseado em projetos

Use o Argo CD Projects (AppProject) para a disponibilização de controle de acesso detalhado e isolamento de recursos para as equipes.

Os projetos fornecem:

- Restrições de origem: limitam quais repositórios do Git podem ser utilizados
- Restrições de destino: limitam quais clusters e namespaces podem ser destinos de implantação
- Restrições de recursos: limitam quais tipos de recursos do Kubernetes podem ser implantados
- Integração de RBAC: mapeamento de projetos para grupos do Centro de Identidade da AWS ou perfis do Argo CD

Exemplo de projeto para isolamento de equipes:

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
  namespace: argocd
spec:
  description: Team A applications

  # Source restrictions
  sourceRepos:
  - https://github.com/myorg/team-a-apps

  # Destination restrictions
  destinations:
  - namespace: team-a-*
    server: arn:aws:eks:us-west-2:111122223333:cluster/production

  # Resource restrictions
  clusterResourceWhitelist:
  - group: ''
    kind: Namespace
  namespaceResourceWhitelist:
  - group: 'apps'
    kind: Deployment
  - group: ''
    kind: Service
  - group: ''
    kind: ConfigMap
```

Atribua usuários a projetos:

É possível restringir usuários com perfis de EDITOR ou VIEWER a projetos específicos. Usuários com perfil de ADMIN têm acesso a todos os projetos.

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
spec:
  # ... project configuration ...

  # Map Identity Center groups to project roles
  roles:
  - name: developer
    description: Team A developers
    policies:
    - p, proj:team-a:developer, applications, *, team-a/*, allow
    groups:
    - TeamADevelopers

  - name: viewer
    description: Team A viewers
    policies:
    - p, proj:team-a:viewer, applications, get, team-a/*, allow
    groups:
    - TeamAViewers
```

Padrões de permissão comuns

Padrão 1: equipe administrativa com acesso total

```
{
  "rbacRoleMapping": {
    "ADMIN": ["PlatformTeam", "SRETeam"]
  }
}
```

Padrão 2: desenvolvedores com capacidade de realizar implantações e outros com capacidade de visualização

```
{
  "rbacRoleMapping": {
    "ADMIN": ["PlatformTeam"],
    "EDITOR": ["DevelopmentTeam", "DevOpsTeam"],
    "VIEWER": ["AllEmployees"]
  }
```

```
}
```

Padrão 3: isolamento baseado em equipe com projetos

1. Mapeie todos os desenvolvedores para o perfil de EDITOR
2. Crie AppProjects separados para cada equipe
3. Use perfis de projeto para restringir o acesso a aplicações específicas de cada equipe

```
{
  "rbacRoleMapping": {
    "ADMIN": ["PlatformTeam"],
    "EDITOR": ["AllDevelopers"]
  }
}
```

Em seguida, crie projetos com restrições exclusivas para cada equipe e seus respectivos mapeamentos de perfis.

Práticas recomendadas

Use grupos em vez de usuários individuais: mapeie grupos do Centro de Identidade da AWS para perfis do Argo CD, em vez de usuários individuais, para facilitar o gerenciamento.

Comece a aplicar permissões com o privilégio mínimo: inicie com acesso com o perfil de VIEWER e conceda acesso de EDITOR ou de ADMIN, conforme necessário.

Use projetos para o isolamento de equipes: crie AppProjects separados para diferentes equipes ou ambientes a fim de aplicar limites.

Aproveite a federação do Centro de Identidade: configure o Centro de Identidade da AWS para federar com seu provedor de identidades corporativo para gerenciamento centralizado de usuários

Análises de acesso regulares: revise periodicamente os mapeamentos de perfis e as atribuições de projetos para garantir níveis de acesso adequados.

Limite o acesso ao cluster: lembre-se de que o RBAC do Argo CD controla o acesso a recursos e operações do Argo CD, mas não corresponde ao RBAC do Kubernetes. Usuários com acesso ao Argo CD podem implantar aplicações em clusters aos quais o Argo CD tem acesso. Limite quais clusters o Argo CD pode acessar e use restrições de destino de projeto para controlar os locais em que as aplicações podem ser implantadas.

AWSPermissões de serviço do

Para usar serviços da AWS diretamente nos recursos da Application (sem a necessidade de criar recursos de Repository), vincule as permissões do IAM obrigatórias ao perfil da funcionalidade.

ECR para charts do Helm:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
      "Resource": "*"
    }
  ]
}
```

Repositórios do CodeCommit:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "arn:aws:codecommit:region:account-id:repository-name"
    }
  ]
}
```

CodeConnections (GitHub, GitLab e Bitbucket):

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection"
      ],
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-id"
    }
  ]
}
```

Consulte the section called “Configuração dos repositórios” para obter detalhes sobre como usar essas integrações.

Próximas etapas

- the section called “Como trabalhar com o Argo CD”: saiba como criar aplicações e gerenciar implantações
- the section called “Conceitos do Argo CD”: compreenda os conceitos do Argo CD, incluindo o Projects
- the section called “Considerações para funcionalidades do EKS”: analise as práticas recomendadas de segurança para as funcionalidades

Como trabalhar com o Argo CD

Ao usar o Argo CD, você define as aplicações em repositórios do Git e a funcionalidade se encarrega de sincronizá-las automaticamente nos clusters do Kubernetes. Isso permite a implantação de aplicações de forma declarativa e com controle de versão, contando com a detecção automática de desvios.

Pré-requisitos

Antes de começar a trabalhar o Argo CD, é necessário ter:

- Um cluster do EKS com a funcionalidade para o Argo CD criada (consulte the section called “Criar o recurso Argo CD”)
- Um repositório do Git que contém manifestos do Kubernetes
- `kubectl` configurado para o estabelecimento de comunicação com o cluster

Tarefas comuns

Os seguintes tópicos orientam você na realização de tarefas comuns do Argo CD:

the section called “Configuração dos repositórios”: configure o Argo CD para acessar seus repositórios do Git usando o AWS Secrets Manager, o AWS CodeConnections ou Kubernetes Secrets.

the section called “Registro de clusters”: registre os clusters de destino nos quais o Argo CD implantará aplicações.

the section called “Projetos”: organize aplicações e aplique limites de segurança usando Projects para ambientes multilocatários.

the section called “Criação de Applications”: crie Applications que realizam a implantação de repositórios do Git com políticas de sincronização automática ou manual.

the section called “ApplicationSets”: use ApplicationSets para implantar aplicações em múltiplos ambientes ou clusters usando modelos e geradores.

Acesso à interface do usuário do Argo CD

Acesse a interface do usuário do Argo CD por meio do console do EKS:

1. Abra o console do Amazon EKS
2. Selecione o cluster
3. Escolha a guia Funcionalidades
4. Escolha Argo CD
5. Escolha Abrir interface do usuário do Argo CD

A interface de usuário fornece a topologia visual da aplicação, o status e o histórico de sincronização, a integridade dos recursos e os eventos, os controles de sincronização manual e o gerenciamento de aplicações.

Documentação original

Para obter informações detalhadas sobre as funcionalidades do Argo CD:

- Documentação do Argo CD: acesse o guia do usuário completo
- Application Spec: acesse a referência de API completa para a Application

- ApplicationSet Guide: acesse padrões e exemplos de ApplicationSet
- Argo CD GitHub: acesse o código-fonte e exemplos

Configuração do acesso ao repositório

Antes de implantar aplicações, configure o Argo CD para acessar os repositórios do Git e os registros do chart do Helm. O Argo CD oferece suporte a diversos métodos de autenticação para GitHub, GitLab, Bitbucket, AWS CodeCommit e AWS ECR.

> **Note**
>
> No caso de integrações diretas com serviços da AWS (como charts do Helm do ECR, repositórios do CodeCommit e CodeConnections), é possível referenciá-los diretamente nos recursos da Application sem a necessidade de criar configurações de Repository. O perfil da funcionalidade deve ter as permissões do IAM necessárias. Para mais detalhes, consulte the section called “Configurar permissões”.

Pré-requisitos

- Um cluster do EKS com a funcionalidade para o Argo CD criada
- Repositórios do Git que contêm manifestos do Kubernetes
- `kubectl` configurado para o estabelecimento de comunicação com o cluster

> **Note**
>
> Para reutilizar credenciais em vários repositórios, você pode usar modelos de credenciais de repositório (repocreds). Para obter mais informações, consulte Private Repositories na documentação do Argo CD.

Métodos de autenticação

Método	Caso de uso	Permissões do IAM necessárias
Integração direta com serviços da AWS		

Método	Caso de uso	Permissões do IAM necessárias
CodeCommit	Integração direta com repositór ios do Git do AWS CodeCommit. Nenhuma configuração de Repositor y é necessária.	`codecommit:GitPull`
CodeConne ctions	Estabeleça conexão com o GitHub, o GitLab ou o Bitbucket usando autenticação gerenciada. Requer configuração da conexão.	`codeconnections:Us eConnection`
Charts do Helm do ECR	Integração direta com o AWS ECR para charts do Helm baseados em OCI. Nenhuma configuração de Repository é necessária.	`ecr:GetAuthorizati onToken` , `ecr:BatchGetImage` , `ecr:GetDownloadUrl ForLayer`
Configuração do repositório com credenciais		
AWS Secrets Manager (nome do usuário/t oken)	Armazene tokens de acesso pessoal ou senhas	`secretsmanager:Get SecretValue`
AWS Secrets Manager (chave SSH)	Use a autenticação por chave SSH	`secretsmanager:Get SecretValue`
AWS Secrets Manager (aplicati vo do GitHub)	Autenticação pelo aplicativo do GitHub com chave privada	`secretsmanager:Get SecretValue`
Kubernetes Secret	Método padrão do Argo CD usando segredos “in-cluster”	Nenhuma (somente política de confiança)

Acesso direto aos serviços da AWS

No caso de serviços da AWS, é possível referenciá-los diretamente nos recursos da Application sem precisar criar configurações de Repository. O perfil da funcionalidade deve ter as permissões do IAM necessárias.

Repositórios do CodeCommit

Referencie repositórios do CodeCommit diretamente nas Applications:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  source:
    repoURL: https://git-codecommit.region.amazonaws.com/v1/repos/repository-name
    targetRevision: main
    path: kubernetes/manifests
```

Permissões obrigatórias do perfil da funcionalidade:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codecommit:GitPull",
      "Resource": "arn:aws:codecommit:region:account-id:repository-name"
    }
  ]
}
```

CodeConnections

Referencie repositórios do GitHub, do GitLab ou do Bitbucket por meio do CodeConnections. O formato do URL do repositório é derivado do ARN da conexão do CodeConnections.

O formato do URL do repositório é:

```
apiVersion: argoproj.io/v1alpha1
```

```
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  source:
    repoURL: https://codeconnections.region.amazonaws.com/git-http/account-
id/region/connection-id/owner/repository.git
    targetRevision: main
    path: kubernetes/manifests
```

Permissões obrigatórias do perfil da funcionalidade:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codeconnections:UseConnection",
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-id"
    }
  ]
}
```

Charts do Helm do ECR

O ECR armazena charts do Helm como artefatos OCI. O Argo CD oferece duas maneiras de referenciá-los:

Formato Helm (recomendado para charts do Helm):

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-helm
  namespace: argocd
spec:
  source:
    repoURL: account-id.dkr.ecr.region.amazonaws.com/repository-name
    targetRevision: chart-version
    chart: chart-name
    helm:
```

```
      valueFiles:
        - values.yaml
```

Observação: não inclua o prefixo `oci://` ao usar o formato Helm. Use o campo `chart` para especificar o nome do chart.

Formato OCI (para artefatos OCI que contêm manifestos do Kubernetes):

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-oci
  namespace: argocd
spec:
  source:
    repoURL: oci://account-id.dkr.ecr.region.amazonaws.com/repository-name
    targetRevision: artifact-version
    path: path-to-manifests
```

Observação: inclua o prefixo `oci://` ao usar o formato OCI. Utilize o campo `path` em vez de `chart`.

Permissões obrigatórias do perfil da funcionalidade:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Resource": "*"
    }
  ]
}
```

Uso do AWS Secrets Manager

Armazene credenciais de repositório no Secrets Manager e faça referência a elas nas configurações de Repository do Argo CD.

Autenticação por nome de usuário e token

Para repositórios HTTPS com tokens de acesso pessoal ou senhas:

Criar o segredo no Secrets Manager:

```
aws secretsmanager create-secret \
  --name argocd/my-repo \
  --description "GitHub credentials for Argo CD" \
  --secret-string '{"username":"your-username","token":"your-personal-access-token"}'
```

Campos opcionais para certificado TLS de cliente (para servidores do Git privados):

```
aws secretsmanager create-secret \
  --name argocd/my-private-repo \
  --secret-string '{
    "username":"your-username",
    "token":"your-token",

 "tlsClientCertData":"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCi4uLgotLS0tLUVORCBDRVJUSUZJQ0FURS0tL

 "tlsClientCertKey":"LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCi4uLgotLS0tLUVORCBQUklWQVRFIEtFWS0tLS
  }'
```

> **Note**
>
> Os valores `tlsClientCertData` e `tlsClientCertKey` devem estar codificados em base64.

Criar um segredo de repositório referenciando o Secrets Manager:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo
  namespace: argocd
```

```
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/your-org/your-repo
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/my-repo-AbCdEf
  project: default
```

Autenticação por chave SSH

Para obter acesso ao Git baseado em SSH, armazene a chave privada em texto não formatado (não em JSON):

Criar o segredo com a chave privada SSH:

```
aws secretsmanager create-secret \
  --name argocd/my-repo-ssh \
  --description "SSH key for Argo CD" \
  --secret-string "-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
...
-----END OPENSSH PRIVATE KEY-----"
```

Criar um segredo de repositório para SSH:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo-ssh
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: git@github.com:your-org/your-repo.git
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/my-repo-ssh-
AbCdEf
  project: default
```

Autenticação por aplicativo do GitHub

Para a autenticação pelo aplicativo do GitHub com uma chave privada:

Criar o segredo com as credenciais do aplicativo do GitHub:

```
aws secretsmanager create-secret \
  --name argocd/github-app \
  --description "GitHub App credentials for Argo CD" \
  --secret-string '{

 "githubAppPrivateKeySecret":"LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQouLi4KLS0tLS1FTkQgUlNBI
    "githubAppID":"123456",
    "githubAppInstallationID":"12345678"
  }'
```

> **Note**
>
> O valor `githubAppPrivateKeySecret` deve estar codificado em base64.

Campo opcional para o GitHub Enterprise:

```
aws secretsmanager create-secret \
  --name argocd/github-enterprise-app \
  --secret-string '{

 "githubAppPrivateKeySecret":"LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQouLi4KLS0tLS1FTkQgUlNBI
    "githubAppID":"123456",
    "githubAppInstallationID":"12345678",
    "githubAppEnterpriseBaseUrl":"https://github.example.com/api/v3"
  }'
```

Criar um segredo de repositório para o aplicativo do GitHub:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo-github-app
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/your-org/your-repo
```

```
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/github-app-
AbCdEf
  project: default
```

> **Important**
>
> Certifique-se de que o perfil da funcionalidade do IAM tenha permissões `secretsmanager:GetSecretValue` para os segredos que você criar. Consulte the section called “Considerações sobre o Argo CD” para obter a configuração da política do IAM.

Uso do AWS CodeConnections

Para a integração com o CodeConnections, consulte the section called “ AWS CodeConnections”.

O CodeConnections fornece autenticação gerenciada para GitHub, GitLab e Bitbucket sem armazenar credenciais.

Uso do Kubernetes Secrets

Armazene credenciais diretamente no Kubernetes usando o método padrão do Argo CD.

Para HTTPS com token de acesso pessoal:

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/your-org/your-repo
  username: your-username
  password: your-personal-access-token
```

Para o SSH:

```
apiVersion: v1
kind: Secret
```

```
metadata:
  name: my-repo-ssh
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: git@github.com:your-org/your-repo.git
  sshPrivateKey: |
    -----BEGIN OPENSSH PRIVATE KEY-----
    ... your private key ...
    -----END OPENSSH PRIVATE KEY-----
```

Repositórios públicos

Nenhuma configuração adicional é necessária para o uso de repositórios públicos:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: public-app
  namespace: argocd
spec:
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps
    targetRevision: HEAD
    path: guestbook
  # ... rest of configuration
```

Repositórios do CodeCommit

Para o AWS CodeCommit, conceda ao perfil da funcionalidade do IAM permissões do CodeCommit (`codecommit:GitPull`).

Configurar o repositório:

```
apiVersion: v1
kind: Secret
metadata:
  name: codecommit-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
```

```
stringData:
  type: git
  url: https://git-codecommit.us-west-2.amazonaws.com/v1/repos/my-repo
  project: default
```

Para obter a configuração detalhada da política do IAM, consulte the section called “Considerações sobre o Argo CD”.

Verificação da conexão do repositório

Verifique o status da conexão pela interface do usuário do Argo CD em “Settings → Repositories”. A interface do usuário apresenta o status da conexão e quaisquer erros de autenticação.

Os segredos do repositório não incluem informações de status.

Recursos adicionais

- the section called “Registro de clusters”: registre os clusters de destino para implantações
- the section called “Criação de Applications”: crie sua primeira Application
- the section called “Considerações sobre o Argo CD”: acesse as permissões do IAM e as configuração de segurança
- Private Repositories: acesse a referência de configuração de repositórios da versão original

Registro de clusters de destino

Registre clusters para permitir que o Argo CD implante aplicações neles. É possível registrar o mesmo cluster em que o Argo CD está em execução (cluster local) ou clusters remotos em diferentes contas ou regiões.

Pré-requisitos

- Um cluster do EKS com a funcionalidade para o Argo CD criada
- `kubectl` configurado para o estabelecimento de comunicação com o cluster
- Para clusters remotos: permissões do IAM e entradas de acesso adequadas

Registro do cluster local

Para implantar aplicações no mesmo cluster em que o Argo CD está em execução, registre-o como um destino de implantação.

> **Note**
>
> A funcionalidade do Argo CD não realiza o registro automático do cluster local. É necessário registrá-lo explicitamente para fazer a implantação de aplicações no mesmo cluster.

Com a CLI do Argo CD:

```
argocd cluster add <cluster-context-name> \
  --aws-cluster-name arn:aws:eks:us-west-2:111122223333:cluster/my-cluster \
  --name local-cluster
```

Com um Kubernetes Secret:

```
apiVersion: v1
kind: Secret
metadata:
  name: local-cluster
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: cluster
stringData:
  name: local-cluster
  server: arn:aws:eks:us-west-2:111122223333:cluster/my-cluster
  project: default
```

Aplique a configuração:

```
kubectl apply -f local-cluster.yaml
```

> **Note**
>
> Use o ARN do cluster do EKS no campo `server`, e não o URL do servidor da API do Kubernetes. A funcionalidade gerenciada requer ARNs para identificar os clusters. O valor padrão `kubernetes.default.svc` não é compatível.

Registro de clusters remotos

Para realizar a implantação em clusters remotos, é necessário:

1. Criar uma entrada de acesso no cluster remoto para o seu perfil da funcionalidade do IAM destinado ao Argo CD
2. Associar uma política de acesso com as permissões apropriadas
3. Registrar o cluster no Argo CD

Etapa 1: criação da entrada de acesso no cluster remoto

Substitua *region-code* pela região da AWS em que seu cluster remoto está, substitua *remote-cluster* pelo nome do cluster remoto e substitua o ARN pelo ARN do perfil da funcionalidade para o Argo CD.

```
aws eks create-access-entry \
  --region region-code \
  --cluster-name remote-cluster \
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --type STANDARD
```

Etapa 2: associação da política de acesso

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name remote-cluster \
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

> **Note**
>
> Em ambientes de produção, é recomendável adotar políticas de acesso mais restritivas. Consulte the section called “Considerações para funcionalidades do EKS” para obter configurações que seguem o princípio do privilégio mínimo.

Etapa 3: registro do cluster no Argo CD

Com a CLI do Argo CD:

```
argocd cluster add <cluster-context-name> \
```

```
  --aws-cluster-name arn:aws:eks:us-west-2:111122223333:cluster/remote-cluster \
  --name remote-cluster
```

Com um Kubernetes Secret:

```
apiVersion: v1
kind: Secret
metadata:
  name: remote-cluster
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: cluster
stringData:
  name: remote-cluster
  server: arn:aws:eks:us-west-2:111122223333:cluster/remote-cluster
  project: default
```

Aplique a configuração:

```
kubectl apply -f remote-cluster.yaml
```

Clusters entre contas e entre regiões

Para realizar a implantação em clusters localizados em contas ou regiões da AWS distintas:

1. Adicione o perfil da funcionalidade do IAM destinado ao Argo CD como uma entrada de acesso no cluster remoto
2. Associe a política de acesso apropriada (normalmente `AmazonEKSClusterAdminPolicy`)
3. Registre o cluster usando seu ARN completo (que inclui a região)

O formato do ARN do cluster inclui a região; portanto, não há diferença entre registros entre contas e entre regiões. Ambos seguem o mesmo processo.

Para obter a configuração detalhada entre contas, incluindo políticas de confiança e de permissões do IAM, consulte the section called “Considerações sobre o Argo CD”.

Verificação do registro do cluster

Consulte os clusters que estão registrados:

```
kubectl get secrets -n argocd -l argocd.argoproj.io/secret-type=cluster
```

Como opção, verifique o status do cluster na interface do Argo CD em Configurações → Clusters.

Clusters privados

A funcionalidade do Argo CD fornece acesso transparente a clusters do EKS totalmente privados, sem exigir emparelhamento da VPC ou configurações de rede especializadas.

A AWS gerencia automaticamente a conectividade entre a funcionalidade do Argo CD e os clusters remotos privados.

Basta registrar o cluster privado usando seu ARN. Nenhuma configuração de rede adicional é necessária.

Restrição do acesso ao cluster com o Projects

Use Projects para controlar em quais clusters as Applications podem ser implantadas:

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: production
  namespace: argocd
spec:
  destinations:
  - server: arn:aws:eks:us-west-2:111122223333:cluster/prod-cluster
    namespace: '*'
  - server: arn:aws:eks:eu-west-1:111122223333:cluster/prod-eu-cluster
    namespace: '*'
  sourceRepos:
  - 'https://github.com/example/production-apps'
```

Para obter detalhes, consulte the section called “Projetos”.

Recursos adicionais

- the section called “Projetos”: organize as Applications e aplique limites de segurança
- the section called “Criação de Applications”: implante sua primeira aplicação
- the section called “ApplicationSets”: realize implantações em vários clusters com ApplicationSets

- the section called “Considerações sobre o Argo CD”: acesse padrões com vários clusters e a configuração entre contas
- Declarative Cluster Setup: acesse a referência de configuração da versão original do cluster

Trabalho com o Argo CD Projects

O Argo CD Projects (AppProject) oferece agrupamento lógico e controle de acesso para Applications. Os projetos definem quais repositórios do Git, clusters de destino e namespaces podem ser usados pelas Applications, viabilizando multilocação e limites de segurança em instâncias compartilhadas do Argo CD.

Situações para o uso do Argo CD Projects

Use o Projects para:

- Separar aplicações por equipe, ambiente ou unidade de negócio
- Restringir de quais repositórios as equipes podem realizar a implantação
- Limitar para quais clusters e namespaces as equipes podem realizar a implantação
- Aplicar cotas de recursos e tipos de recursos permitidos
- Fornecer implantação de aplicações de autoatendimento com barreiras de proteção

Projeto padrão

Cada funcionalidade do Argo CD inclui um projeto `default` que permite acesso a todos os repositórios, clusters e namespaces. Apesar de ser útil para testes iniciais, crie Argo CD Projects dedicados com restrições específicas para o uso em produção.

Para obter detalhes sobre a configuração do projeto padrão e como restringi-lo, consulte The Default Project na documentação do Argo CD.

Criar um projeto

Crie um Argo CD Project por meio da aplicação de um recurso `AppProject` em seu cluster.

Exemplo: projeto específico por equipe

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
```

```
  name: team-a
  namespace: argocd
spec:
  description: Applications for Team A

  # Source repositories this project can deploy from
  sourceRepos:
    - 'https://github.com/my-org/team-a-*'
    - 'https://github.com/my-org/shared-libs'

  # Destination clusters and namespaces
  destinations:
    - name: dev-cluster
      namespace: team-a-dev
    - name: prod-cluster
      namespace: team-a-prod

  # Allowed resource types
  clusterResourceWhitelist:
    - group: ''
      kind: Namespace

  namespaceResourceWhitelist:
    - group: 'apps'
      kind: Deployment
    - group: ''
      kind: Service
    - group: ''
      kind: ConfigMap
```

Aplique o Project:

```
kubectl apply -f team-a-project.yaml
```

Configuração de projetos

Repositórios de origem

Controle quais repositórios do Git podem ser usados por Applications neste projeto:

```
spec:
  sourceRepos:
    - 'https://github.com/my-org/app-*'  # Wildcard pattern
```

```
    - 'https://github.com/my-org/infra'  # Specific repo
```

É possível empregar caracteres curinga e padrões de negação (prefixo `!`) para permitir ou bloquear repositórios específicos. Para obter detalhes, consulte Managing Projects na documentação do Argo CD.

Restrições de destinos

Limite o local em que as Applications podem ser implantadas:

```
spec:
  destinations:
    - name: prod-cluster  # Specific cluster by name
      namespace: production
    - name: '*'  # Any cluster
      namespace: team-a-*  # Namespace pattern
```

> **Important**
>
> Use nomes de clusters específicos e padrões de namespace em vez de caracteres curingas para Projects destinados a ambientes de produção. Isso impede a implantação acidental em clusters ou namespaces não autorizados.

É possível usar caracteres curinga e padrões de negação para controlar os destinos. Para obter detalhes, consulte Managing Projects na documentação do Argo CD.

Restrições de recursos

Controle quais tipos de recursos do Kubernetes podem ser implantados:

Recursos com escopo de cluster:

```
spec:
  clusterResourceWhitelist:
    - group: ''
      kind: Namespace
    - group: 'rbac.authorization.k8s.io'
      kind: Role
```

Recursos com escopo de namespace:

```
spec:
  namespaceResourceWhitelist:
    - group: 'apps'
      kind: Deployment
    - group: ''
      kind: Service
    - group: ''
      kind: ConfigMap
    - group: 's3.services.k8s.aws'
      kind: Bucket
```

Use listas de restrições para negar recursos específicos:

```
spec:
  namespaceResourceBlacklist:
    - group: ''
      kind: Secret  # Prevent direct Secret creation
```

Atribuição de Applications a Projects

Ao criar uma Application, especifique o projeto no campo `spec.project`:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  project: team-a  # Assign to team-a project
  source:
    repoURL: https://github.com/my-org/my-app
    path: manifests
  destination:
    name: prod-cluster
    namespace: team-a-prod
```

As Applications sem um projeto especificado usam o projeto `default`.

Perfis de projeto e RBAC

Os Projects podem definir perfis personalizados para um controle de acesso mais preciso. Mapeie os perfis do projeto para usuários e grupos do Centro de Identidade da AWS na configuração de

funcionalidades para controlar as pessoas com permissões para sincronizar, atualizar ou excluir aplicações.

Exemplo: Project com perfis de desenvolvedor e de administrador

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
  namespace: argocd
spec:
  sourceRepos:
    - '*'
  destinations:
    - name: '*'
      namespace: 'team-a-*'

  roles:
    - name: developer
      description: Developers can sync applications
      policies:
        - p, proj:team-a:developer, applications, sync, team-a/*, allow
        - p, proj:team-a:developer, applications, get, team-a/*, allow
      groups:
        - team-a-developers

    - name: admin
      description: Admins have full access
      policies:
        - p, proj:team-a:admin, applications, *, team-a/*, allow
      groups:
        - team-a-admins
```

Para obter detalhes sobre os perfis de projeto, tokens JWT para pipelines de CI/CD e configuração de RBAC, consulte Project Roles na documentação do Argo CD.

Padrões comuns

Projects com base no ambiente

Crie projetos separados para cada ambiente:

```
apiVersion: argoproj.io/v1alpha1
```

```
kind: AppProject
metadata:
  name: production
  namespace: argocd
spec:
  sourceRepos:
    - 'https://github.com/my-org/*'
  destinations:
    - name: prod-cluster
      namespace: '*'
  # Strict resource controls for production
  clusterResourceWhitelist: []
  namespaceResourceWhitelist:
    - group: 'apps'
      kind: Deployment
    - group: ''
      kind: Service
```

Projects com base em equipe

Isole as equipes com projetos dedicados:

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: platform-team
  namespace: argocd
spec:
  sourceRepos:
    - 'https://github.com/my-org/platform-*'
  destinations:
    - name: '*'
      namespace: 'platform-*'
  # Platform team can manage cluster resources
  clusterResourceWhitelist:
    - group: '*'
      kind: '*'
```

Projects com vários clusters

Faça a implantação em vários clusters utilizando políticas consistentes:

```
apiVersion: argoproj.io/v1alpha1
```

```
kind: AppProject
metadata:
  name: global-app
  namespace: argocd
spec:
  sourceRepos:
    - 'https://github.com/my-org/global-app'
  destinations:
    - name: us-west-cluster
      namespace: app
    - name: eu-west-cluster
      namespace: app
    - name: ap-south-cluster
      namespace: app
```

Práticas recomendadas

Comece com Projects restritivos: comece com permissões limitadas e expanda conforme necessário, em vez de começar com acesso amplo.

Use padrões de namespace: use curingas nas restrições de namespace (como `team-a-*`) para permitir flexibilidade e, ao mesmo tempo, manter os limites.

Projetos separados para ambientes de produção: use projetos dedicados para produção com controles mais rigorosos e políticas de sincronização manual.

Documente os objetivos do Project: use o campo `description` para explicar a finalidade de cada projeto e quem deve usá-lo.

Analise as permissões do Project regularmente: audite os Projects periodicamente para garantir que as restrições ainda estejam alinhadas com as necessidades da equipe e os requisitos de segurança.

Recursos adicionais

- the section called “Configurar permissões”: configure o RBAC e a integração com o Centro de Identidade
- the section called “Criação de Applications”: crie Applications em Projects
- the section called “ApplicationSets”: use ApplicationSets com os Projects para realizar implantações em vários clusters
- Documentação do Argo CD Projects: acesse a referência oficial completa

Criação de Applications

As Applications correspondem a implantações nos clusters de destino. Cada Application define uma origem (repositório do Git) e um destino (cluster e namespace). Quando aplicado, o Argo CD criará os recursos especificados pelos manifestos no repositório do Git para o namespace no cluster. As Applications frequentemente especificam implantações de workloads, mas podem gerenciar quaisquer recursos do Kubernetes disponíveis no cluster de destino.

Pré-requisitos

- Um cluster do EKS com a funcionalidade para o Argo CD criada
- Acesso ao repositório configurado (consulte the section called “Configuração dos repositórios”)
- Cluster de destino registrado (consulte the section called “Registro de clusters”)
- `kubectl` configurado para o estabelecimento de comunicação com o cluster

Criação de uma Application básica

Defina uma Application que realize implantações usando um repositório do Git:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps
    targetRevision: HEAD
    path: guestbook
  destination:
    server: arn:aws:eks:us-west-2:111122223333:cluster/my-cluster
    namespace: default
```

> **Important**
>
> Use o ARN do cluster do EKS no campo `destination.server`, e não o URL do servidor da API do Kubernetes. A funcionalidade gerenciada requer ARNs para identificar os clusters.

Aplique a Application:

```
kubectl apply -f application.yaml
```

Visualize o status da Application:

```
kubectl get application guestbook -n argocd
```

Configuração da origem

Repositório do Git:

```
spec:
  source:
    repoURL: https://github.com/example/my-app
    targetRevision: main
    path: kubernetes/manifests
```

Etiqueta ou confirmação específica do Git:

```
spec:
  source:
    targetRevision: v1.2.0  # or commit SHA
```

Chart do Helm:

```
spec:
  source:
    repoURL: https://github.com/example/helm-charts
    targetRevision: main
    path: charts/my-app
    helm:
      valueFiles:
      - values.yaml
      parameters:
      - name: image.tag
        value: v1.2.0
```

Chart do Helm do ECR:

```
spec:
```

```
  source:
    repoURL: oci://account-id.dkr.ecr.region.amazonaws.com/repository-name
    targetRevision: chart-version
    chart: chart-name
```

Se o perfil da funcionalidade tiver as permissões necessárias do ECR, o repositório será usado diretamente, sem necessidade da configuração de Repository adicional. Para mais detalhes, consulte the section called “Configuração dos repositórios”.

Repositório do Git do CodeCommit:

```
spec:
  source:
    repoURL: https://git-codecommit.region.amazonaws.com/v1/repos/repository-name
    targetRevision: main
    path: kubernetes/manifests
```

Se o perfil da funcionalidade tiver as permissões necessárias do CodeCommit, o repositório será usado diretamente, sem necessidade da configuração de Repository adicional. Para mais detalhes, consulte the section called “Configuração dos repositórios”.

Repositório do Git do CodeConnections:

```
spec:
  source:
    repoURL: https://codeconnections.region.amazonaws.com/git-http/account-
id/region/connection-id/owner/repository.git
    targetRevision: main
    path: kubernetes/manifests
```

O formato do URL do repositório é derivado do ARN da conexão do CodeConnections. Se o perfil da funcionalidade tiver as permissões necessárias do CodeConnections e uma conexão estiver configurada, o repositório será usado diretamente, sem necessidade da configuração de Repository adicional. Para mais detalhes, consulte the section called “Configuração dos repositórios”.

Kustomize:

```
spec:
  source:
    repoURL: https://github.com/example/kustomize-app
```

```
    targetRevision: main
    path: overlays/production
    kustomize:
      namePrefix: prod-
```

Políticas de sincronização

Defina como o Argo CD deve sincronizar as aplicações.

Sincronização manual (padrão):

As aplicações exigem aprovação manual para sincronizar alterações:

```
spec:
  syncPolicy: {}  # No automated sync
```

Acionamento manual da sincronização:

```
kubectl patch application guestbook -n argocd \
  --type merge \
  --patch '{"operation": {"initiatedBy": {"username": "admin"}, "sync": {}}}'
```

Sincronização automática:

As aplicações são sincronizadas automaticamente quando alterações no Git são detectadas:

```
spec:
  syncPolicy:
    automated: {}
```

Autorreparação:

Reverte automaticamente alterações manuais no cluster:

```
spec:
  syncPolicy:
    automated:
      selfHeal: true
```

Quando esta opção está habilitada, o Argo CD reverte quaisquer alterações realizadas diretamente no cluster, garantindo que o Git permaneça como fonte da verdade.

Supressão:

Exclui automaticamente recursos removidos do Git:

```
spec:
  syncPolicy:
    automated:
      prune: true
```

> ⚠ Warning
>
> A supressão excluirá recursos do seu cluster. Use com cautela em ambientes de produção.

Sincronização automática combinada:

```
spec:
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - CreateNamespace=true
```

Opções de sincronização

Configuração de sincronização adicional:

Crie um namespace, se ele não existir:

```
spec:
  syncPolicy:
    syncOptions:
    - CreateNamespace=true
```

Valide os recursos antes de realizar a aplicação:

```
spec:
  syncPolicy:
    syncOptions:
    - Validate=true
```

Realize a aplicação somente quando não estiver em sincronização:

```
spec:
  syncPolicy:
    syncOptions:
    - ApplyOutOfSyncOnly=true
```

Recursos avançados de sincronização

O Argo CD oferece recursos avançados de sincronização para implantações complexas:

- Ciclos de sincronização: controlam a ordem de criação de recursos usando anotações `argocd.argoproj.io/sync-wave`
- Hooks de sincronização: executam trabalhos antes ou depois da sincronização usando as anotações `argocd.argoproj.io/hook` (nomeadamente, PreSync, PostSync e SyncFail)
- Avaliação da integridade dos recursos: verificações de integridade personalizadas são realizadas para recursos específicos da aplicação

Para obter detalhes, consulte Sync Waves e Resource Hooks na documentação do CD Argo.

Desconsideração de diferenças

Evite que o Argo CD sincronize campos específicos gerenciados por outros controladores (como a HPA gerenciando réplicas):

```
spec:
  ignoreDifferences:
  - group: apps
    kind: Deployment
    jsonPointers:
    - /spec/replicas
```

Para obter detalhes sobre como desconsiderar padrões e realizar exclusões de campos, consulte Diffing Customization na documentação do Argo CD.

Implantação em vários ambientes

Implante a mesma aplicação em vários ambientes:

Desenvolvimento do:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-dev
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/example/my-app
    targetRevision: develop
    path: overlays/development
  destination:
    server: arn:aws:eks:us-west-2:111122223333:cluster/dev-cluster
    namespace: my-app
```

Produção:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-prod
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/example/my-app
    targetRevision: main
    path: overlays/production
  destination:
    server: arn:aws:eks:us-west-2:111122223333:cluster/prod-cluster
    namespace: my-app
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

Monitoramento e gerenciamento de Applications

Visualize o status da Application:

```
kubectl get application my-app -n argocd
```

Acesse a interface do usuário do Argo CD:

Abra a interface do usuário do Argo CD pelo console do EKS para visualizar a topologia da aplicação, o status de sincronização, a integridade dos recursos e o histórico de implantações. Consulte the section called “Como trabalhar com o Argo CD” para obter instruções de acesso à interface do usuário.

Reversão de Applications:

Reverta para uma revisão anterior por meio da interface do usuário do Argo CD ou alterando o `targetRevision` na especificação da Application para uma confirmação ou etiqueta anterior no Git.

Recursos adicionais

- the section called “Projetos”: organize aplicações com Projects para ambientes multilocatários
- the section called “ApplicationSets”: realize implantações em vários clusters com modelos
- Application Specification: acesse a referência de API completa para a Application
- Sync Options: acesse a configuração de sincronização avançada

Uso de ApplicationSets

Os ApplicationSets geram diversas Applications a partir de modelos, possibilitando a implantação da mesma aplicação em diversos clusters, ambientes ou namespaces com uma única definição de recurso.

Pré-requisitos

- Um cluster do EKS com a funcionalidade para o Argo CD criada
- Diversos clusters de destino registrados (consulte the section called “Registro de clusters”)
- Acesso ao repositório configurado (consulte the section called “Configuração dos repositórios”)
- `kubectl` configurado para o estabelecimento de comunicação com o cluster

Funcionamento dos ApplicationSets

Os ApplicationSets usam geradores para criar parâmetros e, em seguida, aplicá-los a um modelo de Application. Cada conjunto de parâmetros gerados cria uma Application.

Geradores mais usados em implantações do EKS:

- Gerador de listas: define explicitamente clusters e parâmetros para cada ambiente
- Gerador de cluster: realiza a implantação automática em todos os clusters registrados
- Gerador do Git: gera Applications com base na estrutura do repositório
- Gerador de matriz: combina geradores para implantações multidimensionais

Para obter uma referência completa dos geradores, consulte a Documentação do ApplicationSet.

Gerador de listas

Realize uma implantação em vários clusters usando configuração explícita:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook-all-clusters
  namespace: argocd
spec:
  generators:
  - list:
      elements:
      - cluster: arn:aws:eks:us-west-2:111122223333:cluster/dev-cluster
        environment: dev
        replicas: "2"
      - cluster: arn:aws:eks:us-west-2:111122223333:cluster/staging-cluster
        environment: staging
        replicas: "3"
      - cluster: arn:aws:eks:us-west-2:111122223333:cluster/prod-cluster
        environment: prod
        replicas: "5"
  template:
    metadata:
      name: 'guestbook-{{environment}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/guestbook
        targetRevision: HEAD
        path: 'overlays/{{environment}}'
      destination:
        server: '{{cluster}}'
```

```
        namespace: guestbook
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

> **Note**
>
> Use os ARNs dos clusters do EKS no campo `server` ao direcionar clusters do EKS registrados. Além disso, é possível usar nomes de clusters com `destination.name` em vez de `destination.server`.

Essa ação realiza a criação de três Applications: `guestbook-dev`, `guestbook-staging` e `guestbook-prod`.

Gerador de cluster

Realize a implantação automática de todos os clusters registrados:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: cluster-addons
  namespace: argocd
spec:
  generators:
  - clusters: {}
  template:
    metadata:
      name: '{{name}}-addons'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/cluster-addons
        targetRevision: HEAD
        path: addons
      destination:
        server: '{{server}}'
        namespace: kube-system
      syncPolicy:
        automated:
```

```
          prune: true
          selfHeal: true
```

Essa ação cria automaticamente uma Application para cada cluster registrado.

Clusters de filtros:

```
spec:
  generators:
  - clusters:
      selector:
        matchLabels:
          environment: production
```

Geradores do Git

Os geradores do Git criam Applications com base na estrutura do repositório:

- Gerador de diretório: implanta cada diretório como uma Application separada (útil para microsserviços)
- Gerador de arquivo gera Applications com base em arquivos de parâmetros (útil para implantações multilocatárias)

Exemplo: implantação de microsserviços

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: microservices
  namespace: argocd
spec:
  generators:
  - git:
      repoURL: https://github.com/example/microservices
      revision: HEAD
      directories:
      - path: services/*
  template:
    metadata:
      name: '{{path.basename}}'
    spec:
```

```
      project: default
      source:
        repoURL: https://github.com/example/microservices
        targetRevision: HEAD
        path: '{{path}}'
      destination:
        server: arn:aws:eks:us-west-2:111122223333:cluster/my-cluster
        namespace: '{{path.basename}}'
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
        syncOptions:
        - CreateNamespace=true
```

Para obter mais detalhes sobre os geradores do Git e a configuração baseada em arquivos, consulte Git Generator na documentação do Argo CD.

Gerador de matriz

Use a combinação de vários geradores para realizar implantações em diversas dimensões (ambientes × clusters):

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: multi-env-multi-cluster
  namespace: argocd
spec:
  generators:
  - matrix:
      generators:
      - list:
          elements:
          - environment: dev
          - environment: staging
          - environment: prod
      - clusters:
          selector:
            matchLabels:
              region: us-west-2
  template:
    metadata:
```

```
      name: 'app-{{environment}}-{{name}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/app
        targetRevision: HEAD
        path: 'overlays/{{environment}}'
      destination:
        server: '{{server}}'
        namespace: 'app-{{environment}}'
```

Para obter mais detalhes sobre como combinar geradores, consulte Matrix Generator na documentação do Argo CD.

Implementação progressiva

Realize a implantação nos ambientes de forma sequencial com políticas de sincronização distintas:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: progressive-rollout
  namespace: argocd
spec:
  generators:
  - list:
      elements:
      - environment: dev
        autoSync: "true"
      - environment: staging
        autoSync: "true"
      - environment: prod
        autoSync: "false"
  template:
    metadata:
      name: 'app-{{environment}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/app
        targetRevision: HEAD
        path: 'overlays/{{environment}}'
      destination:
```

```
        server: arn:aws:eks:us-west-2:111122223333:cluster/{{environment}}-cluster
        namespace: app
      syncPolicy:
        automated:
          prune: true
          selfHeal: '{{autoSync}}'
```

Os ambientes de desenvolvimento e de preparação são sincronizados automaticamente, já o ambiente de produção requer aprovação manual.

Implantação multirregião

Realize a implantação em clusters distribuídos por várias regiões:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: global-app
  namespace: argocd
spec:
  generators:
  - list:
      elements:
      - cluster: arn:aws:eks:us-west-2:111122223333:cluster/prod-us-west
        region: us-west-2
      - cluster: arn:aws:eks:us-east-1:111122223333:cluster/prod-us-east
        region: us-east-1
      - cluster: arn:aws:eks:eu-west-1:111122223333:cluster/prod-eu-west
        region: eu-west-1
  template:
    metadata:
      name: 'app-{{region}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/app
        targetRevision: HEAD
        path: kubernetes
        helm:
          parameters:
          - name: region
            value: '{{region}}'
      destination:
```

```
        server: '{{cluster}}'
        namespace: app
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

Gerenciamento de ApplicationSets

Visualize os ApplicationSets e as Applications geradas:

```
kubectl get applicationsets -n argocd
kubectl get applications -n argocd -l argocd.argoproj.io/application-set-
name=<applicationset-name>
```

Atualize um ApplicationSet:

Modifique a especificação do ApplicationSet e a aplique novamente. O Argo CD atualiza automaticamente todas as Applications geradas:

```
kubectl apply -f applicationset.yaml
```

Exclua um ApplicationSet:

```
kubectl delete applicationset <name> -n argocd
```

> **Warning**
>
> Ao excluir um ApplicationSet, todas as Applications geradas são apagadas. Se essas Applications tiverem `prune: true`, os recursos correspondentes também serão removidos dos clusters de destino.

Recursos adicionais

- the section called “Projetos”: organize os ApplicationSets com Projects
- the section called “Criação de Applications”: compreenda a configuração das Applications
- Documentação do ApplicationSet: referência completa de geradores e seus padrões

- Referência de geradores: especificações detalhadas dos geradores

Considerações sobre o Argo CD

Neste tópico, são apresentadas as principais considerações para usar a funcionalidade do EKS para o Argo CD, incluindo planejamento, permissões, autenticação e padrões de implantação em vários clusters.

Planejamento

Antes de implantar o Argo CD, considere o seguinte:

Estratégia de repositório: determine o local em que os manifestos das suas aplicações serão armazenados (por exemplo, CodeCommit, GitHub, GitLab ou Bitbucket). Planeje a estrutura do repositório e a estratégia de ramificação para diferentes ambientes.

Método de autenticação: decida entre usar o Centro de Identidade da AWS para SSO ou gerenciar os usuários do Argo CD de forma direta. O SSO é recomendado para ambientes de produção.

Estratégia de RBAC: defina quais equipes ou usuários terão acesso como administrador, editor ou visualizador. Relacione essas permissões aos grupos do Centro de Identidade da AWS ou aos perfis do Argo CD.

Arquitetura de vários clusters: determine se você gerenciará vários clusters usando uma única instância do Argo CD. Avalie a possibilidade de empregar um cluster de gerenciamento dedicado para o Argo CD.

Organização das aplicações: planeje como você estruturará Applications e ApplicationSets. Avalie a utilização de projetos para estruturar as aplicações de acordo com a equipe ou o ambiente.

Políticas de sincronização: defina se as aplicações serão sincronizadas automaticamente ou se precisarão de aprovação manual. A sincronização automática é comum em ambientes de desenvolvimento, e a manual em ambientes de produção.

Permissões

Para obter informações detalhadas sobre os perfis de funcionalidades do IAM, as políticas de confiança e as práticas recomendadas de segurança, consulte the section called “Perfil do IAM para a funcionalidade” e the section called “Considerações para funcionalidades do EKS”.

Visão geral do perfil da funcionalidade do IAM

Ao criar um recurso de funcionalidade do Argo CD, você fornece um perfil da funcionalidade do IAM. Diferentemente do ACK, o Argo CD gerencia principalmente recursos do Kubernetes e não recursos da AWS diretamente. No entanto, o perfil do IAM é necessário para:

- Acessar repositórios do Git privados no CodeCommit
- Integrar-se ao Centro de Identidade da AWS para autenticação
- Acessar segredos no AWS Secrets Manager (se configurado)
- Realizar implantações entre clusters para outros clusters do EKS

Integração com o CodeCommit

Se você estiver usando repositórios do CodeCommit, anexe uma política com permissões de leitura:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    }
  ]
}
```

Important

Para uso em ambientes de produção, restrinja o campo `Resource` a ARNs específicos de repositórios, em vez de usar `"*"`.
Exemplo:

```
"Resource": "arn:aws:codecommit:us-west-2:111122223333:my-app-repo"
```

Dessa forma, o acesso do Argo CD fica restrito somente aos repositórios necessários para gerenciamento.

Integração do Secrets Manager

Se você estiver armazenando credenciais de repositório no Secrets Manager, anexe uma política com permissões de leitura:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:*:*:secret:argocd/*"
    }
  ]
}
```

Configuração básica

Para a funcionalidade básica do CD Argo com repositórios do Git públicos, nenhuma política do IAM adicional é necessária além da política de confiança.

Autenticação

Integração com o Centro de Identidade da AWS

A funcionalidade gerenciada do Argo CD integra-se diretamente ao Centro de Identidade da AWS (anteriormente AWS SSO), possibilitando o uso do provedor de identidade atual para autenticação.

Ao configurar a integração com o Centro de Identidade da AWS:

1. Os usuários acessam a interface do usuário do Argo CD por meio do console do EKS
2. A autenticação é realizada por meio do Centro de Identidade da AWS (que pode ser federado ao seu provedor de identidades corporativo)
3. O Centro de Identidade da AWS fornece informações de usuários e grupos ao Argo CD
4. O Argo CD mapeia usuários e grupos para perfis de RBAC com base na sua configuração
5. Os usuários visualizam somente as aplicações e os recursos aos quais têm permissão de acessar

Simplificação do acesso com conjuntos de permissões do Centro de Identidade

O Centro de Identidade da AWS fornece dois caminhos distintos de autenticação ao trabalhar com o Argo CD:

Autenticação da API do Argo CD: o Centro de Identidade fornece autenticação de SSO para a interface do usuário do Argo CD e para a API. Essa configuração é realizada por meio dos mapeamentos de perfis de RBAC da funcionalidade do Argo CD.

Acesso ao cluster do EKS: a funcionalidade do Argo CD usa o perfil do IAM fornecido pelo cliente para se autenticar com clusters do EKS por meio de entradas de acesso. Essas entradas de acesso podem ser configuradas manualmente para conceder ou revogar permissões.

É possível usar conjuntos de permissões do Centro de Identidade para simplificar o gerenciamento de identidades, ao permitir que uma única identidade tenha acesso ao Argo CD e aos clusters do EKS. Isso reduz o esforço operacional, pois você precisa gerenciar apenas uma identidade nos dois sistemas, em vez de manter credenciais distintas para o acesso ao Argo CD e ao cluster.

Mapeamentos de perfis de RBAC

O Argo CD tem perfis integrados que você pode mapear para usuários e grupos do Centro de Identidade da AWS:

ADMIN: acesso total a todas as aplicações e configurações. Pode criar, atualizar e excluir aplicações. Pode gerenciar a configuração do Argo CD.

EDITOR: pode criar e modificar aplicações. Não pode alterar as configurações do Argo CD nem excluir aplicações.

VIEWER: acesso somente leitura às aplicações. Pode visualizar o status e o histórico das aplicações. Não pode fazer alterações.

> **Note**
>
> Os nomes dos perfis diferenciam maiúsculas de minúsculas e devem estar em letras maiúsculas (ADMIN, EDITOR e VIEWER).

> **Important**
>
> A integração das funcionalidades do EKS com o Centro de Identidade da AWS fornece suporte para até mil identidades por funcionalidade do Argo CD. Uma identidade pode ser um usuário ou um grupo.

Implantações em vários clusters

A funcionalidade gerenciada do Argo CD fornece suporte a implantações em vários clusters, possibilitando o gerenciamento de aplicações em clusters de desenvolvimento, preparação e produção usando uma única instância do Argo CD.

Funcionamento da estratégia de vários clusters

Ao registrar clusters adicionais com o Argo CD:

1. Você cria segredos de cluster que fazem referência aos clusters de destino do EKS pelo ARN
2. Você cria Applications ou ApplicationSets direcionados a diferentes clusters
3. O Argo CD estabelece conexão com cada cluster para implantar e monitorar os recursos
4. Você visualiza e gerencia todos os clusters usando uma única interface do usuário do Argo CD

Pré-requisitos para a estratégia de vários clusters

Antes de registrar clusters adicionais:

- Crie uma entrada de acesso no cluster de destino para o perfil da funcionalidade para o Argo CD
- Garanta a conectividade de rede entre a funcionalidade do Argo CD e os clusters de destino
- Verifique as permissões do IAM para acessar os clusters de destino

Registro de um cluster

Registre clusters usando o Kubernetes Secrets no namespace `argocd`.

Obtenha o ARN do cluster de destino. Substitua *`region-code`* pela região da AWS em que seu cluster de destino está e substitua *`target-cluster`* pelo nome do cluster de destino.

```
aws eks describe-cluster \
```

```
  --region region-code \
  --name target-cluster \
  --query 'cluster.arn' \
  --output text
```

Crie um segredo de cluster usando o ARN do cluster:

```
apiVersion: v1
kind: Secret
metadata:
  name: target-cluster
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: cluster
type: Opaque
stringData:
  name: target-cluster
  server: arn:aws:eks:us-west-2:111122223333:cluster/target-cluster
  project: default
```

Important

Use o ARN do cluster do EKS no campo `server`, e não o URL do servidor da API do Kubernetes. A funcionalidade gerenciada requer ARNs para identificar os clusters de destino.

Execute a aplicação do segredo:

```
kubectl apply -f cluster-secret.yaml
```

Configuração de uma entrada de acesso no cluster de destino

O cluster de destino deve ter uma entrada de acesso que conceda ao perfil da funcionalidade para o Argo CD permissão para implantar aplicações. Substitua *region-code* pela região da AWS em que seu cluster de destino está, substitua *target-cluster* pelo nome do cluster de destino e substitua o ARN pelo ARN do perfil da funcionalidade para o Argo CD.

```
aws eks create-access-entry \
  --region region-code \
  --cluster-name target-cluster \
```

```
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --type STANDARD \
  --kubernetes-groups system:masters
```

> **Note**
>
> Para uso em ambientes de produção, considere usar grupos do Kubernetes mais restritivos em vez de `system:masters`.

Acesso a clusters privados

A funcionalidade gerenciada do Argo CD pode ser implantada em clusters do EKS totalmente privados, sem exigir emparelhamento da VPC ou configurações de rede especializadas. A AWS gerencia automaticamente a conectividade entre a funcionalidade do Argo CD e os clusters remotos privados.

Implantação entre contas

Para implantações entre contas, adicione o perfil da funcionalidade do IAM para o Argo CD proveniente da conta de origem à entrada de acesso do cluster do EKS da conta de destino:

1. Na conta de destino, crie uma entrada de acesso no cluster de destino do EKS
2. Use o ARN do perfil da funcionalidade do IAM para o Argo CD da conta de origem como a entidade principal
3. Configure as permissões apropriadas de RBAC do Kubernetes para a entrada de acesso
4. Registre o cluster de destino no Argo CD usando o ARN do cluster do EKS

Não é necessária a criação de perfis do IAM adicionais nem a configuração de política de confiança. As entradas de acesso do EKS gerenciam o acesso entre contas.

Práticas recomendadas

Use fontes declarativas como a fonte da verdade: armazene todos os manifestos das suas aplicações em repositórios do Git, registros do Helm ou imagens OCI, permitindo controle de versão, trilhas de auditoria e colaboração.

Implemente o RBAC adequado: use a integração com o Centro de Identidade da AWS para definir quem tem permissão para acessar e gerenciar aplicações no Argo CD. O Argo CD oferece controle

de acesso detalhado para recursos dentro das Applications (Deployments, Pods, ConfigMaps e Secrets).

Use ApplicationSets para realizar implantações em diversos ambientes: use ApplicationSets para implantar aplicações em vários clusters ou namespaces com configurações diferentes.

Gerenciamento de ciclo de vida

Políticas de sincronização de aplicações

Defina como o Argo CD deve sincronizar as aplicações:

Sincronização manual: as aplicações exigem aprovação manual para sincronizar alterações. Recomendado para ambientes de produção.

Sincronização automática: as aplicações são sincronizadas automaticamente quando alterações no Git são detectadas. Comum em ambientes de desenvolvimento e de preparação.

Autorreparação: reverte automaticamente alterações manuais feitas no cluster. Garante que o estado do cluster corresponda ao do Git.

Supressão: exclui automaticamente recursos removidos do Git. Use com cautela, pois isso pode excluir recursos.

Integridade da aplicação

O Argo CD monitora continuamente a integridade das aplicações:

- Íntegro: todos os recursos estão funcionando conforme esperado
- Em progresso: os recursos estão sendo criados ou atualizados
- Degradado: alguns recursos não são íntegros
- Suspenso: a aplicação está pausada
- Ausente: os recursos estão ausentes no cluster

Janelas de sincronização

Configure as janelas de sincronização para gerenciar os períodos em que as aplicações podem ser sincronizadas:

- Permitir sincronizações apenas durante janelas de manutenção
- Bloquear sincronizações durante o horário comercial

- Agendar sincronizações automáticas para horários específicos
- Usar janelas de sincronização para cenários de emergência nos quais é necessário interromper temporariamente todas as sincronizações

Configuração de webhooks para sincronizações mais ágeis

Por padrão, o Argo CD consulta os repositórios do Git a cada seis minutos para detectar alterações. Para tornar as implantações mais responsivas, configure webhooks do Git que acionam sincronizações imediatamente quando houver alterações.

Os webhooks oferecem vários benefícios, como:

- Resposta imediata de sincronização quando o código é enviado (segundos em vez de minutos)
- Redução da sobrecarga da sondagem e melhor performance do sistema
- Uso mais eficiente dos limites de taxa da API
- Melhoria na experiência do usuário com feedback mais rápido

Endpoint do webhook

A funcionalidade do Argo CD disponibiliza um endpoint do webhook para receber notificações provenientes do Git. Encontre o URL do webhook no console do EKS, na guia Funcionalidades do seu cluster, ou recupere-a usando a AWS CLI:

```
aws eks describe-capability \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --query 'capability.configuration.argoCd.webhookUrl' \
  --output text \
  --region region-code
```

Configuração de webhooks por provedor do Git

GitHub: nas configurações do repositório, adicione um webhook com o URL do webhook do Argo CD. Defina o tipo de conteúdo como `application/json` e selecione “Just the push event”.

GitLab: nas configurações do projeto, adicione um webhook com o URL do webhook do Argo CD. Ative “Push events” e, opcionalmente, “Tag push events”.

Bitbucket: nas configurações do repositório, adicione um webhook com o URL do webhook do Argo CD. Selecione “Repository push” como o acionador.

CodeCommit: crie uma regra do Amazon EventBridge que seja acionada por alterações no estado do repositório do CodeCommit e envie notificações para o endpoint do webhook do Argo CD.

Para obter instruções detalhadas de configuração do webhook, consulte Argo CD Webhook Configuration.

> **Note**
>
> Webhooks funcionam como complemento para a sondagem, mas não a substituem. O Argo CD continua a consultar os repositórios como mecanismo de fallback, caso as notificações do webhook não sejam recebidas.

Próximas etapas

- the section called “Como trabalhar com o Argo CD”: aprenda a criar e gerenciar Applications do Argo CD
- the section called “Solução de problemas”: realize a solução de problemas relacionados ao Argo CD
- the section called “Como trabalhar com funcionalidades”: gerencie os recursos da funcionalidade do Argo CD

Solução de problemas em funcionalidades do Argo CD

Este tópico fornece orientações de solução de problemas para a funcionalidade do EKS destinada ao Argo CD, incluindo verificações de integridade da funcionalidade, problemas de sincronização de aplicações, autenticação de repositórios e implantações em vários clusters.

> **Note**
>
> As funcionalidades do EKS são totalmente gerenciadas e executadas de forma externa ao cluster. Você não tem acesso aos logs do servidor do Argo CD nem ao namespace `argocd`. A solução de problemas se concentra na integridade da funcionalidade, no status das aplicações e na configuração.

Funcionalidade está com o status ACTIVE, mas as aplicações não estão sincronizando

Se a funcionalidade do Argo CD apresentar o status `ACTIVE`, mas as aplicações não estiverem sincronizando, verifique a integridade da funcionalidade e o status das aplicações.

Verifique a integridade da funcionalidade:

Você pode visualizar problemas de integridade e de status da funcionalidade no console do EKS ou usando a AWS CLI.

Console do:

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do seu cluster.
3. Escolha a guia Observabilidade.
4. Escolha Monitorar cluster.
5. Escolha a guia Funcionalidades para visualizar a integridade e o status de todas as funcionalidades.

AWS CLI:

```
# View capability status and health
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd

# Look for issues in the health section
```

Causas comuns:

- Repositório não configurado: o repositório do Git não foi adicionado ao Argo CD
- Falha na autenticação: a chave SSH, o token ou as credenciais do CodeCommit estão inválidos
- Aplicação não criada: não existem recursos de Application no cluster
- Política de sincronização: a sincronização manual é necessária (sincronização automática não habilitada)

- Permissões do IAM: as permissões estão ausentes para o CodeCommit ou o Secrets Manager

Verifique o status da aplicação:

```
# List applications
kubectl get application -n argocd

# View sync status
kubectl get application my-app -n argocd -o jsonpath='{.status.sync.status}'

# View application health
kubectl get application my-app -n argocd -o jsonpath='{.status.health}'
```

Verifique as condições da aplicação:

```
# Describe application to see detailed status
kubectl describe application my-app -n argocd

# View application health
kubectl get application my-app -n argocd -o jsonpath='{.status.health}'
```

Aplicações travadas no estado “Progressing”

Se uma aplicação estiver em `Progressing` mas nunca atingir `Healthy`, verifique o status dos recursos da aplicação e os eventos.

Verifique a integridade do recurso:

```
# View application resources
kubectl get application my-app -n argocd -o jsonpath='{.status.resources}'

# Check for unhealthy resources
kubectl describe application my-app -n argocd | grep -A 10 "Health Status"
```

Causas comuns:

- Implantação não está pronta: os pods não conseguem iniciar ou as sondagens de prontidão apresentam falhas
- Dependências entre recursos: existem recursos esperando que outros recursos fiquem prontos

- Erros ao obter imagens: as imagens de contêiner não estão acessíveis
- Recursos insuficientes: o cluster não tem CPU ou memória suficiente para os pods

Verifique a configuração do cluster de destino (para configurações com vários clusters):

```
# List registered clusters
kubectl get secret -n argocd -l argocd.argoproj.io/secret-type=cluster

# View cluster secret details
kubectl get secret cluster-secret-name -n argocd -o yaml
```

Falhas na autenticação do repositório

Se o Argo CD não conseguir acessar seus repositórios do Git, verifique a configuração de autenticação.

Para repositórios do CodeCommit:

Verifique se o perfil da funcionalidade do IAM tem as permissões necessárias para o CodeCommit:

```
# View IAM policies
aws iam list-attached-role-policies --role-name my-argocd-capability-role
aws iam list-role-policies --role-name my-argocd-capability-role

# Get specific policy details
aws iam get-role-policy --role-name my-argocd-capability-role --policy-name policy-name
```

É necessário que o perfil conte com a permissão `codecommit:GitPull` para os repositórios.

Para repositórios do Git privados::

Verifique se as credenciais do repositório estão configuradas corretamente:

```
# Check repository secret exists
kubectl get secret -n argocd repo-secret-name -o yaml
```

Certifique-se de que o segredo contenha as credenciais de autenticação adequadas (por exemplo, chave SSH, token ou usuário/senha).

Para repositórios que usam o Secrets Manager:

```
# Verify IAM Capability Role has Secrets Manager permissions
aws iam list-attached-role-policies --role-name my-argocd-capability-role

# Test secret retrieval
aws secretsmanager get-secret-value --secret-id arn:aws:secretsmanager:region-
code:111122223333:secret:my-secret
```

Problemas relacionados à implantação em vários clusters

Se as aplicações não estiverem sendo implantadas em clusters remotos, verifique o registro do cluster e a configuração de acesso.

Verifique o registro do cluster:

```
# List registered clusters
kubectl get secret -n argocd -l argocd.argoproj.io/secret-type=cluster

# Verify cluster secret format
kubectl get secret CLUSTER_SECRET_NAME -n argocd -o yaml
```

Certifique-se de que o campo `server` contenha o ARN do cluster EKS, e não o URL da API do Kubernetes.

Verifique a entrada de acesso do cluster de destino:

No cluster de destino, confirme que o perfil da funcionalidade do IAM destinado ao Argo CD conta com uma entrada de acesso:

```
# List access entries (run on target cluster or use AWS CLI)
aws eks list-access-entries --cluster-name target-cluster

# Describe specific access entry
aws eks describe-access-entry \
  --cluster-name target-cluster \
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/my-argocd-capability-
role
```

Verifique as permissões do IAM para várias contas:

Para implantações entre contas, confirme que o perfil da funcionalidade do IAM destinado ao Argo CD conta com uma entrada de acesso no cluster de destino. A funcionalidade gerenciada emprega entradas de acesso do EKS para o acesso entre contas, e não a suposição do perfil do IAM.

Para obter mais informações sobre configuração de vários clusters, consulte the section called "Registro de clusters".

Próximas etapas

- the section called "Considerações sobre o Argo CD": acesse considerações e práticas recomendadas do Argo CD
- the section called "Como trabalhar com o Argo CD": crie e gerencie Applications do Argo CD
- the section called "Registro de clusters": configure implantações em vários clusters
- the section called "Solução de problemas das funcionalidades": acesse orientações gerais para solução de problemas de funcionalidades

Comparação da funcionalidade do EKS para o Argo CD em relação ao Argo CD autogerenciado

A funcionalidade do EKS para o Argo CD fornece uma experiência do Argo CD totalmente gerenciada e executada no EKS. Para obter uma comparação geral das funcionalidades do EKS em relação às soluções autogerenciadas, consulte the section called "Considerações". Este tópico aborda as diferenças específicas do Argo CD, incluindo autenticação, gerenciamento de diversos clusters e suporte aos recursos da versão original.

Diferenças em relação à versão original do Argo CD

A funcionalidade do EKS para o Argo CD se baseia na versão original do Argo CD, porém apresenta diferenças na maneira de acesso, configuração e integração com os serviços da AWS.

RBAC e autenticação: a funcionalidade inclui três funções de RBAC (administrador, editor e visualizador) e usa o Centro de Identidade da AWS para autenticação, em vez do sistema de autenticação interno do Argo CD. Configure os mapeamentos de perfis por meio do parâmetro `rbacRoleMapping` da funcionalidade a fim de mapear os grupos do Centro de Identidade para os perfis do Argo CD, e não por meio do ConfigMap `argocd-rbac-cm` do Argo CD. A interface do usuário do Argo CD é hospedada usando um URL direto próprio (que pode ser localizado no console

do EKS, na guia Funcionalidades do seu cluster), e o acesso à API usa autenticação e autorização da AWS por meio do IAM.

Configuração do cluster: a funcionalidade não configura automaticamente clusters locais nem topologias do tipo “hub-and-spoke”. Você deve configurar os clusters de destino da sua implantação e as entradas de acesso ao EKS. A funcionalidade é compatível somente com clusters do Amazon EKS como destinos de implantação, usando ARNs de cluster do EKS (não URLs do servidor da API do Kubernetes). A funcionalidade não adiciona automaticamente o cluster local (`kubernetes.default.svc`) como destino de implantação. Para realizar a implantação no mesmo cluster em que a funcionalidade foi criada, registre explicitamente esse cluster usando seu ARN.

Acesso simplificado a clusters remotos:: a funcionalidade simplifica implantações em vários clusters ao usar entradas de acesso do EKS para conceder ao Argo CD o acesso a clusters remotos, eliminando a necessidade de configurar perfis do IAM para contas de serviço (IRSA, na sigla em inglês) ou estabelecer suposições de perfis do IAM entre contas. A funcionalidade também fornece acesso transparente a clusters do EKS totalmente privados, sem exigir emparelhamento da VPC ou configurações de rede especializadas. A AWS gerencia automaticamente a conectividade entre a funcionalidade do Argo CD e os clusters remotos privados.

Integração direta com serviços da AWS: a funcionalidade oferece integração direta com serviços da AWS por meio das permissões do IAM do perfil da funcionalidade. É possível referenciar repositórios do CodeCommit, charts do Helm do ECR e CodeConnections diretamente nos recursos da Application, sem a necessidade de criar configurações de Repository. Isso simplifica a autenticação e elimina a necessidade de gerenciar credenciais separadas para os serviços da AWS. Para mais detalhes, consulte the section called “Configuração dos repositórios”.

Suporte a namespaces: a funcionalidade oferece suporte inicial para a implantação de aplicações em um único namespace, que deve ser especificado no momento da criação da funcionalidade. O suporte a aplicações distribuídas em vários namespaces poderá ser incluído em futuras versões.

Recursos não suportados: os seguintes recursos não estão disponíveis na funcionalidade gerenciada:

- Plug-ins de gerenciamento de configuração (CMPs, na sigla em inglês) para geração de manifestos personalizados
- Scripts Lua personalizados para avaliação da integridade de recursos (as verificações de integridade nativas para recursos padrão são suportadas)
- O controlador de notificações

- Argo CD Image Updater
- Provedores de SSO personalizados (somente o Centro de Identidade da AWS é compatível, abrangendo também identidades federadas de terceiros por meio do Centro de Identidade da AWS)
- Extensões da interface do usuário e banners personalizados
- Acesso direto aos ConfigMaps de configuração `argocd-cm`, `argocd-params` e outros

Compatibilidade: as Applications e os ApplicationSets funcionam de forma idêntica à versão original do Argo CD, sem alterações nos seus manifestos. A funcionalidade usa as mesmas APIs do Kubernetes e CRDs, portanto, ferramentas como o `kubectl` funcionam de maneira semelhante. A funcionalidade oferece suporte completo às Applications e aos ApplicationSets, bem como a fluxos de trabalho de GitOps com sincronização automática, implantações em vários clusters, políticas de sincronização (automatizada, de supressão e de autorreparação), ciclos de sincronização e hooks, avaliação da integridade de recursos padrão do Kubernetes, funcionalidades de reversão, fontes de repositórios do Git (HTTPS e SSH), Helm, Kustomize e manifestos em YAML sem formatação, credenciais de aplicativo do GitHub, projetos para multilocação, além de exclusões e inclusões de recursos.

Uso da CLI do Argo CD com a funcionalidade gerenciada

Para a maioria das operações, a CLI do Argo CD se comporta de maneira idêntica à versão original do Argo CD, mas há diferenças na autenticação e no registro de clusters.

Pré-requisitos

Instale a CLI do Argo CD conforme as instruções de instalação da versão original.

Configuração

Configure a CLI usando variáveis de ambiente:

1. Obtenha o URL do servidor do Argo CD no console do EKS (na guia Funcionalidades do seu cluster) ou por meio da AWS CLI:

```
export ARGOCD_SERVER=$(aws eks describe-capability \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --query 'capability.configuration.argoCd.serverUrl' \
  --output text \
```

```
  --region region-code)
```

2. Gere um token de conta na interface do usuário do Argo CD (Settings → Accounts → admin → Generate New Token) e, em seguida, defina-o como uma variável de ambiente:

```
export ARGOCD_AUTH_TOKEN="your-token-here"
```

> **Important**
>
> Essa configuração usa o token da conta de administrador para a configuração inicial e os fluxos de trabalho de desenvolvimento. Para casos de uso em ambientes de produção, empregue perfis e tokens com escopo de projeto, seguindo o princípio de privilégio mínimo. Para obter mais informações sobre a configuração de perfis de projeto e RBAC, consulte the section called “Configurar permissões”.

1. Defina a opção gRPC obrigatória:

```
export ARGOCD_OPTS="--grpc-web"
```

Com essas variáveis de ambiente configuradas, é possível utilizar a CLI do Argo CD sem a necessidade de executar o comando `argocd login`.

Principais diferenças

A funcionalidade gerenciada apresenta as seguintes limitações da CLI:

- Os comandos `argocd admin` não são suportados (pois exigem acesso direto ao pod)
- O comando `argocd login` não é suportado (utilize tokens de conta ou de projeto)
- O comando `argocd cluster add` requer o sinalizador `--aws-cluster-name` com o ARN do cluster do EKS

Exemplo: registro de um cluster

Registre um cluster do EKS para a implantação de aplicações:

```
# Get the cluster ARN
```

```
CLUSTER_ARN=$(aws eks describe-cluster \
  --name my-cluster \
  --query 'cluster.arn' \
  --output text)

# Register the cluster
argocd cluster add $CLUSTER_ARN \
  --aws-cluster-name $CLUSTER_ARN \
  --name in-cluster \
  --project default
```

Para obter a documentação completa da CLI do Argo CD, consulte a referência da CLI do Argo CD.

Caminho de migração

Você pode migrar de um Argo CD autogerenciado para a funcionalidade gerenciada:

1. Analise a configuração atual do Argo CD quanto a recursos não suportados (por exemplo, controle de notificações, CMPs, verificações de integridade personalizadas)
2. Reduza a escala dos controladores do Argo CD autogerenciados para zero réplicas
3. Crie um recurso de funcionalidade do Argo CD no cluster
4. Exporte Applications, ApplicationSets e AppProjects existentes
5. Migre as credenciais de repositório, os segredos do cluster e os modelos de credenciais do repositório (repocreds)
6. Se estiver usando chaves GPG, certificados TLS ou hosts conhecidos de SSH, migre também essas configurações
7. Atualize os campos `destination.server` para usar nomes de cluster ou ARNs de clusters do EKS
8. Aplique essas configurações na instância gerenciada do Argo CD
9. Verifique se as aplicações estão sendo sincronizadas corretamente
10. Desative sua instalação autogerenciada do Argo CD

A funcionalidade gerenciada usa as mesmas APIs e definições de recursos do Argo CD, portanto, seus manifestos existentes funcionam com modificações mínimas.

Próximas etapas

- the section called “Criar o recurso Argo CD”: crie um recurso de funcionalidade do Argo CD

- the section called “Como trabalhar com o Argo CD”: implante sua primeira aplicação
- the section called “Considerações sobre o Argo CD”: configure a integração com o Centro de Identidade da AWS

Composição de recursos com o kro (Kube Resource Orchestrator)

O kro (Kube Resource Orchestrator) é um projeto de código aberto nativo do Kubernetes que permite definir APIs personalizadas do Kubernetes usando uma configuração simples e direta. Com o kro, é fácil configurar novas APIs personalizadas para criar grupos de objetos do Kubernetes e definir operações lógicas entre esses recursos.

Com as funcionalidades do EKS, o kro torna-se uma funcionalidade totalmente gerenciada pela AWS, dispensando a instalação, a manutenção e a escalabilidade de controladores do kro nos clusters.

Funcionamento do kro

O kro introduz uma definição de recurso personalizado (CRD, na sigla em inglês), chamada `ResourceGraphDefinition` (RGD), que viabiliza a criação simples e otimizada de APIs personalizadas do Kubernetes. Ao criar uma `ResourceGraphDefinition`, o kro usa extensões nativas do Kubernetes para a criação e o gerenciamento de novas APIs no cluster. A partir desta especificação exclusiva de recurso, o kro criará e registrará uma nova CRD para você com base na sua especificação e se adaptará para gerenciar os recursos personalizados definidos recentemente.

As RGDs podem conter vários recursos, cabendo ao kro determinar as interdependências e a sequência de criação de recursos, para que você não precise fazer isso. É possível usar uma sintaxe simples para injetar configurações de um recurso em outro, simplificando significativamente as composições e eliminando a necessidade de operadores de integração no cluster. Com o kro, os recursos personalizados podem incluir tanto recursos nativos do Kubernetes quanto quaisquer definições de recursos personalizados (CRDs) instaladas no cluster.

O kro oferece suporte a um tipo principal de recurso:

- ResourceGraphDefinition (RGD): define um recurso personalizado do Kubernetes, encapsulando um ou mais recursos nativos ou personalizados subjacentes

Além deste recurso, o kro criará e gerenciará o ciclo de vida dos recursos personalizados criados com ele, bem como de todos os recursos constituintes.

O kro se integra perfeitamente ao AWS Controllers for Kubernetes (ACK), permitindo que você componha recursos de workloads com recursos da AWS para criar abstrações de alto nível. Isso possibilita a criação de seus próprios blocos de criação na nuvem, simplificando o gerenciamento de recursos e permitindo padrões reutilizáveis com configurações padrão e imutáveis, baseadas nos padrões da organização.

Benefícios do kro

Com o kro, as equipes responsáveis pela plataforma podem criar APIs personalizadas do Kubernetes que agrupam vários recursos em abstrações de alto nível. Isso simplifica o gerenciamento de recursos ao permitir que desenvolvedores implantem aplicações complexas por meio de recursos personalizados que são simples, padronizados e versionados. Você define padrões reutilizáveis para combinações comuns de recursos, permitindo a criação consistente de recursos em toda a organização.

O kro faz uso da Common Expression Language (CEL) no Kubernetes para a transferência de valores entre recursos e inclusão de lógica condicional, conferindo maior flexibilidade à composição de recursos. É possível compor tanto recursos do Kubernetes quanto recursos da AWS gerenciados pelo ACK em APIs personalizadas unificadas, permitindo definições completas de aplicação e de infraestrutura.

O kro oferece suporte à configuração declarativa por meio de manifestos do Kubernetes, permitindo fluxos de trabalho de GitOps e práticas de infraestrutura como código que se integram perfeitamente aos processos de desenvolvimento existentes. Como parte das funcionalidades do EKS, o kro é totalmente gerenciado pela AWS, dispensando a instalação, a configuração e a manutenção de controladores do kro nos clusters.

Exemplo: criação de uma ResourceGraphDefinition

O seguinte exemplo apresenta uma `ResourceGraphDefinition` simples para a criação de uma aplicação web contendo um Deployment e um Service:

```
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: web-application
spec:
  schema:
    apiVersion: v1alpha1
```

```
    kind: WebApplication
    spec:
      name: string
      replicas: integer | default=3
  resources:
    - id: deployment
      template:
        apiVersion: apps/v1
        kind: Deployment
        metadata:
          name: ${schema.spec.name}
        spec:
          replicas: ${schema.spec.replicas}
    - id: service
      template:
        apiVersion: v1
        kind: Service
        metadata:
          name: ${schema.spec.name}
```

Quando os usuários criam instâncias do recurso personalizado `WebApplication`, o kro cria automaticamente os recursos correspondentes de Deployment e Service, gerenciando o ciclo de vida junto com o do recurso personalizado.

Integração com outras funcionalidades gerenciadas do EKS

O kro pode ser integrado a outras funcionalidades gerenciadas do EKS.

- AWS Controllers for Kubernetes (ACK): use o kro para compor recursos do ACK em abstrações de alto nível, simplificando o gerenciamento de recursos da AWS.
- Argo CD: use o Argo CD para gerenciar a implantação de recursos personalizados do kro em diversos clusters, possibilitando fluxos de trabalho GitOps para os blocos de criação de plataforma e de pilhas de aplicações.

Conceitos básicos do kro

Para começar a usar a funcionalidade do EKS para o kro:

1. Crie um recurso da funcionalidade do kro em seu cluster do EKS por meio do Console da AWS, da AWS CLI ou da ferramenta de infraestrutura como código de sua preferência.

2. Crie ResourceGraphDefinitions (RGDs) que definam APIs personalizadas e composições de recursos.
3. Aplique instâncias dos recursos personalizados para realizar o provisionamento e o gerenciamento dos recursos subjacentes do Kubernetes e da AWS.

Criação de uma funcionalidade do kro

Este tópico explica como criar uma funcionalidade do kro no cluster do Amazon EKS.

Pré-requisitos

Antes de criar uma funcionalidade do kro, certifique-se de ter:

- Um cluster do Amazon EKS existente executando uma versão compatível do Kubernetes (há suporte para todas as versões nos períodos de suporte padrão e estendido)
- Permissões do IAM suficientes para criar recursos de funcionalidades em clusters do EKS
- (Para a CLI ou o eksctl) A ferramenta de CLI apropriada instalada e configurada

> **Note**
>
> Ao contrário do ACK e do Argo CD, o kro não requer permissões do IAM adicionais além da política de confiança. O kro opera inteiramente no cluster e não faz chamadas de API da AWS. Contudo, ainda é necessário fornecer um perfil do IAM para a funcionalidade com a política de confiança apropriada. Para obter informações sobre como configurar as permissões de RBAC do Kubernetes para o kro, consulte the section called “Configurar permissões”.

Escolha da ferramenta

É possível criar uma funcionalidade do kro por meio do Console de gerenciamento da AWS, da AWS CLI ou do eksctl:

- the section called “Console”: use o Console para obter uma experiência guiada
- the section called “ AWS CLI”: use a AWS CLI para obter scripts e automação
- the section called “eksctl”: use o eksctl para obter uma experiência nativa do Kubernetes

O que ocorre durante a criação de uma funcionalidade do kro

Ao criar uma funcionalidade do kro:

1. O EKS cria o serviço da funcionalidade do kro e o configura para monitorar e gerenciar recursos no cluster
2. As definições de recursos personalizados (CRDs, na sigla em inglês) são instaladas no cluster
3. A funcionalidade assume o perfil do IAM para a funcionalidade, que foi fornecido por você (e é usado somente para a relação de confiança)
4. O kro começa a monitorar recursos da `ResourceGraphDefinition` e as respectivas instâncias
5. O status da funcionalidade é alterado de `CREATING` para `ACTIVE`

Uma vez ativa, é possível criar ResourceGraphDefinitions para definir APIs personalizadas e criar instâncias dessas APIs.

Próximas etapas

Após criar a funcionalidade do kro:

- the section called “Conceitos do kro”: compreenda os conceitos do kro e a composição de recursos
- the section called “Conceitos do kro”: saiba mais sobre o SimpleSchema, expressões CEL e padrões de composição de recursos

Criação de uma funcionalidade do kro por meio do Console

Este tópico descreve como criar uma funcionalidade do kro (Kube Resource Orchestrator) usando o Console de gerenciamento da AWS.

Criação da funcionalidade do kro

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do cluster para abrir a página de detalhes do cluster.
3. Escolha a guia Funcionalidades.
4. No painel de navegação à esquerda, escolha kro (Kube Resource Orchestrator).
5. Selecione Criar funcionalidade do kro.
6. No perfil da funcionalidade do IAM:

- Se você já tiver um perfil da funcionalidade do IAM, selecione-o no menu suspenso
- Se a criação de um perfil for necessária, escolha Criar perfil do kro

  Isso abre o console do IAM em uma nova guia com a política de confiança preenchida previamente. O perfil não requer permissões do IAM adicionais, uma vez que o kro opera inteiramente no cluster.

  Após criar o perfil, retorne ao console do EKS e o perfil será selecionado automaticamente.

  > **Note**
  >
  > Ao contrário do ACK e do Argo CD, o kro não requer permissões do IAM adicionais além da política de confiança. O kro opera inteiramente no cluster e não faz chamadas de API da AWS.

7. Escolha Criar.

O processo de criação da funcionalidade é iniciado.

Verificação da ativação da funcionalidade

1. Na guia Funcionalidades, visualize o status da funcionalidade do kro.
2. Aguarde até que o status mude de `CREATING` para `ACTIVE`.
3. Assim que estiver com o status ativo, a funcionalidade estará pronta para uso.

Para obter informações sobre os status das funcionalidades e sobre a solução de problemas, consulte the section called “Como trabalhar com funcionalidades”.

Concessão de permissões para o gerenciamento de recursos do Kubernetes

Por padrão, o kro pode apenas criar e gerenciar ResourceGraphDefinitions e suas respectivas instâncias. Para permitir que o kro crie e gerencie os recursos subjacentes do Kubernetes definidos em suas ResourceGraphDefinitions, associe a política de acesso `AmazonEKSClusterAdminPolicy` à entrada de acesso da funcionalidade.

1. No console do EKS, acesse a guia Acesso do cluster.
2. Em Entradas de acesso, localize a entrada referente ao perfil da funcionalidade para o kro (ela terá o ARN do perfil que você criou anteriormente).

3. Selecione a entrada de acesso para visualizar os detalhes.
4. Na seção Políticas de acesso, escolha Associar política de acesso.
5. Selecione `AmazonEKSClusterAdminPolicy` na lista de políticas.
6. Em Escopo de acesso, selecione Cluster.
7. Selecione Associar.

> **Important**
>
> A política `AmazonEKSClusterAdminPolicy` concede permissões abrangentes para criar e gerenciar todos os recursos do Kubernetes e tem como objetivo agilizar os primeiros passos. Para uso em ambientes de produção, crie políticas de RBAC mais restritivas que concedam somente as permissões necessárias para os recursos específicos que as ResourceGraphDefinitions gerenciarão. Para obter orientações sobre como configurar permissões de privilégio mínimo, consulte the section called “Configurar permissões” e the section called “Considerações para funcionalidades do EKS”.

Verificação da disponibilidade de recursos personalizados

Após a funcionalidade estar ativa, verifique se os recursos personalizados do kro estão disponíveis no cluster.

Utilizar o console

1. Navegue até o cluster no console do Amazon EKS
2. Escolha a guia Recursos
3. Escolha Extensões
4. Escolha CustomResourceDefinitions

O tipo de recurso `ResourceGraphDefinition` deve aparecer na lista apresentada.

Uso do kubectl

```
kubectl api-resources | grep kro.run
```

O tipo de recurso `ResourceGraphDefinition` deve aparecer na lista apresentada.

Próximas etapas

- the section called “Conceitos do kro”: compreenda os conceitos do kro e a composição de recursos
- the section called “Conceitos do kro”: saiba mais sobre o SimpleSchema, expressões CEL e padrões de composição
- the section called “Como trabalhar com funcionalidades”: gerencie os recursos da funcionalidade do kro

Criação de uma funcionalidade do kro por meio da AWS CLI

Este tópico descreve como criar uma funcionalidade do kro (Kube Resource Orchestrator) usando a AWS CLI.

Pré-requisitos

- AWS CLI: versão `2.12.3` ou em versões posteriores. Para verificar a versão, execute `aws --version`. Para obter mais informações, consulte Instalação no Guia do Usuário da Interface de Linha de Comando AWS.
- **`kubectl`** – uma ferramenta de linha de comando para trabalhar com clusters do Kubernetes. Para obter mais informações, consulte the section called “Configuração do kubectl e eksctl”.

Etapa 1: criação de um perfil da funcionalidade do IAM

Crie um arquivo de política de confiança:

```
cat > kro-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
```

```
}
EOF
```

Crie o perfil do IAM:

```
aws iam create-role \
  --role-name KROCapabilityRole \
  --assume-role-policy-document file://kro-trust-policy.json
```

> **Note**
>
> Ao contrário do ACK e do Argo CD, o kro não requer permissões do IAM adicionais. O kro opera inteiramente no cluster e não faz chamadas de API da AWS. O perfil é exigido exclusivamente para configurar a relação de confiança com o serviço de funcionalidades do EKS.

Etapa 2: criação da funcionalidade do kro

Crie o recurso da funcionalidade do kro no cluster. Substitua *region-code* pela região da AWS em que seu cluster está localizado (por exemplo, `us-west-2`) e *my-cluster* pelo nome do seu cluster.

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --type KRO \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/KROCapabilityRole \
  --delete-propagation-policy RETAIN
```

O comando é concluído de imediato, mas a funcionalidade demora algum tempo para se tornar ativa, conforme o EKS cria a infraestrutura e os componentes necessários para a funcionalidade. O EKS instalará as definições de recursos personalizados do Kubernetes relacionadas a essa funcionalidade no cluster durante o processo de criação.

> **Note**
>
> Caso ocorra um erro indicando a inexistência do cluster ou falta de permissões, verifique o seguinte:

- Se o nome do cluster está correto
- Se a AWS CLI está configurada para a região correta
- Se você tem as permissões do IAM obrigatórias

Etapa 3: verificação da ativação da funcionalidade

Aguarde até que a funcionalidade se torne ativa. Substitua *region-code* pela região AWS em que seu cluster se encontra e substitua *my-cluster* pelo nome do seu cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --query 'capability.status' \
  --output text
```

A funcionalidade estará pronta assim que o status mostrar `ACTIVE`.

Também é possível visualizar os detalhes completos da funcionalidade:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro
```

Etapa 4: concessão de permissões para o gerenciamento de recursos do Kubernetes

Por padrão, o kro pode apenas criar e gerenciar ResourceGraphDefinitions e suas respectivas instâncias. Para permitir que o kro crie e gerencie os recursos subjacentes do Kubernetes definidos em suas ResourceGraphDefinitions, associe a política de acesso `AmazonEKSClusterAdminPolicy` à entrada de acesso da funcionalidade.

Obtenha o ARN do perfil da funcionalidade:

```
CAPABILITY_ROLE_ARN=$(aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
```

```
  --capability-name my-kro \
  --query 'capability.roleArn' \
  --output text)
```

Associe a política de administrador do cluster:

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name my-cluster \
  --principal-arn $CAPABILITY_ROLE_ARN \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

Important

A política `AmazonEKSClusterAdminPolicy` concede permissões abrangentes para criar e gerenciar todos os recursos do Kubernetes e tem como objetivo agilizar os primeiros passos. Para uso em ambientes de produção, crie políticas de RBAC mais restritivas que concedam somente as permissões necessárias para os recursos específicos que as ResourceGraphDefinitions gerenciarão. Para obter orientações sobre como configurar permissões de privilégio mínimo, consulte the section called “Configurar permissões” e the section called “Considerações para funcionalidades do EKS”.

Etapa 5: verificação da disponibilidade de recursos personalizados

Após a funcionalidade estar ativa, verifique se os recursos personalizados do kro estão disponíveis no cluster:

```
kubectl api-resources | grep kro.run
```

O tipo de recurso `ResourceGraphDefinition` deve aparecer na lista apresentada.

Próximas etapas

- the section called “Conceitos do kro”: compreenda os conceitos do kro e a composição de recursos
- the section called “Conceitos do kro”: saiba mais sobre o SimpleSchema, expressões CEL e padrões de composição

- the section called “Como trabalhar com funcionalidades”: gerencie os recursos da funcionalidade do kro

Criação de uma funcionalidade do kro por meio do eksctl

Este tópico descreve como criar uma funcionalidade do kro (Kube Resource Orchestrator) usando o eksctl.

> **Note**
>
> Para realizar as etapas seguintes, você deve estar utilizando o eksctl na versão `0.220.0` ou em versões posteriores. Para verificar a versão, execute `eksctl version`.

Etapa 1: criação de um perfil da funcionalidade do IAM

Crie um arquivo de política de confiança:

```
cat > kro-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Crie o perfil do IAM:

```
aws iam create-role \
  --role-name KROCapabilityRole \
```

```
  --assume-role-policy-document file://kro-trust-policy.json
```

> **Note**
>
> Ao contrário do ACK e do Argo CD, o kro não requer permissões do IAM adicionais além da política de confiança. O kro opera inteiramente no cluster e não faz chamadas de API da AWS.

Etapa 2: criação da funcionalidade do kro

Crie a funcionalidade do kro por meio do eksctl. Substitua *region-code* pela região AWS em que seu cluster se encontra e substitua *my-cluster* pelo nome do seu cluster.

```
eksctl create capability \
  --region region-code \
  --cluster my-cluster \
  --name my-kro \
  --type KRO \
  --role-arn arn:aws:iam::[.replaceable]111122223333:role/KROCapabilityRole
```

O comando é executado de forma imediata, mas a funcionalidade demora algum tempo para se tornar ativa.

Etapa 3: verificação da ativação da funcionalidade

Verifique o status da funcionalidade. Substitua *region-code* pela região AWS em que seu cluster se encontra e substitua *my-cluster* pelo nome do seu cluster.

```
eksctl get capability \
  --region region-code \
  --cluster my-cluster \
  --name my-kro
```

A funcionalidade estará pronta assim que o status mostrar `ACTIVE`.

Etapa 4: concessão de permissões para o gerenciamento de recursos do Kubernetes

Por padrão, o kro pode apenas criar e gerenciar ResourceGraphDefinitions e suas respectivas instâncias. Para permitir que o kro crie e gerencie os recursos subjacentes do

Kubernetes definidos em suas ResourceGraphDefinitions, associe a política de acesso `AmazonEKSClusterAdminPolicy` à entrada de acesso da funcionalidade.

Obtenha o ARN do perfil da funcionalidade:

```
CAPABILITY_ROLE_ARN=$(aws eks describe-capability \
  --region region-code \
  --cluster my-cluster \
  --name my-kro \
  --query 'capability.roleArn' \
  --output text)
```

Associe a política de administrador do cluster:

```
aws eks associate-access-policy \
  --region region-code \
  --cluster my-cluster \
  --principal-arn $CAPABILITY_ROLE_ARN \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

> **Important**
>
> A política `AmazonEKSClusterAdminPolicy` concede permissões abrangentes para criar e gerenciar todos os recursos do Kubernetes e tem como objetivo agilizar os primeiros passos. Para uso em ambientes de produção, crie políticas de RBAC mais restritivas que concedam somente as permissões necessárias para os recursos específicos que as ResourceGraphDefinitions gerenciarão. Para obter orientações sobre como configurar permissões de privilégio mínimo, consulte the section called “Configurar permissões” e the section called “Considerações para funcionalidades do EKS”.

Etapa 5: verificação da disponibilidade de recursos personalizados

Após a funcionalidade estar ativa, verifique se os recursos personalizados do kro estão disponíveis no cluster:

```
kubectl api-resources | grep kro.run
```

O tipo de recurso `ResourceGraphDefinition` deve aparecer na lista apresentada.

Próximas etapas

- the section called “Conceitos do kro”: compreenda os conceitos do kro e a composição de recursos
- the section called “Conceitos do kro”: saiba mais sobre o SimpleSchema, expressões CEL e padrões de composição
- the section called “Como trabalhar com funcionalidades”: gerencie os recursos da funcionalidade do kro

Conceitos do kro

Com o kro, as equipes responsáveis pela plataforma podem criar APIs personalizadas do Kubernetes que agrupam vários recursos em abstrações de alto nível. Este tópico apresenta um exemplo prático e, em seguida, explica os conceitos fundamentais que você precisa entender ao trabalhar com a funcionalidade do EKS para o kro.

Conceitos básicos do kro

Após criar a funcionalidade kro (consulte the section called “Criação da funcionalidade do kro”), você poderá iniciar a criação de APIs personalizadas usando ResourceGraphDefinitions no cluster.

A seguir, apresentamos um exemplo completo que cria uma abstração simples de uma aplicação web:

```
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: webapplication
spec:
  schema:
    apiVersion: v1alpha1
    kind: WebApplication
    group: kro.run
    spec:
      name: string | required=true
      image: string | default="nginx:latest"
      replicas: integer | default=3
  resources:
  - id: deployment
    template:
      apiVersion: apps/v1
```

```
      kind: Deployment
      metadata:
        name: ${schema.spec.name}
      spec:
        replicas: ${schema.spec.replicas}
        selector:
          matchLabels:
            app: ${schema.spec.name}
        template:
          metadata:
            labels:
              app: ${schema.spec.name}
          spec:
            containers:
            - name: app
              image: ${schema.spec.image}
              ports:
              - containerPort: 80
  - id: service
    template:
      apiVersion: v1
      kind: Service
      metadata:
        name: ${schema.spec.name}
      spec:
        selector:
          app: ${schema.spec.name}
        ports:
        - protocol: TCP
          port: 80
          targetPort: 80
```

Após aplicar esta ResourceGraphDefinition, as equipes responsáveis pela aplicação poderão criar aplicações web empregando a API simplificada:

```
apiVersion: kro.run/v1alpha1
kind: WebApplication
metadata:
  name: my-app
spec:
  name: my-app
  replicas: 5
```

O kro cria automaticamente “Deployment” e “Service” com a configuração apropriada. Como `image` não foi especificado, o valor padrão `nginx:latest`, definido no esquema, é usado.

Conceitos principais

> **Important**
>
> O kro valida as ResourceGraphDefinitions no momento da criação, e não no runtime. Quando você cria uma RGD, o kro valida a sintaxe CEL, realiza a checagem de tipos das expressões em relação aos esquemas reais do Kubernetes, verifica a existência dos campos e detecta dependências circulares. Isso significa que os erros são capturados imediatamente ao criar a RGD, antes mesmo que qualquer instância seja implantada.

ResourceGraphDefinition

Uma ResourceGraphDefinition (RGD) define uma API personalizada do Kubernetes ao especificar:

- Esquema: a estrutura da API usando o formato SimpleSchema (que engloba nomes de campo, tipos, valores padrão e validação)
- Recursos: modelos para os recursos subjacentes do Kubernetes ou da AWS que serão criados
- Dependências: maneira como os recursos se relacionam entre si (detectadas automaticamente com base nas referências de campos)

Quando você aplica uma RGD, o kro registra uma nova definição de recurso personalizado (CRD, na sigla em inglês) no cluster. Em seguida, as equipes responsáveis pela aplicação podem criar instâncias da API personalizada, e o kro assume a criação e o gerenciamento de todos os recursos subjacentes.

Para obter mais informações, consulte ResourceGraphDefinition Overview na documentação do kro.

Formato SimpleSchema

O SimpleSchema oferece uma maneira simplificada de definir esquemas de API sem a necessidade da OpenAPI completa:

```
schema:
  apiVersion: v1alpha1
```

```
  kind: Database
  spec:
    name: string | required=true description="Database name"
    size: string | default="small" enum=small,medium,large
    replicas: integer | default=1 minimum=1 maximum=5
```

O SimpleSchema fornece suporte para os tipos `string`, `integer`, `boolean` e `number`, com restrições como `required`, `default`, `minimum`/`maximum`, `enum` e `pattern`.

Para obter mais informações, consulte a seção SimpleSchema na documentação do kro.

Expressões CEL

O kro utiliza a Common Expression Language (CEL) para referenciar valores dinamicamente e adicionar lógica condicional. As expressões CEL são envolvidas por `${` e `}`, e podem ser usadas de duas formas:

Expressões autônomas: o valor total do campo é composto por uma única expressão:

```
spec:
  replicas: ${schema.spec.replicaCount}  # Expression returns integer
  labels: ${schema.spec.labelMap}        # Expression returns object
```

O resultado da expressão substitui o valor total do campo e deve corresponder ao tipo esperado para o campo.

Modelos de string: uma ou mais expressões incorporadas em uma string:

```
metadata:
  name: "${schema.spec.prefix}-${schema.spec.name}"  # Multiple expressions
  annotation: "Created by ${schema.spec.owner}"      # Single expression in string
```

Todas as expressões nos modelos de string devem retornar strings. Use `string()` para converter outros tipos: `"replicas-${string(schema.spec.count)}"`.

Referências de campos: acesse os valores de especificação da instância usando `schema.spec`:

```
template:
  metadata:
    name: ${schema.spec.name}-deployment
```

```
    namespace: ${schema.metadata.namespace}  # Can also reference metadata
  spec:
    replicas: ${schema.spec.replicas}
```

Acesso opcional ao campo: use o operador ? para campos que podem não existir:

```
# For ConfigMaps or Secrets with unknown structure
value: ${configmap.data.?DATABASE_URL}

# For optional status fields
ready: ${deployment.status.?readyReplicas > 0}
```

Se o campo não existir, a expressão retornará `null` em vez de gerar uma falha.

Recursos condicionais: inclua recursos somente quando condições específicas forem atendidas:

```
resources:
- id: ingress
  includeWhen:
    - ${schema.spec.enableIngress == true}
  template:
    # ... ingress configuration
```

O `includeWhen` campo aceita uma lista de expressões booleanas. Todas as condições devem ser verdadeiras para que o recurso seja criado. No momento, `includeWhen` permite referenciar apenas os campos contidos na `schema.spec`.

Transformações: altere valores usando funções e operadores ternários:

```
template:
  spec:
    resources:
      requests:
        memory: ${schema.spec.size == "small" ? "512Mi" : "2Gi"}

    # String concatenation
    image: ${schema.spec.registry + "/" + schema.spec.imageName}

    # Type conversion
    port: ${string(schema.spec.portNumber)}
```

Referências entre recursos: referencie valores provenientes de outros recursos:

```
resources:
- id: bucket
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: Bucket
    spec:
      name: ${schema.spec.name}-data

- id: configmap
  template:
    apiVersion: v1
    kind: ConfigMap
    data:
      BUCKET_NAME: ${bucket.spec.name}
      BUCKET_ARN: ${bucket.status.ackResourceMetadata.arn}
```

Quando você referencia outro recurso em uma expressão CEL, isso cria automaticamente uma dependência. O kro assegura que o recurso de referência seja estabelecido antes dos demais recursos.

Para obter mais informações, consulte CEL Expressions na documentação do kro.

Dependências entre recursos

O kro infere automaticamente as dependências a partir das expressões CEL. Em vez de ditar a ordem, você apenas descreve como os recursos se relacionam. Se um recurso faz referência a outro usando uma expressão CEL, o kro cria uma dependência e determina a ordem de criação correta.

```
resources:
- id: bucket
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: Bucket
    spec:
      name: ${schema.spec.name}-data

- id: notification
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: BucketNotification
```

```
    spec:
      bucket: ${bucket.spec.name}  # Creates dependency: notification depends on bucket
```

A expressão `${bucket.spec.name}` cria uma dependência. O kro desenvolve um grafo acíclico direcionado (DAG, na sigla em inglês) e as dependências e, em seguida, calcula uma ordem topológica para a criação.

Ordem de criação: os recursos são criados em ordem topológica (ou seja, primeiro há a criação das dependências).

Criação paralela: recursos sem dependências são criados simultaneamente.

Ordem de exclusão: os recursos são excluídos em ordem topológica inversa (ou seja, primeiro os recursos dependentes).

Dependências circulares: não são permitidas. O kro rejeita as ResourceGraphDefinitions com dependências circulares durante a validação.

Para conferir a ordem de criação definida pelo sistema, use:

```
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.topologicalOrder}'
```

Para obter mais informações, consulte Graph inference na documentação do kro.

Composição com o ACK

O kro funciona perfeitamente com a funcionalidade do EKS para o ACK para compor recursos da AWS com recursos do Kubernetes:

```
resources:
# Create {aws} S3 bucket with ACK
- id: bucket
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: Bucket
    spec:
      name: ${schema.spec.name}-files

# Inject bucket details into Kubernetes ConfigMap
- id: config
```

```
  template:
    apiVersion: v1
    kind: ConfigMap
    data:
      BUCKET_NAME: ${bucket.spec.name}
      BUCKET_ARN: ${bucket.status.ackResourceMetadata.arn}

# Use ConfigMap in application deployment
- id: deployment
  template:
    apiVersion: apps/v1
    kind: Deployment
    spec:
      template:
        spec:
          containers:
          - name: app
            envFrom:
            - configMapRef:
                name: ${config.metadata.name}
```

Com esse padrão, você consegue criar recursos na AWS, obter detalhes importantes (como ARNs, URLs e endpoints) e inseri-los diretamente na configuração da aplicação, tratando tudo como um único conjunto.

Para obter mais padrões de composição e exemplos avançados, consulte the section called “Considerações”.

Próximas etapas

- the section called “Considerações”: aprenda sobre padrões específicos do EKS, do RBAC e da integração com o ACK e com o Argo CD
- Documentação do kro: acesse a documentação abrangente do kro, incluindo expressões CEL avançadas, padrões de validação e solução de problemas

Configuração de permissões do kro

Ao contrário do ACK e do Argo CD, o kro não requer permissões do IAM. O kro opera inteiramente no cluster do Kubernetes e não faz chamadas de API da AWS. Controle o acesso aos recursos do kro usando o RBAC padrão do Kubernetes.

Funcionamento de permissões com o kro

O kro usa dois tipos de recursos do Kubernetes com escopos diferentes:

ResourceGraphDefinitions: recursos com escopo de cluster para a definição de APIs personalizadas. Geralmente, esses recursos são gerenciados por equipes responsáveis pela plataforma que projetam e mantêm os padrões organizacionais.

Instâncias: recursos personalizados com escopo de namespace criados com base em ResourceGraphDefinitions. Podem ser criados por equipes responsáveis pela aplicação com as permissões de RBAC apropriadas.

Por padrão, a funcionalidade do kro tem permissões para gerenciar ResourceGraphDefinitions e as respectivas instâncias por meio da política de entrada de acesso `AmazonEKSKROPolicy`. No entanto, o kro requer permissões adicionais para criar e gerenciar os recursos subjacentes do Kubernetes definidos em suas ResourceGraphDefinitions (como Deployments, Services ou recursos do ACK). Você deve conceder essas permissões por meio de políticas de entrada de acesso ou de RBAC do Kubernetes. Para detalhes sobre a concessão dessas permissões, consulte a seção Permissões de recursos arbitrários do kro.

Permissões da equipe responsável pela plataforma

As equipes responsáveis pela plataforma precisam de permissões para criar e gerenciar ResourceGraphDefinitions.

Exemplo de ClusterRole para as equipes responsáveis pela plataforma:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kro-platform-admin
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
  verbs: ["*"]
```

Vincule aos membros da equipe responsável pela plataforma:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
```

```
  name: platform-team-kro-admin
subjects:
- kind: Group
  name: platform-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: kro-platform-admin
  apiGroup: rbac.authorization.k8s.io
```

Permissões da equipe responsável pela aplicação

As equipes responsáveis pela aplicação precisam de permissões para criar instâncias de recursos personalizados em seus namespaces.

Exemplo de perfil para as equipes responsáveis pela aplicação:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: kro-app-developer
  namespace: my-app
rules:
- apiGroups: ["kro.run"]
  resources: ["webapps", "databases"]
  verbs: ["create", "get", "list", "update", "delete", "patch"]
```

Vincule aos membros da equipe responsável pela aplicação:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-team-kro-developer
  namespace: my-app
subjects:
- kind: Group
  name: app-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: kro-app-developer
  apiGroup: rbac.authorization.k8s.io
```

> **Note**
>
> O grupo de API no perfil (`kro.run` neste exemplo) deve corresponder à `apiVersion` definida no esquema da ResourceGraphDefinition.

Acesso somente leitura.

Conceda acesso somente de leitura para a visualização de ResourceGraphDefinitions e de instâncias, sem permissões de modificação.

ClusterRole somente de leitura:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kro-viewer
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
  verbs: ["get", "list", "watch"]
```

Perfil somente de leitura para as instâncias:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: kro-instance-viewer
  namespace: my-app
rules:
- apiGroups: ["kro.run"]
  resources: ["webapps", "databases"]
  verbs: ["get", "list", "watch"]
```

Acesso a vários namespaces

Conceda às equipes responsáveis pela aplicação acesso a vários namespaces usando ClusterRoles com RoleBindings.

ClusterRole para acesso a vários namespaces:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kro-multi-namespace-developer
rules:
- apiGroups: ["kro.run"]
  resources: ["webapps"]
  verbs: ["create", "get", "list", "update", "delete"]
```

Vincule a namespaces específicos:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-team-dev-access
  namespace: development
subjects:
- kind: Group
  name: app-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: kro-multi-namespace-developer
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-team-staging-access
  namespace: staging
subjects:
- kind: Group
  name: app-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: kro-multi-namespace-developer
  apiGroup: rbac.authorization.k8s.io
```

Práticas recomendadas

Princípio do privilégio mínimo: conceda somente as permissões mínimas necessárias para que cada equipe execute suas responsabilidades.

Use grupos em vez de usuários individuais: vincule perfis a grupos, em vez de usuários individuais, para facilitar o gerenciamento.

Separação de responsabilidades entre equipes de plataforma e aplicação: equipes responsáveis pela plataforma gerenciam ResourceGraphDefinitions, enquanto equipes responsáveis pela aplicação gerenciam instâncias.

Isolamento de namespace: use namespaces para isolar diferentes equipes ou ambientes, com o RBAC controlando o acesso a cada namespace.

Acesso somente de leitura para auditoria: forneça acesso somente de leitura para equipes responsáveis pela segurança e pela conformidade para fins de auditoria.

Próximas etapas

- the section called “Conceitos do kro”: compreenda os conceitos do kro e a composição de recursos
- the section called “Conceitos do kro”: compreenda sobre o SimpleSchema, expressões CEL e padrões de composição
- the section called “Considerações para funcionalidades do EKS”: analise as práticas recomendadas de segurança para as funcionalidades

Considerações sobre o kro para o EKS

Este tópico aborda considerações importantes para o uso da funcionalidade do EKS para o kro, incluindo quando utilizar a composição de recursos, os padrões de RBAC e a integração com outras funcionalidades do EKS.

Situações para o uso do kro

O kro foi projetado para criar padrões de infraestrutura reutilizáveis e APIs personalizadas que simplificam o gerenciamento de recursos complexos.

Use o kro quando for necessário:

- Criar plataformas de autoatendimento com APIs simplificadas para as equipes destinadas ao trabalho com APIs aplicação
- Padronizar padrões de infraestrutura entre diferentes equipes (banco de dados + backup + monitoramento)
- Gerenciar dependências de recursos e realizar a transferência de valores entre recursos
- Desenvolver abstrações personalizadas que ocultam a complexidade da implementação
- Compor diversos recursos do ACK em blocos de criação de nível superior
- Habilitar fluxos de trabalho de GitOps para pilhas de infraestrutura complexas

Não use kro ao:

- Gerenciar recursos simples e isolados (empregue recursos do Kubernetes ou do ACK diretamente)
- Necessitar de lógica dinâmica de runtime (visto que o kro é declarativo, e não imperativo)
- Lidar com recursos que não têm dependências nem configurações compartilhadas

O kro se destaca na criação de “caminhos pavimentados”, que são padrões opinativos e reutilizáveis que facilitam a implantação correta de infraestruturas complexas pelas equipes.

Padrões de RBAC

O kro permite a separação de responsabilidades entre as equipes responsáveis pela plataforma, que criam as ResourceGraphDefinitions, e as equipes responsáveis pela aplicação, que criam as instâncias.

Responsabilidades da equipe responsável pela plataforma

As equipes responsáveis pela plataforma criam e mantêm as ResourceGraphDefinitions (RGDs) que definem as APIs personalizadas.

Permissões necessárias:

- Criar, atualizar e excluir ResourceGraphDefinitions
- Gerenciar tipos de recursos subjacentes (como Deployments, Services e recursos do ACK)
- Acessar a todos os namespaces nos quais as RGDs serão aplicadas

Exemplo de ClusterRole para a equipe responsável pela plataforma:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: platform-kro-admin
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
  verbs: ["*"]
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["get", "list", "watch"]
```

Para obter a configuração detalhada de RBAC, consulte the section called “Configurar permissões”.

Responsabilidades da equipe responsável pela aplicação

As equipes responsáveis pela aplicação criam instâncias de recursos personalizados definidos por RGDs, sem a necessidade de compreender a complexidade subjacente.

Permissões necessárias:

- Criar, atualizar e excluir instâncias de recursos personalizados
- Ter um acesso de leitura ao próprio namespace
- Não conseguir acessar os recursos subjacentes ou as RGDs

Benefícios:

- As equipes usam APIs simplificadas de alto nível
- As equipes responsáveis pela plataforma detêm o controle dos detalhes de implementação
- Risco reduzido de erros de configuração
- Integração mais rápida para os novos membros da equipe

Integração com outras funcionalidades do EKS

Composição de recursos do ACK

O kro torna-se especialmente eficiente ao ser integrado ao ACK para o desenvolvimento de padrões de infraestrutura.

Padrões comuns:

- Aplicação com armazenamento: bucket do S3 + fila do SQS + configuração de notificação
- Pilha de banco de dados: instância do RDS + grupo de parâmetros + grupo de segurança + segredo do Secrets Manager
- Rede: VPC + sub-redes + tabelas de rotas + grupos de segurança + gateways NAT
- Computação com armazenamento: instância do EC2 + volumes do EBS + perfil de instância do IAM

O kro gerencia a ordenação de dependências, transfere valores entre recursos (como ARNs e strings de conexão) e gerencia o ciclo de vida completo como uma unidade única.

Para obter exemplos da composição de recursos do ACK, consulte the section called “Conceitos do ACK”.

GitOps com Argo CD

Use a funcionalidade do EKS para o Argo CD com a finalidade de implantar tanto RGDs quanto instâncias usando repositórios do Git.

Organização de repositórios:

- Repositório de plataforma: contém as ResourceGraphDefinitions gerenciadas pela equipe responsável pela plataforma
- Repositórios de aplicação: contêm instâncias de recursos personalizados gerenciadas pelas equipes responsáveis pela aplicação
- Repositório compartilhado: contém tanto RGDs quanto instâncias para organizações de menor porte

Considerações:

- Implante as RGDs antes das instâncias (os ciclos de sincronização do Argo CD podem ser úteis)
- Use Projects distintos do CD Argo para as equipes responsáveis pela plataforma e pela aplicação
- A equipe responsável pela plataforma controla o acesso ao repositório de RGDs
- As equipes responsáveis pela aplicação têm acesso somente de leitura às definições de RGDs

Para saber mais sobre o Argo CD, consulte the section called “Como trabalhar com o Argo CD”.

Organização de ResourceGraphDefinitions

Organize as RGDs por finalidade, complexidade e propriedade.

Por finalidade:

- Infraestrutura: pilhas de banco de dados, rede e armazenamento
- Aplicação: aplicativos web, APIs e trabalhos em lote
- Plataforma: serviços compartilhados, monitoramento e registro em log

Por complexidade:

- Simples: dois a três recursos com o mínimo de dependências
- Intermediária: cinco a dez recursos com algumas dependências
- Complexa: mais de dez recursos com dependências complexas

Convenções de nomenclatura:

- Use nomes descritivos: `webapp-with-database` e `s3-notification-queue`
- Inclua a versão no nome para alterações incompatíveis: `webapp-v2`
- Use prefixos consistentes para RGDs relacionadas: `platform-` e `app-`

Estratégia de namespace:

- As RGDs têm escopo de cluster (não pertencem a um namespace)
- As instâncias pertencem a um namespace
- Use seletores de namespace nas RGDs para controlar em qual local as instâncias podem ser criadas

Versionamento e atualizações

Planeje a evolução das RGDs e a migração de instâncias.

Atualizações de RGD:

- Alterações compatíveis: atualize a RGD diretamente (adicione campos opcionais e novos recursos com includeWhen)

- Alterações incompatíveis: crie uma nova RGD com um nome diferente (como webapp-v2)
- Descontinuação: sinalize as RGDs antigas com anotações e comunique o cronograma de migração

Migração de instâncias:

- Crie novas instâncias com a RGD atualizada
- Valide se as novas instâncias funcionam corretamente
- Exclua as instâncias antigas
- O kro gerencia as atualizações dos recursos subjacentes automaticamente

Práticas recomendadas:

- Teste as alterações de RGD primeiro em ambientes que não são de produção
- Use o versionamento semântico nos nomes das RGDs para alterações significativas
- Documente as alterações incompatíveis e os caminhos de migração
- Forneça exemplos de migração para as equipes responsáveis pela aplicação

Validação e testes

Valide as RGDs antes de implantá-las em ambientes de produção.

Estratégias de validação:

- Validação de esquema: o kro valida a estrutura da RGD automaticamente
- Instâncias de simulação: crie instâncias de teste em namespaces de desenvolvimento
- Testes de integração: verifique se os recursos compostos funcionam em conjunto
- Aplicação de políticas: use controladores de admissão para aplicar padrões organizacionais

Problemas comuns em testes:

- Dependências e ordenação de recursos
- Transferência de valores entre recursos (expressões CEL)
- Inclusão condicional de recursos (includeWhen)

- Propagação de status provenientes de recursos subjacentes
- Permissões de RBAC para criação de instâncias

Documentação original

Para obter informações detalhadas sobre o uso do kro:

- Getting Started with kro: criação de ResourceGraphDefinitions
- CEL Expressions: gravação de expressões CEL
- kro Guides: padrões avançados de composição
- Troubleshooting: solução de problemas e depuração

Próximas etapas

- the section called “Configurar permissões”: configure o RBAC para as equipes responsáveis pela plataforma e pela aplicação
- the section called “Conceitos do kro”: compreenda os conceitos do kro e o ciclo de vida dos recursos
- the section called “Solução de problemas”: solucione problemas do kro
- the section called “Conceitos do ACK”: saiba mais sobre os recursos do ACK para a composição
- the section called “Como trabalhar com o Argo CD”: implante RGDs e instâncias utilizando GitOps

Solução de problemas em funcionalidades do kro

Este tópico fornece orientações para a solução de problemas à funcionalidade do EKS para o kro, incluindo verificações de integridade da funcionalidade, permissões de RBAC, erros relacionados à expressão CEL e problemas de composição de recursos.

> **Note**
>
> As funcionalidades do EKS são totalmente gerenciadas e executadas de forma externa ao cluster. Você não tem acesso aos logs do controlador nem ao namespace `kro-system`. A solução de problemas se concentra na integridade da funcionalidade, na configuração de RBAC e no status dos recursos.

A funcionalidade está com o status ACTIVE, mas as ResourceGraphDefinitions não estão funcionando

Se a funcionalidade do kro apresentar o status `ACTIVE`, mas as ResourceGraphDefinitions não estiverem criando recursos subjacentes, verifique a integridade da funcionalidade, as permissões de RBAC e o status do recurso.

Verifique a integridade da funcionalidade:

Você pode visualizar problemas de integridade e de status da funcionalidade no console do EKS ou usando a AWS CLI.

Console do:

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do seu cluster.
3. Escolha a guia Observabilidade.
4. Escolha Monitorar cluster.
5. Escolha a guia Funcionalidades para visualizar a integridade e o status de todas as funcionalidades.

AWS CLI:

```
# View capability status and health
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro

# Look for issues in the health section
```

Causas comuns:

- Permissões de RBAC ausentes: o kro não tem permissões para criar recursos subjacentes do Kubernetes
- Expressões CEL inválidas: erros de sintaxe na ResourceGraphDefinition
- Dependências de recursos: recursos dependentes não estão prontos

- Validação do esquema: a instância não corresponde ao esquema da RGD

Verifique as permissões do RBAC:

```
# Check if capability has cluster admin policy
kubectl get accessentry -A | grep kro
```

Se a funcionalidade não tiver as permissões necessárias, associe a política `AmazonEKSClusterAdminPolicy` à entrada de acesso da funcionalidade do kro ou crie políticas de RBAC mais restritivas para uso em ambientes de produção. Para mais detalhes, consulte the section called “Configurar permissões”.

Verifique o status da ResourceGraphDefinition:

```
# List all RGDs
kubectl get resourcegraphdefinition

# Describe specific RGD
kubectl describe resourcegraphdefinition my-rgd

# Check for validation errors
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.conditions}'
```

As ResourceGraphDefinitions contam com três condições de status fundamentais:

- `ResourceGraphAccepted`: se a RGD foi aprovada na validação (por exemplo, por sintaxe CEL, verificação de tipos e existência de campos)
- `KindReady`: se a CRD para a API personalizada foi gerado e registrado
- `ControllerReady`: se o kro está monitorando ativamente as instâncias da API personalizada

Se `ResourceGraphAccepted` for `False`, verifique a mensagem da condição para erros de validação, como campos desconhecidos, incompatibilidades de tipo ou dependências circulares.

As instâncias foram criadas, porém os recursos subjacentes não estão sendo exibidos

Se existirem instâncias de recursos personalizados, mas os recursos subjacentes do Kubernetes (como Deployments, Services e ConfigMaps) não estiverem sendo criados, valide as permissões do kro e verifique se há erros na composição.

Verifique o status da instância:

```
# Describe the instance (replace with your custom resource kind and name)
kubectl describe custom-kind
         my-instance

# View instance events
kubectl get events --field-selector involvedObject.name=my-instance

# Check instance status conditions
kubectl get custom-kind
         my-instance -o jsonpath='{.status.conditions}'

# Check instance state
kubectl get custom-kind
         my-instance -o jsonpath='{.status.state}'
```

As instâncias têm um campo `state` que indica o status geral:

- `ACTIVE`: a instância está sendo executada com êxito
- `IN_PROGRESS`: a instância está sendo processada ou reconciliada
- `FAILED`: a instância apresentou falha na reconciliação
- `DELETING`: a instância está sendo excluída
- `ERROR`: ocorreu um erro durante o processamento

Além disso, as instâncias contam com quatro condições de status:

- `InstanceManaged`: finalizadores e rótulos estão configurados corretamente
- `GraphResolved`: gráfico de runtime criado e recursos resolvidos
- `ResourcesReady`: todos os recursos foram criados e estão prontos
- `Ready`: integridade geral da instância (torna-se `True` apenas quando todas as subcondições são `True`)

Monitore a condição `Ready` para determinar a integridade da instância. Se `Ready` for `False`, verifique as subcondições para identificar em qual fase ocorreu a falha.

Verifique as permissões do RBAC:

A funcionalidade do kro precisa de permissões para criar os recursos subjacentes do Kubernetes, definidos nas ResourceGraphDefinitions.

```
# Check if the capability has the AmazonEKSClusterAdminPolicy
kubectl get accessentry -A | grep kro
```

Se as permissões estiverem ausentes, associe a política `AmazonEKSClusterAdminPolicy` à entrada de acesso da funcionalidade do kro ou crie políticas de RBAC mais restritivas para uso em ambientes de produção. Para mais detalhes, consulte the section called “Configurar permissões”.

Erros em expressões CEL

Os erros em expressões CEL são detectados no momento da criação da ResourceGraphDefinition, e não quando as instâncias são criadas. O kro valida toda a sintaxe do CEL, verifica os tipos das expressões em relação aos esquemas do Kubernetes e verifica a existência de campos ao criar a RGD.

Erros comuns de validação em expressões CEL:

- Referência de campo indefinida: referência a um campo que não existe no esquema ou no recurso
- Incompatibilidade de tipo: a expressão retorna o tipo incorreto (por exemplo, string em que o número inteiro é esperado)
- Sintaxe inválida: ausência de colchetes, aspas ou operadores na expressão CEL
- Tipo de recurso desconhecido: referência a uma CRD que não está presente no cluster

Verifique o status de validação da RGD:

```
# Check if RGD was accepted
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.conditions[?
(@.type=="ResourceGraphAccepted")]}'

# View detailed validation errors
kubectl describe resourcegraphdefinition my-rgd
```

Se `ResourceGraphAccepted` for `False`, a mensagem da condição conterá o erro de validação.

Exemplos de expressões CEL válidas:

```
# Reference schema field
${schema.spec.appName}

# Conditional expression
${schema.spec.replicas > 1}

# String template (expressions must return strings)
name: "${schema.spec.appName}-service"

# Standalone expression (can be any type)
replicas: ${schema.spec.replicaCount}

# Resource reference
${deployment.status.availableReplicas}

# Optional field access (returns null if field doesn't exist)
${configmap.data.?DATABASE_URL}
```

As dependências de recursos não estão sendo resolvidas

O kro identifica dependências automaticamente por meio das expressões CEL, garantindo a criação dos recursos na sequência correta. Se os recursos não estiverem sendo criados conforme o esperado, verifique a ordem de dependência e a prontidão dos recursos.

Confira a ordem de criação definida pelo sistema:

```
# See the order kro will create resources
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.topologicalOrder}'
```

Isso mostra a ordem definida pelo sistema com base nas referências de expressões CEL entre os recursos.

Verifique a prontidão dos recursos:

```
# View instance status to see which resources are ready
kubectl get custom-kind
         my-instance -o jsonpath='{.status}'

# Check specific resource status
kubectl get deployment my-deployment -o jsonpath='{.status.conditions}'
```

Verifique as condições readyWhen (se houver):

O campo `readyWhen` é opcional. Se não for especificado, os recursos serão considerados prontos imediatamente após a criação. Se você definiu condições `readyWhen`, confirme se elas estão verificando o estado de prontidão do recurso de forma correta:

```
resources:
  - id: deployment
    readyWhen:
      - ${deployment.status.availableReplicas == deployment.spec.replicas}
```

Verifique os eventos do recurso:

```
# View events for the underlying resources
kubectl get events -n namespace --sort-by='.lastTimestamp'
```

Ocorrem falhas de validação do esquema

Se a criação de instâncias falhar devido a erros de validação do esquema, verifique se a instância corresponde aos requisitos de esquema da RGD.

Verifique os erros de validação:

```
# Attempt to create instance and view error
kubectl apply -f instance.yaml

# View existing instance validation status
kubectl describe custom-kind
         my-instance | grep -A 5 "Validation"
```

Problemas comuns de validação:

- Campos obrigatórios ausentes: a instância não fornece todos os campos obrigatórios do esquema
- Incompatibilidade de tipo: fornecimento de string em um local em que um número inteiro é esperado
- Valor de enumeração inválido: uso de um valor não constante na lista permitida
- Incompatibilidade de padrões: a string não corresponde ao padrão regex

Analise o esquema da RGD:

```
# View the schema definition
```

```
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.spec.schema}'
```

Certifique-se de que sua instância contenha todos os campos requeridos com os respectivos tipos configurados corretamente.

Próximas etapas

- the section called “Considerações”: obtenha considerações e práticas recomendadas para o kro
- the section called “Configurar permissões”: configure o RBAC para as equipes responsáveis pela plataforma e pela aplicação
- the section called “Conceitos do kro”: compreenda os conceitos do kro e o ciclo de vida dos recursos
- the section called “Solução de problemas das funcionalidades”: acesse orientações gerais para solução de problemas de funcionalidades

Comparação da funcionalidade do EKS para o kro em relação ao kro autogerenciado

A funcionalidade do EKS para o kro é equivalente à do kro autogerenciado, porém apresenta vantagens operacionais significativas. Para obter uma comparação geral das funcionalidades do EKS em relação às soluções autogerenciadas, consulte the section called “Considerações”.

A funcionalidade do EKS para o kro usa os mesmos controladores da versão original do kro e é totalmente compatível com a versão original do projeto. As ResourceGraphDefinitions, as expressões CEL e a composição de recursos funcionam de maneira idêntica. Para obter a documentação completa do kro e exemplos, consulte a documentação oficial do kro.

Caminho de migração

Você pode migrar de um kro autogerenciado para a funcionalidade gerenciada sem tempo de inatividade.

> **Important**
>
> Antes de realizar a migração, certifique-se de que o controlador kro autogerenciado esteja executando a mesma versão da funcionalidade do EKS para o kro. Verifique a versão da funcionalidade no console do EKS ou usando o comando `aws eks describe-`

`capability` e, em seguida, atualize a instalação do recurso autogerenciado para que seja compatível. Isso evita problemas de compatibilidade durante a migração.

1. Atualize o controlador do kro autogerenciado para usar `kube-system` para concessões de eleição de líder:

```
helm upgrade --install kro \
  oci://ghcr.io/awslabs/kro/kro-chart \
  --namespace kro \
  --set leaderElection.namespace=kube-system
```

   Isso transfere a concessão do controlador para o `kube-system`, permitindo que a funcionalidade gerenciada se coordene com ele.

2. Crie a funcionalidade do kro no cluster (consulte the section called “Criação da funcionalidade do kro”).
3. A funcionalidade gerenciada reconhece as ResourceGraphDefinitions e as instâncias existentes, assumindo o processo de reconciliação.
4. Reduza gradualmente a escala verticalmente ou remova as implantações do kro autogerenciado:

```
helm uninstall kro --namespace kro
```

Com essa abordagem, ambos os controladores podem coexistir com segurança durante a migração. A funcionalidade gerenciada assume automaticamente as ResourceGraphDefinitions e as instâncias que antes eram gerenciadas pelo kro autogerenciado, assegurando reconciliação contínua sem gerar conflitos.

Próximas etapas

- the section called “Criação da funcionalidade do kro”: crie um recurso de funcionalidade do kro
- the section called “Conceitos do kro”: compreenda os conceitos do kro e a composição de recursos

Solução de problemas das funcionalidades do EKS

Este tópico fornece orientações gerais para a solução de problemas das funcionalidades do EKS, incluindo verificações de integridade das funcionalidades, problemas comuns e links para o acesso à solução de problemas específica de cada funcionalidade.

> **Note**
>
> As funcionalidades do EKS são totalmente gerenciadas e executadas de forma externa ao cluster. Você não tem acesso aos logs do controlador nem aos namespaces do controlador. A solução de problemas se concentra na integridade da funcionalidade, no status dos recursos e na configuração.

Abordagem geral para a solução de problemas

Ao solucionar problemas das funcionalidades do EKS, siga esta abordagem geral:

1. Verifique a integridade da funcionalidade: use `aws eks describe-capability` para visualizar o status da funcionalidade e possíveis problemas de integridade
2. Verifique o status dos recursos: confira os recursos do Kubernetes (CRDs) que você criou, observando condições de status e eventos
3. Analise as permissões do IAM: garanta que o perfil da funcionalidade conte com as permissões necessárias
4. Verifique a configuração: confirme se a configuração específica da funcionalidade está correta

Verificação da integridade da funcionalidade

Todas as funcionalidades do EKS fornecem informações de integridade por meio do console do EKS e da API `describe-capability`.

Console do:

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. Selecione o nome do seu cluster.
3. Escolha a guia Observabilidade.
4. Escolha Monitorar cluster.

5. Escolha a guia Funcionalidades para visualizar a integridade e o status de todas as funcionalidades.

A guia Funcionalidades apresenta:

- Nome e tipo da funcionalidade
- Status atual
- Problemas de integridade, acompanhados de descrição

AWS CLI:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability-name
```

A resposta inclui:

- status: estado atual da funcionalidade (`CREATING`, `ACTIVE`, `UPDATING`, `DELETING`, `CREATE_FAILED` ou `UPDATE_FAILED`)
- integridade: informações de integridade, incluindo quaisquer problemas detectados pela funcionalidade

Status comuns das funcionalidades

CREATING: a funcionalidade está sendo configurada.

ACTIVE: a funcionalidade está em execução e pronta para uso. Se os recursos não estiverem funcionando conforme esperado, verifique o status do recurso e as permissões de IAM.

UPDATING: as alterações de configuração estão sendo aplicadas. Aguarde até que o status retorne para `ACTIVE`.

CREATE_FAILED ou UPDATE_FAILED: o processo de criação ou de atualização apresentou um erro. Confira a seção destinada à integridade para obter mais detalhes. Causas comuns:

- Política de confiança do perfil do IAM incorreta ou ausente
- O perfil do IAM não existe ou não está acessível

- Problemas de acesso ao cluster
- Parâmetros de configuração inválidos

Verificação do status dos recursos do Kubernetes

As funcionalidades do EKS criam e gerenciam definições de recursos personalizados (CRDs, na sigla em inglês) do Kubernetes em seu cluster. Ao solucionar problemas, verifique o status dos recursos que você criou:

```
# List resources of a specific type
kubectl get resource-kind -A

# Describe a specific resource to see conditions and events
kubectl describe resource-kind
         resource-name -n namespace

# View resource status conditions
kubectl get resource-kind
         resource-name -n namespace -o jsonpath='{.status.conditions}'

# View events related to the resource
kubectl get events --field-selector involvedObject.name=resource-name -n namespace
```

As condições de status do recurso fornecem informações sobre:

- A prontidão do recurso
- Quaisquer erros encontrados
- O estado atual de reconciliação

Análise das permissões do IAM e do acesso ao cluster

Diversos problemas relacionados às funcionalidades são causados por permissões inadequadas no IAM ou pela falta de configuração de acesso ao cluster. Verifique tanto as permissões do perfil da funcionalidade quanto as entradas de acesso ao cluster.

Verificação das permissões do perfil do IAM

Confirme se o perfil da funcionalidade detém todas as permissões necessárias:

```
# List attached managed policies
aws iam list-attached-role-policies --role-name my-capability-role

# List inline policies
aws iam list-role-policies --role-name my-capability-role

# Get specific policy details
aws iam get-role-policy --role-name my-capability-role --policy-name policy-name

# View the role's trust policy
aws iam get-role --role-name my-capability-role --query 'Role.AssumeRolePolicyDocument'
```

É necessário que a política de confiança autorize a entidade principal do serviço `capabilities.eks.amazonaws.com`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Verificação das entradas de acesso e das políticas de acesso do EKS

Todas as funcionalidades necessitam de entradas de acesso e de políticas de acesso apropriadas no cluster em que são executadas.

Verifique se a entrada de acesso existe:

```
aws eks list-access-entries \
  --cluster-name my-cluster \
  --region region-code
```

Procure o ARN do perfil da funcionalidade na lista. Se estiver ausente, a funcionalidade não poderá acessar o cluster.

Verifique as políticas de acesso anexadas à entrada:

```
aws eks list-associated-access-policies \
  --cluster-name my-cluster \
  --principal-arn arn:aws:iam::111122223333:role/my-capability-role \
  --region region-code
```

Todas as funcionalidades requerem políticas de acesso apropriadas:

- ACK: necessita de permissões para criar e gerenciar recursos do Kubernetes
- kro: necessita de permissões para criar e gerenciar recursos do Kubernetes
- Argo CD: necessita de permissões para criar e gerenciar Applications e requer entradas de acesso nos clusters remotos para implantações em vários clusters

Para implantações de vários clusters do Argo CD:

Ao realizar implantações em clusters remotos, assegure-se de que o perfil da funcionalidade disponha de uma entrada de acesso em cada cluster de destino:

```
# Check Access Entry on target cluster
aws eks describe-access-entry \
  --cluster-name target-cluster \
  --principal-arn arn:aws:iam::111122223333:role/argocd-capability-role \
  --region region-code
```

Se a entrada de acesso estiver ausente em algum cluster de destino, o Argo CD não conseguirá implantar aplicações nele. Consulte the section called “Registro de clusters” para obter detalhes sobre a configuração.

Solução de problemas específicas por funcionalidade

Para obter orientações detalhadas de solução de problemas específicas relacionadas a cada tipo de funcionalidade:

- the section called “Solução de problemas”: solução de problemas na criação de recursos do ACK, permissões do IAM e acesso entre contas
- the section called “Solução de problemas”: solução de problemas na sincronização de aplicações, autenticação de repositórios e implantações em vários clusters
- the section called “Solução de problemas”: solução de problemas em ResourceGraphDefinitions, expressões CEL e permissões RBAC

Problemas comuns em todas as funcionalidades

Funcionalidade travada no estado CREATING

Se uma funcionalidade permanecer no estado `CREATING` por mais tempo do que o esperado:

1. Verifique a integridade da funcionalidade em busca de problemas específicos no console (Observabilidade > Monitorar cluster > guia Funcionalidades) ou usando a AWS CLI:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability-name \
  --query 'capability.health'
```

2. Verifique se o perfil do IAM existe e conta com a política de confiança correta
3. Certifique-se de que o cluster esteja acessível e íntegro
4. Verifique se há algum problema no nível do cluster que possa impedir a configuração de recursos

Recursos não estão sendo criados ou atualizados

Se a funcionalidade estiver `ACTIVE`, mas os recursos não estiverem sendo criados ou atualizados:

1. Verifique o status do recurso em busca de erros
2. Verifique as permissões do IAM para os serviços da AWS específicos (ACK) ou para os repositórios (Argo CD)
3. Verifique as permissões RBAC para a criação dos recursos subjacentes (kro)
4. Analise as especificações do recurso em busca de erros de validação

Problemas de integridade da funcionalidade

Se o comando `describe-capability` apresentar problemas de integridade:

1. Leia atentamente as descrições dos problemas, pois elas geralmente indicam a causa específica
2. Corrija a causa raiz (por exemplo, permissões do IAM, erros de configuração e entre outros)
3. A funcionalidade será recuperada automaticamente assim que o problema for resolvido

Próximas etapas

- the section called “Como trabalhar com funcionalidades”: gerenciamento de recursos da funcionalidade
- the section called “Solução de problemas”: solução de problemas específica do ACK
- the section called “Solução de problemas”: solução de problemas específica do Argo CD
- the section called “Solução de problemas”: solução de problemas específica do kro
- the section called “Considerações para funcionalidades do EKS”: práticas recomendadas de segurança para as funcionalidades

Organizar e monitorar recursos de clusters

Este capítulo inclui os tópicos a seguir para ajudar você a gerenciar o cluster. Também é possível visualizar informações sobre recursos do Kubernetes com o Console de gerenciamento da AWS.

- O Kubernetes Dashboard é uma interface de usuário de uso geral baseada na web para clusters do Kubernetes. Com ele, os usuários podem gerenciar aplicações em execução no cluster, solucionar problemas relacionados a essas aplicações e gerenciar o próprio cluster. Para obter mais informações, consulte o repositório do Kubernetes Dashboard no GitHub.
- the section called “Servidor de métricas” – O servidor de métricas do Kubernetes é um agregador de dados de uso de recursos no cluster. Ele não é implantado por padrão no cluster, mas é usado por complementos do Kubernetes, como o Kubernetes Dashboard e o the section called “Horizontal Pod Autoscaler”. Neste tópico, você aprenderá como instalar o Metrics Server.
- the section called “Implantar aplicações com o Helm” - O gerenciador de pacotes Helm para Kubernetes ajuda a instalar e gerenciar aplicações no cluster do Kubernetes. Este tópico ajuda a instalar e executar os binários Helm para que você possa instalar e gerenciar gráficos usando a CLI do Helm no computador local.
- the section called “Marcando seus Recursos” – Para ajudar a gerenciar os recursos do Amazon EKS, você pode atribuir seus próprios metadados a cada recurso na forma de etiquetas. Este tópico descreve as etiquetas e mostra como criá-las.
- the section called “Cotas de serviço” – Sua conta da AWS tem cotas padrão, anteriormente chamadas de limites, para cada serviço da AWS. Conheça as cotas do Amazon EKS e saiba como aumentá-las.

Monitorar e otimizar os custos de clusters do Amazon EKS

O monitoramento de custos é um aspecto essencial do gerenciamento dos clusters do Kubernetes no Amazon EKS. Ao obter visibilidade dos custos do cluster, você pode otimizar a utilização de recursos, definir orçamentos e tomar decisões baseadas em dados sobre suas implantações. O Amazon EKS fornece duas soluções de monitoramento de custos, cada uma com suas vantagens exclusivas, para ajudar você a acompanhar e alocar seus custos de forma eficaz:

AWS Dados de alocação de custos de divisão de faturamento para o Amazon EKS- Esse recurso nativo se integra perfeitamente ao Console de faturamento do AWS, permitindo que você analise e aloque custos usando a mesma interface familiar e fluxos de trabalho que você usa para outros

serviços do AWS. Com a alocação de custos divididos, você pode obter insights dos custos do Kubernetes diretamente junto com seus outros gastos com a AWS, o que facilita a otimização holística dos custos em todo o seu ambiente da AWS. Também é possível utilizar os recursos de faturamento da AWS existentes, como Categorias de Custos e Detecção de Anomalias em Custos, para aprimorar ainda mais seus recursos de gerenciamento de custos. Para obter mais informações, consulte Compreender os dados de alocação de custos divididos no Guia do usuário de faturamento da AWS.

Kubecost: o Amazon EKS é compatível com o Kubecost, uma ferramenta de monitoramento de custos do Kubernetes. O Kubecost oferece uma abordagem rica em recursos e nativa do Kubernetes para monitoramento de custos, fornecendo detalhamentos granulares de custos por recursos do Kubernetes, recomendações de otimização de custos e painéis e relatórios prontos para uso. O Kubecost também recupera dados de preços precisos por meio da integração com o Relatório de Custos e Uso da AWS, garantindo que você tenha uma visão precisa dos custos do Amazon EKS. Saiba como Instalar o Kubecost. Consulte a página do Kubecost do AWS Marketplace para obter informações sobre como obter uma assinatura gratuita do Kubecost.

Visualizar os custos por pod na cobrança da AWS com alocação de custos divididos

Monitoramento de custos usando dados de alocação de custos divididos da AWS para o Amazon EKS

É possível usar dados de alocação de custos divididos da AWS para o Amazon EKS para obter visibilidade granular de custos para seus clusters do Amazon EKS. Isso permite que você analise, otimize e estorne o custo e o uso das aplicações do Kubernetes. Você aloca os custos da aplicação a unidades de negócios e equipes individuais com base nos recursos de CPU e memória do Amazon EC2 consumidos pela aplicação do Kubernetes. Os dados de alocação de custos divididos para o Amazon EKS oferecem visibilidade do custo por pod e permitem agregar os dados de custos por pod usando um namespace, um cluster e outras primitivas do Kubernetes. Veja a seguir exemplos de primitivas do Kubernetes que você pode usar para analisar dados de alocação de custos do Amazon EKS.

- Nome do cluster
- Implantação
- Namespace
- Nó

- Nome da workload
- Tipo de workload

As tags de alocação de custos definidas pelo usuário também são compatíveis. Para obter mais informações sobre como usar dados de alocação de custos divididos, consulte Noções básicas de dados de alocação de custos divididos no Guia do usuário de faturamento da AWS.

Configurar relatórios de custos e utilização

Você pode ativar a divisão de dados de alocação de custos para EKS no Console de gerenciamento de custos, na interface de linha de comando AWS ou nos SDKs AWS.

Siga estas etapas para usar os Dados de alocação de custos divididos:

1. Opte por Dados de alocação de custos divididos. Para obter mais informações, consulte Ativação de dados de alocação de custo dividido no Guia do usuário do relatório de custo e uso AWS.
2. Inclua os dados em um relatório novo ou existente.
3. Visualize o relatório. É possível usar o console do Gerenciamento de Faturamento e Custos ou visualizar os arquivos do relatório no Amazon Simple Storage Service.

Instalar o Kubecost

O Amazon EKS é compatível com o Kubecost, que você pode usar para monitorar os custos detalhados por recursos do Kubernetes, incluindo pods, nós, namespaces e rótulos. Este tópico aborda a instalação do Kubecost e o acesso ao painel dele.

O Amazon EKS fornece um pacote otimizado da AWS do Kubecost para visibilidade de custos do cluster. É possível usar seus acordos de suporte da AWS existentes para obter suporte. Para obter mais informações sobre as versões disponíveis do Kubecost, consulte the section called “Saiba mais sobre o Kubecost”.

> **Note**
>
> O Kubecost v2 apresenta vários novos recursos importantes. Saiba mais sobre o Kubecost v2.

Para obter mais informações sobre o Kubecost, consulte a documentação do Kubecost e as Perguntas frequentes.

Instalar pacote otimizado do Kubecost para Amazon EKS

Você pode usar um dos seguintes procedimentos para instalar o pacote otimizado do Kubecost para Amazon EKS:

- Antes de começar, é recomendável revisar a Visão geral da arquitetura do Kubecost para entender como ele funciona no Amazon EKS.
- Se for iniciante no Amazon EKS, recomendamos usar o complemento do Amazon EKS para a instalação porque ele simplifica a instalação do pacote otimizado do Kubecost para o Amazon EKS. Para obter mais informações, consulte Deploying Kubecost on an Amazon EKS cluster using Amazon EKS add-on.
- Para personalizar a instalação, você pode configurar o pacote otimizado do Kubecost para Amazon EKS com o Helm. Para obter mais informações, consulte Deploying Kubecost on an Amazon EKS cluster using Helm na documentação do Kubecost.

Acessar o painel do Kubecost

Depois que a configuração do pacote otimizado do Kubecost para Amazon EKS estiver concluída, você terá acesso ao painel do Kubecost. Para obter mais informações, consulte the section called “Acessar o painel do Kubecost”.

Acessar o painel do Kubecost

Pré-requisitos

1. Certifique-se de que o estado dos pods relacionados ao Kubecost seja “Running”.

```
kubectl get pods --namespace kubecost
```

Acessar o painel do Kubecost

1. No seu dispositivo, habilite o encaminhamento de portas para expor o painel do Kubecost.
   - Se o kubecost for instalado usando o Helm:

```
kubectl port-forward deployment/kubecost-cost-analyzer 9090 --namespace kubecost
```

- Se o kubecost for instalado usando o complemento do Amazon EKS:

```
kubectl port-forward deployment/cost-analyzer 9090 --namespace kubecost
```

Como alternativa, você pode usar o AWS Load Balancer Controller para expor o Kubecost e usar o Amazon Cognito para autenticação, autorização e gerenciamento de usuários. Para obter mais informações, consulte Como usar o Application Load Balancer e o Amazon Cognito para autenticar usuários para suas aplicações Web do Kubernetes.

2. No mesmo dispositivo em que você concluiu a etapa anterior, abra um navegador da Web e digite o seguinte endereço.

```
http://localhost:9090
```

Você verá a página de visão geral do Kubecost em seu navegador. Pode levar de cinco a dez minutos (ou mais) para o Kubecost coletar métricas, dependendo do tamanho do seu cluster. Você pode ver seus gastos com o Amazon EKS, incluindo custos cumulativos de cluster, custos de ativos associados do Kubernetes e gastos mensais agregados.

3. Para rastrear os custos em um nível de cluster, marque seus recursos do Amazon EKS para cobrança. Para obter mais informações, consulte the section called “Marcar recursos para faturamento”.
   - Cost allocation (Alocação de custos): visualize os custos mensais do Amazon EKS e os custos cumulativos de cada um dos seus namespaces e outras dimensões nos últimos sete dias. Isso é útil para entender quais partes da sua aplicação estão contribuindo para os gastos do Amazon EKS.
   - Assets (Ativos): visualize os custos dos ativos de infraestrutura da AWS que estão associados aos recursos do Amazon EKS.

Saiba mais sobre o Kubecost

O Amazon EKS fornece um pacote otimizado da AWS do Kubecost para visibilidade de custos do cluster. O Amazon EKS é compatível com o Kubecost, que você pode usar para monitorar os custos detalhados por recursos do Kubernetes, incluindo pods, nós, namespaces e rótulos.

Este tópico aborda as versões disponíveis do Kubecost e as diferenças entre os níveis disponíveis. O EKS é compatível com a versão 1 e 2 do Kubecost. Cada versão está disponível em níveis diferentes. Você pode usar o pacote otimizado do Kubecost para Amazon EKS nos clusters do Amazon EKS sem custo adicional. Pode haver cobranças pelo uso de serviços da AWS associados, como o Amazon Managed Service para Prometheus. Além disso, é possível usar seus acordos de suporte da AWS existentes para obter suporte.

Como um administrador da plataforma do Kubernetes e líder financeiro, você pode usar o Kubecost para visualizar um detalhamento das cobranças do Amazon EKS, alocar custos e cobrar de forma retroativa unidades organizacionais, como as equipes de aplicações. É possível fornecer às suas equipes internas e unidades de negócios dados de custos transparentes e precisos com base em seu faturamento real da AWS. Além disso, você também pode obter recomendações personalizadas para otimização de custos com base no ambiente de infraestrutura e nos padrões de uso em seus clusters. Para obter mais informações sobre o Kubecost, consulte Kubecost na documentação.

Qual é a diferença entre o pacote personalizado e a versão gratuita do Kubecost (também conhecido como OpenCost)?

A AWS e o Kubecost colaboraram para oferecer uma versão personalizada do Kubecost. Esta versão inclui um subconjunto de recursos comerciais sem custo adicional. Consulte as tabelas a seguir para conferir os recursos incluídos no pacote personalizado do Kubecost.

Kubecost v2

Qual é a diferença entre o Kubecost v1 e v2?

O Kubecost 2.0 é uma grande atualização das versões anteriores e traz novos recursos importantes, incluindo um backend de API. Observe que as APIs de alocação e ativos são totalmente compatíveis com versões anteriores. Consulte a documentação da Kubecost para garantir uma transição tranquila. Para obter a lista completa de aprimoramentos, consulte o anúncio do Kubecost v2.0 e as notas completas da release do Kubecost.

> **Important**
>
> Consulte a documentação do Kubecost antes de fazer a atualização. A atualização pode afetar a disponibilidade do relatório.

Comparação dos principais recursos:

Recurso	Nível gratuito do Kubecost 2.0	Pacote Kubecost 2.0 otimizado para Amazon EKS	Kubecost Enterprise 2.0
Visibilidade dos custos do cluster	Clusters ilimitados de até 250 núcleos	Multicluster unificado sem limites de núcleo quando integrado ao Amazon Managed Service para Prometheus	Número unificado e ilimitado de clusters em um número ilimitado de ambientes (ou seja, multinuvem)
Implantação	Hospedado pelo usuário	Hospedado pelo usuário	Hospedado pelo usuário, hospedado pelo Kubecost (locatário dedicado), SaaS
Bancos de dados compatíveis	Prometheus local	Amazon Managed Service for Prometheus ou Prometheus local	Qualquer variante do Prometheus e bancos de dados personalizados
Suporte à retenção de banco de dados (métricas brutas)	15 dias	Dados históricos ilimitados	Dados históricos ilimitados
Retenção de API e interface de usuário do Kubecost (ETL)	15 dias	15 dias	Ilimitado
Visibilidade da nuvem híbrida	-	Clusters Amazon EKS e Amazon EKS Anywhere	Multinuvem e nuvem híbrida
Alertas e relatórios recorrentes	Compatível apenas com o cluster primário, limitado a 250 núcleos	Alertas de eficiência, alertas de orçamento, alertas de alteração de gastos e outros	Alertas de eficiência, alertas de orçamento, alertas de alteração de gastos e outros

Recurso	Nível gratuito do Kubecost 2.0	Pacote Kubecost 2.0 otimizado para Amazon EKS	Kubecost Enterprise 2.0
		recursos com suporte em todos os clusters	recursos com suporte em todos os clusters
Relatórios salvos	-	Relatórios usando 15 dias de métricas	Relatórios usando métricas e dados históricos ilimitados
Integração de faturamento na nuvem	Compatível apenas com o cluster primário, limitado a 250 núcleos	Suporte personalizado de preços para AWS (incluindo vários clusters e várias contas)	Suporte personalizado de preços para qualquer nuvem
Recomendações de economia	Compatível apenas com o cluster primário, limitado a 250 núcleos	Insights primários do cluster, mas não há limite de 250 núcleos	Insights de vários clusters
Governança: auditorias	-	-	Auditoria de eventos históricos de custos
Compatibilidade com autenticação única (SSO)	-	Compatível com Amazon Cognito	Okta, Auth0, PingID, KeyCloak e qualquer outro item personalizado
Controle de acesso baseado em função (RBAC) com SAML 2.0	-	-	Okta, Auth0, PingID, KeyCloak e qualquer outro item personalizado
Treinamento e integração corporativos	-	-	Serviço completo de treinamento e integração para FinOps

Recurso	Nível gratuito do Kubecost 2.0	Pacote Kubecost 2.0 otimizado para Amazon EKS	Kubecost Enterprise 2.0
Equipes	-	-	Sim

Novos recursos:

Os seguintes recursos têm limites de métricas:

- Agregador do Kubecost
- Monitoramento de rede
- Ações do Kubecost
- Coleções
- Detecção de anomalias
- Dimensionamento correto da solicitação de contêiner
- Previsão do Kubecost
- Preenchimento automático para filtragem e agregação

Limites de métricas:

Métrica	Nível gratuito do Kubecost 2.0	Pacote Kubecost 2.0 otimizado para Amazon EKS	Kubecost Enterprise 2.0
Tamanho do cluster	Clusters ilimitados de até 250 núcleos	Ilimitado	Ilimitado
Retenção de métricas	15 dias	15 dias	Ilimitado
Suporte a vários clusters	Indisponível	Available (Disponível)	Available (Disponível)
Limites de núcleo	250 núcleos por cluster	Sem limites de núcleos	Sem limites de núcleos

Kubecost v1

Recurso	Nível gratuito do Kubecost	Pacote otimizado do Kubecost para Amazon EKS	Empresa Kubecost
Implantação	Hospedado pelo usuário	Hospedado pelo usuário	Hospedado pelo usuário ou hospedado pelo Kubecost (SaaS)
Número compatível de clusters	Ilimitado	Ilimitado	Ilimitado
Bancos de dados compatíveis	Prometheus local	Prometheus local ou Amazon Managed Service for Prometheus	Prometheus, Amazon Managed Service for Prometheus, Cortex ou Thanos
Compatibilidade com retenção de banco de dados	15 dias	Dados históricos ilimitados	Dados históricos ilimitados
Retenção de API do Kubecost (ETL)	15 dias	15 dias	Dados históricos ilimitados
Visibilidade dos custos do cluster	Clusters únicos	Vários clusters unificados	Vários clusters unificados
Visibilidade da nuvem híbrida	-	Clusters Amazon EKS e Amazon EKS Anywhere	Compatibilidade com multinuvem e nuvem híbrida
Alertas e relatórios recorrentes	-	Compatibilidade com alertas de eficiência, alertas de orçamento, alertas de alteração de gastos e outros	Compatibilidade com alertas de eficiência, alertas de orçamento, alertas de alteração de gastos e outros

Recurso	Nível gratuito do Kubecost	Pacote otimizado do Kubecost para Amazon EKS	Empresa Kubecost
Relatórios salvos	-	Relatórios usando dados de 15 dias	Relatórios usando dados históricos ilimitados
Integração de faturamento na nuvem	Obrigatório para cada cluster individual	Suporte personalizado de preços para AWS (incluindo vários clusters e várias contas)	Suporte personalizado de preços para AWS (incluindo vários clusters e várias contas)
Recomendações de economia	Insights de um único cluster	Insights de um único cluster	Insights de vários clusters
Governança: auditorias	-	-	Auditoria de eventos históricos de custos
Compatibilidade com autenticação única (SSO)	-	Compatível com Amazon Cognito	Okta, Auth0, PingID, KeyCloak
Regras de controle de acesso com base em função (RBAC) com SAML **2.0**	-	-	Okta, Auth0, PingID, Keycloak
Treinamento e integração corporativos	-	-	Serviço completo de treinamento e integração para FinOps

Perguntas frequentes

Veja as perguntas e respostas comuns a seguir sobre o uso do Kubecost com o Amazon EKS.

O que é o recurso de retenção de API do Kubecost (ETL)?

O recurso de ETL do Kubecost agrega e organiza métricas para promover a visibilidade dos custos em vários níveis de granularidade (como `namespace-level`, `pod-level`, e `deployment-level`). No pacote otimizado do Kubecost para Amazon EKS, os clientes obtêm dados e insights das métricas dos últimos 15 dias.

O que é o recurso de alertas e relatórios recorrentes? Quais alertas e relatórios estão incluídos?

Os alertas do Kubecost permitem que as equipes recebam atualizações em tempo real de gastos do Kubernetes, bem como despesas com a nuvem. Os relatórios recorrentes permitem que as equipes recebam exibições personalizadas de gastos históricos com o Kubernetes e a nuvem. Ambos são configuráveis usando a interface de usuário do Kubecost ou os valores do Helm. Eles oferecem suporte a e-mail, ao Slack e ao Microsoft Teams.

O que os relatórios salvos incluem?

Os relatórios salvos do Kubecost são exibições predefinidas das métricas de custo e eficiência. Eles incluem custo por cluster, namespace, rótulo e muito mais.

O que é a integração de faturamento na nuvem?

A integração com as APIs de faturamento da AWS permitem que o Kubecost exiba custos fora do cluster (como os do Amazon S3). Além disso, ela permite que o Kubecost reconcilie as previsões em cluster do Kubecost com dados reais de faturamento para contabilizar o uso de spot, Savings Plans e descontos corporativos.

O que as recomendações de economia incluem?

O Kubecost fornece insights e automação para ajudar os usuários a otimizar a infraestrutura e os gastos do Kubernetes.

Há alguma cobrança por essa funcionalidade?

Não. Você pode usar o pacote otimizado do Kubecost para Amazon EKS sem custo adicional. Caso queira mais recursos do Kubecost que não estão incluídos, você pode comprar uma licença Enterprise do Kubecost por meio do AWS Marketplace ou diretamente no Kubecost.

Há suporte disponível para o pacote otimizado do Kubecost para Amazon EKS?

Sim, somente se você estiver usando o pacote otimizado do Kubecost para Amazon EKS.

Como obtenho suporte para o pacote otimizado do Kubecost para Amazon EKS?

Você pode abrir um caso de suporte com a equipe de suporte da AWS em Entre em contato com a AWS.

Eu preciso de uma licença para usar os recursos do Kubecost fornecidos pela integração com o Amazon EKS?

Não.

É possível integrar o Kubecost com o Relatório de custos e uso da AWS para obter relatórios mais precisos?

Sim. É possível configurar o Kubecost para ingerir dados do Relatório de custos e uso da AWS e obter uma visibilidade precisa das despesas, incluindo descontos, preços de spot, preços de instâncias reservadas e outros. Para obter mais informações, consulte AWS Cloud Billing Integration na documentação do Kubecost.

Essa versão é compatível com o gerenciamento de custos de clusters autogerenciados do Kubernetes no Amazon EC2?

Não. O pacote otimizado do Kubecost para Amazon EKS só é compatível com clusters do Amazon EKS.

O Kubecost pode rastrear custos do Amazon EKS no AWS Fargate?

O Kubecost se esforça para mostrar a visibilidade dos custos do cluster do Amazon EKS no Fargate, mas com menor precisão do que com o Amazon EKS no Amazon EC2. Isso se deve principalmente à diferença na forma de cobrança do uso. Com o Amazon EKS no Fargate, você paga pelos recursos consumidos. Com o Amazon EKS em nós do Amazon EC2, você paga pelos recursos provisionados. O Kubecost calcula o custo de um nó do Amazon EC2 com base na especificação do nó, que inclui CPU, RAM e armazenamento temporário. Com o Fargate, os custos são calculados com base nos recursos solicitados para os pods do Fargate.

Como posso obter atualizações e novas versões do Kubecost?

Você pode atualizar sua versão do Kubecost usando procedimentos padrão de atualização do Helm. As versões mais recentes estão na Galeria pública do Amazon ECR.

A CLI **`kubectl-cost`** é compatível? Como faço para instalar?

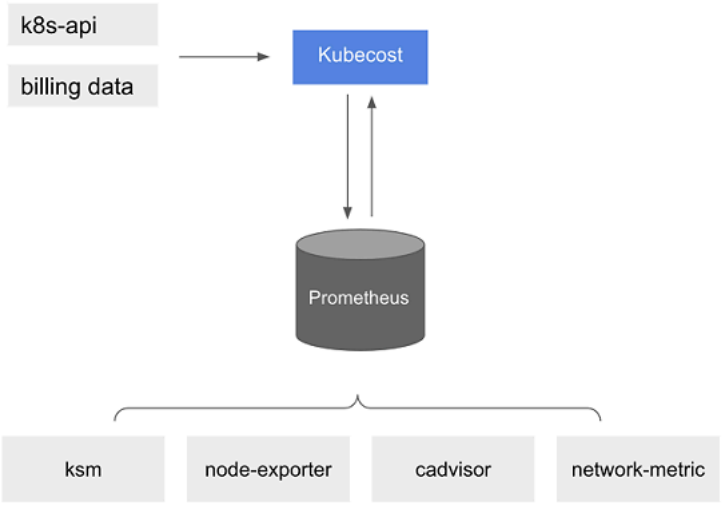
Sim. O `Kubectl-cost` é uma ferramenta de código aberto do Kubecost (licença do Apache 2.0) que fornece acesso da CLI a métricas de alocação de custos do Kubernetes. Para instalar o `kubectl-cost`, consulte Instalação no GitHub.

A interface de usuário do Kubecost é compatível? Como faço para acessá-la?

O Kubecost fornece um painel da Web que você pode acessar por meio do encaminhamento de portas do `kubectl`, de uma entrada ou de um balanceador de carga. Também é possível usar o AWS Load Balancer Controller para expor o Kubecost e o Amazon Cognito para autenticação, autorização e gerenciamento de usuários. Para obter mais informações, consulte How to use Application Load Balancer and Amazon Cognito to authenticate users for your Kubernetes web apps no blog da AWS.

Recursos adicionais do Kubecost

- Os recursos a seguir estão disponíveis na v1 e na v2 do Kubecost.
  - Exportar métricas de custos: o monitoramento otimizado de custos do Amazon EKS é implantado com o Kubecost e o Prometheus, que é um sistema de monitoramento de código aberto e banco de dados de séries temporais. O Kubecost lê as métricas do Prometheus e, em seguida, realiza cálculos de alocação de custos e grava as métricas de volta no Prometheus. O frontend do Kubecost lê métricas do Prometheus e as mostra na interface do usuário do Kubecost. A arquitetura está ilustrada no diagrama a seguir



    Com o Prometheus pré-instalado, é possível escrever consultas para ingerir dados do Kubecost no sistema atual de business intelligence para análise posterior. Você também pode usá-lo como uma fonte de dados para o seu painel Grafana atual para exibir os custos de cluster

do Amazon EKS com os quais suas equipes internas estão familiarizadas. Para saber mais sobre como escrever consultas do Prometheus, consulte o arquivo `readme` https://opencost.io/docs/installation/prometheus/ no GitHub ou use os modelos JSON do Grafana de exemplo no repositório do Kubecost para GitHub como referências.

- Integração do Relatório de Custos e Uso da AWS: para realizar cálculos de alocação de custos para o cluster do Amazon EKS, o Kubecost recupera as informações de preços públicos dos serviços da AWS e dos recursos da AWS da API Price List da AWS. Você também pode integrar o Kubecost com o Relatório de Custos e Uso da AWS para aumentar a precisão das informações de preços específicas da sua conta da AWS. Essas informações incluem programas de descontos corporativos, uso de instâncias reservadas, Savings Plans e uso de spot. Para saber mais sobre como funciona a integração do Relatório de Custos e Uso da AWS, consulte AWS Cloud Billing Integration na documentação do Kubecost.

Visualizar o uso de recursos com o servidor de métricas do Kubernetes

O Metrics Server do Kubernetes é um agregador de dados de uso de recursos no cluster, e não é implantado por padrão em clusters do Amazon EKS. Para obter mais informações, consulte Kubernetes Metrics Server no GitHub. O Metrics Server é comumente usado por outros complementos do Kubernetes, como implantações de escala de pods com o Horizontal Pod Autoscaler ou o painel do Kubernetes. Para obter mais informações, consulte Pipeline de métricas de recursos na documentação do Kubernetes. Este tópico explica como implantar o Kubernetes Metrics Server no cluster do Amazon EKS.

> **Important**
>
> As métricas destinam-se à análise pontual e não são uma fonte precisa para análise histórica. Eles não podem ser usados como uma solução de monitoramento ou para outros fins que não sejam de ajuste de escala automático. Para obter informações sobre as ferramentas de monitoramento, consulte Monitorar clusters.

Considerações

- Se estiver implantando manualmente o Kubernetes Metrics Server nos nós do Fargate usando o manifesto, configure a implantação do `metrics-server` para usar uma porta diferente da padrão

de `10250`. Essa porta é reservada para o Fargate. A versão do complemento do Amazon EKS do Metrics Server está pré-configurada para usar a porta `10251`.

- Certifique-se de que os grupos de segurança e ACLs de rede permitam a porta `10250` entre os pods do `metrics-server` e todos os outros nós e pods. O Kubernetes Metrics Server ainda usa a porta `10250` para coletar métricas de outros endpoints no cluster. Se você implantar nos nós do Fargate, permita tanto a porta alternativa configurada do Metrics Server quanto a porta `10250`.

Implantação como um complemento da comunidade com os complementos do Amazon EKS

Novidade: agora é possível implantar o Metrics Server como um complemento da comunidade usando o console da AWS ou as APIs do Amazon EKS.

Implantação com o console da AWS

1. Abra o cluster do EKS no console da AWS
2. Na guia “Complementos”, selecione Obter mais complementos.
3. Na seção “Complementos da comunidade”, selecione Metrics Server e, em seguida, clique em Próximo.
4. O EKS determina a versão apropriada do complemento para o seu cluster. É possível alterar a versão usando o menu suspenso Versão.
5. Selecione Próximo e, em seguida, Criar para instalar o complemento.

Recursos adicionais

Saiba mais sobre o the section called “Complementos da comunidade”.

Você instala ou atualiza complementos da comunidade da mesma forma que os outros complementos do Amazon EKS.

- the section called “Criação de um complemento”
- the section called “Atualização de um complemento”
- the section called “Remover um complemento”

Implantação com manifesto

Novidade: agora é possível implantar o Metrics Server como um complemento da comunidade usando o console da AWS ou as APIs do Amazon EKS. Essas instruções de instalação com manifesto serão arquivadas.

1. Implante o servidor de métricas com o seguinte comando:

   ```
   kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
   ```

   Se estiver usando o Fargate, será necessário alterar esse arquivo. Na configuração padrão, o servidor de métricas usa a porta 10250. Essa porta é reservada no Fargate. Substitua as referências à porta 10250 em components.yaml por outra porta, por exemplo, 10251.

2. Verifique se a implantação do `metrics-server` está executando o número desejado de pods com o comando a seguir.

   ```
   kubectl get deployment metrics-server -n kube-system
   ```

   Veja abaixo um exemplo de saída.

   ```
   NAME             READY   UP-TO-DATE   AVAILABLE   AGE
   metrics-server   1/1     1            1           6m
   ```

3. Teste se o servidor de métricas está funcionando ao exibir o uso de recursos (CPU/memória) dos nós.

   ```
   kubectl top nodes
   ```

4. Caso receba a mensagem de erro `Error from server (Forbidden)`, você precisará atualizar sua configuração do RBAC do Kubernetes. Sua identidade do RBAC do Kubernetes precisa de permissões suficientes para ler as métricas do cluster. Analise as permissões mínimas exigidas da API do Kubernetes para leitura de métricas no GitHub. Aprenda como conceder acesso às APIs do Kubernetes para identidades do AWS IAM, como perfis.

Implantar aplicações com o Helm no Amazon EKS

O gerenciador de pacotes Helm para Kubernetes ajuda a instalar e gerenciar aplicações no cluster do Kubernetes. Para obter mais informações, consulte a documentação do Helm. Este tópico ajuda a instalar e executar os binários Helm para que você possa instalar e gerenciar gráficos usando a CLI do Helm no sistema local.

Important

Antes de instalar os gráficos Helm no cluster do Amazon EKS, é necessário configurar o `kubectl` para funcionar com o Amazon EKS. Se você ainda não fez isso, consulte the section called “Acessar meu cluster com o kubectl” antes de continuar. Se o comado a seguir for bem-sucedido para o cluster, a configuração estará correta.

```
kubectl get svc
```

1. Execute o comando apropriado para o sistema operacional cliente.
   - Se estiver usando o MacOS com o Homebrew, instale os binários com o comando a seguir.

     ```
     brew install helm
     ```

   - Para obter mais opções de instalação, consulte Installing Helm na documentação do Helm.

     Note

     Se você receber uma mensagem de que o `openssl` deve ser instalado primeiro, poderá instalá-lo com o comando a seguir.

     ```
     sudo yum install openssl
     ```

1. Para escolher os novos arquivos binários no `PATH`, feche a janela atual do terminal e abra uma nova.
2. Veja a versão do Helm que você instalou.

```
helm version --template='{{ .Version }}{{ "\n" }}'
```

Veja um exemplo de saída abaixo.

```
v3.17.2
```

3. Verifique se a versão instalada é compatível com a versão do cluster. Verifique Supported Version Skew para saber mais. Por exemplo, se você estiver executando com `3.17.x`, a versão compatível do Kubernetes não deverá estar fora do intervalo de `1.29.x ~ 1.32.x`.
4. Neste ponto, você pode executar quaisquer comandos do Helm (como `helm install` *chart-name*) para instalar, modificar, excluir ou consultar gráficos do Helm no seu cluster. Caso esteja começando a usar o Helm e ainda não tenha um chart específico para instalar, você pode:
   - Experimentar instalando um gráfico de exemplo. Consulte Instalar um gráfico de exemplo no Guia de início rápido do Helm.
   - Crie um gráfico de exemplo e envie-o para o Amazon ECR. Para obter mais informações, consulte os Pushing a Helm chart (Enviar um gráfico Helm) no Manual do usuário do Amazon Elastic Container Registry.
   - Instale um gráfico do Amazon EKS no repositório eks-charts do GitHub ou do ArtifactHub.

Organizar recursos do Amazon EKS com tags

É possível usar tags para ajudar você a gerenciar seus recursos do Amazon EKS. Este tópico fornece uma visão geral da função de etiquetas e mostra como você pode criá-las.

Tópicos

- Conceitos Básicos de Tags
- Marcando seus Recursos
- Restrições de tags
- Marcar recursos para faturamento
- Trabalhando com tags usando o console
- Trabalhar com etiquetas usando a CLI, a API ou o eksctl

> **Note**
>
> As tags são um tipo de metadados separados de rótulos e anotações do Kubernetes. Para saber mais sobre esses outros tipos de metadados, consulte as seguintes seções na documentação do Kubernetes:
>
> - Etiquetas e seletores
> - Anotações

Conceitos Básicos de Tags

Uma tag um rótulo atribuído a um recurso AWS. Cada tag consiste em uma chave e um valor opcional.

Com tags, é possível categorizar seus recursos da AWS. Por exemplo, você pode categorizar recursos por finalidade, proprietário ou ambiente. Quando você tem muitos recursos do mesmo tipo; é possível identificar rapidamente um recurso específico com base nas tags que você atribuiu a ele. Por exemplo, você pode definir um conjunto de etiquetas para os clusters do Amazon EKS para ajudar a monitorar o proprietário e o nível da pilha de cada cluster. Recomendamos planejar um conjunto consistente de chaves de etiquetas para cada tipo de recurso. Você pode pesquisar e filtrar os recursos de acordo com as tags que adicionar.

Após adicionar uma tag, você pode editar as chaves e os valores das tags ou removê-las de um recurso a qualquer momento. Caso exclua um recurso, todas as respectivas tags também serão excluídas.

As etiquetas não têm significado semântico no Amazon EKS e são interpretadas estritamente como uma sequência de caracteres. É possível definir o valor de uma tag como uma string vazia. No entanto, você não pode definir o valor de uma tag como nulo. Se você adicionar uma etiqueta que tenha a mesma chave de uma etiqueta existente nesse recurso, o novo valor substituirá o antigo.

Se você usar o AWS Identity and Access Management (IAM), poderá controlar quais usuários da sua conta AWS têm permissão para gerenciar tags.

Marcando seus Recursos

As seguintes etiquetas de suporte para recursos do Amazon EKS:

- clusters

- grupos de nós gerenciados
- Perfis do Fargate

Você pode etiquetar esses recursos usando o seguinte:

- Se você estiver usando o console do Amazon EKS, poderá aplicar tags a recursos novos ou existentes a qualquer momento. Para isso, você pode usar a guia Tags (Etiquetas) na página de recursos relevante. Para ter mais informações, consulte the section called “Trabalhando com tags usando o console”.
- Se você estiver usando o `eksctl`, poderá aplicar tags a recursos quando eles forem criados usando a opção `--tags`.
- Se estiver usando a AWS CLI, a API do Amazon EKS ou um SDK do AWS, você poderá aplicar tags a novos recursos usando o parâmetro `tags` na ação relevante da API. Você também pode aplicar etiquetas aos recursos usando a ação da API `TagResource`. Para mais informações, consulte TagResource.

Ao usar algumas ações de criação de recursos, você também pode especificar tags para o recurso ao mesmo tempo em que o cria. Se as tags não puderem ser aplicadas enquanto um recurso estiver sendo criado, o recurso falhará ao ser criado. Esse mecanismo garante que os recursos que você pretende aplicar tags sejam criados com as etiquetas especificadas por você ou não sejam criados. Se você aplicar etiquetas nos recursos no momento da criação, não precisará executar os scripts de aplicação de tags personalizados após a criação do recurso.

As tags não são propagadas para nenhum outro recurso associado ao recurso criado. Por exemplo, as tags de perfil do Fargate não se propagam para outros recursos associados ao perfil do Fargate, como os pods que são agendados com ele.

Restrições de tags

As restrições a seguir se aplicam às tags:

- Podem ser associadas no máximo 50 tags a um recurso.
- As chaves de tag não podem ser repetidas para um recurso. As chaves de tag devem ser exclusivas, e cada chave só pode ter um valor.
- As chaves podem ter até 128 caracteres em UTF-8.
- Os valores podem ter até 256 caracteres em UTF-8.

- Se vários serviços e recursos do AWS usarem seu esquema de marcação, limite os tipos de caracteres que você usa. Alguns serviços podem ter restrições quanto aos caracteres permitidos. Em geral, os caracteres permitidos são letras, números, espaços e os seguintes caracteres: `+ - = . _ : / @`.
- As chaves e valores das tags diferenciam maiúsculas de minúsculas.
- Não use `aws:`, `AWS:` ou qualquer combinação de letras maiúsculas e minúsculas como prefixo para chaves ou valores. Esses são reservados para uso pela AWS. Você não pode editar nem excluir chaves nem valores de tags com esse prefixo. As tags com esse prefixo não contam para seu limite de tags por recurso.

Marcar recursos para faturamento

Ao aplicar tags aos clusters do Amazon EKS, é possível pode usá-las para alocação de custos em seus Cost & Usage Reports (Relatórios de custos e uso). Os dados de medição do seu Cost & Usage Reports (Relatório de custos e uso) mostram o uso em todos ps clusters do Amazon EKS. Para obter mais informações, consulte AWS cost and usage report no AWS Billing User Guide.

A tag de alocação de custos gerada pela AWS, especificamente `aws:eks:cluster-name`, permite que você divida os custos de instância do Amazon EC2 pelo cluster individual do Amazon EKS no Explorador de custos. No entanto, essa tag não captura as despesas do ambiente de gerenciamento. A tag é automaticamente adicionada às instâncias do Amazon EC2 que participam de um cluster do Amazon EKS. Esse comportamento acontece independentemente de as instâncias serem provisionadas usando os grupos de nós gerenciados do Amazon EKS, o Karpenter ou diretamente com o Amazon EC2. Essa tag específica não conta para o limite de 50 tags. Para usar a tag, o proprietário da conta deve ativá-la no console AWS Billing ou usando a API. Quando o proprietário de uma conta de gerenciamento de AWS Organizations ativa a tag, ela também é ativada para todas as contas de membros da organização.

É possível organizar suas informações de faturamento com base nos recursos com os mesmos valores da chave da tag. Por exemplo, é possível aplicar tags a vários recursos com um nome de aplicação específico, e depois organizar suas informações de faturamento. Dessa forma, é possível ver o custo total daquela aplicação em vários serviços. Para obter mais informações sobre a configuração de um relatório de alocação de custos com tags, consulte O relatório mensal de alocação de custos no AWS Billing User Guide.

> **Note**
>
> Se você tiver acabado de habilitar a criação de relatórios, os dados do mês atual estarão disponíveis para visualização após 24 horas.

O Cost Explorer é uma ferramenta de geração de relatórios que está disponível como parte do nível gratuito da AWS. É possível usar o Cost Explorer para visualizar gráficos dos seus recursos do Amazon EKS dos últimos 13 meses. Também é possível prever o provável valor que você gastará nos próximos três meses. É possível exibir os padrões de quanto você gasta nos recursos da AWS ao longo do tempo. Por exemplo, você pode usar o Cost Explorer para identificar áreas que precisam de uma investigação mais profunda e observar tendências que podem ser usadas para o entendimento dos custos. Também é possível especificar os períodos dos dados e visualizar os dados de tempo por dia ou mês.

Trabalhando com tags usando o console

Usando o console do Amazon EKS, é possível gerenciar as etiquetas associadas aos clusters e grupos de nós gerenciados novos ou existentes.

Quando você selecionar uma página específica do recurso no console do Amazon EKS, a página exibirá uma lista desses recursos. Por exemplo, se você selecionar Clusters no painel de navegação esquerdo, o console exibirá uma lista dos clusters do Amazon EKS. Ao selecionar um recurso de uma dessas listas (por exemplo, um cluster específico), que seja compatível com as etiquetas, será possível vê-las e gerenciá-las na guia Tags (Etiquetas).

Também é possível usar o Editor de etiquetas no Console de gerenciamento da AWS que fornece uma maneira unificada de gerenciar etiquetas. Para obter mais informações, consulte Marcação dos recursos do AWS com o Tag Editor no Guia do Usuário do AWS Tag Editor.

Adição de tags a um recurso na sua criação

Você pode adicionar etiquetas aos clusters, grupos de nós gerenciados e perfis do Fargate do Amazon EKS quando os criar. Para ter mais informações, consulte the section called “Criar um cluster”.

Adição e exclusão de tags a um recurso

É possível adicionar ou excluir tags associadas aos seus clusters diretamente a partir da página do recurso.

1. Abra o console do Amazon EKS.
2. Na barra de navegação, selecione a região AWS a ser usada.
3. No painel de navegação à esquerda, escolha Clusters.
4. Escolha um cluster específico.
5. Escolha a guia Tags e escolha Gerenciar tags.
6. Na página Manage tags (Gerenciar etiquetas), adicione ou exclua as etiquetas conforme necessário.
   - Para adicionar uma tag, escolha Adicionar tag. Em seguida, especifique a chave e o valor de cada tag.
   - Para remover uma tag, escolha Remove tag (Remover tag).
7. Repita esse processo para cada tag que você deseja adicionar ou excluir.
8. Selecione Update (Atualizar) para concluir.

Trabalhar com etiquetas usando a CLI, a API ou o `eksctl`

Use os seguintes comandos da AWS CLI ou operações da API do Amazon EKS para adicionar, atualizar, listar e excluir as etiquetas dos seus recursos. Somente é possível usar `eksctl` para adicionar etiquetas durante a criação dos novos recursos com um único comando.

Tarefa	AWS CLI	AWS Tools for Windows PowerShell	Ação da API
Adicione ou sobrescreva uma ou mais tags.	tag-resource	Add-EKSResourceTag	TagResource
Exclua uma ou mais tags.	untag-resource	Remove-EKSResourceTag	UntagResource

Os exemplos a seguir mostram como marcar ou desmarcar recursos usando a AWS CLI.

Exemplo 1: marcar um cluster existente

O comando a seguir marca um cluster existente.

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

Exemplo 2: desmarcar um cluster existente

O comando a seguir exclui uma etiqueta de um cluster existente.

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Exemplo 3: listar etiquetas para um recurso

O comando a seguir lista as tags que estão associadas a um recurso existente.

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

Ao usar algumas ações de criação de recursos, você também pode especificar tags ao mesmo tempo em que cria o recurso. As ações a seguir são compatíveis com a especificação de tags na criação de um recurso.

Tarefa	AWS CLI	AWS Tools for Windows PowerShell	Ação da API	eksctl
Criar um cluster	create-cluster	New-EKSCluster	CreateCluster	`create cluster`
Crie um grupo de nós gerenciados*	create-nodegroup	New-EKSNodegroup	CreateNodegroup	`create nodegroup`
Criar um perfil do Fargate	criar perfil-fargata	New-EKSFargateProfile	CreateFargateProfile.html	`create fargatepr ofile`

- Se você quiser aplicar tags também a instâncias do Amazon EC2 ao criar um grupo de nós gerenciados, crie o grupo de nós gerenciados usando um modelo de execução. Para ter mais informações, consulte the section called “Etiquetar instâncias do Amazon EC2”. Se as instâncias já existirem, você pode etiquetá-las manualmente. Para obter mais informações, consulte Marcar seus recursos, no Guia do usuário do Amazon EC2.

Visualizar e gerenciar as cotas de serviço do Amazon EKS e do Fargate

O Amazon EKS foi integrado ao Service Quotas, um serviço da AWS que permite visualizar e gerenciar as cotas de um local central. Para obter mais informações, consulte What Is Service Quotas? (O que são cotas de serviço) no Manual do usuário de cotas de serviço. Com a integração do Service Quotas, você pode procurar rapidamente o valor das cotas de serviço do Amazon EKS e do AWS Fargate usando a Console de gerenciamento da AWS CLI e o AWS.

Visualizar cotas de serviço do EKS com o Console de gerenciamento da AWS

1. Abra o console do Service Quotas.
2. No painel de navegação à esquerda, escolha Serviços da AWS.
3. Na lista de serviços AWS, pesquise e selecione Amazon Elastic Kubernetes Service (Amazon EKS) ou AWS Fargate.

   Na lista Service quotas (Cotas de serviço) é possível ver o nome da cota de serviço, o valor aplicado (se estiver disponível), a cota da AWS padrão e se o valor da cota for ajustável.
4. Para visualizar informações adicionais sobre uma service quota, como descrição, escolha o nome da cota.
5. (Opcional) Para solicitar um aumento de cota, selecione a cota que deseja aumentar, selecione Solicitar Aumento de Cota, insira ou selecione as informações necessárias e, por fim, selecione Solicitar.

Para trabalhar mais com cotas de serviço usando o Console de gerenciamento da AWS, consulte o Guia do usuário do Service Quotas. Para solicitar o aumento da cota, consulte Solicitar um aumento de cota no Guia do usuário do Service Quotas.

Visualizar service quotas do EKS com a CLI AWS

Execute o comando a seguir para visualizar as cotas do Amazon EKS.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code eks \
```

```
    --output table
```

Execute o comando a seguir para visualizar as cotas do Fargate.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

> **Note**
>
> A cota retornada é o número de tarefas do Amazon ECS ou dos pods do Amazon EKS que podem ser executados simultaneamente no Fargate nessa conta na região atual da AWS.

Para trabalhar mais com cotas de serviço usando a AWS CLI, consulte service-quotas na Referência de comandos da AWS CLI. Para solicitar um aumento de cota, consulte o comando request-service-quota-increase, na Referência de comandos da AWS CLI.

Cotas de serviço do Amazon EKS

A AWS recomenda o uso do Console de gerenciamento da AWS para visualizar as cotas atuais. Para obter mais informações, consulte the section called “Visualizar cotas de serviço do EKS com o Console de gerenciamento da AWS”.

Para visualizar service quotas padrão do EKS, consulte Endpoints e cotas do Amazon Elastic Kubernetes Service na Referência geral da AWS.

Essas cotas de serviço estão listadas em Amazon Elastic Kubernetes Service (Amazon EKS) no console de Service Quotas. Para solicitar um aumento de cota para os valores que são mostrados como ajustáveis, consulte Solicitar um aumento de cota no Guia do usuário do Service Quotas.

> **Note**
>
> Ajustes nos seguintes componentes não são compatíveis no Service Quotas: * associações de Identidades de Pods por cluster. Para obter os limites, consulte the section called “Identidade de Pods”. * CIDRs para redes remotas de nós ou redes remotas de pods para nós híbridos. Para obter os limites, consulte the section called “Nós híbridos”.

Cotas de serviço do AWS Fargate

O serviço AWS Fargate no console Service Quotas lista várias cotas de serviço. Também é possível configurar alarmes que alertem você quando o uso se aproximar de uma cota de serviço. Para obter mais informações, consulte the section called “Criar um alarme do CloudWatch para monitorar métricas de uso dos recursos do Fargate”.

As novas contas AWS podem ter cotas iniciais mais baixas que podem aumentar com o tempo. A Fargate monitora constantemente o uso da conta em cada AWS Region e, em seguida, aumenta automaticamente as cotas com base no uso. Também é possível solicitar um aumento de cota para valores que sejam mostrados como ajustáveis. Para obter mais informações, consulte Solicitando um Aumento de Cota no Guia do usuário do Service Quotas.

A AWS recomenda usar o Console de gerenciamento da AWS para visualizar as cotas atuais. Para obter mais informações, consulte the section called “Visualizar cotas de serviço do EKS com o Console de gerenciamento da AWS”.

Para visualizar o AWS Fargate padrão nas service quotas do EKS, consulte Service quotas do Fargate na Referência geral da AWS.

> **Note**
>
> Além disso, o Fargate aplica tarefas do Amazon ECS e cotas de taxa de inicialização de pods do Amazon EKS. Para obter mais informações, consulte AWS Cotas de limitação do Fargate no guia do Amazon ECS.

Segurança no Amazon EKS

A segurança da nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você contará com um data center e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O modelo de responsabilidade compartilhada descreve isto como segurança da nuvem e segurança na nuvem:

- Segurança da nuvem – a AWS é responsável por proteger a infraestrutura que executa os serviços da AWS na Nuvem AWS. Para o Amazon EKS, a AWS é responsável pelo plano de controle do Kubernetes, que inclui os nós do plano de controle e o banco de dados `etcd`. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos programas de conformidade da AWS. Para saber mais sobre os programas de conformidade que se aplicam ao Amazon EKS, consulte Serviços da AWS no escopo pelo programa de conformidade.
- Segurança na nuvem: sua responsabilidade inclui as seguintes áreas:
  - A configuração de segurança do plano de dados, incluindo a configuração dos grupos de segurança que permitem que o tráfego passe do plano de controle do Amazon EKS para a VPC do cliente
  - A configuração dos nós e dos contêineres
  - O sistema operacional do nó (incluindo atualizações e patches de segurança)
  - Outros softwares de aplicações associadas:
    - Configurar e gerenciar os controles de rede, como regras de firewall
    - Administrar o gerenciamento de acesso e identidade no nível da plataforma, com ou além do IAM.
  - A confidencialidade dos dados, os requisitos da sua empresa e as leis e regulamentos aplicáveis

O Amazon EKS é certificado por vários programas de conformidade para aplicações regulamentadas e confidenciais. O Amazon EKS está em conformidade com SOC, PCI, ISO, FedRAMP-Moderate, IRAP, C5, K-ISMS, ENS High, OSPAR, HITRUST CSF, e é um serviço qualificado para HIPAA. Para obter mais informações, consulte Gerenciar acesso.

Esta documentação ajuda a entender como aplicar o modelo de responsabilidade compartilhada ao usar o Amazon EKS. Os tópicos a seguir mostram como configurar o Amazon EKS para atender aos

seus objetivos de segurança e de conformidade. Saiba também como usar outros serviços da AWS que ajudam você a monitorar e proteger os recursos do Amazon EKS.

> **Note**
>
> Os contêineres Linux são compostos de grupos de controle (cgroups) e namespaces que ajudam a limitar o que um contêiner pode acessar, mas todos os contêineres compartilham o mesmo kernel Linux que a instância host do Amazon EC2. Executar um contêiner como usuário raiz (UID 0) ou conceder acesso a um contêiner aos recursos do host ou namespaces, como a rede host ou o namespace PID do host são fortemente desencorajados, pois isso reduz a eficácia do isolamento fornecido pelos contêineres.

Tópicos

- Proteger os clusters do Amazon EKS com as práticas recomendadas
- Analisar vulnerabilidades no Amazon EKS
- Validação de conformidade para clusters do Amazon EKS
- Considerações de segurança sobre o Amazon Elastic Kubernetes Service
- Considerações de segurança para o Kubernetes
- Considerações de segurança sobre o Modo Automático do Amazon EKS
- Considerações sobre segurança para funcionalidades do EKS
- Identity and Access Management para o Amazon EKS

Proteger os clusters do Amazon EKS com as práticas recomendadas

As práticas recomendadas de segurança do Amazon EKS estão em Práticas recomendadas de segurança, no Guia de práticas recomendadas do Amazon EKS.

Analisar vulnerabilidades no Amazon EKS

A segurança é uma consideração essencial para configurar e manter clusters e aplicações do Kubernetes. A seguir são listados recursos para você analisar a configuração de segurança dos seus

clusters do EKS, recursos para verificar vulnerabilidades e integrações com serviços da AWS que podem fazer essa análise para você.

Referência do Center for Internet Security (CIS) para Amazon EKS

O Kubernetes Benchmark do Center for Internet Security (CIS) fornece orientação para as configurações de segurança do Amazon EKS. O parâmetro de referência:

- É aplicável aos nós do Amazon EC2 (gerenciados e autogerenciados) nos quais você é responsável pelas configurações de segurança dos componentes do Kubernetes.
- Fornece uma maneira padrão aprovada pela comunidade de garantir que você configurou seu cluster e nós do Kubernetes com segurança ao usar o Amazon EKS.
- Consiste em quatro seções: configuração de log de plano de controle, configurações de segurança do nó, políticas e serviços gerenciados.
- Oferece suporte a todas as versões do Kubernetes atualmente disponíveis no Amazon EKS e podem ser executadas usando kube-bench, uma ferramenta de código aberto padrão para verificar a configuração usando o benchmark do CIS em clusters do Kubernetes.

Para saber mais, consulteApresentando o CIS Amazon EKS Benchmark.

Para obter um pipeline `aws-sample` automatizado para atualizar o grupo de nós com uma AMI de referência da CIS, consulte EKS-Optimized AMI Hardening Pipeline.

Versões da plataforma do Amazon EKS

As versões da plataforma do Amazon EKS representam os recursos do ambiente de gerenciamento do cluster, incluindo quais sinalizadores do servidor de API do Kubernetes estão habilitados e a versão de patch atual do Kubernetes. Novos clusters são implantados com a versão mais recente da plataforma. Para obter detalhes, consulte as versões da plataforma do EKS.

Você pode atualizar um cluster do Amazon EKS para versões mais recentes do Kubernetes. Conforme novas versões do Kubernetes são disponibilizadas no Amazon EKS, recomendamos que você atualize proativamente seus clusters para usarem a versão mais recente disponível. Para obter mais informações sobre as versões do Kubernetes no EKS, consulte Versões compatíveis com o Amazon EKS.

Lista de vulnerabilidades do sistema operacional

Lista de vulnerabilidades do AL2023

Rastreie eventos de segurança ou privacidade para o Amazon Linux 2023 no Centro de segurança do Amazon Linux ou assine o feed RSS associado. Eventos de segurança e privacidade incluem uma visão geral do problema afetado, pacotes e instruções para atualizar suas instâncias para corrigir o problema.

Lista de vulnerabilidades do Amazon Linux 2

Rastreie eventos de segurança ou privacidade para o Amazon Linux 2 no Centro de segurança do Amazon Linux ou assine o feed RSS associado. Eventos de segurança e privacidade incluem uma visão geral do problema afetado, pacotes e instruções para atualizar suas instâncias para corrigir o problema.

Detecção de nós com o Amazon Inspector

É possível usar o Amazon Inspector para verificar a acessibilidade de rede não intencional dos nós e as vulnerabilidades nessas instâncias do Amazon EC2.

Detecção de clusters e nós com o Amazon GuardDuty

O Amazon GuardDuty é um serviço de detecção de ameaças que ajuda a proteger contas, contêineres, workloads e dados no ambiente da AWS. Entre outros recursos, o GuardDuty oferece os dois recursos a seguir que detectam possíveis ameaças aos seus clusters do EKS: Proteção do EKS e Monitoramento de runtime.

Para obter mais informações, consulte the section called “Amazon GuardDuty”.

Validação de conformidade para clusters do Amazon EKS

Para saber se um serviço AWS está dentro do escopo de programas de conformidade específicos, consulte AWS services in Scope by Compliance Program e escolha o programa de conformidade do seu interesse. Para obter informações gerais, consulte Programas de conformidade da AWS.

Você pode fazer download de relatórios de auditoria de terceiros usando o AWS Artifact. Para obter mais informações, consulte Fazer o download de Relatórios no AWS Artifact.

Sua responsabilidade com relação à conformidade ao usar os serviços da AWS é determinada pela confidencialidade dos dados, pelos objetivos de conformidade da empresa e pelos regulamentos e leis aplicáveis. A AWS fornece os seguintes recursos para ajudar com a conformidade:

- Governança e conformidade de segurança: esses guias de implementação de solução abordam considerações sobre a arquitetura e fornecem etapas para implantar recursos de segurança e conformidade.
- Referência de serviços qualificados para HIPAA: lista os serviços qualificados para HIPAA. Nem todos os serviços da AWS são qualificados para a HIPAA.
- Recursos de compatibilidade da AWS – Esta coleção de guias e pastas de trabalho pode ser aplicada ao seu setor e local.
- Guias de conformidade do cliente da AWS: entenda o modelo de responsabilidade compartilhada sob a ótica da conformidade. Os guias resumem as práticas recomendadas para proteger os serviços do AWS e mapeiam as orientações para os controles de segurança em várias estruturas (incluindo National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI) e International Organization for Standardization (ISO)).
- Avaliação de recursos com regras no AWS Config Developer Guide - O serviço AWS Config avalia a conformidade das configurações do seus recursos com as práticas internas, as diretrizes do setor e as normas.
- AWS Security Hub - Esse serviço AWS fornece uma visão abrangente do seu estado de segurança em AWS. O Security Hub usa controles de segurança para avaliar os recursos da AWS e verificar a conformidade com os padrões e as práticas recomendadas do setor de segurança. Para obter uma lista dos serviços e controles aceitos, consulte a Referência de controles do Security Hub.
- Amazon GuardDuty - Este serviço AWS detecta possíveis ameaças às suas contas AWS, workloads, contêineres e dados, monitorando seu ambiente em busca de atividades suspeitas e mal-intencionadas. O GuardDuty pode ajudar você a atender a diversos requisitos de conformidade, como o PCI DSS, com o cumprimento dos requisitos de detecção de intrusões requeridos por determinadas estruturas de conformidade.
- Audit Manager da AWS: esse serviço da AWS ajuda a auditar continuamente o uso da AWS para simplificar a forma como você gerencia os riscos e a conformidade com regulamentos e padrões do setor.

Considerações de segurança sobre o Amazon Elastic Kubernetes Service

A seguir estão algumas considerações sobre a segurança da nuvem, pois elas afetam o Amazon EKS.

Tópicos

- Segurança da infraestrutura no Amazon EKS
- Entenda a resiliência nos clusters do Amazon EKS
- Prevenção do problema de confused deputy entre serviços no Amazon EKS

Segurança da infraestrutura no Amazon EKS

Como um serviço gerenciado, o Amazon Elastic Kubernetes Service é protegido pela segurança de rede global da AWS. Para obter informações sobre serviços de segurança da AWS e como a AWS protege a infraestrutura, consulte Segurança na Nuvem AWS. Para projetar seu ambiente da AWS usando as práticas recomendadas de segurança da infraestrutura, consulte Proteção de Infraestrutura em Pilar de Segurança: AWS Estrutura bem arquitetada.

Você usa chamadas à API publicadas pela AWS para acessar o Amazon EKS por meio da rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o serviço de token de segurança da AWS (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Ao criar um cluster do Amazon EKS, você especifica as sub-redes da VPC para uso do cluster. O Amazon EKS exige sub-redes em, pelo menos, duas zonas de disponibilidade. Recomendamos uma VPC com sub-redes públicas e privadas a fim de que o Kubernetes possa criar balanceadores de

carga públicos nas sub-redes públicas que fazem o balanceamento de carga do tráfego para pods em execução nos nós localizados em sub-redes privadas.

Para obter mais informações sobre as considerações da VPC, consulte the section called “Requisitos para sub-rede e VPC”.

Se você criar os grupos de nós e VPCs com os modelos do AWS CloudFormation fornecidos no passo a passo Introdução ao Amazon EKS, os grupos de segurança do ambiente de gerenciamento e dos nós serão configurados com nossas definições recomendadas.

Para obter mais informações sobre considerações de grupos de segurança, consulte the section called “Requisitos para grupos de segurança”.

Quando você cria um cluster, o Amazon EKS cria um endpoint para o servidor gerenciado de API do Kubernetes usado para se comunicar com o cluster (usando as ferramentas de gerenciamento do Kubernetes, como `kubectl`). Por padrão, esse endpoint do servidor de API é público para a internet, e o acesso ao servidor de API é protegido usando uma combinação do AWS Identity and Access Management (IAM) e do Role Based Access Control (RBAC) nativo do Kubernetes.

Você pode habilitar o acesso privado ao servidor de API do Kubernetes para que todas as comunicações entre os nós e o servidor de API fiquem na VPC. Você pode limitar os endereços IP que podem acessar o servidor de API pela Internet ou desativar completamente o acesso à Internet para o servidor de API.

Para obter mais informações sobre como modificar o acesso do endpoint do cluster, consulte the section called “Modificar o acesso ao endpoint do cluster”:

Você pode implementar políticas de rede do Kubernetes com a CNI da Amazon VPC ou com ferramentas de terceiros, como o Project Calico. Para obter mais informações sobre o uso do Amazon VPC CNI para políticas de rede, consulte the section called “Políticas do Kubernetes”. O Project Calico é um projeto de código-fonte aberto de terceiros. Para obter mais informações, consulte a documentação do Project Calico.

Acessar o Amazon EKS usando o AWS PrivateLink

Você pode usar o AWS PrivateLink para criar uma conexão privada entre seu VPC e o Amazon Elastic Kubernetes Service. Você pode acessar o Amazon EKS como se ele estivesse em sua VPC, sem o uso de um gateway de Internet, dispositivo NAT, conexão VPN ou conexão AWS Direct Connect. As instâncias na VPC não precisam de endereços IP públicos para acessar o Amazon EKS.

Você estabelece essa conexão privada criando um endpoint de interface alimentado pelo AWS PrivateLink. Criaremos um endpoint de interface de rede em cada sub-rede que você habilitar para o endpoint de interface. Essas são interfaces de rede gerenciadas pelo solicitante que servem como ponto de entrada para o tráfego destinado ao Amazon EKS.

Para obter mais informações, consulte Acessar serviços da AWS por meio do AWS PrivateLink no Guia do AWS PrivateLink.

Antes de começar

Antes de começar, certifique-se de ter realizado as seguintes tarefas:

- Revise Access an AWS service using an interface VPC endpoint no Guia do AWS PrivateLink

Considerações

- Suporte e limitações: os endpoints da interface do Amazon EKS permitem acesso seguro a todas as ações da API do Amazon EKS da sua VPC, mas vêm com limitações específicas: eles não oferecem suporte ao acesso às APIs do Kubernetes, pois estas têm um endpoint privado separado. Você não pode configurar o Amazon EKS para ser acessível somente por meio do endpoint da interface.
- Preços: o uso de endpoints de interface para o Amazon EKS gera cobranças padrão do AWS PrivateLink: cobranças por hora para cada endpoint provisionado em cada zona de disponibilidade e cobranças de processamento de dados do tráfego pelo endpoint. Para saber mais, consulte Preço do AWS PrivateLink.
- Segurança e controle de acesso: recomendamos aprimorar a segurança e controlar o acesso com estas configurações adicionais: use políticas de endpoints da VPC para controlar o acesso ao Amazon EKS por meio do endpoint da interface, associe grupos de segurança a interfaces de rede de endpoints para gerenciar tráfego e use logs de fluxo da VPC para capturar e monitorar o tráfego IP de e para os endpoints da interface, com logs publicáveis no Amazon CloudWatch ou no Amazon S3. Para saber mais, consulte Control access to VPC endpoints using endpoint policies e Como registrar tráfego IP em log com logs de fluxo da VPC.
- Opções de conectividade: os endpoints de interface oferecem opções flexíveis de conectividade usando acesso on-premises (conecte seu data center on-premises a uma VPC com o endpoint de interface usando o AWS Direct Connect ou o AWS Site-to-Site VPN) ou via conectividade entre VPCs (use o AWS Transit Gateway ou o emparelhamento da VPC para conectar outras VPCs à VPC com o endpoint de interface, mantendo o tráfego na rede da AWS).

- Suporte para a versão IP: endpoints criados antes de agosto de 2024 são compatíveis somente com o IPv4 usando eks.region.amazonaws.com. Novos endpoints criados após agosto de 2024 são compatíveis com o IPv4 e IPv6 de pilha dupla (por exemplo, eks.region.amazonaws.com, eks.region.api.aws).
- Disponibilidade regional: o AWS PrivateLink para a API do EKS não está disponível nas regiões Ásia-Pacífico (Malásia) (ap-southeast-5), Ásia-Pacífico (Tailândia) (ap-southeast-7), México (Centro) (mx-central-1) e Ásia-Pacífico (Taipei) (ap-east-2). O suporte do AWS PrivateLink para o eks-auth (Identidade de Pods do EKS) está disponível na região Ásia-Pacífico (Malásia) (ap-southeast-5).

Criar um endpoint de interface de VPC para o Amazon EKS

Você pode criar um endpoint de interface para o Amazon EKS usando o console do Amazon VPC ou a interface de linha de comando AWS (AWS CLI). Para obter mais informações, consulte Criar um endpoint VPC no AWS PrivateLink Guide.

Crie um endpoint de interface para o Amazon EKS usando os seguintes nomes de serviço:

API do EKS

- com.amazonaws.region-code.eks
- com.amazonaws.region-code.eks-fips (para endpoints compatíveis com FIPS)

API de autenticação do EKS (Identidade de Pods do EKS)

- com.amazonaws.region-code.eks-auth

Recurso de DNS privado para endpoints de interface do Amazon EKS

O recurso de DNS privado, habilitado por padrão para endpoints de interface do Amazon EKS e outros serviços da AWS, viabiliza solicitações de API seguras e privadas usando nomes de DNS regionais padrão. Esse recurso garante que as chamadas de API sejam roteadas pelo endpoint de interface pela rede privada da AWS, aprimorando a segurança e a performance.

O recurso de DNS privado é ativado automaticamente quando você cria um endpoint de interface para o Amazon EKS ou outros serviços da AWS. Para habilitar, você precisa configurar sua VPC corretamente definindo atributos específicos:

- enableDnsHostnames: permite que as instâncias dentro da VPC tenham nomes de host DNS.
- enableDnsSupport: habilita a resolução de DNS em toda a VPC.

Para obter instruções passo a passo para verificar ou modificar essas configurações, consulte Visualizar e atualizar atributos de DNS para sua VPC.

Nomes de DNS e tipos de endereço IP

Com o recurso de DNS privado habilitado, você pode usar nomes DNS específicos para se conectar ao Amazon EKS, e essas opções evoluem com o tempo:

- eks.region.amazonaws.com: o nome DNS tradicional, resolvido apenas para endereços IPv4 antes de agosto de 2024. Para endpoints existentes atualizados para pilha dupla, esse nome resolve para endereços IPv4 e IPv6.
- eks.region.api.aws: disponível para novos endpoints criados após agosto de 2024, esse nome DNS de pilha dupla resolve para endereços IPv4 e IPv6.

Depois de agosto de 2024, os novos endpoints de interface vêm com dois nomes DNS, e você pode optar pelo tipo de endereço IP de pilha dupla. Para endpoints existentes, a atualização para pilha dupla modifica eks.region.amazonaws.com para ser compatível tanto com IPv4 quanto com IPv6.

Como usar o recurso DNS privado

Depois de configurado, o recurso de DNS privado pode ser integrado aos seus fluxos de trabalho, oferecendo os seguintes recursos:

- Solicitações de API: use os nomes DNS regionais padrão, `eks.region.amazonaws.com` ou `eks.region.api.aws`, com base na configuração do seu endpoint para fazer solicitações de API para o Amazon EKS.
- Compatibilidade de aplicações: suas aplicações existentes que chamam as APIs do EKS não precisam de alterações para aproveitar esse recurso.
- AWS CLI com pilha dupla: para usar os endpoints de pilha dupla com a AWS CLI, consulte a configuração de endpoints de pilha dupla e FIPS no Guia de referência de ferramentas e SDKs da AWS.
- Roteamento automático: qualquer chamada para o endpoint de serviço padrão do Amazon EKS é automaticamente direcionada pelo endpoint de interface, garantindo conectividade privada e segura.

Entenda a resiliência nos clusters do Amazon EKS

A infraestrutura global da AWS é criada com base em regiões da AWS e zonas de disponibilidade. As regiões da AWS As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, as quais são conectadas com baixa latência, alto throughput e redes altamente redundantes. Com as zonas de disponibilidade, é possível projetar e operar aplicações e bancos de dados que executam o failover automaticamente entre as zonas de disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data center tradicionais.

O Amazon EKS executa e escala planos de controle do Kubernetes em várias zonas de disponibilidade da AWS para garantir a alta disponibilidade. O Amazon EKS escala automaticamente instâncias do ambiente de gerenciamento baseadas em carga, detecta e substitui instâncias não íntegras do ambiente de gerenciamento e aplica patches automaticamente ao ambiente de gerenciamento. Depois de iniciar uma atualização de versão, o Amazon EKS atualiza seu ambiente de gerenciamento para você, mantendo a alta disponibilidade do ambiente de gerenciamento durante a atualização.

Esse ambiente de gerenciamento consiste em pelo menos duas instâncias de servidor de API e três instâncias de `etcd` que são executadas em três zonas de disponibilidade em uma região AWS. Amazon EKS:

- Monitora ativamente a carga nas instâncias do plano de controle e as escala automaticamente para garantir alta performance.
- Detecta e substitui automaticamente instâncias não íntegras do plano de controle, reiniciando-as nas zonas de disponibilidade da região da AWS, conforme a necessidade.
- Aproveita a arquitetura de Regiões da AWS a fim de manter alta disponibilidade. Por isso, o Amazon EKS é capaz de oferecer um SLA para disponibilidade do endpoint do servidor de API.

Para obter mais informações sobre regiões e zonas de disponibilidade da AWS, consulte Infraestrutura global da AWS.

Prevenção do problema de confused deputy entre serviços no Amazon EKS

O problema de confused deputy é uma questão de segurança em que uma entidade que não tem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executar a ação. Na AWS, a personificação entre serviços pode resultar no problema do ‘confused deputy’. A

personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). O serviço de chamada pode ser manipulado a usar suas permissões para atuar nos recursos de outro cliente de modo a não ter permissão de acesso. Para evitar isso, a AWS fornece ferramentas que ajudam você a proteger seus dados para todos os serviços com entidades principais de serviço que receberam acesso aos recursos em sua conta.

Recomendamos o uso das chaves de contexto de condição global `aws:SourceArn`, `aws:SourceAccount` em políticas de recursos para limitar as permissões que o Amazon Elastic Kubernetes Service (Amazon EKS) concede para o recurso a outro serviço.

`aws:SourceArn`

Use `aws:SourceArn` se quiser associar apenas um recurso ao acesso entre serviços.

`aws:SourceAccount`

Use `aws:SourceAccount` se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

A maneira mais eficaz de se proteger contra o problema do confused deputy é usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. Se você não souber o ARN completo do recurso ou especificar vários recursos, use a chave de condição de contexto global `aws:SourceArn` com caracteres curinga (*) para as partes desconhecidas do ARN. Por exemplo, `arn:aws:<servicename>:*:<123456789012>:*`.

Se o valor do `aws:SourceArn` não contiver o ID da conta, como um ARN de bucket do Amazon S3, você deverá usar ambos, a `aws:SourceAccount` e o `aws:SourceArn` para limitar as permissões.

Prevenção contra o problema confused deputy entre serviços no perfil do cluster do Amazon EKS

É necessário um perfil do IAM de cluster do Amazon EKS para cada cluster. Os clusters do Kubernetes gerenciados pelo Amazon EKS usam esse perfil para gerenciar nós, e o provedor de nuvem legado usa esse perfil para criar balanceadores de carga com o Elastic Load Balancing para serviços. Essas ações de cluster só podem afetar a mesma conta; portanto, recomendamos que você limite cada perfil de cluster a esse cluster e a essa conta. Essa é uma aplicação específica da AWS recomendação de seguir o princípio do privilégio mínimo em sua conta.

Formato do ARN de origem

O valor `aws:SourceArn` deve ser o ARN de um cluster do EKS no formato `arn:aws:eks:region:account:cluster/cluster-name` . Por exemplo, ., `arn:aws:eks:us-west-2:123456789012:cluster/my-cluster` .

Formato de política de confiança para perfis de cluster do EKS

O exemplo a seguir mostra como é possível usar as chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` no Amazon EKS, a fim de evitar o problema de confused deputy.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:eks:us-west-2:123456789012:cluster/my-cluster"
          },
        "StringEquals": {
            "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Considerações de segurança para o Kubernetes

Veja abaixo algumas considerações sobre a segurança na nuvem, pois elas afetam o Kubernetes nos clusters do Amazon EKS. Para uma análise mais aprofundada dos controles e práticas de segurança no Kubernetes, consulte Cloud Native Security and Kubernetes na documentação do Kubernetes.

Tópicos

- Proteger workloads com certificados do Kubernetes

- Entender as funções e os usuários do RBAC criados pelo Amazon EKS
- Criptografar segredos do Kubernetes com o KMS em clusters existentes
- Uso dos segredos do AWS Secrets Manager com os pods do Amazon EKS
- Criptografia envelopada padrão para todos os dados de API do Kubernetes

Proteger workloads com certificados do Kubernetes

A API Certificates do Kubernetes automatiza o provisionamento de credenciais X.509. Ela tem uma interface de linha de comandos para os clientes de APIs do Kubernetes solicitarem e receberem certificados X.509 de uma autoridade de certificação (CA). Você pode usa o recurso `CertificateSigningRequest` (CSR) para solicitar que um signatário indicado assine o certificado. Suas solicitações são aprovadas ou negadas antes de serem assinadas. O Kubernetes é compatível com signers incorporados e personalizados com comportamentos bem definidos. Dessa maneira, os clientes podem prever o que acontece com suas CSRs. Para saber mais sobre assinatura de certificados, consulte solicitações de assinatura.

Um dos signatários integrados é `kubernetes.io/legacy-unknown`. A API `v1beta1` do recurso CSR respeitou esse signatário legacy-unknown. Porém, a API estável `v1` de CSR não permite que o `signerName` seja definido como `kubernetes.io/legacy-unknown`.

Se você desejar usar a CA do Amazon EKS para gerar certificados em seus clusters, deverá usar um signatário personalizado. Para usar a versão `v1` da API de CSR e gerar um novo certificado, você deverá migrar todos os manifestos e clientes de API existentes. Certificados existentes criados com a API `v1beta1` existente são válidos e funcionarão até o certificado expirar. Essa transmissão inclui o seguinte:

- Distribuição de confiança: nada. Não há confiança ou distribuição padrão para esse signer em um cluster do Kubernetes.
- Requerentes permitidos: qualquer
- Extensões x509 permitidas: respeita subjectAltName e extensões de uso de chaves e descarta outras extensões
- Usos de chaves permitidos: não deve incluir usos além de ["criptografia de chaves”, “assinatura digital”, “autenticação de servidor"]

> **Note**
>
> A assinatura de certificados de clientes não é possível.

- Vida útil de expiração/certificado: um ano (padrão e máximo)
- Bit de CA permitido/proibido: não permitido

Exemplo de geração de CSR com signerName

Estas etapas mostram como gerar um certificado de serviço para o nome de DNS `myserver.default.svc` usando `signerName: beta.eks.amazonaws.com/app-serving`. Use isso como orientação para seu próprio ambiente.

1. Execute o comando `openssl genrsa -out myserver.key 2048` para gerar uma chave RSA privada.

```
openssl genrsa -out myserver.key 2048
```

2. Use o seguinte comando para gerar uma solicitação de certificado:

```
openssl req -new -key myserver.key -out myserver.csr -subj "/CN=myserver.default.svc"
```

3. Gere um valor em `base64` para a solicitação de CSR e armazene-o em uma variável para uso em uma etapa posterior.

```
base_64=$(cat myserver.csr | base64 -w 0 | tr -d "
")
```

4. Execute o seguinte comando para criar um arquivo chamado `mycsr.yaml`. No exemplo a seguir, `beta.eks.amazonaws.com/app-serving` é `signerName`.

```
cat >mycsr.yaml <<EOF
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: $base_64
  signerName: beta.eks.amazonaws.com/app-serving
```

```
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

5. Envie a CSR.

```
kubectl apply -f mycsr.yaml
```

6. Aprove o certificado de serviço.

```
kubectl certificate approve myserver
```

7. Confira se o certificado foi emitido.

```
kubectl get csr myserver
```

   Veja um exemplo de saída abaixo.

```
NAME       AGE     SIGNERNAME                              REQUESTOR          CONDITION
myserver   3m20s   beta.eks.amazonaws.com/app-serving   kubernetes-admin
 Approved,Issued
```

8. Exporte o certificado emitido.

```
kubectl get csr myserver -o jsonpath='{.status.certificate}'| base64 -d >
 myserver.crt
```

Entender as funções e os usuários do RBAC criados pelo Amazon EKS

Quando você cria um cluster do Kubernetes, várias identidades padrão do Kubernetes são criadas nesse cluster para o funcionamento adequado do Kubernetes. O Amazon EKS cria identidades do Kubernetes para cada um de seus componentes padrão. As identidades fornecem controle de autorização baseado em perfil (RBAC) do Kubernetes para os componentes do cluster. Para obter mais informações, consulte Usar a autorização de RBAC na documentação do Kubernetes.

Quando você instala complementos opcionais no cluster, identidades adicionais do Kubernetes talvez sejam adicionadas a ele. Para obter mais informações sobre as identidades não abordadas neste tópico, consulte a documentação do complemento.

Você pode ver a lista de identidades do Kubernetes criadas pelo Amazon EKS no cluster usando o Console de gerenciamento da AWS ou a ferramenta de linha de comandos do `kubectl`. Todas as identidades de usuário aparecem nos logs de auditoria do `kube`, disponíveis a você por meio do Amazon CloudWatch.

Console de gerenciamento da AWS

Pré-requisito

A entidade principal do IAM que você usar deve ter as permissões descritas em Permissões necessárias.

Para visualizar as identidades criadas pelo Amazon EKS usando o Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Na lista Clusters, escolha o cluster que contém as identidades do que você deseja visualizar.
3. Escolha a guia Recursos.
4. Em Resource types (Tipos de recursos), escolha Authorization (Autorização).
5. Escolha ClusterRoles, ClusterRoleBindings, Roles ou RoleBindings. Todos os recursos prefaciados com eks são criados pelo Amazon EKS. Outros recursos de identidade criados pelo Amazon EKS são:
   - O ClusterRole e o ClusterRoleBinding denominados aws-node. Os recursos aws-node são compatíveis com o plug-in CNI da Amazon VPC para Kubernetes, que o Amazon EKS instala em todos os clusters.
   - Um ClusterRole denominado vpc-resource-controller-role e um ClusterRoleBinding denominado vpc-resource-controller-rolebinding. Esses recursos são compatíveis com o controlador de recursos da Amazon VPC, que o Amazon EKS instala em todos os clusters.

   Além dos recursos que você vê no console, as seguintes identidades de usuário especiais existem no cluster, embora não estejam visíveis na sua configuração:
   - **`eks:cluster-bootstrap`** : usado para operações `kubectl` durante a inicialização do cluster.
   - **`eks:support-engineer`** – usado para operações de gerenciamento de cluster.
6. Escolha um recurso específico para visualizar detalhes sobre ele. Por padrão, as informações são mostradas na visualização estruturada. No canto superior direito da página de detalhes, você pode escolher Raw view (Visualização bruta) para ver todas as informações sobre o recurso.

Kubectl

Pré-requisito

A entidade que você usa (AWS Identity and Access Management [IAM] ou OpenID Connect [OIDC]) para listar os recursos do Kubernetes no cluster deve ser autenticada pelo IAM ou pelo seu provedor de identidade OIDC. A entidade deve receber permissões para usar os verbos `get` e `list` do Kubernetes para os recursos `Role`, `ClusterRole`, `RoleBinding` e `ClusterRoleBinding` no cluster com o qual você deseja que a entidade trabalhe. Para obter mais informações sobre conceder às entidades do IAM acesso ao cluster, consulte the section called “Acesso à API do Kubernetes”. Para obter mais informações sobre conceder às entidades autenticadas por seu próprio provedor OIDC acesso ao cluster, consulte the section called “Vincular um provedor OIDC”.

Para visualizar as identidades criadas pelo Amazon EKS usando o **`kubectl`**

Execute o comando para o tipo de recurso que você deseja ver. Todos os recursos retornados precedidos por eks são criados pelo Amazon EKS. Além dos recursos que retornados na saída dos comandos, as seguintes identidades de usuário especiais existem no cluster, embora não estejam visíveis na sua configuração:

- **`eks:cluster-bootstrap`** : usado para operações `kubectl` durante a inicialização do cluster.
- **`eks:support-engineer`** – usado para operações de gerenciamento de cluster.

ClusterRoles: `ClusterRoles` estão no escopo do cluster, portanto, qualquer permissão concedida a um perfil se aplicará aos recursos em qualquer namespace do Kubernetes no cluster.

O comando a seguir retorna todos os `ClusterRoles` do Kubernetes criados pelo Amazon EKS no cluster.

```
kubectl get clusterroles | grep eks
```

Além dos `ClusterRoles` retornados na saída que são precedidos por eks, existem os `ClusterRoles` a seguir.

- **`aws-node`** : este `ClusterRole` é compatível com o plug-in CNI da Amazon VPC para Kubernetes, que o Amazon EKS instala em todos os clusters.
- **`vpc-resource-controller-role`** este `ClusterRole` é compatível com o controlador de recursos do Amazon VPC, que o Amazon EKS instala em todos os clusters.

Para ver a especificação de um `ClusterRole`, substitua *`eks:k8s-metrics`* no comando a seguir por um `ClusterRole` retornado na saída do comando anterior. O exemplo a seguir retorna a especificação para `ClusterRole` *`eks:k8s-metrics`*.

```
kubectl describe clusterrole eks:k8s-metrics
```

Veja um exemplo de saída abaixo.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names  Verbs
  ---------         -----------------  --------------  -----
                    [/metrics]         []              [get]
  endpoints         []                 []              [list]
  nodes             []                 []              [list]
  pods              []                 []              [list]
  deployments.apps  []                 []              [list]
```

ClusterRoleBindings: `ClusterRoleBindings` estão no escopo do cluster.

O comando a seguir retorna todos os `ClusterRoleBindings` do Kubernetes criados pelo Amazon EKS no cluster.

```
kubectl get clusterrolebindings | grep eks
```

Além dos `ClusterRoleBindings` retornados na saída, existem os `ClusterRoleBindings` a seguir.

- **`aws-node`** : este `ClusterRoleBinding` é compatível com o plug-in CNI da Amazon VPC para Kubernetes, que o Amazon EKS instala em todos os clusters.
- **`vpc-resource-controller-rolebinding`** este `ClusterRoleBinding` é compatível com o controlador de recursos do Amazon VPC, que o Amazon EKS instala em todos os clusters.

Para ver a especificação de um `ClusterRoleBinding`, substitua *`eks:k8s-metrics`* no comando a seguir por um `ClusterRoleBinding` retornado na saída do comando anterior. O exemplo a seguir retorna a especificação para `ClusterRoleBinding` *`eks:k8s-metrics`*.

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

Veja abaixo um exemplo de saída.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Perfis: `Roles` estão no escopo de um namespace do Kubernetes. Todos os `Roles` criados pelo Amazon EKS estão no escopo do namespace do `kube-system`.

O comando a seguir retorna todos os `Roles` do Kubernetes criados pelo Amazon EKS no cluster.

```
kubectl get roles -n kube-system | grep eks
```

Para ver a especificação de um `Role`, substitua *`eks:k8s-metrics`* no comando a seguir pelo nome de um `Role` retornado na saída do comando anterior. O exemplo a seguir retorna a especificação para `Role` *`eks:k8s-metrics`*.

```
kubectl describe role eks:k8s-metrics -n kube-system
```

Veja abaixo um exemplo de saída.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names             Verbs
  ---------         -----------------  --------------             -----
  daemonsets.apps   []                 [aws-node]                 [get]
  deployments.apps  []                 [vpc-resource-controller]  [get]
```

RoleBindings: `RoleBindings` estão no escopo de um namespace do Kubernetes. Todos os `RoleBindings` criados pelo Amazon EKS estão no escopo do namespace do `kube-system`.

O comando a seguir retorna todos os `RoleBindings` do Kubernetes criados pelo Amazon EKS no cluster.

```
kubectl get rolebindings -n kube-system | grep eks
```

Para ver a especificação de um `RoleBinding`, substitua *`eks:k8s-metrics`* no comando a seguir por um `RoleBinding` retornado na saída do comando anterior. O exemplo a seguir retorna a especificação para `RoleBinding` *`eks:k8s-metrics`*.

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

Veja um exemplo de saída abaixo.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Criptografar segredos do Kubernetes com o KMS em clusters existentes

> **Important**
>
> Este procedimento se aplica somente aos clusters do EKS que executam o Kubernetes versão 1.27 ou mais antiga. Se você estiver executando o Kubernetes versão 1.28 ou mais recente, os segredos do Kubernetes serão protegidos com criptografia envelopada por padrão. Para obter mais informações, consulte the section called “Criptografia envelopada padrão para todos os dados de API do Kubernetes”.

Se você habilitar a criptografia de segredos, os segredos do Kubernetes serão criptografados usando a chave do AWS KMS selecionada. A chave do KMS deve atender às seguintes condições:

- Simétrica
- Pode criptografar e descriptografar dados
- Criado na mesma região AWS que o cluster
- Se a chave do KMS tiver sido criada em uma conta diferente, a entidade principal do IAM deverá ter acesso à chave do KMS.

Para obter mais informações, consulte Como permitir que as entidades principais do IAM em outras contas usem uma chave do KMS no Guia do desenvolvedor do serviço de gerenciamento de chaves AWS .

> **Warning**
>
> Não é possível desativar a criptografia de segredos depois de habilitá-la. Essa ação é irreversível.

eksctl

Este procedimento se aplica somente aos clusters do EKS que executam o Kubernetes versão 1.27 ou mais antiga. Para obter mais informações, consulte the section called “Criptografia envelopada padrão para todos os dados de API do Kubernetes”.

É possível ativar a criptografia de duas formas:

- Adicione criptografia ao cluster com um único comando.

  Para recriptografar seus segredos automaticamente, execute o comando a seguir.

```
eksctl utils enable-secrets-encryption \
    --cluster my-cluster \
    --key-arn arn:aws:kms:region-code:account:key/key
```

  Para optar por não recriptografar seus segredos automaticamente, execute o comando a seguir.

```
eksctl utils enable-secrets-encryption
```

```
    --cluster my-cluster \
    --key-arn arn:aws:kms:region-code:account:key/key \
    --encrypt-existing-secrets=false
```

- Adicione criptografia ao cluster com um arquivo `kms-cluster.yaml`.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

secretsEncryption:
  keyARN: arn:aws:kms:region-code:account:key/key
```

Para que seus segredos sejam recriptografados automaticamente, execute o comando a seguir.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

Para optar por não recriptografar seus segredos automaticamente, execute o comando a seguir.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-
secrets=false
```

Console de gerenciamento da AWS

a. Este procedimento se aplica somente aos clusters do EKS que executam o Kubernetes versão 1.27 ou mais antiga. Para obter mais informações, consulte the section called “Criptografia envelopada padrão para todos os dados de API do Kubernetes”.

b. Abra o console do Amazon EKS.

c. Escolha o cluster ao qual você deseja adicionar a criptografia do KMS.

d. Escolha a guia Overview (Visão geral) (selecionada por padrão).

e. Role para baixo até Secrets encryption (Criptografia de segredos) e escolha Enable (Habilitar).

f. Selecione uma chave na lista suspenso e escolha Enable (Habilitar). Se nenhuma chave estiver listada, primeiro você deve criar uma. Para obter mais informações, consulte Criar chaves

g. Escolha o botão Confirm (Confirmar) para usar a chave escolhida.

AWS CLI

a. Este procedimento se aplica somente aos clusters do EKS que executam o Kubernetes versão 1.27 ou mais antiga. Para obter mais informações, consulte the section called “Criptografia envelopada padrão para todos os dados de API do Kubernetes”.

b. Associe a configuração de criptografia de segredos ao seu cluster usando o seguinte comando da CLI AWS. Substitua os valores de exemplo pelos seus próprios.

```
aws eks associate-encryption-config \
    --cluster-name my-cluster \
    --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws:kms:region-code:account:key/key"}}]'
```

Veja abaixo um exemplo de saída.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734,
    "errors": []
  }
}
```

c. Você pode monitorar o status da atualização da criptografia com o comando a seguir. Utilize o `cluster name` e o `update ID` específicos que foram retornados na saída anterior. Quando um status de `Successful` for exibido, significa que a atualização está concluída.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id 3141b835-8103-423a-8e68-12c2521ffa4d
```

Veja um exemplo de saída abaixo.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734>,
    "errors": []
  }
}
```

d. Para verificar se a criptografia está habilitada no cluster, execute comando `describe-cluster`. A resposta contém uma string `EncryptionConfig`.

```
aws eks describe-cluster --region region-code --name my-cluster
```

Depois de habilitar a criptografia no cluster, você precisa criptografar todos os segredos existentes com a nova chave:

> **Note**
>
> Se você utilizar `eksctl`, apenas será necessário executar o seguinte comando se você optar por não recriptografar seus segredos automaticamente:

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-
encryption-timestamp="time value"
```

> **Warning**
>
> Se você habilitar a criptografia de segredos para um cluster existente e se algum dia a chave do KMS usada for excluída, não haverá maneira de recuperar o cluster. Se você excluir a chave do KMS, colocará permanentemente o cluster em um estado degradado. Para informações, consulte Exclusão das chaves do AWS KMS.

> **Note**
>
> Por padrão, o comando `create-key` cria uma chave KMS de criptografia simétrica com uma política de chave que concede à conta acesso de administrador raiz às ações e aos recursos do KMS AWS. Se você quiser reduzir o escopo das permissões, certifique-se de que as ações `kms:DescribeKey` e `kms:CreateGrant` sejam permitidas na política para a entidade principal que chama a API `create-cluster`.
> Para clusters que usam criptografia envelopada do KMS, são necessárias permissões `kms:CreateGrant`. A condição `kms:GrantIsForAWSResource` não tem suporte para a ação CreateCluster e não deve ser utilizada nas políticas do KMS para controlar permissões `kms:CreateGrant` de usuários que executam CreateCluster.

Uso dos segredos do AWS Secrets Manager com os pods do Amazon EKS

Para mostrar os segredos do Secrets Manager e os parâmetros do Parameter Store como arquivos montados nos pods do Amazon EKS, você pode usar o AWS Secrets and Configuration Provider (ASCP) para o driver CSI do armazenamento de segredos do Kubernetes.

Com o ASCP, você pode armazenar e gerenciar segredos no Secrets Manager e, em seguida, recuperá-los por meio de workloads executadas no Amazon EKS. Você pode usar os perfis e as políticas do IAM para limitar a pods específicos do Amazon EKS em um cluster o acesso aos seus segredos. O ASCP recupera a identidade de pods e troca a identidade por um perfil do IAM. O ASCP assume o perfil do IAM do pod e, então, consegue recuperar segredos do Secrets Manager autorizados para esse perfil.

Se você usar a alternância automática do Secrets Manager para seus segredos, também poderá usar o recurso de reconciliador de alternância do driver da CSI do armazenamento de segredos para garantir que está recuperando o segredo mais recente do Secrets Manager.

> **Note**
>
> Não há suporte para grupos de nós do AWS Fargate (Fargate).

Para obter mais informações, consulte Uso dos segredos do Secrets Manager o Amazon EKS no Guia do usuário do AWS Secrets Manager.

Criptografia envelopada padrão para todos os dados de API do Kubernetes

O Amazon Elastic Kubernetes Service (Amazon EKS) fornece criptografia envelopada padrão para todos os dados de API do Kubernetes nos clusters do EKS em execução no Kubernetes versão 1.28 ou mais recente.

A criptografia envelopada protege os dados que você armazena com o servidor de API do Kubernetes. Por exemplo, a criptografia envelopada se aplica à configuração do cluster do Kubernetes, como `ConfigMaps`. A criptografia envelopada não se aplica aos dados em nós ou volumes do EBS. Anteriormente, o EKS era compatível com a criptografia de segredos do Kubernetes, e agora essa criptografia envelopada se estende a todos os dados de API do Kubernetes.

Ela oferece uma experiência padrão gerenciada que implementa a defesa profunda para as aplicações do Kubernetes e não requer nenhuma ação de sua parte.

O Amazon EKS usa o AWS Key Management Service (KMS) com o provedor v2 do Kubernetes para KMS para essa camada adicional de segurança com uma chave de propriedade da Amazon Web Services e a opção de você trazer sua própria chave gerenciada pelo cliente (CMK) do AWS KMS.

Como compreender a criptografia envelopada

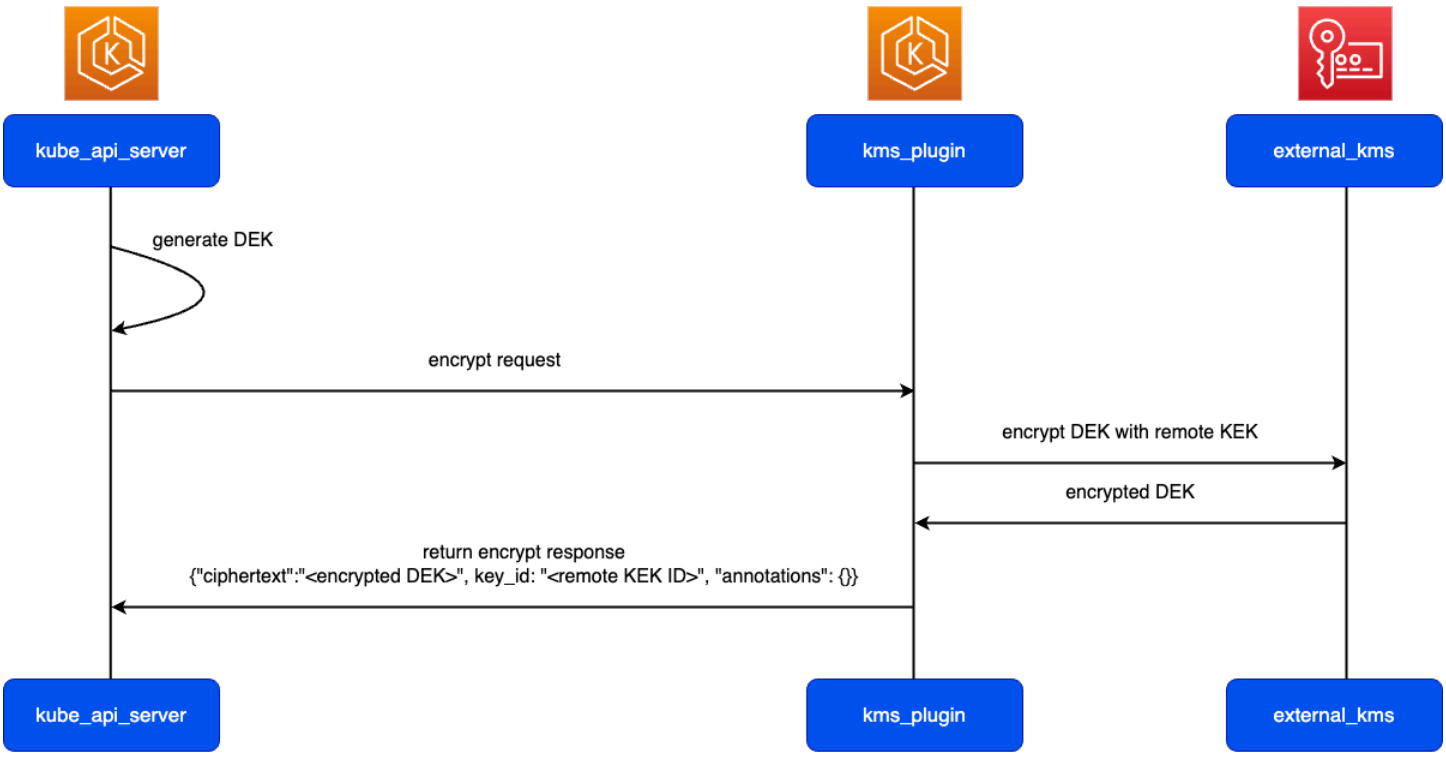
A criptografia envelopada é o processo de criptografar dados de texto simples com uma chave de criptografia de dados (DEK) antes de serem enviados ao datastore (etcd), e depois criptografando a DEK com uma chave raiz do KMS que é armazenada em um sistema KMS remoto e gerenciado de forma centralizada (AWS KMS). Esta é uma estratégia de defesa em profundidade porque protege os dados com uma chave de criptografia (DEK) e, depois, adiciona outra camada de segurança, protegendo essa DEK com uma chave de criptografia separada e armazenada com segurança, denominada chave de criptografia de chave (KEK).

Como o Amazon EKS habilita a criptografia envelopada padrão com o KMS v2 e o AWS KMS

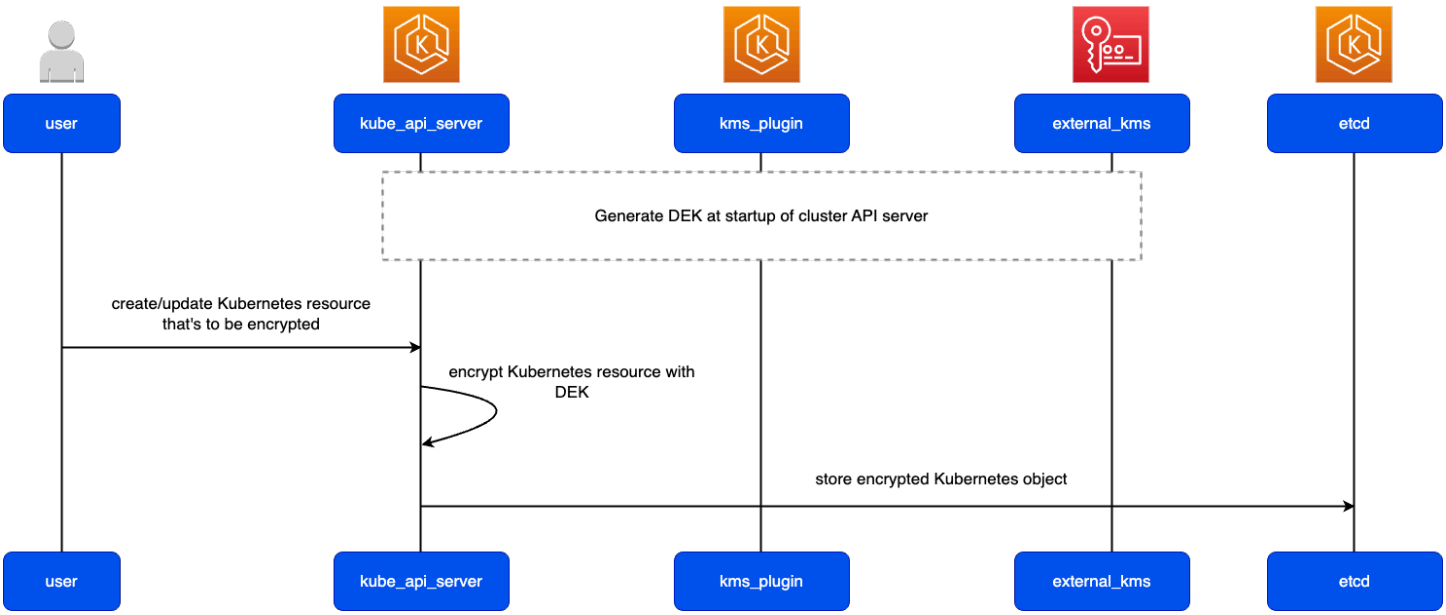
O Amazon EKS usa o KMS v2 para implementar a criptografia envelopada padrão para todos os dados de API no ambiente de gerenciamento controlado do Kubernetes antes que persistam no banco de dados etcd. Na inicialização, o servidor de API do cluster gera uma chave de criptografia de dados (DEK) de uma semente secreta combinada com dados gerados aleatoriamente. Também na inicialização, o servidor de API faz uma chamada para o plug-in do KMS para criptografar a semente da DEK usando uma chave de criptografia de chave (KEK) remota do AWS KMS. Essa é uma chamada única executada na inicialização do servidor de API e na alternância da KEK. Em seguida, o servidor de API armazena em cache a semente criptografada da DEK. Depois, o servidor de API usa a semente em cache da DEK para gerar outras DEKs de uso único com base em uma Função de Derivação de Chave (KDF). Cada uma dessas DEKs geradas é usada apenas uma vez para criptografar um único recurso do Kubernetes antes de ser armazenado no etcd. Com o uso de uma semente em cache criptografada da DEK no KMS v2, o processo de criptografia dos recursos do Kubernetes no servidor de API é mais eficiente e econômico.

Por padrão, essa KEK é de propriedade da AWS, mas, opcionalmente, você pode trazer sua própria KEK do AWS KMS.

O diagrama abaixo mostra a geração e a criptografia de uma DEK na inicialização do servidor de API.



O diagrama de alto nível abaixo mostra a criptografia de um recurso do Kubernetes antes de ser armazenado no etcd.



Perguntas frequentes

Como a criptografia envelopada padrão melhora a postura de segurança do meu cluster do EKS?

Esse recurso reduz a área de superfície e o tempo em que os metadados e o conteúdo do cliente não são criptografados. Com a criptografia envelopada padrão, os metadados e o conteúdo do cliente só ficam não criptografados temporariamente na memória do kube-apiserver antes de serem armazenados no etcd. A memória do kube-apiserver é protegida pelo sistema Nitro. O Amazon EKS usa somente instâncias do EC2 baseadas em Nitro para o ambiente de gerenciamento controlado do Kubernetes. Essas instâncias têm designs de controle de segurança que impedem que qualquer sistema ou pessoa acesse sua memória.

Qual versão do Kubernetes eu preciso executar para ter esse recurso?

Para que a criptografia envelopada padrão seja habilitada, o cluster do Amazon EKS precisa estar executando a versão 1.28 ou posterior do Kubernetes.

Meus dados ainda estarão seguros se eu estiver executando uma versão do cluster do Kubernetes não compatível com esse recurso?

Sim. Na AWS, a segurança é a nossa maior prioridade. Baseamos toda a nossa transformação e inovação digital nas mais elevadas práticas operacionais de segurança e mantemos o compromisso de elevar esse nível.

Todos os dados armazenados no etcd são criptografados no nível de disco para cada cluster do EKS, independentemente da versão do Kubernetes que está sendo executada. O EKS usa chaves raiz que geram chaves de criptografia de volume gerenciadas pelo serviço do EKS. Além disso, todos os clusters do Amazon EKS são executados em uma VPC isolada usando máquinas virtuais específicas do cluster. Por causa dessa arquitetura e de nossas práticas em relação à segurança operacional, o Amazon EKS alcançou várias classificações e padrões de conformidade, incluindo a elegibilidade para SOC 1,2,3, PCI-DSS, ISO e HIPAA. Esses padrões e classificações de conformidade são mantidos para todos os clusters do EKS com ou sem criptografia envelopada padrão.

Como a criptografia envelopada funciona no Amazon EKS?

Na inicialização, o servidor de API do cluster gera uma chave de criptografia de dados (DEK) de uma semente secreta combinada com dados gerados aleatoriamente. Também na inicialização, o servidor de API faz uma chamada para o plug-in do KMS para criptografar a DEK usando uma chave de criptografia de chave (KEK) remota do AWS KMS. Essa é uma chamada única executada na

inicialização do servidor de API e na alternância da KEK. Em seguida, o servidor de API armazena em cache a semente criptografada da DEK. Depois, o servidor de API usa a semente em cache da DEK para gerar outras DEKs de uso único com base em uma Função de Derivação de Chave (KDF). Cada uma dessas DEKs geradas é usada apenas uma vez para criptografar um único recurso do Kubernetes antes de ser armazenado no etcd.

É importante observar que o servidor de API realiza chamadas adicionais para verificar a integridade e a funcionalidade normal da integração do AWS KMS. Essas verificações de integridade adicionais ficam visíveis no AWS CloudTrail.

Preciso fazer alguma coisa ou alterar qualquer permissão para que esse recurso funcione no cluster do EKS?

Não, você não precisa tomar nenhuma medida. A criptografia envelopada no Amazon EKS agora é uma configuração padrão habilitada em todos os clusters que executam a versão 1.28 ou mais recente do Kubernetes. A integração com o AWS KMS é estabelecida pelo servidor de API do Kubernetes gerenciado pela AWS. Isso significa que você não precisa configurar nenhuma permissão para começar a usar a criptografia do KMS no cluster.

Como posso saber se a criptografia envelopada padrão está habilitada no cluster?

Se migrar para usar sua própria CMK, você verá o ARN da chave do KMS associada ao cluster. Além disso, você pode visualizar os logs de eventos do AWS CloudTrail associados ao uso da CMK do cluster.

Se o cluster usar uma chave de propriedade da AWS, isso será detalhado no console do EKS (excluindo o ARN da chave).

A AWS pode acessar a chave de propriedade da AWS usada para criptografia envelopada padrão no Amazon EKS?

Não. A AWS tem controles de segurança rigorosos no Amazon EKS que impedem que qualquer pessoa acesse qualquer chave de criptografia de texto simples usada para proteger dados no banco de dados do etcd. Essas medidas de segurança também se aplicam à chave do KMS de propriedade da AWS.

A criptografia envelopada padrão está habilitada no meu cluster existente do EKS?

Se você estiver executando um cluster do Amazon EKS com a versão 1.28 ou mais recente do Kubernetes, a criptografia envelopada de todos os dados de API do Kubernetes estará habilitada. Para os clusters existentes, o Amazon EKS usa o RBAC ClusterRole `eks:kms-storage-`

`migrator` para migrar dados que antes não tinham criptografia envelopada no etcd para esse novo estado de criptografia.

O que isso significa se eu já habilitei a criptografia envelopada para os segredos no cluster do EKS?

Se você tiver uma chave gerenciada pelo cliente (CMK) existente no KMS que foi usada para a criptografia envelopada dos seus segredos do Kubernetes, essa mesma chave será usada como KEK para a criptografia envelopada de todos os tipos de dados de API do Kubernetes no cluster.

Há algum custo adicional em executar um cluster do EKS com a criptografia envelopada padrão?

Não haverá custo adicional associado ao ambiente de gerenciamento controlado do Kubernetes se você estiver usando uma chave de propriedade da Amazon Web Services para a criptografia envelopada padrão. Por padrão, todos os clusters do EKS em execução no Kubernetes versão 1.28 ou posterior usam uma chave de propriedade da Amazon Web Service. No entanto, se você usar sua própria chave do AWS KMS, o preço do KMS será aplicado.

Quanto custa para usar minha própria chave do AWS KMS para criptografar dados de API do Kubernetes no cluster?

Você paga US$ 1 por mês para armazenar qualquer chave personalizada criada ou importada para o KMS. O KMS cobra por solicitações de criptografia e descriptografia. Existe um nível gratuito de 20 mil solicitações por mês por conta, e você paga US$ 0,03 por cada 10 mil solicitações acima do nível gratuito por mês. Isso se aplica a todo o uso do KMS em uma conta e, portanto, o custo de usar sua própria chave do AWS KMS no cluster será afetado pelo uso dessa chave em outros clusters ou recursos da AWS em sua conta.

Minhas cobranças do KMS serão mais altas agora que minha chave gerenciada pelo cliente (CMK) está sendo usada para a criptografia envelopada de todos os dados de API do Kubernetes e não apenas dos segredos?

Não. Nossa implementação com o KMS v2 reduz significativamente o número de chamadas feitas para o AWS KMS. Por sua vez, ele reduzirá os custos associados à CMK, independentemente de os dados adicionais do Kubernetes serem criptografados ou descriptografados no cluster do EKS.

Como detalhado anteriormente, a semente da DEK gerada usada para criptografia de recursos do Kubernetes é armazenada localmente no cache do servidor de API do Kubernetes depois de ser criptografada com a KEK remota. Se a semente criptografada da DEK não estiver no cache do servidor de API, o servidor de API chamará o AWS KMS para criptografar a semente da DEK. Em seguida, o servidor de API armazena em cache a semente criptografada da DEK para uso futuro no cluster sem chamar o KMS. Da mesma forma, para solicitações de descriptografia, o servidor de

API chamará o AWS KMS para a primeira solicitação de descriptografia e, então, a semente de DEK descriptografada será armazenada em cache e usada para futuras operações de descriptografia.

Para obter mais informações, consulte KEP-3299: KMS v2 Improvements em Kubernetes Enhancements no GitHub.

Posso usar a mesma chave CMK para vários clusters do Amazon EKS?

Sim. Para usar uma chave novamente, você pode vinculá-la a um cluster na mesma região, associando o ARN ao cluster durante a criação. No entanto, se estiver usando a mesmo CMK para vários clusters do EKS, você deverá adotar as medidas necessárias para evitar a desabilitação arbitrária da CMK. Caso contrário, uma CMK desabilitada associada a vários clusters do EKS terá um maior impacto sobre os clusters que dependem dessa chave.

O que acontece com meu cluster do EKS se minha CMK ficar indisponível após a ativação da criptografia envelopada padrão?

Se você desabilitar uma chave do KMS, ela não poderá ser usada em nenhuma operação criptográfica. Sem acesso a uma CMK existente, o servidor de API não conseguirá criptografar e manter quaisquer objetos do Kubernetes recém-criados, bem como descriptografar quaisquer objetos do Kubernetes criptografados anteriormente armazenados no etcd. Se a CMK for desabilitada, o cluster será colocado imediatamente em um estado não íntegro/degradado, situação na qual não poderemos cumprir nosso Compromisso de Serviço até que você reabilite a CMK associada.

Quando uma CMK for desabilitada, você receberá notificações sobre a integridade degradada do cluster do EKS e a necessidade de reabilitar a CMK em até 30 dias após a desabilitação para garantir a restauração bem-sucedida dos recursos do ambiente de gerenciamento do Kubernetes.

Como posso proteger o cluster do EKS contra o impacto de uma CMK desabilitada ou excluída?

Para proteger seus clusters do EKS contra essa ocorrência, os administradores de chave devem gerenciar o acesso às operações de chaves do KMS usando as políticas do IAM com um princípio de privilégio mínimo para reduzir o risco de qualquer desabilitação ou exclusão arbitrária de chaves associadas a clusters do EKS. Além disso, você pode definir um alarme do CloudWatch para ser notificado sobre o estado da sua CMK.

Meu cluster do EKS será restaurado se eu reativar a CMK?

Para garantir a restauração bem-sucedida do cluster do EKS, é altamente recomendável reabilitar a CMK nos primeiros 30 dias após sua desabilitação. No entanto, a restauração bem-sucedida do

cluster do EKS também dependerá de ele sofrer ou não alguma alteração significativa na API devido a uma atualização automática do Kubernetes que pode ocorrer enquanto o cluster está em um estado não íntegro ou degradado.

Por que meu cluster do EKS é colocado em um estado não íntegro ou degradado após a desabilitação da CMK?

O servidor de API do ambiente de gerenciamento do EKS usa uma chave DEK que é criptografada e armazenada em cache na memória do servidor de API para criptografar todos os objetos durante as operações de criação e atualização antes de serem armazenados no etcd. Quando um objeto existente está sendo recuperado do etcd, o servidor DEK API usa a mesma chave DEK em cache e descriptografa o objeto de recurso do Kubernetes. Se você desativar a CMK, o servidor de API não detectará nenhum impacto imediato por causa da chave DEK em cache na memória do servidor de API. No entanto, quando a instância do servidor de API for reiniciada, ela não terá uma DEK em cache e precisará chamar o AWS KMS para operações de criptografia e descriptografia. Sem uma CMK, esse processo falhará com um código de erro KMS_KEY_DISABLED, impedindo que o servidor da API seja inicializado com êxito.

O que acontecerá com meu cluster do EKS se eu excluir minha CMK?

Excluir a chave CMK associada ao seu cluster do EKS degradará sua integridade além da recuperação. Sem sua CMK, o servidor de API não conseguirá mais criptografar e manter quaisquer novos objetos do Kubernetes, bem como descriptografar quaisquer objetos do Kubernetes criptografados anteriormente armazenados no banco de dados etcd. Somente prossiga com a exclusão de uma chave CMK para o cluster do EKS quando tiver certeza de que não vai precisar mais usá-lo.

Observe que, se a CMK não for encontrada (KMS_KEY_NOT_FOUND) ou as concessões da CMK associada ao seu cluster forem revogadas (KMS_GRANT_REVOKED), seu cluster não será recuperável. Para obter mais informações sobre a integridade e os códigos de erro do cluster, consulte Perguntas frequentes sobre a integridade do cluster e códigos de erro com caminhos de resolução.

Ainda serei cobrado por um cluster do EKS degradado e não íntegro porque desabilitei ou excluí a CMK?

Sim. Embora o ambiente de gerenciamento do EKS não possa ser usado no caso de uma CMK desabilitada, a AWS ainda estará executando recursos de infraestrutura dedicados alocados ao cluster do EKS até que ele seja excluído pelo cliente. Além disso, nosso Compromisso de Serviço

não se aplicará nesta circunstância porque será uma ação ou inação voluntária do cliente que impedirá a integridade e a operação normais do cluster do EKS.

O cluster do EKS pode ser atualizado automaticamente quando está em um estado não íntegro ou degradado devido a uma CMK desabilitada?

Sim. No entanto, se o cluster tiver uma CMK desabilitada, você terá um período de 30 dias para reabilitá-la. Nesses 30 dias, o cluster do Kubernetes não será atualizado automaticamente. No entanto, se esse período expirar e você não tiver reabilitado a CMK, o cluster será automaticamente atualizado para a próxima versão (n+1) que está no suporte padrão, seguindo o ciclo de vida da versão do Kubernetes no EKS.

É altamente recomendável reabilitar rapidamente uma CMK desabilitada quando você tomar conhecimento de um cluster afetado. É importante observar que, embora o EKS atualize automaticamente esses clusters afetados, não há garantia de que eles serão recuperados com sucesso, especialmente se o cluster passar por várias atualizações automáticas, já que essas atualizações podem incluir alterações na API do Kubernetes e um comportamento inesperado no processo de inicialização do servidor de API.

Posso usar um alias de chave do KMS?

Sim. O Amazon EKS é compatível com o uso de alias de chave do KMS. Um alias é um nome amigável para uma chave do Amazon Web Service KMS. Por exemplo, um alias permite fazer referência a uma chave do KMS como my-key em vez de **`1234abcd-12ab-34cd-56ef-1234567890ab`**.

Ainda posso fazer backup dos meus recursos de cluster e restaurá-los usando minha própria solução de backup do Kubernetes?

Sim. Você pode usar uma solução de backup do Kubernetes (como o Velero) para recuperação de desastres, migração e proteção de dados do cluster do Kubernetes. Se você executar uma solução de backup do Kubernetes que acessa os recursos do cluster por meio do servidor de API, todos os dados recuperados pela aplicação serão descriptografados antes de chegar ao cliente. Com isso, você poderá recuperar os recursos do cluster em outro cluster do Kubernetes.

Considerações de segurança sobre o Modo Automático do Amazon EKS

Este tópico descreve a arquitetura de segurança, os controles e as práticas recomendadas do Modo Automático do Amazon EKS. À medida que as organizações implantam aplicações em contêineres em grande escala, manter uma postura de segurança forte torna-se cada vez mais complexo. O Modo Automático do EKS implementa controles de segurança automatizados e se integra aos serviços de segurança da AWS para ajudar você a proteger a infraestrutura de clusters, workloads e dados. Por meio de recursos de segurança integrados, como gerenciamento forçado do ciclo de vida dos nós e implantação automatizada de patches, o Modo Automático do EKS ajuda você a manter as práticas recomendadas de segurança e, ao mesmo tempo, reduzir a sobrecarga operacional.

Antes de continuar com este tópico, certifique-se de estar familiarizado com os conceitos básicos do Modo Automático do EKS e ter analisado os pré-requisitos para habilitar o Modo Automático do EKS nos clusters. Para obter informações gerais sobre a segurança do Amazon EKS, consulte Segurança.

O Modo Automático do Amazon EKS se baseia nas bases de segurança existentes do Amazon EKS e, ao mesmo tempo, introduz controles de segurança automatizados adicionais para instâncias gerenciadas pelo EC2.

Segurança e autenticação da API

O Modo Automático do Amazon EKS usa mecanismos de segurança da plataforma da AWS para proteger e autenticar chamadas para a API do Amazon EKS.

- O acesso à API do Kubernetes é protegido por meio de entradas de acesso ao EKS, que se integram às identidades do AWS IAM.
  - Para ter mais informações, consulte the section called “Entradas de acesso”.
- Os clientes podem implementar um controle de acesso refinado ao endpoint da API do Kubernetes por meio da configuração das entradas de acesso do EKS.

Segurança de rede

O Modo Automático do Amazon EKS oferece suporte a várias camadas de segurança de rede:

- Integração com a VPC

  - Opera na Amazon Virtual Private Cloud (VPC)
  - Compatível com as configurações personalizadas da VPC e layouts de sub-rede
  - Habilita a rede privada entre os componentes do cluster
  - Para obter mais informações, consulte Gerenciar responsabilidades pela segurança na Amazon Virtual Private Cloud
- Políticas de rede
  - Suporte nativo para políticas de rede do Kubernetes
  - Capacidade de definir regras granulares de tráfego de rede
  - Para ter mais informações, consulte the section called “Políticas do Kubernetes”.

Segurança da instância gerenciada do EC2

O Modo Automático do Amazon EKS opera instâncias gerenciadas pelo EC2 com os seguintes controles de segurança:

Segurança do EC2

- As instâncias gerenciadas pelo EC2 mantêm os recursos de segurança do Amazon EC2.
- Para obter mais informações sobre instâncias gerenciadas pelo EC2, consulte Segurança no Amazon EC2.

Gerenciamento do ciclo de vida da instância

As instâncias gerenciadas pelo EC2 operadas pelo Modo Automático do EKS têm vida útil máxima de 21 dias. O Modo Automático do Amazon EKS encerra automaticamente as instâncias que excedem essa vida útil. Esse limite de ciclo de vida ajuda a evitar desvios de configuração e mantém a postura de segurança.

Proteção de dados

- O armazenamento de instâncias do Amazon EC2 é criptografado, sendo um armazenamento diretamente anexado à instância. Para obter mais informações, consulte Proteção de dados no Amazon EC2.
- O Modo Automático do EKS gerencia os volumes anexados às instâncias do EC2 no momento da criação, incluindo volumes raiz e de dados. O Modo Automático do EKS não gerencia totalmente os volumes do EBS criados usando os recursos de armazenamento persistente do Kubernetes.

Gerenciamento de patches

- O Modo Automático do Amazon EKS aplica automaticamente os patches às instâncias gerenciadas.
- Os patches incluem:
  - Atualizações do sistema operacional
  - Patches de segurança
  - Componentes do Modo Automático do Amazon EKS

> **Note**
>
> Os clientes mantêm a responsabilidade de proteger e atualizar as workloads em execução nessas instâncias.

Controles de acesso

- O acesso direto à instância é restrito:
  - O acesso SSH não está disponível.
  - O acesso ao AWS Systems Manager Session Manager (SSM) não está disponível.
- As operações de gerenciamento são realizadas por meio da API do Amazon EKS e da API do Kubernetes.

Automatizar o gerenciamento de recursos.

O Modo Automático do Amazon EKS não gerencia totalmente os volumes do Amazon Elastic Block Store (Amazon EBS) criados usando recursos de armazenamento persistente do Kubernetes. O Modo Automático do EKS também não gerencia Elastic Load Balancers (ELB). O Modo Automático do Amazon EKS automatiza as tarefas de rotina desses recursos.

Segurança de armazenamento

- A AWS recomenda que você habilite a criptografia para volumes do EBS provisionados pelos recursos de armazenamento persistente do Kubernetes. Para ter mais informações, consulte the section called “Criar StorageClass”.

- Criptografia em repouso usando o AWS KMS
- Será possível configurar sua conta da AWS para impor a criptografia das novas cópias de snapshots e volumes do EBS que criar. Para obter mais informações, consulte Enable Amazon EBS encryption by default no Guia do usuário do Amazon EBS.
- Para obter mais informações, consulte Security in Amazon EBS.

Segurança do balanceador de carga

- Configuração automatizada de Elastic Load Balancers
- Gerenciamento de certificados SSL e TLS por meio da integração com o AWS Certificate Manager
- Automação de grupos de segurança para controle de acesso ao balanceador de carga
- Para obter mais informações, consulte Security in Elastic Load Balancing.

Práticas recomendadas de segurança

A seção a seguir descreve as práticas recomendadas de segurança do Modo Automático do Amazon EKS.

- Analise regularmente as políticas do AWS IAM e as entradas de acesso ao EKS.
- Implemente padrões de acesso de privilégio mínimo para workloads.
- Monitore a atividade dos clusters por meio do AWS CloudTrail e do Amazon CloudWatch. Para ter mais informações, consulte the section called “ AWS CloudTrail” e the section called “Amazon CloudWatch”.
- Use o AWS Security Hub para avaliação da postura de segurança.
- Implemente padrões de segurança de pods apropriados para as workloads.

Considerações sobre segurança para funcionalidades do EKS

Este tópico aborda considerações importantes de segurança para as funcionalidades do EKS, incluindo a configuração de perfis do IAM, as permissões do Kubernetes e os padrões arquiteturais para implantações em vários clusters e para o gerenciamento de recursos da AWS entre contas.

As funcionalidades do EKS empregam uma combinação de perfis do IAM, entradas de acesso do EKS e RBAC do Kubernetes para fornecer acesso seguro a serviços da AWS, recursos do

Kubernetes no cluster e integrações com o AWS CodeConnections, o AWS Secrets Manager e outros serviços da AWS.

Perfil do IAM para a funcionalidade

Ao criar uma funcionalidade, você fornece um perfil do IAM da funcionalidade que o EKS usa para executar ações em seu nome. Este perfil deve:

- Estar na mesma conta da AWS que o cluster e o recurso da funcionalidade
- Contar com uma política de confiança que permite que a entidade principal do serviço `capabilities.eks.amazonaws.com` assuma o perfil
- Ter permissões do IAM apropriadas para o tipo de funcionalidade e de caso de uso, dependendo dos seus requisitos. Para informações detalhadas sobre permissões do IAM necessárias, consulte the section called " AWS CodeConnections", the section called " AWS Secrets Manager" e the section called "Configurar permissões"

É uma prática recomendada considerar o escopo de privilégios necessário para seu caso de uso específico e conceder apenas as permissões essenciais para atender aos seus requisitos. Por exemplo, ao usar a funcionalidade do EKS para o Kube Resource Orchestrator, nenhuma permissão do IAM pode ser necessária, enquanto ao usar a funcionalidade do EKS para o AWS Controllers for Kubernetes, você pode conceder acesso total a um ou mais serviços da AWS.

> **Important**
>
> Embora alguns casos de uso possam justificar o uso de privilégios administrativos abrangentes, siga o princípio de privilégio mínimo, concedendo apenas as permissões do IAM mínimas necessárias para seu caso de uso específico e restringindo o acesso a recursos específicos usando ARNs e chaves de condição, em vez de permissões curinga.

Para obter informações detalhadas sobre como criar e configurar perfis do IAM de funcionalidades, consulte the section called "Perfil do IAM para a funcionalidade".

Entradas de acesso do EKS

Quando você cria uma funcionalidade com um perfil do IAM, o Amazon EKS gera automaticamente uma entrada de acesso para esse perfil no cluster. Essa entrada de acesso concede à funcionalidade as permissões básicas do Kubernetes necessárias para seu funcionamento.

> **Note**
>
> As entradas de acesso são geradas no cluster em que a funcionalidade foi criada. Para implantações do Argo CD em clusters remotos, é necessário criar entradas de acesso nesses clusters com as permissões adequadas, permitindo que a funcionalidade do Argo CD implante e gerencie aplicações.

A entrada de acesso inclui:

- O ARN do perfil do IAM como entidade principal
- As políticas de entrada de acesso específicas da funcionalidade que concedem permissões básicas do Kubernetes
- O escopo apropriado (abrangendo todo o cluster ou limitado a um namespace) com base no tipo da funcionalidade

> **Note**
>
> No caso do Argo CD, as permissões limitadas a namespace são concedidas ao namespace especificado na configuração da funcionalidade (por padrão, `argocd`).

Políticas padrão de entrada de acesso por funcionalidade

Cada tipo de funcionalidade concede ao perfil da funcionalidade as permissões necessárias, definindo diferentes políticas padrão de entrada de acesso da seguinte forma:

kro

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSKROPolicy` (com escopo de cluster)

  Concede permissões para monitorar e gerenciar ResourceGraphDefinitions, bem como criar instâncias de recursos personalizados definidos pelas RGDs.

ACK

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSACKPolicy` (com escopo de cluster)

Concede permissões para criar, ler, atualizar e excluir recursos personalizados do ACK em todos os namespaces.

Argo CD

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSArgoCDClusterPolicy` (com escopo de cluster)

  Concede permissões em nível de cluster para o Argo CD descobrir recursos e gerenciar objetos com escopo de cluster.
- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSArgoCDPolicy` (com escopo de namespace)

  Concede permissões em nível de namespace para o Argo CD implantar e gerenciar aplicações. Aplica-se ao namespace especificado na configuração da funcionalidade (por padrão, `argocd`).

Consulte the section called “Analisar as políticas de acesso” para obter informações mais detalhadas.

Permissões adicionais do Kubernetes

Algumas funcionalidades podem exigir permissões adicionais do Kubernetes, além das políticas padrão de entrada de acesso. Essas permissões podem ser concedidas por meio de:

- Políticas de entrada de acesso: associe políticas gerenciadas adicionais à entrada de acesso
- RBAC do Kubernetes: crie associações `Role` ou `ClusterRole` para o usuário do Kubernetes da funcionalidade

Permissões de leitura de segredos do ACK

Alguns controladores do ACK precisam ler segredos do Kubernetes para obter dados sensíveis, como senhas de banco de dados. Os controladores do ACK listados abaixo precisam de permissão de leitura de segredos:

- `acm`, `acmpca`, `documentdb`, `memorydb`, `mq`, `rds`, `secretsmanager`

Para conceder permissões de leitura de segredos:

1. Associe a política de entrada de acesso `arn:aws:eks::aws:cluster-access-policy/AmazonEKSSecretReaderPolicy` à entrada de acesso da funcionalidade

2. Defina o escopo da política para namespaces específicos nos quais os recursos do ACK farão referência a segredos, ou conceda acesso em todo o cluster

> **Important**
>
> As permissões de leitura de segredos são limitadas aos namespaces que você especificar ao associar a política de entrada de acesso. Isso possibilita restringir o acesso da funcionalidade apenas aos segredos desejados.

Permissões de recursos arbitrários do kro

O kro requer permissões para criar e gerenciar os recursos definidos em ResourceGraphDefinitions. Por padrão, o kro só pode monitorar e gerenciar as próprias RGDs.

Para conceder permissões ao kro para criar recursos:

Opção 1: políticas de entrada de acesso

Associe políticas de entrada de acesso definidas previamente, como `AmazonEKSAdminPolicy` ou `AmazonEKSEditPolicy`, à entrada de acesso da funcionalidade.

Opção 2: RBAC do Kubernetes

Crie uma `ClusterRoleBinding` que conceda ao usuário do Kubernetes da funcionalidade as permissões necessárias:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kro-cluster-admin
subjects:
- kind: User
  name: arn:aws:sts::111122223333:assumed-role/my-kro-role/kro
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
```

> **Note**
>
> Para o kro, o nome do usuário do Kubernetes adota o padrão:
> `arn:aws:sts::ACCOUNT_ID:assumed-role/ROLE_NAME/kro`
> O nome da sessão `/kro` é definido automaticamente pela funcionalidade kro do EKS.

Permissões do IAM exigidas pela funcionalidade

kro (Kube Resource Orchestrator)

Nenhuma permissão do IAM é necessária. É possível criar um perfil da funcionalidade sem políticas vinculadas. O kro requer apenas permissões de RBAC do Kubernetes.

ACK (AWS Controllers for Kubernetes)

Requer permissões para gerenciar os recursos da AWS que o ACK criará e gerenciará. Você deve restringir o escopo das permissões a serviços, ações e recursos específicos, de acordo com suas necessidades. Para obter informações detalhadas sobre como configurar permissões do ACK, incluindo as práticas recomendadas para ambientes de produção com seletores de perfil do IAM, consulte the section called “Configurar permissões”.

Argo CD

Por padrão, nenhuma permissão do IAM é necessária. Entretanto, permissões adicionais podem ser exigidas para:

- AWS Secrets Manager: se armazenar credenciais de repositórios do Git no Secrets Manager
- AWS CodeConnections: se usar o CodeConnections para realizar autenticação em repositórios do Git
- Amazon ECR: se usar charts do Helm armazenados em formato OCI no Amazon ECR

Práticas recomendadas de segurança

Princípio de privilégio mínimo do IAM

Conceda aos recursos da funcionalidade apenas as permissões necessárias para o caso de uso. Isso não significa que você não possa conceder permissões administrativas abrangentes às funcionalidades, se necessário. Nesses casos, você deve gerenciar o acesso a esses recursos de forma adequada.

Perfis da funcionalidade:

- ACK: sempre que possível, limite as permissões do IAM aos serviços e recursos da AWS que suas equipes precisam, com base no caso de uso e nos requisitos
- Argo CD: restrinja o acesso a repositórios do Git específicos e a namespaces do Kubernetes
- kro: requer um perfil da funcionalidade para a política de confiança, mas não são necessárias permissões do IAM (apenas RBAC do cluster é usado)

Exemplo: em vez de `"Resource": "*"`, especifique padrões para recursos ou grupos de recursos específicos.

```
"Resource": [
  "arn:aws:s3:::my-app-*",
  "arn:aws:rds:us-west-2:111122223333:db:prod-*"
]
```

Use chaves de condição do IAM para aplicar restrições adicionais de acesso:

```
"Condition": {
  "StringEquals": {
    "aws:ResourceTag/Environment": "production"
  }
}
```

Para obter informações adicionais sobre a configuração do IAM, consulte a seção de considerações destinada a cada funcionalidade.

Isolamento de namespace para segredos do Argo CD

A funcionalidade gerenciada do Argo CD tem acesso a todos os segredos do Kubernetes no namespace configurado (padrão: `argocd`). Para manter uma postura de segurança ideal, siga estas práticas de isolamento de namespace:

- Mantenha apenas segredos relevantes para o Argo CD no namespace do Argo CD
- Evite armazenar segredos de aplicações não relacionadas no mesmo namespace do Argo CD
- Use namespaces separados para segredos de aplicações que não sejam necessários para as operações do Argo CD

Esse isolamento garante que o acesso do Argo CD a segredos seja limitado apenas às credenciais necessárias à autenticação em repositórios do Git e a outras operações específicas do Argo CD.

RBAC do Kubernetes

Controle quais usuários e contas de serviço podem criar e gerenciar recursos de funcionalidades. É uma prática recomendada implantar recursos de funcionalidades em namespaces dedicados, com políticas de RBAC apropriadas.

Exemplo: perfil de RBAC para trabalhar com ACK, permitindo o gerenciamento de recursos de bucket do S3 no namespace `app-team`:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ack-s3-manager
  namespace: app-team
rules:
- apiGroups: ["s3.services.k8s.aws"]
  resources: ["buckets"]
  verbs: ["get", "list", "create", "update", "delete"]
```

Registro em log de auditoria

CloudTrail: todas as operações de API das funcionalidades do EKS (por exemplo, criar, atualizar e excluir) são registradas no AWS CloudTrail.

Habilite o registro em log do CloudTrail para acompanhar:

- Quem criou ou modificou funcionalidades
- Quando as configurações das funcionalidades foram alteradas
- Quais perfis de funcionalidades estão em uso

Acesso à rede e endpoints da VPC

Acesso privado à API do Argo CD

Você pode restringir o acesso ao servidor de API do Argo CD associando um ou mais endpoints da VPC ao endpoint hospedado do Argo CD. Isso habilita conectividade privada com a interface do usuário do Argo CD e com a API do Argo CD usando a VPC, sem trafegar pela internet pública.

> **Note**
>
> Endpoints da VPC conectados a endpoints hospedados da API do Argo CD (usando eks-capabilities.*region*.amazonaws.com) não oferecem suporte a políticas de endpoint da VPC.

Implantação em clusters privados

A funcionalidade do Argo CD pode implantar aplicações em clusters do EKS totalmente privados, oferecendo um benefício operacional significativo ao eliminar a necessidade de emparelhamento da VPC ou de configurações de rede complexas. No entanto, ao projetar esta arquitetura, é importante considerar que o Argo CD busca configurações em repositórios do Git (que podem ser públicos) e as aplica aos seus clusters privados.

Garanta que você:

- Utilize repositórios do Git privados para workloads sensíveis
- Implemente controles de acesso e de autenticação adequados nos repositórios do Git
- Analise e aprove alterações por meio de solicitações de pull antes de realizar a mesclagem
- Considere usar as janelas de sincronização do Argo CD para controlar o momento em que as implantações podem ocorrer
- Monitore os logs de auditoria do Argo CD em busca de alterações de configuração não autorizadas

Compliance

As funcionalidades do EKS são totalmente gerenciadas e contam com as certificações de conformidade do Amazon EKS.

Para obter informações atualizadas sobre conformidade, consulte AWS Services in Scope by Compliance Program.

Próximas etapas

- the section called “Configurar permissões”: configure permissões do IAM para o ACK
- the section called “Configurar permissões”: configure o RBAC do Kubernetes para o kro
- the section called “Configurar permissões”: configure a integração com o Centro de Identidade para o Argo CD

- the section called “Solução de problemas das funcionalidades”: solucione problemas de segurança e de permissões

Identity and Access Management para o Amazon EKS

O AWS Identity and Access Management (IAM) é um serviço da AWS que ajuda administradores a controlar o acesso aos recursos da AWS de forma segura. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) para usar os recursos do Amazon EKS. O IAM é um AWS serviço da que pode ser usado sem custo adicional.

Público

O modo como você usa o AWS Identity and Access Management (IAM) é diferente, dependendo do trabalho que você faz no Amazon EKS.

Usuário do serviço: se você usar o serviço Amazon EKS para fazer sua tarefa, o administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais recursos do Amazon EKS para realizar o trabalho, talvez sejam necessárias permissões adicionais. Entender como o acesso é gerenciado pode ajudar a solicitar as permissões corretas ao administrador. Se você não puder acessar um recurso na Amazon EKS, consulte the section called “Solução de problemas”.

Administrador do serviço: se você for o responsável pelos recursos do Amazon EKS em sua empresa, você provavelmente terá acesso total ao Amazon EKS. Cabe a você determinar quais funcionalidades e recursos do Amazon EKS os usuários do seu serviço devem acessar. Envie as solicitações ao administrador do IAM para alterar as permissões dos usuários de serviço. Revise as informações nesta página para compreender os conceitos básicos do IAM. Para saber mais sobre como a empresa pode usar o IAM com o Amazon EKS, consulte the section called “Amazon EKS e IAM”.

Administrador do IAM: se você for um administrador do IAM, talvez queira saber detalhes sobre como pode escrever políticas para gerenciar o acesso à Amazon EKS. Para visualizar exemplos de políticas baseadas em identidade do Amazon EKS que podem ser usadas no IAM, consulte the section called “Políticas baseadas em identidade”.

Autenticação com identidades

A autenticação é a forma como fazer login na AWS usando suas credenciais de identidade. É necessário estar autenticado (conectado à AWS) como usuário raiz da conta AWS, como usuário do IAM ou assumindo um perfil do IAM.

É possível fazer login na AWS como uma identidade federada usando credenciais fornecidas por uma fonte de identidades. AWS Os usuários do Centro de Identidade do IAM, a autenticação de logon único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já configurou anteriormente a federação de identidades usando perfis do IAM. Quando você acessa a AWS usando a federação, está indiretamente assumindo um perfil.

A depender do tipo de usuário, você pode fazer login no Console de gerenciamento da AWS ou no portal de acesso AWS. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login na sua conta AWS no Guia do usuário de login AWS.

Se você acessar a AWS de forma programática, a AWS fornecerá um kit de desenvolvimento de software (SDK) e uma interface de linha de comandos (CLI) para você assinar de forma criptográfica as solicitações usando as suas credenciais. Se você não utilizar as ferramentas AWS, deverá designar as solicitações por conta própria. Para obter mais informações sobre como usar o método recomendado para designar solicitações por conta própria, consulte Designando solicitações de API AWS no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, a AWS recomenda o uso da autenticação multifator (MFA) para aumentar a segurança de sua conta. Para saber mais, consulte Autenticação multifator no Guia do usuário do Centro de Identidade do AWS IAM e Usar a autenticação multifator (MFA) em AWS no Guia do usuário do IAM.

Usuário raiz de conta da AWS

Ao criar uma conta AWS, você começa com uma identidade de login que tem acesso completo a todos os serviços e recursos da conta AWS. Essa identidade é denominada usuário-raiz da conta da AWS e é acessada pelo login com o endereço de e-mail e a senha usados para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário-raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário-raiz, consulte Tarefas que exigem credenciais de usuário-raiz no Guia do Usuário do IAM.

Grupos e usuários do IAM

Um usuário do IAM é uma identidade dentro da sua conta da AWS que tem permissões específicas para uma única pessoa ou aplicação. Sempre que possível, é recomendável contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, é recomendável alternar as chaves de acesso. Para obter mais informações, consulte Alternar as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do Usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de um perfil) no Guia do usuário do IAM.

Funções do IAM

Perfil do IAM é uma identidade dentro da sua conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. É possível presumir temporariamente um perfil do IAM no Console de gerenciamento da AWS alternando perfis. É possível assumir um perfil chamando uma operação da AWS CLI ou da AWS API, ou usando um URL personalizado. Para obter mais informações sobre métodos para o uso de perfis, consulte Utilizar perfis do IAM no Guia do usuário do IAM.

Perfis do IAM com credenciais temporárias são úteis nas situações a seguir:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, é possível criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas por ele. Para obter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidades de terceiros no Guia do usuário do IAM. Se usar o Centro de Identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a

autenticação, o Centro de Identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões no Guia do usuário do Centro de Identidade do AWS IAM.

- Permissões temporárias para usuários do IAM: um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas: é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, alguns serviços da AWS permitem que você anexe uma política diretamente a um recurso (em vez de usar um perfil como proxy). Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.
- Acesso entre serviços: alguns serviços da AWS usam recursos em outros serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicações no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões da entidade principal de chamada, usando um perfil de serviço ou um perfil vinculado ao serviço.
  - Sessões de acesso direto (FAS): qualquer pessoa que utilizar um perfil ou usuário do IAM para executar ações na AWS é considerada uma entidade principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. A FAS usa as permissões do entidade principal que chama um serviço da AWS, combinadas com o serviço da AWS solicitante, para fazer solicitações a serviços downstream. As solicitações FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros serviços ou recursos do AWS para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Sessões de acesso direto.
  - Perfil de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para executar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir uma função de serviço do IAM. Para obter mais informações, consulte Criação de uma função para delegar permissões a um AWS serviço no Guia do usuário do IAM.
  - Função vinculada a serviço: uma função vinculada a serviço é um tipo de função de serviço vinculada a um serviço da AWS. O serviço pode assumir o perfil de executar uma ação em seu nome. Os Perfis vinculados a serviços aparecem em sua conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para perfis vinculados ao serviço.

- Aplicações em execução no Amazon EC2: é possível usar um perfil do IAM para gerenciar credenciais temporárias para aplicações em execução em uma instância do EC2 e fazer solicitações da AWS CLI ou da AWS API. É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir um perfil da AWS a uma instância do EC2 e disponibilizá-la para todas as suas aplicações, crie um perfil de instância que esteja anexado a ela. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para obter mais informações, consulte Utilizar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar perfis do IAM, consulte Quando criar um perfil do IAM (em vez de um usuário) no Guia do usuário do IAM.

Gerenciando acesso usando políticas

Você controla o acesso na AWS criando políticas e anexando-as a identidades ou recursos da AWS. Uma política é um objeto na AWS que, quando associado a uma identidade ou recurso, define suas permissões. A AWS avalia essas políticas quando uma entidade principal (usuário, usuário-raiz ou sessão de perfil) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada na AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do usuário do IAM.

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e perfis não têm permissões. Para conceder permissão aos usuários para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem assumir os perfis.

As políticas do IAM definem permissões para uma ação independentemente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações sobre funções usando o Console de gerenciamento da AWS, a AWS CLI ou a API da AWS.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou função. As políticas gerenciadas são políticas independentes que podem ser anexadas a vários usuários, grupos e funções em sua conta da AWS. As políticas gerenciadas incluem políticas gerenciadas pela AWS e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do Usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o atributo ao qual a política está anexada, a política define quais ações uma entidade principal especificado pode executar nesse atributo e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os principais podem incluir contas, usuários, funções, usuários federados ou serviços da AWS.

Políticas baseadas em recursos são políticas embutidas que estão localizadas nesse serviço. Não é possível usar as políticas do IAM gerenciadas por AWS em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Amazon S3, AWS WAF e Amazon VPC são exemplos de serviços que suportam ACLs. Para saber mais sobre ACLs, consulte Visão geral da lista de controle de acesso (ACL) no Guia do Desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

A AWS oferece compatibilidade com tipos de política menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um recurso avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recursos que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do usuário do IAM.
- Políticas de controle de serviço (SCPs) - SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em AWS Organizations. AWS O Organizations é um serviço para agrupar e gerenciar de forma centralizada várias contas do AWS que sua empresa possui. Se você habilitar todos os recursos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. Uma SCP limita as permissões para entidades em contas-membro, incluindo cada usuário root da conta da AWS. Para obter mais informações sobre Organizações e SCPs, consulte Políticas de controle de serviços no Guia do Usuário de Organizações AWS.
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou da função e as políticas da sessão. As permissões também podem ser provenientes de uma política baseada em recursos. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como a AWS determina se deve permitir ou não uma solicitação quando há vários tipos de política envolvidos, consulte Lógica da avaliação de políticas no Guia do usuário do IAM.

Como o Amazon EKS funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao Amazon EKS, você deve entender quais recursos do IAM estão disponíveis para uso com o Amazon EKS. Para obter uma visualização de alto nível de como o Amazon EKS e outros serviços da AWS funcionam com o IAM, consulte Serviços da AWS compatíveis com o IAM no Manual do usuário do IAM.

Tópicos

- Políticas baseadas em identidade do Amazon EKS
- Políticas baseadas em recursos do Amazon EKS
- Autorização baseada em etiquetas do Amazon EKS
- Funções do IAM no Amazon EKS

Políticas baseadas em identidade do Amazon EKS

Com as políticas baseadas em identidade do IAM, é possível especificar ações e recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. A Amazon EKS oferece suporte a ações, chaves de condição e recursos específicos. Para saber mais sobre todos os elementos usados em uma política JSON, consulte Referência de elementos de política JSON do IAM no Guia do usuário do IAM.

Ações

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Action` de uma política JSON descreve as ações que podem ser usadas para permitir ou negar acesso em uma política. As ações de políticas geralmente têm o mesmo nome que a operação de API da AWS associada. Existem algumas exceções, como Ações somente de permissão, que não têm uma operação de API correspondente. Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

As ações de política na Amazon EKS usam o seguinte prefixo antes da ação: `eks:`. Por exemplo, para conceder a alguém permissão para obter informações descritivas sobre um cluster do Amazon EKS, inclua a ação `DescribeCluster` na política dessa pessoa. As instruções de política devem incluir um elemento `Action` ou `NotAction`.

Para especificar várias ações em uma única instrução, separe-as com vírgulas, como segue:

```
"Action": ["eks:action1", "eks:action2"]
```

Você também pode especificar várias ações usando caracteres curinga (*). Por exemplo, para especificar todas as ações que começam com a palavra `Describe`, inclua a seguinte ação:

```
"Action": "eks:Describe*"
```

Para ver uma lista das ações do Amazon EKS, consulte Ações definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço.

Recursos

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem suporte a permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

O recurso de cluster do Amazon EKS tem o seguinte ARN.

```
arn:aws:eks:region-code:account-id:cluster/cluster-name
```

Para obter mais informações sobre o formato de ARNs, consulte Nomes de recursos da Amazon (ARNs) e namespaces de serviços da AWS.

Por exemplo, para especificar o cluster chamado *`my-cluster`* em sua instrução, use o seguinte ARN:

```
"Resource": "arn:aws:eks:region-code:111122223333:cluster/my-cluster"
```

Para especificar todos os clusters que pertencem a uma conta e região da AWS específicas, use o caractere curinga (*):

```
"Resource": "arn:aws:eks:region-code:111122223333:cluster/*"
```

Algumas ações do Amazon EKS, como as que servem para a criação de recursos, não podem ser executadas em um recurso específico. Nesses casos, você deve utilizar o caractere curinga (*).

```
"Resource": "*"
```

Para ver uma lista dos tipos de recursos do Amazon EKS e seus ARNs, consulte Recursos definidos pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Ações definidas pelo Amazon Elastic Kubernetes Service.

Chaves de condição

O Amazon EKS define seu próprio conjunto de chaves de condição e também oferece suporte ao uso de algumas chaves de condição globais. Para ver todas as chaves de condição globais da AWS, consulte Chaves de contexto de condição globais da AWS no Guia do usuário do IAM.

Você pode definir chaves de condição ao associar um provedor OpenID Connect ao cluster. Para obter mais informações, consulte the section called “Exemplo de política do IAM”.

Todas as ações do Amazon EC2 oferecem suporte às chaves de condição `aws:RequestedRegion` e `ec2:Region`. Para obter mais informações, consulte Exemplo: Restrição de acesso a uma região específica do AWS.

Para obter uma lista de chaves de condição do Amazon EKS, consulte Condições definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar a chave de condição, consulte Ações definidas pelo Amazon Elastic Kubernetes Service.

Exemplos

Para visualizar exemplos de políticas baseadas em identidade do Amazon EKS, consulte the section called “Políticas baseadas em identidade”.

Quando você cria um cluster do Amazon EKS, a entidade principal do IAM que cria o cluster, recebe automaticamente permissões `system:masters` na configuração de controle de acesso baseado em perfil (RBAC) no ambiente de gerenciamento do Amazon EKS. Como essa entidade principal

não é exibida em nenhuma configuração visível, mantenha o controle de qual entidade principal criou o cluster originalmente. Para conceder a outras entidades principais do IAM a capacidade de interagir com o cluster, edite o `aws-auth ConfigMap` no Kubernetes e crie um `rolebinding` ou `clusterrolebinding` do Kubernetes com o nome de um `group` especificado no `aws-auth ConfigMap`.

Para obter mais informações sobre como trabalhar com o ConfigMap, consulte the section called "Acesso à API do Kubernetes".

Políticas baseadas em recursos do Amazon EKS

O Amazon EKS não oferece suporte a políticas baseadas em recursos.

Autorização baseada em etiquetas do Amazon EKS

É possível anexar etiquetas aos recursos do Amazon EKS ou passar etiquetas em uma solicitação para o Amazon EKS. Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`. Para obter mais informações sobre recursos de marcação do Amazon EKS, consulte the section called "Marcando seus Recursos". Para obter mais informações sobre com quais ações você pode usar tags em chaves de condição, consulte Actions defined by Amazon EKS (Ações definidas pelo Amazon EKS) na Referência de autorização do serviço.

Funções do IAM no Amazon EKS

Perfil do IAM é uma entidade dentro da sua conta da AWS que tem permissões específicas.

Usar credenciais temporárias com o Amazon EKS

É possível usar credenciais temporárias para fazer login com federação, assumir um perfil do IAM ou assumir um perfil entre contas. Você obtém credenciais de segurança temporárias chamando operações de API do AWS STS, como AssumeRole ou GetFederationToken.

A Amazon EKS oferece suporte ao uso de credenciais temporárias.

Perfis vinculados a serviço

Perfis vinculados ao serviço permitem que os serviços da AWS acessem recursos em outros serviços para concluir uma ação em seu nome. Os perfis vinculados a serviço aparecem em sua conta do IAM e são de propriedade do serviço. Um administrador pode visualizar, mas não pode editar as permissões de perfis vinculados ao serviço.

Amazon EKS oferece suporte às funções de serviço. Para obter detalhes sobre como criar ou gerenciar funções vinculadas ao serviço do Amazon EKS, consulte the section called “Perfis vinculados a serviço”.

Perfis de serviço

Esse atributo permite que um serviço assuma um perfil de serviço em seu nome. O perfil permite que o serviço acesse recursos em outros serviços para concluir uma ação em seu nome. Os perfis de serviço aparecem em sua conta do IAM e são de propriedade da conta. Isso significa que um administrador do IAM pode alterar as permissões para esse perfil. Porém, fazer isso pode alterar a funcionalidade do serviço.

Amazon EKS oferece suporte às funções de serviço. Para obter mais informações, consulte the section called “Função do IAM do cluster” e the section called “Perfil do IAM do nó”.

Escolher uma função IAM no Amazon EKS

Ao criar um recurso de cluster no Amazon EKS, é necessário escolher uma função para permitir que o Amazon EKS acesse outros recursos da AWS em seu nome. Se você já tiver criado uma função de serviço, o Amazon EKS fornecerá uma lista de funções da qual escolher. É importante escolher uma função que tenha as políticas gerenciadas do Amazon EKS anexadas. Para obter mais informações, consulte the section called “Verificar se há uma função de cluster existente” e the section called “Verificar se há uma função existente do nó”.

Exemplos de políticas baseadas em identidade do Amazon EKS

Por padrão, os usuários e as funções do IAM não têm permissão para criar ou modificar recursos do Amazon EKS. Eles também não podem executar tarefas usando o Console de gerenciamento da AWS, a CLI da AWS ou a API da AWS. Um administrador do IAM deve criar políticas do IAM que concedam aos usuários e perfis permissão para executarem operações de API específicas nos recursos especificados de que precisam. O administrador deve anexar essas políticas aos usuários ou grupos do IAM que exigem essas permissões.

Para saber como criar uma política baseada em identidade do IAM utilizando esses exemplos de documentos de política JSON, consulte Criar políticas na guia JSON no Guia do usuário do IAM.

Quando você cria um cluster do Amazon EKS, a entidade principal do IAM que cria o cluster, recebe automaticamente permissões `system:masters` na configuração de controle de acesso baseado em perfil (RBAC) no ambiente de gerenciamento do Amazon EKS. Como essa entidade principal

não é exibida em nenhuma configuração visível, mantenha o controle de qual entidade principal criou o cluster originalmente. Para conceder a outras entidades principais do IAM a capacidade de interagir com o cluster, edite o `aws-auth ConfigMap` no Kubernetes e crie um `rolebinding` ou `clusterrolebinding` do Kubernetes com o nome de um `group` especificado no `aws-auth ConfigMap`.

Para obter mais informações sobre como trabalhar com o ConfigMap, consulte the section called “Acesso à API do Kubernetes”.

Tópicos

- Práticas recomendadas de política
- Usar o console do Amazon EKS
- Permitir que os usuários do IAM visualizem suas próprias permissões
- Crie um cluster do Kubernetes na Nuvem AWS
- Criar um cluster local do Kubernetes em um Outpost
- Atualizar um cluster do Kubernetes
- Listar ou descrever todos os clusters

Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do Amazon EKS em sua conta. Essas ações podem incorrer em custos para sua conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece a usar as políticas gerenciadas do AWS e avance para as permissões de privilégios mínimos - Para começar a conceder permissões aos seus usuários e workloads, use as políticas gerenciadas do AWS que concedem permissões para muitos casos de uso comuns. Elas estão disponíveis em sua conta AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da AWS que são específicas para seus casos de uso. Para obter mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do usuário do IAM.

- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso a ações de serviço se elas forem usadas por meio de um serviço específico do AWS, como o AWS CloudFormation. Para obter mais informações, consulte Elementos da política JSON do IAM: condição no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de 100 verificações de política e recomendações acionáveis para ajudá-lo a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do usuário do IAM.
- Exigir autenticação multifator (MFA) - se você tiver um cenário que exija usuários do IAM ou um usuário raiz na sua conta AWS, ative a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte Práticas Recomendadas de Segurança no IAM no Guia do usuário do IAM.

Usar o console do Amazon EKS

Para acessar o console da Amazon EKS, uma entidade principal do IAM deve ter um conjunto mínimo de permissões. Essas permissões devem garantir à entidade principal concessão para visualizar os detalhes sobre os recursos da Amazon EKS na sua conta da AWS. Se você criar uma política baseada em identidade que seja mais restritiva que as permissões mínimas necessárias, o console não funcionará como pretendido para as entidades principais com essa política anexada.

Para garantir que as entidades principais do IAM ainda possam usar o console do Amazon EKS, crie uma política com seu próprio nome exclusivo, como `AmazonEKSAdminPolicy`. Anexe a política às entidades principais. Para obter mais informações, consulte Adicionar e remover permissões de identidade do IAM no Guia do usuário do IAM.

> **Important**
>
> A política de exemplo a seguir permite que uma entidade principal visualize informações na guia Configuração do console. Para visualizar as informações nas guias Visão geral e Recursos no Console de gerenciamento da AWS, a entidade principal também precisa de permissões do Kubernetes. Para obter mais informações, consulte the section called "Permissões obrigatórias".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

Não é necessário permitir permissões mínimas de console para os diretores que fazem chamadas apenas para a AWS CLI ou para a API da AWS. Em vez disso, permita o acesso somente às ações que correspondem à operação da API que você está tentando executar.

Permitir que os usuários do IAM visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui

permissões para concluir essa ação no console ou de forma programática usando a AWS CLI ou a API da AWS.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Crie um cluster do Kubernetes na Nuvem AWS

Esta política de exemplo inclui as permissões mínimas necessárias para criar um cluster do Amazon EKS chamado *my-cluster* na região *us-west-2* AWS. Você pode substituir a região da AWS pela região da AWS na qual deseja criar um cluster. Se você receber um aviso informando que As ações na sua política não são compatíveis com permissões em nível de recurso e exigem que

você escolha **`All resources`** no Console de gerenciamento da AWS, é possível ignorá-lo com segurança. Se sua conta já tiver o perfil *`AWSServiceRoleForAmazonEKS`*, você poderá remover a ação `iam:CreateServiceLinkedRole` da política. Se você já tiver criado um cluster do Amazon EKS em sua conta, esse perfil já existirá, a menos que você o tenha excluído.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
eks.amazonaws.com/AWSServiceRoleForAmazonEKS",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iam:AWSServiceName": "eks"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        }
    ]
}
```

Criar um cluster local do Kubernetes em um Outpost

Essa política de exemplo inclui as permissões mínimas necessárias para criar um cluster local do Amazon EKS chamado *`my-cluster`* em um Outpost na região *`us-west-2`* da AWS. Você pode substituir a região da AWS pela região da AWS na qual deseja criar um cluster. Se você receber um aviso informando que As ações na sua política não são compatíveis com permissões em nível de recurso e exigem que você escolha **`All resources`** no Console de gerenciamento da AWS, é possível ignorá-lo com segurança. Se sua conta já tiver o perfil `AWSServiceRoleForAmazonEKSLocalOutpost`, você poderá remover a ação

`iam:CreateServiceLinkedRole` da política. Se você já tiver criado um cluster local do Amazon EKS na sua conta, esse perfil já existirá, a menos que você o tenha excluído.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "iam:GetRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/outposts.eks-
local.amazonaws.com/AWSServiceRoleForAmazonEKSLocalOutpost"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
```

```
            "Resource": "arn:aws:iam::*:instance-profile/eks-local-*",
            "Effect": "Allow"
        }
    ]
}
```

Atualizar um cluster do Kubernetes

Este exemplo de política inclui a permissão mínima necessária para atualizar um cluster chamado *my-cluster* na região us-west-2 AWS.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:UpdateClusterVersion",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        }
    ]
}
```

Listar ou descrever todos os clusters

Este exemplo de política inclui as permissões mínimas necessárias para listar e descrever todos os clusters na sua conta. Uma entidade principal do IAM deve ser capaz de listar e descrever clusters para usar o comando `update-kubeconfig` da AWS CLI.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

Usar funções vinculadas ao serviço para o Amazon EKS

O Amazon Elastic Kubernetes Service usa perfis vinculados a serviço do AWS Identity and Access Management (IAM). A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Tópicos

- Usar funções para clusters do Amazon EKS
- Usar funções para grupos de nós do Amazon EKS
- Usar funções para os perfis do Amazon EKS Fargate
- Usar funções para conectar um cluster do Kubernetes ao Amazon EKS
- Usar funções para clusters locais do Amazon EKS no Outpost
- Como usar perfis para o painel do Amazon EKS

Usar funções para clusters do Amazon EKS

O Amazon Elastic Kubernetes Service usa perfis vinculados a serviço do AWS Identity and Access Management (IAM). A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Uma função vinculada ao serviço poderá ser excluída somente após excluir seus recursos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com funções vinculadas aos serviços, consulte Serviços da AWS que funcionam com o IAM e procure os serviços que apresentam

Sim na coluna Funções vinculadas aos serviços. Escolha um Sim com um link para visualizar a documentação do perfil vinculado para esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa o perfil vinculado a serviço chamado `AWSServiceRoleForAmazonEKS`. O perfil permite que o Amazon EKS gerencie clusters em sua conta. As políticas anexadas permitem que a função gerencie os seguintes recursos: interfaces de rede, grupos de segurança, logs e VPCs.

> **Note**
>
> A função vinculada ao serviço `AWSServiceRoleForAmazonEKS` é distinta da função necessária para a criação do cluster. Para ter mais informações, consulte the section called "Função do IAM do cluster".

A função vinculada ao serviço `AWSServiceRoleForAmazonEKS` confia nos seguintes serviços para assumir a função:

- `eks.amazonaws.com`

A política de permissões da função permite que o Amazon EKS conclua as seguintes ações nos recursos especificados:

- AmazonEKSServiceRolePolicy

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para obter mais informações, consulte Permissões de perfil vinculado ao serviço no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon EKS

Não é necessário criar manualmente uma função vinculada ao serviço. Quando você cria um cluster no Console de gerenciamento da AWS, na AWS CLI ou na API AWS, o Amazon EKS cria o perfil vinculado a serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria um cluster, o Amazon EKS cria uma função vinculada ao serviço para você novamente.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKS`. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para obter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um atributo ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, escolha Clusters.
3. Se o cluster tiver grupos de nós ou perfis do Fargate, você deverá excluí-los antes de excluir o cluster. Para ter mais informações, consulte the section called “Excluir” e the section called “Excluir perfis”.
4. NoClustersEscolha o cluster que você quer excluir e escolhaExcluir.
5. Digite o nome do cluster na janela de confirmação de exclusão e escolha Delete (Excluir) para excluir.
6. Repita esse procedimento para qualquer outro cluster na sua conta. Aguarde até que todas as operações de exclusão sejam concluídas.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API da AWS para excluir a função vinculada ao serviço `AWSServiceRoleForAmazonEKS`. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte a funções vinculadas a serviço do Amazon EKS

O Amazon EKS oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço estiver disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon EKS.

Usar funções para grupos de nós do Amazon EKS

O Amazon EKS usa perfis vinculados a serviços do AWS Identity and Access Management (IAM). A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Uma função vinculada ao serviço poderá ser excluída somente após excluir seus recursos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com funções vinculadas aos serviços, consulte Serviços da AWS que funcionam com o IAM e procure os serviços que apresentam Sim na coluna Funções vinculadas aos serviços. Escolha um Sim com um link para visualizar a documentação do perfil vinculado para esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa o perfil vinculado a serviço chamado `AWSServiceRoleForAmazonEKSNodegroup`. O perfil permite que o Amazon EKS gerencie grupos de nós em sua conta. A política `AWSServiceRoleForAmazonEKSNodegroup` anexada permite que o perfil gerencie os recursos a seguir: grupos do Auto Scaling, grupos de segurança, modelos

de execução e perfis de instância do IAM. Para ter mais informações, consulte the section called "Política gerenciada pela AWS: AWSServiceRoleForAmazonEKSNodegroup".

A função vinculada ao serviço `AWSServiceRoleForAmazonEKSNodegroup` confia nos seguintes serviços para assumir a função:

- `eks-nodegroup.amazonaws.com`

A política de permissões da função permite que o Amazon EKS conclua as seguintes ações nos recursos especificados:

- AWSServiceRoleForAmazonEKSNodegroup

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para obter mais informações, consulte Permissões de perfil vinculado ao serviço no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon EKS

Não é necessário criar manualmente uma função vinculada ao serviço. Quando você cria o CreateNodegroup no Console de gerenciamento da AWS, na CLI AWS ou na API AWS, o Amazon EKS cria o perfil vinculada ao serviço para você.

> **Important**
>
> Esse perfil vinculado ao serviço pode aparecer em sua conta se você concluiu uma ação em outro serviço que usa os atributos compatíveis com esse perfil. Se você estava usando o serviço do Amazon EKS antes de 1º de janeiro de 2017, quando ele começou a oferecer suporte a funções vinculadas ao serviço, o Amazon EKS criou a função AWSServiceRoleForAmazonEKSNodegroup na sua conta. Para saber mais, consulte Uma nova função apareceu na minha conta do IAM.

Criação de uma função vinculada ao serviço no Amazon EKS (API da AWS)

Não é necessário criar manualmente uma função vinculada ao serviço. Quando você cria um grupo de nós gerenciados no Console de gerenciamento da AWS, na CLI AWS ou na API AWS, o Amazon EKS cria o perfil vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria outro grupo de nós gerenciados, o Amazon EKS cria a função vinculada ao serviço novamente para você.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKSNodegroup`. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para obter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um atributo ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, escolha Clusters.
3. Selecione a guia Compute (Computação).
4. Na seção Node groups (Grupos de nós), escolha o grupo de nós a ser excluído.
5. Digite o nome do grupo de nós na janela de confirmação de exclusão e selecione Delete (Excluir) para excluir.
6. Repita esse procedimento para qualquer outro grupo de nós no cluster. Aguarde até que todas as operações de exclusão sejam concluídas.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API da AWS para excluir a função vinculada ao serviço `AWSServiceRoleForAmazonEKSNodegroup`. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte a funções vinculadas a serviço do Amazon EKS

O Amazon EKS oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço estiver disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon EKS.

Usar funções para os perfis do Amazon EKS Fargate

O Amazon EKS usa perfis vinculados a serviços do AWS Identity and Access Management (IAM). A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Uma função vinculada ao serviço poderá ser excluída somente após excluir seus recursos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com funções vinculadas aos serviços, consulte Serviços da AWS que funcionam com o IAM e procure os serviços que apresentam Sim na coluna Funções vinculadas aos serviços. Escolha um Sim com um link para visualizar a documentação do perfil vinculado para esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa o perfil vinculado a serviço chamado `AWSServiceRoleForAmazonEKSForFargate`. O perfil permite que o Fargate para Amazon EKS configure a rede da VPC necessária para pods do Fargate. As políticas anexadas permitem que a função crie e exclua interfaces de rede elástica e descreva recursos e interfaces de rede elástica.

O perfil vinculado ao serviço `AWSServiceRoleForAmazonEKSForFargate` confia nos seguintes serviços para aceitar o perfil:

- `eks-fargate.amazonaws.com`

A política de permissões da função permite que o Amazon EKS conclua as seguintes ações nos recursos especificados:

- AmazonEKSForFargateServiceRolePolicy

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para obter mais informações, consulte Permissões de perfil vinculado ao serviço no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon EKS

Não é necessário criar manualmente uma função vinculada ao serviço. Quando você cria um perfil do Fargate no Console de gerenciamento da AWS, na AWS CLI ou na API da AWS, o Amazon EKS cria o perfil vinculado a serviço para você.

> **Important**
>
> Esse perfil vinculado ao serviço pode aparecer em sua conta se você concluiu uma ação em outro serviço que usa os atributos compatíveis com esse perfil. Se você estava usando o serviço do Amazon EKS antes de 13 de dezembro de 2019, quando ele começou a oferecer suporte a funções vinculadas ao serviço, o Amazon EKS criou a função AWSServiceRoleForAmazonEKSForFargate na sua conta. Para saber mais, consulte Uma nova função apareceu na minha conta do IAM.

Criação de uma função vinculada ao serviço no Amazon EKS (API da AWS)

Não é necessário criar manualmente uma função vinculada ao serviço. Quando você cria um perfil do Fargate no Console de gerenciamento da AWS, na AWS CLI ou na API da AWS, o Amazon EKS cria o perfil vinculado a serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria outro grupo de nós gerenciados, o Amazon EKS cria a função vinculada ao serviço novamente para você.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKSForFargate`. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para obter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um atributo ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, escolha Clusters.
3. Na página Clusters, selecione seu cluster.
4. Selecione a guia Compute (Computação).
5. Se houver algum perfil do Fargate na seção Fargate profiles (Perfis do Fargate), selecione cada um individualmente e, depois, Delete (Excluir).
6. Digite o nome do perfil na janela de confirmação de exclusão e escolha Delete (Excluir) para excluir.
7. Repita o procedimento para todos os perfis do Fargate no cluster e para outros clusters na sua conta.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API AWS para excluir o perfil vinculada ao serviço AWSServiceRoleForAmazonEKSForFargate. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte a funções vinculadas a serviço do Amazon EKS

O Amazon EKS oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço estiver disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon EKS.

Usar funções para conectar um cluster do Kubernetes ao Amazon EKS

O Amazon EKS usa perfis vinculados a serviços do AWS Identity and Access Management (IAM). A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros serviços que oferecem suporte a perfis vinculados ao serviço, consulte Serviços da AWS que funcionam com o IAM e procure pelos serviços que apresentam a afirmativa Sim na coluna Perfil vinculado ao serviço. Escolha um Sim com um link para visualizar a documentação do perfil vinculado para esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa o perfil vinculado a serviço chamado `AWSServiceRoleForAmazonEKSConnector`. O perfil permite que o Amazon EKS conecte clusters do Kubernetes. As políticas anexadas permitem que o perfil gerencie os recursos necessários para se conectar ao cluster do Kubernetes registrado.

O perfil vinculado ao serviço `AWSServiceRoleForAmazonEKSConnector` confia nos seguintes serviços para aceitar o perfil:

- `eks-connector.amazonaws.com`

A política de permissões da função permite que o Amazon EKS conclua as seguintes ações nos recursos especificados:

- AmazonEKSConnectorServiceRolePolicy

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para obter mais informações, consulte Permissões de perfil vinculado a serviços no Guia do usuário do IAM.

Este perfil usa permissões do SSM (Systems Manager) para estabelecer conexões seguras e gerenciar clusters do Kubernetes conectados.

Criar uma função vinculada ao serviço para o Amazon EKS

Você não precisa criar manualmente uma função vinculada a serviço para se conectar a um cluster. Quando você conecta um cluster no Console de gerenciamento da AWS, no AWS CLI, no `eksctl` ou na API AWS, o Amazon EKS cria o perfil vinculada ao serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando você conecta um cluster, o Amazon EKS cria uma função vinculada a serviço para você novamente.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKSConnector`. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para obter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um recurso ou serviço que requer um perfil vinculado ao serviço, é recomendável excluí-lo. Dessa forma, você não tem uma entidade não utilizada que não seja

monitorada ativamente ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, escolha Clusters.
3. Na página Clusters, selecione seu cluster.
4. Selecione a guia Deregister (Cancelar registro) e, em seguida, selecione a guia OK.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API do AWS para excluir o perfil vinculada ao serviço AWSServiceRoleForAmazonEKSConnector. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Usar funções para clusters locais do Amazon EKS no Outpost

O Amazon Elastic Kubernetes Service usa perfis vinculados a serviço do AWS Identity and Access Management (IAM). A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode

assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Uma função vinculada ao serviço poderá ser excluída somente após excluir seus recursos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com funções vinculadas aos serviços, consulte Serviços da AWS que funcionam com o IAM e procure os serviços que apresentam Sim na coluna Funções vinculadas aos serviços. Escolha um Sim com um link para visualizar a documentação do perfil vinculado para esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa o perfil vinculado a serviço chamado `AWSServiceRoleForAmazonEKSLocalOutpost`. O perfil permite que o Amazon EKS gerencie clusters locais em sua conta. As políticas anexadas permitem que a função gerencie os seguintes recursos: interfaces de rede, grupos de segurança, logs e instâncias do Amazon EC2.

> **Note**
>
> A função vinculada ao serviço `AWSServiceRoleForAmazonEKSLocalOutpost` é distinta da função necessária para a criação do cluster. Para ter mais informações, consulte the section called “Função do IAM do cluster”.

A função vinculada ao serviço `AWSServiceRoleForAmazonEKSLocalOutpost` confia nos seguintes serviços para assumir a função:

- `outposts.eks-local.amazonaws.com`

A política de permissões da função permite que o Amazon EKS conclua as seguintes ações nos recursos especificados:

- AmazonEKSServiceRolePolicy

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para obter mais informações, consulte Permissões de perfil vinculado ao serviço no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon EKS

Não é necessário criar manualmente uma função vinculada ao serviço. Quando você cria um cluster no Console de gerenciamento da AWS, na AWS CLI ou na API AWS, o Amazon EKS cria o perfil vinculado a serviço para você.

Se excluir esse perfil vinculado ao serviço e precisar criá-lo novamente, será possível usar esse mesmo processo para recriar o perfil em sua conta. Quando você cria um cluster, o Amazon EKS cria uma função vinculada ao serviço para você novamente.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKSLocalOutpost`. Depois que você criar um perfil vinculado ao serviço, não poderá alterar o nome do perfil, pois várias entidades podem fazer referência ao perfil. No entanto, será possível editar a descrição da função usando o IAM. Para obter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um atributo ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

1. Abra o console do Amazon EKS.
2. No painel de navegação à esquerda, escolha Clusters do Amazon EKS.

3. Se o cluster tiver grupos de nós ou perfis do Fargate, você deverá excluí-los antes de excluir o cluster. Para ter mais informações, consulte the section called “Excluir” e the section called “Excluir perfis”.
4. NoClustersEscolha o cluster que você quer excluir e escolhaExcluir.
5. Digite o nome do cluster na janela de confirmação de exclusão e escolha Delete (Excluir) para excluir.
6. Repita esse procedimento para qualquer outro cluster na sua conta. Aguarde até que todas as operações de exclusão sejam concluídas.

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API da AWS para excluir a função vinculada ao serviço `AWSServiceRoleForAmazonEKSLocalOutpost`. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte a funções vinculadas a serviço do Amazon EKS

O Amazon EKS oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço estiver disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon EKS.

Como usar perfis para o painel do Amazon EKS

O painel do Amazon EKS usa esse perfil vinculado ao serviço para agregar informações sobre os recursos de clusters de várias regiões e contas. O painel se integra ao AWS Organizations para ler com segurança as informações sobre várias contas em sua organização.

Para saber mais sobre o painel do Amazon EKS, consulte the section called “Painel do Amazon EKS”.

Contexto

O Amazon EKS usa perfis vinculados a serviços do AWS Identity and Access Management (IAM). A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada diretamente ao Amazon EKS. As funções vinculadas a serviços são predefinidas pelo Amazon EKS e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do Amazon EKS porque você não precisa adicionar as permissões necessárias manualmente. O Amazon EKS define as permissões das

funções vinculadas ao serviço e, exceto se definido de outra forma, somente o Amazon EKS pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, que não pode ser anexada a nenhuma outra entidade do IAM.

Um perfil vinculado ao serviço poderá ser excluído somente após excluir seus atributos relacionados. Isso protege seus recursos do Amazon EKS, pois você não pode remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com funções vinculadas aos serviços, consulte Serviços da AWS que funcionam com o IAM e procure os serviços que apresentam Sim na coluna Funções vinculadas aos serviços. Escolha um Sim com um link para visualizar a documentação do perfil vinculado para esse serviço.

Permissões de função vinculada ao serviço para o Amazon EKS

O Amazon EKS usa o perfil vinculado a serviço chamado `AWSServiceRoleForAmazonEKSDashboard`. O perfil permite que o Amazon EKS gere e exiba o painel do EKS, incluindo informações agregadas sobre os recursos de clusters. A política `AmazonEKSDashboardServiceRolePolicy` anexada permite que o perfil gerencie os recursos a seguir: grupos do Auto Scaling, grupos de segurança, modelos de execução e perfis de instância do IAM. Para obter mais informações, consulte the section called “Política gerenciada pela AWS: AmazonEKSDashboardServiceRolePolicy”.

A função vinculada ao serviço `AWSServiceRoleForAmazonEKSDashboard` confia nos seguintes serviços para assumir a função:

- `dashboard.eks.amazonaws.com`

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSDashboardServiceRolePolicy no Guia de referência de políticas gerenciadas pela AWS.

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua um perfil vinculado a serviço. Para obter mais informações, consulte Permissões de perfil vinculado ao serviço no Guia do usuário do IAM.

Criar uma função vinculada ao serviço para o Amazon EKS

Não é necessário criar manualmente uma função vinculada ao serviço. Ao habilitar o painel no Console da AWS, o Amazon EKS cria o perfil vinculado ao serviço para você.

Para obter mais informações sobre como habilitar o painel do EKS, consulte the section called "Configurar organizações".

> **Important**
>
> Esse perfil vinculado ao serviço pode aparecer em sua conta se você concluiu uma ação em outro serviço que usa os atributos compatíveis com esse perfil.

Editar uma função vinculada ao serviço do Amazon EKS

O Amazon EKS não permite que você edite a função vinculada ao serviço do `AWSServiceRoleForAmazonEKSDashboard`. Depois que criar um perfil vinculado ao serviço, você não poderá alterar o nome do perfil, pois várias entidades podem fazer referência a ele. No entanto, será possível editar a descrição do perfil usando o IAM. Para obter mais informações, consulte Editar um perfil vinculado ao serviço no Guia do usuário do IAM.

Excluir uma função vinculada ao serviço do Amazon EKS

Se você não precisar mais usar um recurso ou serviço que requer um perfil vinculado ao serviço, é recomendável excluí-lo. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar suo perfil vinculado ao serviço para excluí-la manualmente.

Limpar um perfil vinculado ao serviço

Antes de usar o IAM para excluir uma função vinculada ao serviço, você deverá excluir qualquer recurso usado pela função.

> **Note**
>
> Se o serviço do Amazon EKS estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

Para obter mais informações sobre como desabilitar o painel do EKS, consulte the section called "Configurar organizações".

Excluir manualmente o perfil vinculado ao serviço

Use o console do IAM, a AWS CLI ou a API da AWS para excluir a função vinculada ao serviço `AWSServiceRoleForAmazonEKSDashboard`. Para obter mais informações, consulte Excluir um perfil vinculado ao serviço no Guia do usuário do IAM.

Regiões com suporte a funções vinculadas a serviço do Amazon EKS

O Amazon EKS oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço estiver disponível. Para obter mais informações, consulte Endpoints e cotas do Amazon EKS.

Perfil do IAM de execução de pods do Amazon EKS

O perfil de execução de pods do Amazon EKS é necessário para executar pods na infraestrutura do AWS Fargate.

Quando o cluster cria pods na infraestrutura do AWS Fargate, os componentes que estão em execução nessa infraestrutura do Fargate devem fazer as chamadas para as APIs da AWS em seu nome. Isso é para que eles possam realizar ações como extrair imagens de contêiner do Amazon ECR ou encaminhar logs para outros serviços da AWS. O perfil de execução de pods do Amazon EKS fornece as permissões do IAM para fazer isso.

Ao criar um perfil do Fargate, é necessário especificar um perfil de execução de pods para os componentes do Amazon EKS que são executados na infraestrutura do Fargate usando o perfil. Esse perfil é adicionado ao Controle de acesso baseado em perfil (RBAC) do Kubernetes do cluster para autorização. Isso permite que o `kubelet` que está sendo executado na infraestrutura do Fargate seja registrado no cluster do Amazon EKS para aparecer no cluster como um nó.

> **Note**
>
> O perfil do Fargate deve ter uma função do IAM diferente dos grupos de nós do Amazon EC2.

> **Important**
>
> Os contêineres em execução no pod do Fargate não podem assumir as permissões do IAM associadas a um perfil de execução do pods. Para conceder aos contêineres no pod do

Fargate permissões de acesso a outros serviços da AWS, é necessário usar os perfis do IAM das contas de serviço.

Antes de criar um perfil do Fargate, você deve criar um perfil do IAM com AmazonEKSFargatePodExecutionRolePolicy.

Verifique se existe um perfil de execução de pods existente e configurado corretamente

Você pode usar o procedimento a seguir para verificar se a conta já tem um perfil de execução de pods do Amazon EKS corretamente configurado. Para evitar um problema de segurança de representante confuso, é importante que a função restrinja o acesso com base no `SourceArn`. É possível modificar a função de execução conforme necessário de forma a incluir suporte para perfis do Fargate em outros clusters.

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Na página Roles (Funções), procure AmazonEKSFargatePodExecutionRole na lista de funções. Se a função não existir, consulte the section called “Criar o perfil de execução de pods do Amazon EKS” para criá-la. Se a função existir, escolha-a.
4. Na página AmazonEKSFargatePodExecutionRole, faça o seguinte:
   a. Escolha Permissões.
   b. Certifique-se de que a política gerenciada pela Amazon AmazonEKSFargatePodExecutionRolePolicy esteja associada ao perfil.
   c. Escolha Relações de Confiança.
   d. Escolha Edit trust policy (Editar política de confiança).
5. Na página Edit trust policy (Editar política de confiança), verifique se a relação de confiança contém a política a seguir e tem uma linha para perfis do Fargate no cluster. Em caso afirmativo, escolha Cancel (Cancelar).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
```

```
          "ArnLike": {
             "aws:SourceArn": "arn:aws:eks:us-east-1:111122223333:fargateprofile/my-
cluster/*"
          }
       },
       "Principal": {
         "Service": "eks-fargate-pods.amazonaws.com"
       },
       "Action": "sts:AssumeRole"
     }
  ]
}
```

Se a política corresponder, mas não tiver uma linha especificando os perfis do Fargate no seu cluster, será possível adicionar a seguinte linha sobre o objeto `ArnLike`. Substitua *`region-code`* pela região da AWS em que seu cluster está localizado, *`111122223333`* pelo ID da sua conta e *`my-cluster`* pelo nome do seu cluster.

```
"aws:SourceArn": "arn:aws:eks:region-code:111122223333:fargateprofile/my-cluster/*",
```

Se a política não corresponder, copie a política anterior completa para o formulário e escolha Atualizar política. Substitua *`region-code`* pela região da AWS em que seu cluster se encontra. Se você quiser usar o mesmo perfil em todas as regiões da AWS da sua conta, substitua *`region-code`* por *. Substitua *`111122223333`* pelo ID da sua conta e *`my-cluster`* pelo nome do seu cluster. Se quiser usar o mesmo perfil para todos os clusters da sua conta, substitua *`my-cluster`* por *.

Criar o perfil de execução de pods do Amazon EKS

Se ainda não tiver o perfil de execução de pods do Amazon EKS para o cluster, você poderá usar o Console de gerenciamento da AWS ou a AWS CLI para criá-lo.

Console de gerenciamento da AWS

a. Abra o console do IAM em https://console.aws.amazon.com/iam/.
b. No painel de navegação à esquerda, escolha Funções.
c. Na página Perfis, selecione Criar perfil.
d. Na página Selecionar entidade confiável, faça o seguinte:
   i. Na seção Tipo de entidade confiável), escolha Service da AWS.

ii. Na lista suspensa Casos de uso para outros serviços AWS, escolha EKS.

iii. Escolha EKS - Pod do Fargate.

iv. Escolha Próximo.

e. Na página Add permissions (Adicionar permissões), escolha Next (Próximo).

f. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:

i. Em Nome do perfil, insira um nome exclusivo para o perfil, como `AmazonEKSFargatePodExecutionRole`.

ii. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.

iii. Selecione Criar perfil.

g. Na página Roles (Funções), procure AmazonEKSFargatePodExecutionRole na lista de funções. Escolha a função.

h. Na página AmazonEKSFargatePodExecutionRole, faça o seguinte:

i. Escolha Relações de Confiança.

ii. Escolha Edit trust policy (Editar política de confiança).

i. Na página Edit trust policy (Editar política de confiança), faça o seguinte:

i. Copie e cole o conteúdo a seguir no formulário Edit trust policy (Editar política de confiança). Substitua *region-code* pela região da AWS em que seu cluster se encontra. Se você quiser usar o mesmo perfil em todas as regiões da AWS da sua conta, substitua *region-code* por *. Substitua *111122223333* pelo ID da sua conta e *my-cluster* pelo nome do seu cluster. Se quiser usar o mesmo perfil para todos os clusters da sua conta, substitua *my-cluster* por *.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:us-
east-1:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
```

```
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

ii. Escolha Atualizar política.

AWS CLI

a. Copie e cole o conteúdo a seguir em um arquivo denominado `pod-execution-role-trust-policy.json`. Substitua *`region-code`* pela região da AWS em que seu cluster se encontra. Se você quiser usar o mesmo perfil em todas as regiões da AWS da sua conta, substitua *`region-code`* por *. Substitua *`111122223333`* pelo ID da sua conta e *`my-cluster`* pelo nome do seu cluster. Se quiser usar o mesmo perfil para todos os clusters da sua conta, substitua *`my-cluster`* por *.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:us-east-1:111122223333:fargateprofile/
my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Crie um perfil do IAM de execução de pods.

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

c. Anexe a política do IAM gerenciada pelo Amazon EKS ao perfil.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Função do IAM do conector do Amazon EKS

Você pode conectar clusters do Kubernetes para visualizá-los no Console de gerenciamento da AWS. Para se conectar a um cluster do Kubernetes, crie uma função do IAM.

Verificar se já existe um perfil de conector do EKS

Use o procedimento a seguir para verificar se a conta já tem a função de conector do Amazon EKS.

1. Abra o console do IAM, em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Pesquise na lista de perfis `AmazonEKSConnectorAgentRole`. Se não houver uma função que inclua `AmazonEKSConnectorAgentRole`, consulte the section called “Criar a função do IAM para o agente de conector do Amazon EKS” para criar a função. Se houver uma função que inclua `AmazonEKSConnectorAgentRole`, selecione-a para visualizar as políticas anexadas.
4. Escolha Permissões.
5. Verifique se a política gerenciada AmazonEKSConnectorAgentPolicy está anexada ao perfil. Se a política estiver anexada, o perfil de conector do Amazon EKS estará configurado corretamente.
6. Escolha Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).
7. Verifique se o relacionamento de confiança contém a seguinte política: Se o relacionamento de confiança corresponder à seguinte política, escolha Cancel (Cancelar). Se o relacionamento de confiança não corresponder, copie a política para a janela Edit trust policy (Editar política de confiança) e escolha Update policy (Atualizar política).

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
```

```
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Criar a função do IAM para o agente de conector do Amazon EKS

Você pode usar o Console de gerenciamento da AWS ou o AWS CloudFormation para criar o perfil de agente conector.

AWS CLI

a. Crie um arquivo denominado `eks-connector-agent-trust-policy.json` que contenha o seguinte JSON para ser usado para a função do IAM:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Crie um arquivo denominado `eks-connector-agent-policy.json` que contenha o seguinte JSON para ser usado para a função do IAM.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SsmControlChannel",
            "Effect": "Allow",
```

```
            "Action": [
                "ssmmessages:CreateControlChannel"
            ],
            "Resource": "arn:aws:eks:*:*:cluster/*"
        },
        {
            "Sid": "ssmDataplaneOperations",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenDataChannel",
                "ssmmessages:OpenControlChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

c. Crie a função de agente do Amazon EKS Connector usando a política de confiança e a política que você criou nos itens de lista anteriores.

```
aws iam create-role \
     --role-name AmazonEKSConnectorAgentRole \
     --assume-role-policy-document file://eks-connector-agent-trust-policy.json
```

d. Anexe a política à função do agente do Amazon EKS Connector.

```
aws iam put-role-policy \
     --role-name AmazonEKSConnectorAgentRole \
     --policy-name AmazonEKSConnectorAgentPolicy \
     --policy-document file://eks-connector-agent-policy.json
```

AWS CloudFormation

a. Salve o seguinte modelo do AWS CloudFormation em um arquivo de texto em seu sistema local.

> **Note**
>
> Esse modelo também cria a função vinculada ao serviço que seria criada quando a API `registerCluster` fosse chamada. Para mais detalhes, consulte the section called “Perfil do conector de cluster”.

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Provisions necessary resources needed to register clusters in EKS'
Parameters: {}
Resources:
  EKSConnectorSLR:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: eks-connector.amazonaws.com

  EKSConnectorAgentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action: [ 'sts:AssumeRole' ]
            Principal:
              Service: 'ssm.amazonaws.com'

  EKSConnectorAgentPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: EKSConnectorAgentPolicy
      Roles:
        - {Ref: 'EKSConnectorAgentRole'}
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateControlChannel' ]
            Resource:
            - Fn::Sub: 'arn:${AWS::Partition}:eks:*:*:cluster/*'
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateDataChannel',
 'ssmmessages:OpenDataChannel', 'ssmmessages:OpenControlChannel' ]
            Resource: "*"
Outputs:
  EKSConnectorAgentRoleArn:
    Description: The agent role that EKS connector uses to communicate with AWS
 services.
```

```
    Value: !GetAtt EKSConnectorAgentRole.Arn
```

b. Abra o console do AWS CloudFormation.

c. Escolha Criar pilha com novos recursos (padrão).

d. Para Specify template (Especificar modelo), selecione Upload a template file (Fazer upload de um arquivo de modelo) e depois Choose file (Escolher arquivo).

e. Selecione o arquivo que você criou anteriormente e, em seguida, selecione Next (Próximo).

f. Em Stack name (Nome da pilha), insira um nome para a função, como `eksConnectorAgentRole`, e selecione Next (Próximo).

g. Na página Configurar opções de pilha, selecione Avançar.

h. Na página Review (Revisão), reveja as informações, confirme se a pilha pode criar recursos do IAM e, em seguida, selecione Create (Criar).

Políticas gerenciadas pela AWS para o Amazon Elastic Kubernetes Service

Uma política gerenciada pela AWS é uma política autônoma criada e administrada pela AWS. As políticas gerenciadas pela AWS são criadas para fornecer permissões a vários casos de uso comuns e permitir a atribuição de permissões a usuários, grupos e perfis.

Lembre-se de que as políticas gerenciadas pela AWS podem não conceder permissões de privilégio mínimo para casos de uso específicos porque estão disponíveis para todos os clientes da AWS usarem. Recomendamos que você reduza ainda mais as permissões definindo as políticas gerenciadas pelo cliente que são específicas para seus casos de uso.

Você não pode alterar as permissões definidas em políticas gerenciadas pela AWS. Caso a AWS faça atualizações nas permissões estabelecidas em uma política gerenciada pela AWS, estas mudanças impactarão todas as identidades das entidades principais (usuários, grupos e funções) vinculadas à esta política. A AWS é mais propensa a atualizar uma política gerenciada pela AWS durante o lançamento de um novo serviço da AWS ou quando novas operações de API estiverem disponíveis para serviços existentes.

Para obter mais informações, consulte Políticas gerenciadas pela AWS no Manual do usuário do IAM.

Política gerenciada pela AWS: AmazonEKS_CNI_Policy

Você pode anexar a `AmazonEKS_CNI_Policy` às suas entidades do IAM. Antes de criar um grupo de nós do Amazon EC2, essa política deve ser anexada ao perfil do IAM do nó ou a um perfil do IAM

que seja usado especificamente pelo plug-in CNI da Amazon VPC para Kubernetes. Isso é para que ele possa realizar ações em seu nome. Recomendamos que você anexe a política a uma função que é usada apenas pelo plugin. Para obter mais informações, consulte the section called “CNI da Amazon VPC” e the section called “Configuração para IRSA”.

Detalhes das permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2:*NetworkInterface`** e **`ec2:*PrivateIpAddresses`** : permite que o plug-in CNI da Amazon VPC execute ações, como o provisionamento de interfaces de rede elástica e de endereços IP para pods a fim de fornecer redes para as aplicações executadas no Amazon EKS.
- Ações de leitura do **`ec2`**: permite que o plug-in CNI da Amazon VPC execute ações, como a descrição de instâncias e de sub-redes, para visualizar a quantidade de endereços IP disponíveis nas sub-redes da Amazon VPC. O plug-in CNI da VPC pode usar os endereços IP disponíveis em cada sub-rede para escolher as sub-redes com a maior quantidade de endereços IP disponíveis para usar ao criar uma interface de rede elástica.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKS_CNI_Policy no Guia de referência de políticas gerenciadas da AWS.

AWSPolítica gerenciada da : AmazonEKSClusterPolicy

Você pode anexar `AmazonEKSClusterPolicy` às entidades do IAM. Antes de criar um cluster, é preciso ter um perfil do IAM para o cluster com esta política anexada. Os clusters do Kubernetes gerenciados pelo Amazon EKS fazem chamadas para outros serviços da AWS em seu nome. Eles fazem isso para gerenciar os recursos que você usa junto com o serviço.

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`autoscaling`**: leia e atualize a configuração de um grupo do Auto Scaling. Essas permissões não são usadas pelo Amazon EKS, mas permanecem na política para compatibilidade com versões anteriores.
- **`ec2`** : trabalhe com volumes e recursos de rede associados aos nós do Amazon EC2. Isso é necessário para que o ambiente de gerenciamento do Kubernetes possa unir instâncias a um cluster e provisionar e gerenciar dinamicamente os volumes do Amazon EBS solicitados por volumes persistentes do Kubernetes.

- **`ec2`** : exclua interfaces de rede elástica criadas pela CNI da VPC. Isso é necessário para que o EKS possa limpar as interfaces de rede elástica que serão esquecidas se a CNI da VPC for encerrada inesperadamente.
- **`elasticloadbalancing`** : trabalhe com Elastic Load Balancers e adicione nós a eles como alvos. Isso é necessário para que o plano de controle do Kubernetes possa provisionar dinamicamente o Elastic Load Balancers solicitado pelos serviços do Kubernetes.
- **`iam`** – Crie uma função vinculada a serviço. Isso é necessário para que o ambiente de gerenciamento do Kubernetes possa provisionar dinamicamente o Elastic Load Balancers solicitado pelos serviços do Kubernetes.
- **`kms`** : leia uma chave do AWS KMS. Isso é necessário para que o ambiente de gerenciamento do Kubernetes suporte a criptografia de segredos dos segredos do Kubernetes armazenados no `etcd`.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSClusterPolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada pela AWS: AmazonEKSDashboardConsoleReadOnly

Você pode anexar `AmazonEKSDashboardConsoleReadOnly` às entidades do IAM.

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`eks`** : acesso somente leitura às informações de versão do cluster, aos recursos e aos dados do painel do EKS. Isso permite visualizar as métricas relacionadas ao EKS e os detalhes da configuração do cluster.
- **`organizations`** : acesso somente leitura às informações do AWS Organizations, incluindo:
  - Visualização de detalhes da organização e acesso ao serviço
  - Listagem de raízes organizacionais, contas e unidades organizacionais
  - Visualização da estrutura da organização

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSDashboardConsoleReadOnly no Guia de referência de políticas gerenciadas pela AWS.

AWSPolítica gerenciada da : AmazonEKSFargatePodExecutionRolePolicy

Você pode anexar `AmazonEKSFargatePodExecutionRolePolicy` às entidades do IAM. Antes de criar um perfil do Fargate, você deve criar um perfil de execução de pod do Fargate e anexar essa

política a ele. Para obter mais informações, consulte the section called “Etapa 2: criar um perfil de execução de pods do Fargate” e the section called “Definir perfis”.

Essa política concede ao perfil as permissões que fornecem acesso a outros recursos de serviços da AWS necessários para executar pods do Amazon EKS no Fargate.

Detalhes das permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ecr`** – permite que pods em execução no Fargate extraiam imagens de contêiner armazenadas no Amazon ECR.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSFargatePodExecutionRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada pela AWS: AmazonEKSConnectorServiceRolePolicy

Não é possível anexar `AmazonEKSConnectorServiceRolePolicy` às entidades do IAM. Esta política é anexada a uma função vinculada ao serviço que permite ao Amazon EKS realizar ações em seu nome. Para obter mais informações, consulte the section called “Perfil do conector de cluster”.

O perfil permite que o Amazon EKS conecte clusters do Kubernetes. As políticas anexadas permitem que o perfil gerencie os recursos necessários para se conectar ao cluster do Kubernetes registrado.

Detalhes relacionados às permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`SSM Management`**: permite criar, descrever e excluir ativações do SSM, bem como cancelar o registro de instâncias gerenciadas. Isso viabiliza operações básicas do Systems Manager.
- **`Session Management`**: permite iniciar sessões do SSM específicas para clusters do EKS e executar comandos não interativos por meio do documento AmazonEKS.
- **`IAM Role Passing`**: permite a transferência de perfis do IAM especificamente para o serviço SSM, controlado por uma condição que restringe os perfis transferidos para `ssm.amazonaws.com`.

- **`EventBridge Rules`**: permite criar regras e destinos no EventBridge somente quando gerenciados por `eks-connector.amazonaws.com`. As regras são especificamente limitadas ao AWS SSM como origem do evento.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSConnectorServiceRolePolicy no Guia de referência de políticas gerenciadas pela AWS.

AWSPolítica gerenciada da : AmazonEKSForFargateServiceRolePolicy

Não é possível anexar `AmazonEKSForFargateServiceRolePolicy` às entidades do IAM. Esta política é anexada a uma função vinculada ao serviço que permite ao Amazon EKS realizar ações em seu nome. Para obter mais informações, consulte `AWSServiceRoleforAmazonEKSForFargate`.

Esta política concede permissões necessárias ao Amazon EKS para executar tarefas do Fargate. A política só é usada se você tiver nós Fargate.

Detalhes relacionados às permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`** – crie e exclua interfaces de rede elástica e descreva os recursos e interfaces de rede elástica. Isso é necessário para que o serviço Amazon EKS Fargate possa configurar a rede VPC necessária para Fargate Pods.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSForFargateServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada pela AWS: AmazonEKSComputePolicy

Você pode anexar `AmazonEKSComputePolicy` às entidades do IAM. Você pode anexar essa política ao perfil do IAM do cluster para expandir os recursos que o EKS pode gerenciar na sua conta.

Essa política concede as permissões necessárias para o Amazon EKS criar e gerenciar instâncias do EC2 para o cluster do EKS, bem como as permissões do IAM necessárias para configurar o EC2. Além disso, essa política concede ao Amazon EKS as permissões para criar o perfil vinculado ao serviço de instâncias spot do EC2 em seu nome.

Detalhes das permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`** Permissões do:
  - `ec2:CreateFleet` e `ec2:RunInstances`: permitem criar instâncias do EC2 e usar recursos específicos do EC2 (imagens, grupos de segurança, sub-redes) para nós do cluster do EKS.
  - `ec2:CreateLaunchTemplate`: permite criar modelos de execução do EC2 para nós de cluster do EKS.
  - A política também inclui condições para restringir o uso dessas permissões do EC2 a recursos com tags com o nome do cluster do EKS e outras tags relevantes.
  - `ec2:CreateTags`: permite adicionar tags aos recursos do EC2 criados pelas ações `CreateFleet`, `RunInstances` e `CreateLaunchTemplate`.
- **`iam`** Permissões do:
  - `iam:AddRoleToInstanceProfile`: permite adicionar um perfil do IAM ao perfil de instância de computação do EKS.
  - `iam:PassRole`: permite passar os perfis do IAM necessários ao serviço do EC2.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSComputePolicy no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada da AWS: AmazonEKSNetworkingPolicy

Você pode anexar `AmazonEKSNetworkingPolicy` às entidades do IAM. Você pode anexar essa política ao perfil do IAM do cluster para expandir os recursos que o EKS pode gerenciar na sua conta.

Essa política foi projetada para conceder as permissões necessárias para o Amazon EKS criar e gerenciar interfaces de rede para o cluster do EKS, permitindo que o ambiente de gerenciamento e os nós de processamento se comuniquem e funcionem corretamente.

Detalhes da permissão

Essa política concede as seguintes permissões para permitir que o Amazon EKS gerencie interfaces de rede para o cluster:

- **`ec2`** Permissões de interface de rede:

- `ec2:CreateNetworkInterface`: permite a criação de interfaces de rede do EC2.
- A política inclui condições para restringir o uso dessa permissão a interfaces de rede marcadas com o nome do cluster do EKS e o nome do nó CNI do Kubernetes.
- `ec2:CreateTags`: permite adicionar etiquetas às interfaces de rede criadas pela ação `CreateNetworkInterface`.

- **`ec2`** Permissões de gerenciamento de interface de rede do:
  - `ec2:AttachNetworkInterface`, `ec2:DetachNetworkInterface`: permite anexar e desanexar interfaces de rede a instâncias do EC2.
  - `ec2:UnassignPrivateIpAddresses`, `ec2:UnassignIpv6Addresses`, `ec2:AssignPrivateIpAddresses`, `ec2:AssignIpv6Addresses`: permite gerenciar as atribuições de endereço IP das interfaces de rede.
  - Essas permissões são restritas às interfaces de rede marcadas com o nome do cluster do EKS.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSNetworkingPolicy no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AmazonEKSBlockStoragePolicy

Você pode anexar `AmazonEKSBlockStoragePolicy` às entidades do IAM. Você pode anexar essa política ao perfil do IAM do cluster para expandir os recursos que o EKS pode gerenciar na sua conta.

Essa política concede as permissões necessárias para o Amazon EKS criar, gerenciar e manter volumes e snapshots do EC2 para o cluster EKS, permitindo que o ambiente de gerenciamento e os nós de processamento provisionem e usem armazenamento persistente conforme exigido pelas workloads do Kubernetes.

Detalhes da permissão

Essa política do IAM concede as permissões a seguir para permitir que o Amazon EKS gerencie volumes e snapshots do EC2:

- **`ec2`** Permissões de gerenciamento de volume do:
  - `ec2:AttachVolume`, `ec2:DetachVolume`, `ec2:ModifyVolume`, `ec2:EnableFastSnapshotRestores`: permitem anexar, desanexar, modificar e habilitar restaurações rápidas de instantâneos para volumes do EC2.

  - Essas permissões são restritas aos volumes com tag do nome do cluster do EKS.
  - `ec2:CreateTags`: permite adicionar tags aos volumes e snapshots do EC2 criados pelas ações `CreateVolume` e `CreateSnapshot`.
- **`ec2`** Permissões de criação de volume do:
  - `ec2:CreateVolume`: permite criar novos volumes do EC2.
  - A política inclui condições para restringir o uso dessa permissão a volumes com tags com o nome do cluster do EKS e outras tags relevantes.
  - `ec2:CreateSnapshot`: permite criar novos snapshots de volumes do EC2.
  - A política inclui condições para restringir o uso dessa permissão a snapshots com tags com o nome do cluster do EKS e outras tags relevantes.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSBlockStoragePolicy no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AmazonEKSLoadBalancingPolicy

Você pode anexar `AmazonEKSLoadBalancingPolicy` às entidades do IAM. Você pode anexar essa política ao perfil do IAM do cluster para expandir os recursos que o EKS pode gerenciar na sua conta.

Essa política do IAM concede as permissões necessárias para que o Amazon EKS trabalhe com vários serviços da AWS para gerenciar o Elastic Load Balancing (ELB) e recursos relacionados.

Detalhes da permissão

As principais permissões concedidas por essa política são:

- **`elasticloadbalancing`** : permite criar, modificar e gerenciar balanceadores de carga elásticos e grupos de destino. Isso inclui permissões para criar, atualizar e excluir balanceadores de carga, grupos de destino, receptores e regras.
- **`ec2`** : permite criar e gerenciar grupos de segurança, necessários para que o ambiente de gerenciamento do Kubernetes possa unir instâncias a um cluster e gerenciar os volumes do Amazon EBS. Além disso, possibilita descrever e listar recursos do EC2, como instâncias, VPCs, sub-redes, grupos de segurança e outros recursos de rede.
- **`iam`** : permite criar um perfil vinculado a serviço para o Elastic Load Balancing, que é necessário para que o ambiente de gerenciamento do Kubernetes provisione ELBs dinamicamente.

- **`kms`**: permite a leitura de uma chave do AWS KMS, necessária para que o ambiente de gerenciamento do Kubernetes ofereça suporte à criptografia de segredos do Kubernetes armazenados no etcd.
- **`wafv2`** e **`shield`**: permitem associar e desassociar ACLs da Web e criar/excluir proteções do AWS Shield para os balanceadores de carga elásticos.
- **`cognito-idp`**, **`acm`** e **`elasticloadbalancing`**: concede permissões para descrever clientes do grupo de usuários, listar e descrever certificados e descrever grupos de destino, que são necessários para que o ambiente de gerenciamento do Kubernetes gerencie os balanceadores de carga elásticos.

A política também inclui várias verificações de condição para garantir que as permissões tenham como escopo o cluster do EKS específico que está sendo gerenciado, usando a tag `eks:eks-cluster-name`.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSLoadBalancingPolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada pela AWS: AmazonEKSMCPReadOnlyAccess

Você pode anexar `AmazonEKSMCPReadOnlyAccess` às entidades do IAM. Essa política fornece acesso somente de leitura aos recursos do Amazon EKS e aos serviços da AWS relacionados, permitindo que o servidor do protocolo de contexto para modelos (MCP) do Amazon EKS execute operações de observabilidade e de solução de problemas sem realizar quaisquer modificações na infraestrutura.

Detalhes relacionados às permissões

Essa política inclui as seguintes permissões que possibilitam que as entidades principais realizem as seguintes tarefas:

- **`eks`**: permite que as entidades principais descrevam e listem clusters do EKS, grupos de nós, complementos, entradas de acesso, insights, e acessem a API do Kubernetes para operações somente de leitura.
- **`iam`**: permite que as entidades principais recuperem informações sobre perfis do IAM, políticas e seus anexos para entender as permissões associadas aos recursos do EKS.
- **`ec2`**: permite que as entidades principais descrevam VPCs, sub-redes e tabelas de rotas para compreender a configuração de rede dos clusters do EKS.

- **`sts`**: permite que as entidades principais recuperem informações de identidade do chamador para fins de autenticação e autorização.
- **`logs`**: permite que as entidades principais iniciem consultas e recuperem os resultados de consultas do CloudWatch Logs para solução de problemas e monitoramento.
- **`cloudwatch`**: permite que as entidades principais recuperem dados de métricas para monitorar a performance de clusters e workloads.
- **`eks-mcp`**: permite que as entidades principais invoquem operações MCP e usem ferramentas somente de leitura dentro do servidor MCP do Amazon EKS.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSMCPReadOnlyAccess no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AmazonEKSServicePolicy

Você pode anexar `AmazonEKSServicePolicy` às entidades do IAM. Os clusters criados antes de 16 de abril de 2020 exigiam que você criasse um perfil do IAM e associasse essa política a ele. Os clusters criados em ou após 16 de abril de 2020 não exigem que você crie uma função, nem que você atribua essa política. Quando você cria um cluster usando uma entidade principal do IAM que tem a permissão `iam:CreateServiceLinkedRole`, o perfil vinculado a serviço AWSServiceRoleforAmazonEKS é criado automaticamente para você. O perfil vinculada a serviço tem a política gerenciada: AmazonEKSServiceRolePolicy anexada a ela.

Essa política permite que o Amazon EKS crie e gerencie os recursos necessários para operar clusters do Amazon EKS.

Detalhes relacionados às permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`eks`** – atualize a versão do Kubernetes do cluster depois de iniciar uma atualização. Essa permissão não é usada pelo Amazon EKS, mas permanece na política para compatibilidade com versões anteriores.
- **`ec2`** – trabalhe com interfaces de rede elásticas e outros recursos e tags de rede. Isso é exigido pelo Amazon EKS para configurar redes que facilitem a comunicação entre nós e o ambiente de gerenciamento do Kubernetes. Leia informações sobre grupos de segurança. Atualize etiquetas em grupos de segurança.
- **`route53`** – associe uma VPC a uma zona hospedada. Isso é exigido pelo Amazon EKS para habilitar a rede privada de endpoint para o seu servidor de API de cluster do Kubernetes.

- **`logs`** – Eventos de log Isso é necessário para que o Amazon EKS possa enviar logs do plano de controle do Kubernetes para o CloudWatch.
- **`iam`** – crie um perfil vinculado a serviço. Isso é necessário para que o Amazon EKS crie o perfil vinculado ao serviço the section called “Permissões de função vinculada ao serviço para o Amazon EKS” em seu nome.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSServicePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada pela AWS: AmazonEKSServiceRolePolicy

Não é possível anexar `AmazonEKSServiceRolePolicy` às entidades do IAM. Esta política é anexada a uma função vinculada ao serviço que permite ao Amazon EKS realizar ações em seu nome. Para obter mais informações, consulte the section called “Permissões de função vinculada ao serviço para o Amazon EKS”. Quando você cria um cluster usando uma entidade principal do IAM que tem a permissão `iam:CreateServiceLinkedRole`, o perfil vinculado a serviço AWSServiceRoleforAmazonEKS é criado automaticamente para você e essa política é anexada a ele.

Esta política permite que o perfil vinculado ao serviço chame serviços da AWS da em seu nome.

Detalhes relacionados às permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`** – crie e descreva as interfaces de rede elástica e as instâncias do Amazon EC2, o grupo de segurança de cluster e VPC que são necessários para a criação de um cluster. Para obter mais informações, consulte the section called “Requisitos para grupos de segurança”. Leia informações sobre grupos de segurança. Atualize tags em grupos de segurança. Leia mais informações sobre reservas de capacidade sob demanda. Leia a configuração da VPC, incluindo tabelas de rotas e ACLs de rede, a fim de identificar problemas de configuração nos insights do cluster.
- Modo automático do **`ec2`**: encerre instâncias do EC2 criadas pelo Modo automático do EKS. Para obter mais informações, consulte Modo Automático do EKS.
- **`iam`** – liste todas as políticas gerenciadas anexadas a um perfil do IAM. Isso é necessário para que o Amazon EKS possa listar e validar todas as políticas gerenciadas e permissões necessárias para criar um cluster.
- Associar uma VPC a uma zona hospedada: isso é exigido pelo Amazon EKS para habilitar a rede privada de endpoint para o seu servidor de API de cluster do Kubernetes.

- Evento de logs: isso é necessário para que o Amazon EKS possa enviar logs do ambiente de gerenciamento do Kubernetes para o CloudWatch.
- Put metric: isto é necessário para que o Amazon EKS possa enviar logs do ambiente de gerenciamento do Kubernetes para o CloudWatch.
- **`eks`** - gerencia entradas e políticas de acesso ao cluster, permitindo um controle refinado sobre quem pode acessar os recursos do EKS e quais ações eles podem realizar. Isso inclui a associação de políticas de acesso padrão para operações de computação, rede, balanceamento de carga e armazenamento.
- **`elasticloadbalancing`** - cria, gerencia e exclui balanceadores de carga e seus componentes (receptores, grupos de destino, certificados) associados aos clusters do EKS. Visualiza os atributos e o status de integridade do balanceador de carga.
- **`events`**: cria e gerencia as regras do EventBridge para monitorar eventos do EC2 e do AWS Health relacionados aos clusters do EKS, permitindo respostas automatizadas às mudanças na infraestrutura e alertas de saúde.
- **`iam`** - gerencia perfis de instância do EC2 com o prefixo "eks", incluindo a criação, exclusão e associação de perfis, o que é necessário para o gerenciamento de nós do EKS.
- **`pricing shield`**: acessa as informações de preços da AWS e o status de proteção do Shield, permitindo o gerenciamento de custos e recursos avançados de segurança para os recursos do EKS.
- Limpeza de recursos: exclui com segurança recursos marcados com tag do EKS, incluindo volumes, instantâneos, modelos de lançamento e interfaces de rede durante as operações de limpeza do cluster.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada pela AWS: AmazonEKSVPCResourceController

É possível anexar a política `AmazonEKSVPCResourceController` às suas identidades do IAM. Se estiver usando grupos de segurança para Pods, você deverá anexar essa política à perfil IAM do cluster do Amazon EKS para executar ações em seu nome.

Esta política concede ao perfil do cluster permissões para gerenciar interfaces de rede elástica e endereços IP para nós.

Detalhes das permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`** – gerencie interfaces de rede elástica e endereços IP para ter compatibilidade com grupos de segurança de pods e nós do Windows.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSVPCResourceController no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AmazonEKSWorkerNodePolicy

Você pode anexar a `AmazonEKSWorkerNodePolicy` às suas entidades do IAM. É preciso anexar essa política a um perfil do IAM do nó que você especifica ao criar nós do Amazon EC2 que permitem ao Amazon EKS realizar ações em seu nome. Se você criar um grupo de nós usando`eksctl`, ele cria a função do nó do IAM e anexa essa política à função automaticamente.

Esta política concede aos nós Amazon EKS Amazon EC2 permissões para se conectar a clusters do Amazon EKS.

Detalhes das permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`** – leia o volume da instância e as informações de rede. Isso é necessário para que os nós do Kubernetes possam descrever informações sobre os recursos do Amazon EC2 necessários para o nó ingressar no cluster do Amazon EKS.
- **`eks`** – opcionalmente, descreva o cluster como parte do bootstrapping do nó.
- **`eks-auth:AssumeRoleForPodIdentity`** – permita a recuperação de credenciais para workloads do EKS no nó. Essa API é necessária para a Identidade de Pods do EKS funcionar corretamente.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSWorkerNodePolicy no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AmazonEKSWorkerNodeMinimalPolicy

Você pode anexar o AmazonEKSWorkerNodeMinimalPolicy às suas entidades IAM. É possível anexar essa política a um perfil do IAM do nó que você especifica ao criar nós do Amazon EC2 que permitem ao Amazon EKS realizar ações em seu nome.

Esta política concede aos nós Amazon EKS Amazon EC2 permissões para se conectar a clusters do Amazon EKS. Essa política tem menos permissões em comparação com a AmazonEKSWorkerNodePolicy.

Detalhes das permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- `eks-auth:AssumeRoleForPodIdentity`: permita a recuperação de credenciais para workloads do EKS no nó. Essa API é necessária para a Identidade de Pods do EKS funcionar corretamente.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSWorkerNodePolicy no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AWSServiceRoleForAmazonEKSNodegroup

Não é possível anexar `AWSServiceRoleForAmazonEKSNodegroup` às entidades do IAM. Esta política é anexada a uma função vinculada ao serviço que permite ao Amazon EKS realizar ações em seu nome. Para obter mais informações, consulte the section called “Permissões de função vinculada ao serviço para o Amazon EKS”.

Esta política concede o`AWSServiceRoleForAmazonEKSNodegroup`permissões de função que permitem que ele crie e gerencie grupos de nós do Amazon EC2 em sua conta.

Detalhes das permissões

Esta política inclui as permissões abaixo, que permitem ao Amazon EKS concluir as tarefas a seguir.

- **`ec2`** – trabalhe com grupos de segurança, tags, reservas de capacidade e modelos de lançamento. Isso é necessário para grupos de nós gerenciados pelo Amazon EKS a fim de habilitar a configuração de acesso remoto e descrever reservas de capacidade que podem ser usadas em grupos de nós gerenciados. Além disso, os grupos de nós gerenciados do Amazon EKS criam um modelo de execução em seu nome. Isso serve para configurar o grupo do Amazon EC2 Auto Scaling que oferece suporte a cada grupo de nós gerenciados.
- **`iam`** – crie um perfil vinculado a serviço e transmita um perfil. Isso é exigido pelos grupos de nós gerenciados do Amazon EKS para gerenciar perfis de instância para a função que está sendo passada ao criar um grupo de nós gerenciados. Esse perfil de instância é usado pelas instâncias do Amazon EC2 executadas como parte de um grupo de nós gerenciados. O Amazon EKS precisa

criar funções vinculadas ao serviço para outros serviços, como os grupos do Amazon EC2 Auto Scaling. Essas permissões são usadas na criação de um grupo de nós gerenciados

- **`autoscaling`** – trabalhe com grupos de segurança Auto Scaling. Isso é exigido pelos grupos de nós gerenciados do Amazon EKS para gerenciar o grupo do Amazon EC2 Auto Scaling que apoia cada grupo de nós gerenciados. Ele também é usado para oferecer suporte a funcionalidades como remover pods quando os nós são encerrados ou reciclados durante as atualizações de grupo de nós.

Para visualizar a versão mais recente do documento de política JSON, consulte AWSServiceRoleForAmazonEKSNodegroup no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AmazonEKSDashboardServiceRolePolicy

Não é possível anexar `AmazonEKSDashboardServiceRolePolicy` às entidades do IAM. Esta política é anexada a uma função vinculada ao serviço que permite ao Amazon EKS realizar ações em seu nome. Para obter mais informações, consulte the section called “Permissões de função vinculada ao serviço para o Amazon EKS”.

Esta política concede o`AWSServiceRoleForAmazonEKSDashboard`permissões de função que permitem que ele crie e gerencie grupos de nós do Amazon EC2 em sua conta.

Detalhes relacionados às permissões

Esta política inclui as permissões abaixo que permitem ao serviço realizar as seguintes tarefas:

- **`organizations`**: visualize informações sobre a estrutura e as contas do AWS Organizations. Isso inclui permissões para listar contas em sua organização, visualizar unidades organizacionais e raízes, listar administradores delegados, visualizar serviços que têm acesso à sua organização e recuperar informações detalhadas sobre sua organização e contas.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSDashboardServiceRolePolicy no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AmazonEBSCSIDriverPolicy

A política `AmazonEBSCSIDriverPolicy` permite que o driver Container Storage Interface (CSI) do Amazon EBS crie, modifique, copie, anexe, desanexe e exclua volumes em seu nome. Isso inclui a

modificação de etiquetas em volumes existentes e a habilitação da restauração rápida de snapshot (FSR) em volumes do EBS. Além disso, concede permissões ao driver da CSI do EBS para criar, restaurar e excluir snapshots, bem como listar as instâncias, os volumes e os snapshots.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEBSCSIDriverServiceRolePolicy no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AmazonEFSCSIDriverPolicy

A política `AmazonEFSCSIDriverPolicy` permite que a Container Storage Interface (CSI) do Amazon EFS crie e exclua pontos de acesso em seu nome. Ela também concede ao driver CSI do Amazon EFS permissões para listar os sistemas de arquivos, destinos de montagem e zonas de disponibilidade do Amazon EC2 dos seus pontos de acesso.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEFSCSIDriverServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Política gerenciada pela AWS: AmazonEKSLocalOutpostClusterPolicy

Essa política pode ser anexada a entidades do IAM. Antes de criar um cluster local, você precisa anexar essa política ao perfil do cluster. Os clusters do Kubernetes gerenciados pelo Amazon EKS fazem chamadas para outros serviços da AWS em seu nome. Eles fazem isso para gerenciar os recursos que você usa junto com o serviço.

A política `AmazonEKSLocalOutpostClusterPolicy` inclui as seguintes permissões:

- Ações de leitura do **`ec2`**: permite que as instâncias do ambiente de gerenciamento descrevam as propriedades da zona de disponibilidade, da tabela de rotas, da instância e da interface de rede. Permissões necessárias para que as instâncias do Amazon EC2 ingressem com êxito no cluster como instâncias do ambiente de gerenciamento.
- **`ssm`** – permite a conexão do Amazon EC2 Systems Manager com a instância do ambiente de gerenciamento, que é usada pelo Amazon EKS para se comunicar e gerenciar o cluster local na sua conta.
- **`logs`** – permite que as instâncias enviem logs para o Amazon CloudWatch.
- **`secretsmanager`**: permite que as instâncias obtenham e excluam dados de bootstrap para as instâncias do ambiente de gerenciamento com segurança no AWS Secrets Manager .
- **`ecr`** – permite que pods e contêineres em execução nas instâncias do ambiente de gerenciamento extraiam imagens de contêineres armazenadas no Amazon Elastic Container Registry.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSLocalOutpostClusterPolicy no Guia de referência de políticas gerenciadas pela AWS.

Política gerenciada pela AWS: AmazonEKSLocalOutpostServiceRolePolicy

Não é possível anexar essa política a suas entidades do IAM. Quando você cria um cluster usando uma entidade principal do IAM que tem a permissão `iam:CreateServiceLinkedRole`, o Amazon EKS cria automaticamente o perfil vinculado a serviço AWSServiceRoleforAmazonEKS para você e essa política é anexada a ele. Essa política permite que o perfil vinculado ao serviço chame produtos da AWS em seu nome para clusters locais.

A política `AmazonEKSLocalOutpostServiceRolePolicy` inclui as seguintes permissões:

- **`ec2`** – permite que o Amazon EKS trabalhe com segurança, com a rede e com outros recursos para iniciar e gerenciar com êxito instâncias do ambiente de gerenciamento na sua conta.
- **`ssm`**, **`ssmmessages`** – permite a conexão do Amazon EC2 Systems Manager com as instâncias do ambiente de gerenciamento, que é usada pelo Amazon EKS para se comunicar e gerenciar o cluster local na sua conta.
- **`iam`** – permite que o Amazon EKS gerencie o perfil de instância associado às instâncias do ambiente de gerenciamento.
- **`secretsmanager`** : permite que o Amazon EKS coloque dados de bootstrap para as instâncias do ambiente de gerenciamento no AWS Secrets Manager para que ele possa ser referenciado com segurança durante o bootstrapping da instância.
- **`outposts`** – permite que o Amazon EKS obtenha informações do Outpost de sua conta para lançar com êxito um cluster local em um Outpost.

Para visualizar a versão mais recente do documento de política JSON, consulte AmazonEKSLocalOutpostServiceRolePolicy no Guia de referência de políticas gerenciadas da AWS.

Atualizações do Amazon EKS para políticas gerenciadas pela AWS

Visualize detalhes sobre atualizações em políticas gerenciadas pela AWS para o Amazon EKS, desde que este serviço começou a monitorar essas alterações.

Para receber notificações de todas as alterações do arquivo de origem nesta página específica da documentação, você pode se inscrever no seguinte URL com um leitor de RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/security/iam-reference/security-iam-awsmanpol.adoc.atom
```

Alteração	Descrição	Data
Introduzido em the section called “Política gerenciada pela AWS: AmazonEKSMCPReadOnlyAccess”.	O Amazon EKS introduziu a nova política gerenciada `AmazonEKSMCPReadOnlyAccess` para habilitar ferramentas somente de leitura no servidor MCP do Amazon EKS para observabilidade e solução de problemas.	21 de novembro de 2025
Permissões adicionadas à AmazonEBSCSIDriverPolicy.	Foi adicionada a permissão `ec2:CopyVolumes`, possibilitando que o driver CSI do EBS realize a cópia direta de volumes do EBS.	17 de novembro de 2025
Adicionada uma permissão a the section called “Política gerenciada pela AWS: AmazonEKSServiceRolePolicy”.	Foram adicionadas as permissões `ec2:DescribeRouteTables` e `ec2:DescribeNetworkAcls` para a política `AmazonEKSServiceRolePolicy`. Isso permite que o Amazon EKS detecte problemas de configuração em tabelas de rotas da VPC e ACLs de rede para nós híbridos, como parte dos insights do cluster.	22 de outubro de 2025
Foi adicionada uma permissão à política AWSServic	Foi adicionada a permissão `ssmmessages:OpenDa`	15 de outubro de 2025

Alteração	Descrição	Data
eRoleForAmazonEKSConnector.	taChannel para a política AmazonEKSConnector ServiceRolePolicy .	
Adicionada uma permissão a the section called “Política gerenciada pela AWS: AmazonEKSServiceRolePolicy”	Esse perfil pode anexar uma nova política de acesso AmazonEKSEventPolicy . Permissões restritas para ec2:DeleteLaunchTemplate e ec2:TerminateInstances .	26 de agosto de 2025
Adicionada uma permissão a the section called “Política gerenciada pela AWS: AmazonEKSLocalOutpostServiceRolePolicy”	Adicionada a permissão ssmmessages:OpenDataChannel a AmazonEKSLocalOutpostServiceRolePolicy .	26 de junho de 2025

Alteração	Descrição	Data
Adicionada uma permissão a the section called “Política gerenciada pela AWS: AmazonEKSComputePolicy”.	Permissões de recursos atualizadas para as ações `ec2:RunInstances` e `ec2:CreateFleet` para incluir reservas de capacidade `arn:aws:ec2:*:*:capacity-reservation/*`. Isso permite que o Modo automático do Amazon EKS inicialize instâncias usando as reservas de capacidade sob demanda do EC2 em sua conta. Adicionado `iam:CreateServiceLinkedRole` para permitir que o Modo automático do Amazon EKS crie o perfil `AWSServiceRoleForEC2Spot` vinculado ao serviço de instâncias spot do EC2 em seu nome.	20 de junho de 2025
Adicionada uma permissão à AmazonEKSServiceRolePolicy.	Adicionada a permissão `ec2:DescribeCapacityReservations` para permitir que o Modo automático do Amazon EKS inicialize instâncias usando as reservas de capacidade sob demanda do EC2 em sua conta.	20 de junho de 2025
Introduzido em the section called “Política gerenciada pela AWS: AmazonEKSDashboardConsoleReadOnly”.	Introdução da nova política `AmazonEKSDashboardConsoleReadOnly`.	19 de junho de 2025

Alteração	Descrição	Data
Introduzido em the section called “Política gerenciada pela AWS: AmazonEKSDashboardServiceRolePolicy”.	Introduziu a nova política `AmazonEKSDashboardServiceRolePolicy` .	21 de maio de 2025
Permissões adicionadas à AmazonEKSClusterPolicy.	Adicionada a permissão `ec2:DeleteNetworkInterfaces` para possibilitar que o Amazon EKS exclua interfaces de rede elásticas que serão esquecidas se a CNI da VPC for encerrada inesperadamente.	16 de abril de 2025
Adicionada uma permissão à política AmazonEKSServiceRolePolicy.	Adicionadas as permissões `ec2:RevokeSecurityGroupEgress` e `ec2:AuthorizeSecurityGroupEgress` para possibilitar que os clientes de IA e ML do EKS adicionem regras de saída de grupos de segurança ao grupo de segurança padrão do cluster do EKS que sejam compatíveis com o EFA, como parte do lançamento da versão 1.33 do EKS.	14 de abril de 2025
Permissões adicionadas à AmazonEKSServiceRolePolicy.	Adicionada permissão para encerrar instâncias do EC2 criadas pelo Modo automático do EKS.	28 de fevereiro de 2025

Alteração	Descrição	Data
Permissões adicionadas à AmazonEBSCSIDriverPolicy.	Foi adicionada uma nova declaração autorizando o driver da CSI do EBS a restaurar todos os snapshots . Anteriormente, isso era permitido pela política existente, mas uma nova declaração explícita se faz necessária devido a uma alteração no gerenciamento do IAM para `CreateVolume` . Foi adicionada a capacidade para o driver da CSI do EBS modificar tags em volumes existentes. O driver CSI do EBS pode modificar as tags de volumes existentes por meio dos parâmetros em VolumeAttributesClasses do Kubernetes. Foi adicionada a capacidade para o driver da CSI do EBS habilitar a restauração rápida de snapshot (FSR) em volumes do EBS. O driver CSI do EBS pode habilitar a FSR em novos volumes por meio de parâmetros nas classes de armazenamento do Kubernetes.	13 de janeiro de 2025

Alteração	Descrição	Data
Permissões adicionadas à the section called “Política gerenciada pela AWS: AmazonEKSLoadBalancingPolicy”.	A política `AmazonEKSLoadBalancingPolicy` foi atualizada para possibilitar a listagem e descrição de recursos de rede e endereços IP.	26 de dezembro de 2024
Foram adicionadas permissões a the section called “Política gerenciada pela AWS: AWSServiceRoleForAmazonEKSNodegroup”.	Foi atualizado o `AWSServiceRoleForAmazonEKSNodegroup` para compatibilidade com as regiões da China.	22 de novembro de 2024
Permissões adicionadas à the section called “Política gerenciada pela AWS: AmazonEKSLocalOutpostClusterPolicy”	Foi adicionada a permissão `ec2:DescribeAvailabilityZones` a `AmazonEKSLocalOutpostClusterPolicy` para que o AWS Cloud Controller Manager no ambiente de gerenciamento do cluster possa identificar a zona de disponibilidade em que cada nó está.	21 de novembro de 2024

Alteração	Descrição	Data
Foram adicionadas permissões a the section called “Política gerenciada pela AWS: AWSServiceRoleForAmazonEKSNodegroup”.	Foi atualizada a política `AWSServiceRoleForAmazonEKSNodegroup` para permitir `ec2:RebootInstances` para instâncias criadas por grupos de nós gerenciados pelo Amazon EKS. Foram restringidas as permissões `ec2:CreateTags` para recursos do Amazon EC2.	20 de novembro de 2024
Foram adicionadas permissões a the section called “Política gerenciada pela AWS: AmazonEKSServiceRolePolicy”.	O EKS atualizou a política gerenciada `AmazonEKSServiceRolePolicy` da AWS. Foram adicionadas permissões para políticas de acesso ao EKS, gerenciamento de balanceador de carga e limpeza automatizada de recursos de cluster.	16 de novembro de 2024
Introduzido em the section called “Política gerenciada pela AWS: AmazonEKSComputePolicy”.	O EKS atualizou a política gerenciada `AmazonEKSComputePolicy` da AWS. Foram atualizadas as permissões de recurso para a ação `iam:AddRoleToInstanceProfile` .	7 de novembro de 2024
Introduzido em the section called “Política gerenciada pela AWS: AmazonEKSComputePolicy”.	A AWS introduziu a `AmazonEKSComputePolicy` .	1.º de novembro de 2024

Alteração	Descrição	Data
Permissões adicionadas à `AmazonEKSClusterPolicy`	Foi adicionada a permissão `ec2:DescribeInstanceTopology` para permitir que o Amazon EKS anexe informações de topologia ao nó como rótulos.	1.º de novembro de 2024
Introduzido em the section called “Política gerenciada pela AWS: AmazonEKSBlockStoragePolicy”.	AWS A introduziu a `AmazonEKSBlockStoragePolicy` .	30 de outubro de 2024
Introduzido em the section called “Política gerenciada pela AWS: AmazonEKSLoadBalancingPolicy”.	AWS A introduziu a `AmazonEKSLoadBalancingPolicy` .	30 de outubro de 2024
Foram adicionadas permissões ao AmazonEKSServiceRolePolicy.	Foram adicionadas permissões `cloudwatch:PutMetricData` para permitir que o Amazon EKS publique métricas no Amazon CloudWatch.	29 de outubro de 2024
Introduzido em the section called “Política gerenciada da AWS: AmazonEKSNetworkingPolicy”.	AWS A introduziu a `AmazonEKSNetworkingPolicy` .	28 de outubro de 2024
Foram adicionadas permissões para `AmazonEKSServicePolicy` e `AmazonEKSServiceRolePolicy`	Adição de `ec2:GetSecurityGroupsForVpc` e permissões de etiqueta associadas para permitir que o EKS leia as informações do grupo de segurança e atualize as etiquetas relacionadas.	10 de outubro de 2024

Alteração	Descrição	Data
Introdução da AmazonEKS WorkerNodeMinimalPolicy.	AWS A introduziu a `AmazonEKSWorkerNodeMinimalPolicy` .	3 de outubro de 2024
Permissões adicionadas para AWSServiceRoleForAmazonEKSNodegroup.	Foram adicionadas as permissões `autoscaling:ResumeProcesses` e `autoscaling:SuspendProcesses` para permitir que o Amazon EKS suspenda e retome o `AZRebalance` em grupos do Auto Scaling gerenciados pelo Amazon EKS.	21 de agosto de 2024
Permissões adicionadas para AWSServiceRoleForAmazonEKSNodegroup.	A permissão `ec2:DescribeCapacityReservations` foi adicionada para que o Amazon EKS possa descrever a reserva de capacidade na conta do usuário. A permissão `autoscaling:PutScheduledUpdateGroupAction` foi adicionada para permitir a configuração da escalabilidade programada em grupos de nós `CAPACITY_BLOCK` .	27 de junho de 2024

Alteração	Descrição	Data
AmazonEKS_CNI_Policy: atualização de uma política existente	O Amazon EKS adicionou novas permissões `ec2:DescribeSubnets` para permitir que o plug-in CNI da Amazon VPC para Kubernetes identifique a quantidade de endereços IP disponíveis nas sub-redes da Amazon VPC. O plug-in CNI da VPC pode usar os endereços IP disponíveis em cada sub-rede para escolher as sub-redes com a maior quantidade de endereços IP disponíveis para usar ao criar uma interface de rede elástica.	4 de março de 2024
AmazonEKSWorkerNodePolicy: atualização de uma política existente	O Amazon EKS adicionou novas permissões para permitir a Identidade de Pods do EKS. O atentende de Identidade de Pods do Amazon EKS usa o perfil de nó.	26 de novembro de 2023
Introdução da AmazonEFSCSIDriverPolicy.	AWS A introduziu a `AmazonEFSCSIDriverPolicy` .	26 de julho de 2023

Alteração	Descrição	Data
Permissões adicionadas à AmazonEKSClusterPolicy.	A permissão `ec2:DescribeAvailabilityZones` foi adicionada para permitir que o Amazon EKS obtenha os detalhes da AZ durante a descoberta automática de sub-redes ao criar balanceadores de carga.	7 de fevereiro de 2023
Condições da política AmazonEBSCSIDriverPolicy foram atualizadas.	Foram removidas as condições de política inválidas com caracteres curingas no campo de chave `StringLike`. Também foi adicionada uma nova condição `ec2:ResourceTag/kubernetes.io/created-for/pvc/name: "*"` a `ec2:DeleteVolume`, que permite que o driver da CSI do EBS exclua volumes criados pelo plug-in em árvore.	17 de novembro de 2022

Alteração	Descrição	Data
Adicionadas permissões à AmazonEKSLocalOutpostServiceRolePolicy.	Adicionados `ec2:DescribeVPCAttribute` , `ec2:GetConsoleOutput` e `ec2:DescribeSecret` para permitir melhor validação de pré-requisitos e controle gerenciado do ciclo de vida. Também foram adicionados `ec2:DescribePlacementGroups` e `"arn:aws:ec2:*:*:placement-group/*"` a `ec2:RunInstances` para permitir controle de colocação das instâncias do Amazon EC2 do ambiente de gerenciamento no Outposts.	24 de outubro de 2022
Atualize as permissões do Amazon Elastic Container Registry em AmazoneKSLocalOutpostClusterPolicy.	Movida a ação `ecr:GetDownloadUrlForLayer` de todas as seções de recurso para uma seção com escopo definido. Adicionado o recurso `arn:aws:ecr:*:*:repository/eks/` . Removido o recurso `arn:aws:ecr:` . Esse recurso é coberto pelo recurso `arn:aws:ecr:*:*:repository/eks/*` adicionado.	20 de outubro de 2022

Alteração	Descrição	Data
Permissões adicionadas à AmazonEKSLocalOutpostClusterPolicy.	Adicionado o repositório do Amazon Elastic Container Registry `arn:aws:ecr:*:*:repository/kubelet-config-updater` para que as instâncias do ambiente de gerenciamento do cluster possam atualizar alguns argumentos `kubelet`.	31 de agosto de 2022
Apresentada a AmazonEKSLocalOutpostClusterPolicy.	AWS A introduziu a `AmazonEKSLocalOutpostClusterPolicy` .	24 de agosto de 2022
Apresentada a AmazonEKSLocalOutpostServiceRolePolicy.	AWS A introduziu a `AmazonEKSLocalOutpostServiceRolePolicy` .	23 de agosto de 2022
Introdução da AmazonEBSCSIDriverPolicy.	A AWS apresentou o `AmazonEBSCSIDriverPolicy` .	4 de abril de 2022
Adição de permissões a AmazonEKSWorkerNodePolicy.	Adição de `ec2:DescribeInstanceTypes` para habilitar AMIs otimizadas para do Amazon EKS que podem detectar propriedades em nível de instâncias automaticamente.	21 de março de 2022

Alteração	Descrição	Data
Permissões adicionadas para AWSServiceRoleForAmazonEKSNodegroup.	A permissão `autoscaling:EnableMetricsCollection` foi adicionada para que o Amazon EKS possa habilitar a coleta de métricas.	13 de dezembro de 2021
Permissões adicionadas para AmazonEKSClusterPolicy.	As permissões `ec2:DescribeAccountAttributes`, `ec2:DescribeAddresses` e `ec2:DescribeInternetGateways` foram adicionadas para permitir que o Amazon EKS crie um perfil vinculado ao serviço para um Network Load Balancer.	17 de junho de 2021
O Amazon EKS passou a monitorar as alterações.	O Amazon EKS passou a controlar as alterações para as políticas gerenciadas da AWS.	17 de junho de 2021

Solução de problemas do IAM

Este tópico aborda alguns erros comuns que você pode encontrar ao usar o Amazon EKS com o IAM e como resolvê-los.

AccessDeniedException

Se você receber um `AccessDeniedException` ao chamar uma operação de API da AWS, é porque as credenciais da entidade principal do IAM que estão sendo usadas não têm as permissões necessárias para faze essa chamada.

```
An error occurred (AccessDeniedException) when calling the DescribeCluster operation:
```

```
User: arn:aws:iam::111122223333:user/user_name is not authorized to perform:
eks:DescribeCluster on resource: arn:aws:eks:region:111122223333:cluster/my-cluster
```

No exemplo de mensagem anterior, o usuário não tem permissões para chamar a operação da API `DescribeCluster` do Amazon EKS. Para conceder permissões de administrador do Amazon EKS a uma entidade principal do IAM, consulte the section called “Políticas baseadas em identidade”.

Para obter mais informações sobre o IAM, consulte Controlling access using policies (Controlar o acesso usando políticas) no Manual do usuário do IAM.

Não é possível ver Nodes (Nós) na guia Compute (Computação) ou qualquer coisa na guia Resources (Recursos), e você recebe um erro no Console de gerenciamento da AWS

Você pode ver uma mensagem de erro do console informando `Your current user or role does not have access to Kubernetes objects on this EKS cluster`. Verifique se o usuário da entidade principal do IAM com o qual você está usando o Console de gerenciamento da AWS tem as permissões necessárias. Para obter mais informações, consulte the section called “Permissões obrigatórias”.

O aws-auth **`ConfigMap`** não concede acesso ao cluster

O AWS IAM Authenticator não permite um caminho no ARN da função usado no `ConfigMap`. Portanto, antes de especificar `rolearn`, remova o caminho. Por exemplo, altere `arn:aws:iam::111122223333:role/team/developers/eks-admin` para `arn:aws:iam::111122223333:role/eks-admin` .

Não estou autorizado a executar iam:PassRole

Se receber uma mensagem de erro informando que você não tem autorização para executar a ação `iam:PassRole`, suas políticas devem ser atualizadas para permitir a transmissão de um perfil ao Amazon EKS.

Alguns serviços da AWS permitem que você passe uma função existente para o serviço, em vez de criar um novo perfil de serviço ou perfil vinculado ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O erro exemplificado a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no Amazon EKS. No entanto, a ação exige que o serviço tenha

permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: {arn-aws}iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se você precisar de ajuda, entre em contato com seu administrador AWS. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha conta da AWS acessem meus recursos do Amazon EKS

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Amazon EKS oferece suporte a esses recursos, consulte the section called “Amazon EKS e IAM”.
- Para saber como conceder acesso a seus recursos em todas as contas da AWS pertencentes a você, consulte Fornecimento de acesso a um usuário do IAM em outra conta da AWS pertencente a você no Guia de Usuário do IAM.
- Para saber como conceder acesso aos recursos para contas da AWS de terceiros, consulte Fornecer acesso a contas da AWS pertencentes a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Os contêineres do pod recebem o seguinte erro: **`An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`**

Seus contêineres receberão esse erro se a aplicação estiver explicitamente fazendo solicitações ao endpoint global do AWS STS (`https://sts.amazonaws`) e sua conta de serviço do Kubernetes estiver configurada para usar um endpoint regional. É possível resolver o problema com uma das seguintes opções:

- Atualize o código da aplicação para remover chamadas explícitas para o endpoint global AWS STS.
- Atualize o código da aplicação para fazer chamadas explícitas para endpoints regionais, como `https://sts.us-west-2.amazonaws.com` . Seu aplicação deve ter redundância incorporada para escolher uma região AWS diferente em caso de falha do serviço na região AWS. Para obter mais informações, consulte Gerenciar o AWS STS em uma região da AWS, no Guia do usuário do IAM.
- Configure as suas contas de serviço para utilizar o endpoint global. Os clusters usam o endpoint regional por padrão. Para obter mais informações, consulte the section called “Endpoints do STS”.

Função do IAM do cluster do Amazon EKS

É necessário um perfil do IAM de cluster do Amazon EKS para cada cluster. Os clusters do Kubernetes gerenciados pelo Amazon EKS usam esse perfil para gerenciar nós, e o provedor de nuvem legado usa esse perfil para criar balanceadores de carga com o Elastic Load Balancing para serviços.

Antes de criar clusters do Amazon EKS, você deve criar uma função do IAM com uma das seguintes políticas do IAM:

- AmazonEKSClusterPolicy
- Uma política do IAM personalizada. As permissões mínimas a seguir permitem que o cluster do Kubernetes gerencie nós, mas não permitem que o provedor de nuvem legado crie balanceadores de carga com o Elastic Load Balancing. Sua política do IAM personalizada deve ter pelo menos as seguintes permissões:

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": "kubernetes.io/cluster/*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInstanceTopology",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

Note

Antes de 3 de outubro de 2023, AmazoneksClusterPolicy era exigido no perfil do IAM para cada cluster.
Antes de 16 de abril de 2020, AmazonEKSServicePolicy e AmazonEKSClusterPolicy eram obrigatórios, e o nome sugerido para o perfil era `eksServiceRole`. Com o perfil vinculado ao serviço `AWSServiceRoleForAmazonEKS`, a política AmazonEKSServicePolicy não será mais necessária para clusters criados em ou após 16 de abril de 2020.

Verificar se há uma função de cluster existente

É possível usar o procedimento a seguir para verificar se a conta já tem a função do cluster do Amazon EKS.

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Pesquise na lista de perfis `eksClusterRole`. Se não houver uma função que inclua `eksClusterRole`, consulte the section called “Criar uma função para o cluster do Amazon EKS” para criar a função. Se houver uma função que inclua `eksClusterRole`, selecione-a para visualizar as políticas anexadas.
4. Escolha Permissões.
5. Verifique se a política gerenciada AmazonEKSClusterPolicy está anexada à função. Se a política estiver anexada, a função do cluster do Amazon EKS estará configurada corretamente.
6. Escolha Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).
7. Verifique se o relacionamento de confiança contém a seguinte política: Se o relacionamento de confiança corresponder à seguinte política, escolha Cancel (Cancelar). Se o relacionamento de confiança não corresponder, copie a política para a janela Edit trust policy (Editar política de confiança) e escolha Update policy (Atualizar política).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Criar uma função para o cluster do Amazon EKS

Você pode usar o Console de gerenciamento da AWS ou a CLI AWS para criar o perfil de cluster.

Console de gerenciamento da AWS

a. Abra o console do IAM em https://console.aws.amazon.com/iam/.
b. Escolha Roles (Funções) e, em seguida, Create Role (Criar função).
c. Em Tipo de entidade confiável, selecione Serviço da AWS.
d. Na lista suspensa Casos de uso para outros serviços AWS, escolha EKS.
e. Escolha EKS - Cluster para o caso de uso e escolha Next (Próximo).
f. Na guia Add permissions (Adicionar permissões), escolha Next (Próximo).
g. Em Nome do perfil, insira um nome exclusivo para o perfil, como `eksClusterRole`.
h. Em Description (Descrição), insira um texto descritivo, como `Amazon EKS - Cluster role`.
i. Selecione Criar perfil.

AWS CLI

a. Copie o conteúdo a seguir em um arquivo chamado *`cluster-trust-policy.json`*.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Crie a função. Você pode substituir *`eksClusterRole`* por qualquer nome que desejar.

```
aws iam create-role \
  --role-name eksClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

c. Anexe a política do IAM necessária à função.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name eksClusterRole
```

Perfil do IAM em nós do Amazon EKS

O daemon `kubelet` do nó do Amazon EKS chama as APIs da AWS em seu nome. Os nós recebem permissões para essas chamadas de API por meio de um perfil de instância do IAM e políticas associadas. Antes de iniciar os nós e registrá-los em um cluster, você deve criar um perfil do IAM para uso desses nós quando eles forem iniciados. Esse requisito se aplica a nós executados com a AMI otimizada para Amazon EKS fornecida pela Amazon ou com qualquer outra AMI do nó que você pretende usar. Além disso, este requisito se aplica tanto aos grupos de nós gerenciados quanto aos nós autogerenciados.

> **Note**
>
> Não é possível usar a mesma função usada para criar clusters.

Antes de criar os nós, é necessário criar um perfil do IAM com as seguintes permissões:

- Permissões para que o `kubelet` descreva os recursos do Amazon EC2 na VPC, como as fornecidas pela política AmazonEKSWorkerNodePolicy. Esta política também fornece as permissões para o Amazon EKS Pod Identity Agent.
- Permissões para que o `kubelet` use imagens de contêiner do Amazon Elastic Container Registry (Amazon ECR), como as fornecidas pela política AmazonEC2ContainerRegistryPullOnly. As permissões para usar imagens de contêiner do Amazon Elastic Container Registry (Amazon ECR) são necessárias porque os complementos integrados para a rede executam pods que usam imagens de contêiner do Amazon ECR.
- (Opcional) Permissões para que o Amazon EKS Pod Identity Agent use a ação `eks-auth:AssumeRoleForPodIdentity` para recuperar credenciais para pods. Se você não usa a AmazonEKSWorkerNodePolicy, então forneça esta permissão adicionalmente às permissões do EC2 para usar a Identidade de Pods do EKS.
- (Opcional) Caso não use a Identidade de Pods do EKA ou IRSA para conceder permissões aos pods da CNI da VPC, forneça permissões para a CNI da VPC no perfil da instância. Você pode usar a política gerenciada `AmazonEKS_CNI_Policy` (caso tenha criado o cluster com a família `IPv4`) ou uma política destinada a IPv6 criada por você (caso tenha criado o cluster com a família `IPv6`). Contudo, em vez de anexar a política à essa função, recomendamos que você anexe a política a uma função separada usada especificamente para o complemento CNI da Amazon VPC. Para obter mais informações sobre a criação de uma função separada para o complemento CNI da Amazon VPC, consulte the section called “Configuração para IRSA”.

> **Note**
>
> Os grupos de nós do Amazon EC2 devem ter uma função do IAM diferente do perfil do Fargate. Para obter mais informações, consulte the section called “Função do IAM de execução de pod”.

Verificar se há uma função existente do nó

Use o procedimento a seguir para verificar se a conta já tem a função de nó do Amazon EKS.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Procure `eksNodeRole`, `AmazonEKSNodeRole` ou `NodeInstanceRole` na lista de funções. Se não houver uma função com um desses nomes, consulte the section called “Criação de uma função para a função do IAM de nó do Amazon EKS” para criar a função. Se houver uma função que contenha `eksNodeRole`, `AmazonEKSNodeRole` ou `NodeInstanceRole`, selecione-a para visualizar as políticas anexadas.
4. Escolha Permissões.
5. Certifique-se de que as políticas gerenciadas AmazonEKSWorkerNodePolicy e AmazonEC2ContainerRegistryPullOnly estejam anexadas à perfil ou que uma política personalizada esteja anexada com as permissões mínimas.

   > **Note**
   >
   > Se oAmazonEKS_CNI_Policyestiver anexada à função, recomendamos removê-la e anexá-la a uma função do IAM mapeada para `oaws-node`Em vez disso, a conta do serviço Kubernetes. Para obter mais informações, consulte the section called “Configuração para IRSA”.

6. Escolha Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).
7. Verifique se o relacionamento de confiança contém a seguinte política: Se o relacionamento de confiança corresponder à seguinte política, escolha Cancel (Cancelar). Se o relacionamento de confiança não corresponder, copie a política para a janela Edit trust policy (Editar política de confiança) e escolha Update policy (Atualizar política).

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

Criação de uma função para a função do IAM de nó do Amazon EKS

Você pode criar o perfil IAM do nó com o Console de gerenciamento da AWS ou com a CLI AWS.

Console de gerenciamento da AWS

a. Abra o console do IAM em https://console.aws.amazon.com/iam/.
b. No painel de navegação à esquerda, escolha Funções.
c. Na página Perfis, selecione Criar perfil.
d. Na página Selecionar entidade confiável, faça o seguinte:
    i. Na seção Tipo de entidade confiável), escolha Service da AWS.
    ii. Em Use case (Caso de uso), selecione EC2.
    iii. Escolha Próximo.
e. Na página Add permissions (Adicionar permissões), faça o seguinte:
    i. Na caixa Filtrar políticas insira `AmazonEKSWorkerNodePolicy`.
    ii. Marque a caixa de seleção à esquerda de AmazonEKSWorkerNodePolicy nos resultados da pesquisa.
    iii. Escolha Clear filters (Limpar filtros).
    iv. Na caixa Filtrar políticas insira `AmazonEC2ContainerRegistryPullOnly`.

v. Marque a caixa de seleção à esquerda de AmazonEC2ContainerRegistryPullOnly nos resultados da pesquisa.

A política gerenciada AmazonEKS_CNI_Policy ou uma política de IPv6 que você criar também deverá ser anexada a esse perfil ou a um perfil diferente que seja mapeado para a conta de serviço `aws-node` do Kubernetes. Recomendamos atribuir a política à função associada à conta de serviço do Kubernetes em vez de atribuí-la a essa função. Para obter mais informações, consulte the section called “Configuração para IRSA”.

vi. Escolha Próximo.

f. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:

i. Em Nome do perfil, insira um nome exclusivo para o perfil, como `AmazonEKSNodeRole`.

ii. Em Description (Descrição), substitua o texto atual por um texto descritivo como `Amazon EKS - Node role`.

iii. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.

iv. Selecione Criar perfil.

AWS CLI

a. Execute o seguinte comando para criar o arquivo `node-role-trust-relationship.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

b. Crie o perfil do IAM.

```
aws iam create-role \
  --role-name AmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-relationship.json"
```

c. Anexe duas políticas do IAM gerenciadas necessárias à função do IAM.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly \
  --role-name AmazonEKSNodeRole
```

d. Anexe uma das seguintes políticas do IAM à função do IAM, dependendo de com qual família de IP você criou o seu cluster. A política deve ser anexada a esse perfil ou a um perfil associado à conta de serviço `aws-node` do Kubernetes, que é usada para o plug-in CNI da Amazon VPC para Kubernetes. Recomendamos atribuir a política à função associada à conta de serviço do Kubernetes. Para atribuir a política à função associada à conta de serviço do Kubernetes, consulte the section called “Configuração para IRSA”.

- IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSNodeRole
```

- IPv6

A. Copie o texto a seguir e salve-o em um arquivo chamado `vpc-cni-ipv6-policy.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

B. Crie a política do IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

C. Anexe a política do IAM à função do IAM. Substitua *111122223333* pelo ID da sua conta.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

Perfil do IAM do cluster do Modo Automático do Amazon EKS

É necessário um perfil do IAM de cluster do Amazon EKS para cada cluster. Os clusters do Kubernetes gerenciados pelo Amazon EKS usam esse perfil para automatizar tarefas de rotina de armazenamento, rede e ajuste de escala automático de computação.

Antes de criar clusters do Amazon EKS, é necessário criar um perfil do IAM com as políticas necessárias a seguir do Modo Automático do EKS. Você pode anexar as políticas gerenciadas pelo AWS IAM sugeridas ou criar políticas personalizadas com permissões equivalentes.

- AmazonEKSComputePolicy
- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- AmazonEKSNetworkingPolicy

- AmazonEKSClusterPolicy

Verificar se há uma função de cluster existente

É possível usar o procedimento a seguir para verificar se a conta já tem a função do cluster do Amazon EKS.

1. Abra o console IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Pesquise na lista de perfis `AmazonEKSAutoClusterRole`. Se um perfil que inclua `AmazonEKSAutoClusterRole` não existir, consulte as instruções na próxima seção para criar o perfil. Se houver uma função que inclua `AmazonEKSAutoClusterRole`, selecione-a para visualizar as políticas anexadas.
4. Escolha Permissões.
5. Verifique se a política gerenciada AmazonEKSClusterPolicy está anexada à função. Se a política estiver anexada, a função do cluster do Amazon EKS estará configurada corretamente.
6. Escolha Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).
7. Verifique se o relacionamento de confiança contém a seguinte política: Se o relacionamento de confiança corresponder à seguinte política, escolha Cancel (Cancelar). Se o relacionamento de confiança não corresponder, copie a política para a janela Edit trust policy (Editar política de confiança) e escolha Update policy (Atualizar política).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

> **Note**
>
> A AWS não exige o nome `AmazonEKSAutoClusterRole` para esse perfil.

Criar uma função para o cluster do Amazon EKS

Você pode usar o Console de gerenciamento da AWS ou a CLI AWS para criar o perfil de cluster.

Console de gerenciamento da AWS

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. Escolha Roles (Funções) e, em seguida, Create Role (Criar função).
3. Em Tipo de entidade confiável, selecione Serviço da AWS.
4. Na lista suspensa Casos de uso para outros serviços AWS, escolha EKS.
5. Escolha EKS - Cluster para o caso de uso e escolha Next (Próximo).
6. Na guia Adicionar permissões, selecione as políticas e escolha Próximo.
   - AmazonEKSComputePolicy
   - AmazonEKSBlockStoragePolicy
   - AmazonEKSLoadBalancingPolicy
   - AmazonEKSNetworkingPolicy
   - AmazonEKSClusterPolicy
7. Em Nome do perfil, insira um nome exclusivo para o perfil, como `AmazonEKSAutoClusterRole`.
8. Em Description (Descrição), insira um texto descritivo, como `Amazon EKS - Cluster role`.
9. Selecione Criar perfil.

AWS CLI

1. Copie o conteúdo a seguir em um arquivo chamado *`cluster-trust-policy.json`*.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

2. Crie a função. Você pode substituir *AmazonEKSAutoClusterRole* por qualquer nome que desejar.

```
aws iam create-role \
  --role-name AmazonEKSAutoClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. Anexe as políticas do IAM necessárias ao perfil:

AmazonEKSClusterPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
```

AmazonEKSComputePolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSComputePolicy
```

AmazonEKSBlockStoragePolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSBlockStoragePolicy
```

AmazonEKSLoadBalancingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
```

```
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

AmazonEKSNetworkingPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSNetworkingPolicy
```

Perfil do IAM do nó do Modo Automático do Amazon EKS

> **Note**
>
> Não é possível usar a mesma função usada para criar clusters.

Antes de criar os nós, é necessário criar um perfil do IAM com as seguintes políticas, ou permissões equivalentes:

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

Verificar se há uma função existente do nó

Use o procedimento a seguir para verificar se a conta já tem a função de nó do Amazon EKS.

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Pesquise na lista de perfis `AmazonEKSAutoNodeRole`. Se não houver um perfil com um desses nomes, consulte as instruções na próxima seção para criar o perfil. Se houver uma função que contenha `AmazonEKSAutoNodeRole`, selecione a função para visualizar as políticas anexadas.
4. Escolha Permissões.
5. Certifique-se de que as políticas necessárias acima estejam anexadas, ou políticas personalizadas equivalentes.
6. Escolha Trust relationships (Relacionamentos de confiança) e, em seguida, escolha Edit trust policy (Editar política de confiança).
7. Verifique se o relacionamento de confiança contém a seguinte política: Se o relacionamento de confiança corresponder à seguinte política, escolha Cancel (Cancelar). Se o relacionamento

de confiança não corresponder, copie a política para a janela Edit trust policy (Editar política de confiança) e escolha Update policy (Atualizar política).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Criação de uma função para a função do IAM de nó do Amazon EKS

Você pode criar o perfil IAM do nó com o Console de gerenciamento da AWS ou com a CLI AWS.

Console de gerenciamento da AWS

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Na página Perfis, selecione Criar perfil.
4. Na página Selecionar entidade confiável, faça o seguinte:
   a. Na seção Tipo de entidade confiável), escolha Service da AWS.
   b. Em Use case (Caso de uso), selecione EC2.
   c. Escolha Próximo.
5. Na página Adicionar permissões, anexe as seguintes políticas:
   - AmazonEKSWorkerNodeMinimalPolicy
   - AmazonEC2ContainerRegistryPullOnly
6. Na página Name, review, and create (Nomear, revisar e criar), faça o seguinte:
   a. Em Nome do perfil, insira um nome exclusivo para o perfil, como `AmazonEKSAutoNodeRole`.
   b. Em Description (Descrição), substitua o texto atual por um texto descritivo como `Amazon EKS - Node role`.

c. Em Adicionar tags (Opcional), adicione metadados ao perfil anexando tags como pares chave-valor. Para obter mais informações sobre o uso de tags no IAM, consulte Marcar recursos do IAM no Guia do usuário do IAM.

d. Selecione Criar perfil.

AWS CLI

Criar o perfil do IAM do nó

Use o arquivo node-trust-policy.json da etapa anterior para definir quais entidades podem assumir o perfil. Execute o seguinte comando para criar o perfil do IAM do nó:

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

Registrar o ARN do perfil

Depois de criar o perfil, recupere e salve o ARN do perfil do IAM do nó. Você precisará desse ARN nas etapas subsequentes. Use o seguinte comando para obter o ARN:

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Anexar as políticas obrigatórias

Anexe as seguintes políticas gerenciadas pela AWS ao perfil do IAM do nó para fornecer as permissões necessárias:

Para anexar a política AmazonEKSWorkerNodeMinimalPolicy:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Para anexar a política AmazonEC2ContainerRegistryPullOnly:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
```

```
    --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Perfil do IAM para a funcionalidade do Amazon EKS

As funcionalidades do EKS exigem a configuração de um perfil do IAM para a funcionalidade (ou perfil da funcionalidade). As funcionalidades usam esse perfil para realizar ações em serviços da AWS e para acessar recursos do Kubernetes no cluster por meio de entradas de acesso criadas automaticamente.

Antes que você possa especificar um perfil da funcionalidade durante a criação desta, é necessário criar o perfil do IAM com a política de confiança e as permissões adequadas ao tipo de funcionalidade. Assim que este perfil do IAM for criado, ele poderá ser reaproveitado para qualquer quantidade de recursos da funcionalidade.

Requisitos do perfil da funcionalidade

O perfil da funcionalidade deve atender aos seguintes requisitos:

- O perfil deve estar na mesma conta da AWS que o cluster e o recurso da funcionalidade
- O perfil deve ter uma política de confiança que permita ao serviço de funcionalidades do EKS assumir o perfil
- O perfil deve ter permissões apropriadas para o tipo de funcionalidade e para os requisitos do caso de uso (consulte the section called “Permissões por tipo de funcionalidade”)

Política de confiança para perfis da funcionalidade

Todos os perfis da funcionalidade devem incluir a seguinte política de confiança:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
```

```
      ]
    }
  ]
}
```

Esta política de confiança autoriza o EKS a:

- Assumir o perfil para executar operações de API da AWS
- Marcar sessões para fins de auditoria e rastreamento

Permissões por tipo de funcionalidade

As permissões do IAM exigidas variam conforme a funcionalidade que você usa e o modelo de implantação.

> **Note**
>
> Para implantações em ambientes de produção usando seletores de perfil do IAM com o ACK, ou ao usar o kro ou o Argo CD sem integração com serviços da AWS, o perfil da funcionalidade talvez não requeira permissões do IAM além da política de confiança.

kro (Kube Resource Orchestrator)

Não são necessárias permissões do IAM. É possível criar um perfil da funcionalidade sem políticas vinculadas. O kro requer apenas permissões de RBAC do Kubernetes para a criação e o gerenciamento de recursos do Kubernetes.

AWS Controlllers for Kubernetes (ACK) da)

O ACK fornece suporte a dois modelos de permissão:

- Configuração simples para ambientes de desenvolvimento e teste: adicione permissões para os serviços da AWS diretamente ao perfil da funcionalidade. Esta opção é ideal para começar a usar, em implantações realizadas em uma única conta ou quando todos os usuários necessitam das mesmas permissões.
- Prática recomendada para ambientes de produção: use seletores de perfil do IAM para implementar o acesso de privilégio mínimo. Com essa abordagem, o perfil da funcionalidade precisa apenas da permissão `sts:AssumeRole` para assumir perfis específicos de serviço.

Você não adiciona permissões de serviços da AWS (como o S3 ou o RDS) diretamente ao perfil da funcionalidade. Essas permissões são concedidas a perfis do IAM individuais que são mapeados para namespaces específicos.

Os seletores de perfil do IAM permitem:

- Isolamento de permissões em nível de namespace
- Gerenciamento de recursos entre contas
- Perfis do IAM exclusivos para cada equipe
- Modelo de segurança com privilégio mínimo

Exemplo de política de perfil da funcionalidade para a abordagem de seletor de perfil do IAM:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": [
        "arn:aws:iam::111122223333:role/ACK-S3-Role",
        "arn:aws:iam::111122223333:role/ACK-RDS-Role",
        "arn:aws:iam::444455556666:role/ACKCrossAccountRole"
      ]
    }
  ]
}
```

Para obter detalhes sobre a configuração de permissões do ACK, incluindo os seletores de perfil do IAM, consulte the section called “Configurar permissões”.

Argo CD

Por padrão, nenhuma permissão do IAM é necessária. Entretanto, permissões adicionais podem ser exigidas para:

- AWS Secrets Manager: se estiver usando o Secrets Manager para armazenar credenciais de repositório do Git
- AWS CodeConnections: se usar o CodeConnections para realizar autenticação em repositórios do Git

Exemplo de política para o Secrets Manager e para o CodeConnections:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:region:account-id:secret:argocd/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection",
        "codeconnections:GetConnection"
      ],
      "Resource": "arn:aws:codeconnections:region:account-id:connection/*"
    }
  ]
}
```

Para obter detalhes sobre os requisitos da permissão do Argo CD, consulte the section called “Considerações sobre o Argo CD”.

Verificação da existência de um perfil da funcionalidade

Você pode usar o procedimento apresentado a seguir para verificar se a conta já tem um perfil do IAM para a funcionalidade que seja adequado ao seu caso de uso.

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação à esquerda, escolha Funções.
3. Pesquise na lista de perfis pelo nome do perfil da funcionalidade (por exemplo, `ACKCapabilityRole` ou `ArgoCDCapabilityRole`).
4. Se um perfil existir, selecione-o para visualizar as políticas anexadas e a relação de confiança.
5. Escolha Relações de confiança e, em seguida, escolha Editar política de confiança.
6. Verifique se a relação de confiança corresponde à política de confiança da funcionalidade. Se não houver correspondência, atualize a política de confiança.

7. Selecione a guia Permissões e valide se o perfil conta com as permissões adequadas ao seu tipo de funcionalidade e às necessidades do caso de uso.

Criação de um perfil do IAM para a funcionalidade

É possível usar o Console de gerenciamento da AWS ou a AWS CLI para a criação de um perfil da funcionalidade.

Console de gerenciamento da AWS

a. Abra o console do IAM em https://console.aws.amazon.com/iam/.
b. Escolha Roles (Funções) e, em seguida, Create Role (Criar função).
c. Em Tipo de entidade confiável, selecione Política de confiança personalizada.
d. Copie e cole a política de confiança da funcionalidade no editor de política de confiança.
e. Escolha Próximo.
f. Na guia Adicionar permissões, selecione ou crie as políticas apropriadas para o seu tipo de funcionalidade (consulte the section called “Permissões por tipo de funcionalidade”). Para o kro, você pode pular esta etapa.
g. Escolha Próximo.
h. Em Nome do perfil, insira um nome exclusivo para o seu perfil, como`ACKCapabilityRole`, `ArgoCDCapabilityRole` ou `kroCapabilityRole`.
i. Em Description (Descrição), insira um texto descritivo, como `Amazon EKS - ACK capability role`.
j. Selecione Criar perfil.

AWS CLI

a. Copie a política de confiança da funcionalidade para um arquivo chamado `capability-trust-policy.json`.
b. Crie a função. Substitua `ACKCapabilityRole` pelo nome de perfil desejado.

```
aws iam create-role \
  --role-name ACKCapabilityRole \
  --assume-role-policy-document file://capability-trust-policy.json
```

c. Anexe as políticas do IAM necessárias ao perfil. Para o ACK, anexe as políticas dos serviços da AWS que você deseja gerenciar. Para o Argo CD, anexe políticas para o Secrets Manager ou CodeConnections, se necessário. Para o kro, você pode pular esta etapa.

Exemplo para o ACK com permissões de S3:

```
aws iam put-role-policy \
  --role-name ACKCapabilityRole \
  --policy-name S3Management \
  --policy-document file://s3-policy.json
```

Solução de problemas em perfis da funcionalidade

Criação da funcionalidade apresentou falhas com o erro “Perfil do IAM inválido”

Verificar se:

- O perfil existe na mesma conta que o cluster
- A política de confiança corresponde à política de confiança da funcionalidade
- Você conta com a permissão `iam:PassRole` para o perfil

Funcionalidade apresenta erros de permissão

Verificar se:

- O perfil tem as permissões do IAM necessárias para o tipo de funcionalidade
- A entrada de acesso existe no cluster para o perfil
- As permissões adicionais do Kubernetes estão configuradas, se necessário (consulte the section called “Permissões adicionais do Kubernetes”)

Recursos do ACK apresentam falhas com erros de “Permissão negada”

Verificar se:

- O perfil tem as permissões do IAM necessárias para o caso de uso
- Para controladores do ACK que referenciam segredos, certifique-se de ter associado a política de entrada de acesso `AmazonEKSSecretReaderPolicy` com escopo para os namespaces apropriados.

Para obter mais orientações sobre a solução de problemas, consulte the section called “Considerações para funcionalidades do EKS”.

Monitorar a performance de clusters e visualizara logs

É possível observar seus dados no Amazon EKS usando várias ferramentas de monitoramento ou registro em log disponíveis. Os dados de log do Amazon EKS podem ser transmitidos para os serviços da AWS ou para ferramentas de parceiros para análise de dados. Existem muitos serviços disponíveis no Console de gerenciamento da AWS que fornecem dados para solucionar problemas do Amazon EKS. Você também pode usar uma solução de código aberto compatível com a AWS para monitorar a infraestrutura do Amazon EKS.

Depois de selecionar Clusters no painel de navegação esquerdo do console do Amazon EKS, será possível visualizar a integridade e os detalhes do cluster escolhendo o nome do cluster e a guia Observabilidade. Para visualizar detalhes sobre todos os recursos existentes do Kubernetes que estão implantados no cluster, consulte the section called “Acessar recursos do cluster”.

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e a performance do Amazon EKS e das soluções da AWS. Recomendamos coletar dados de monitoramento de todas as partes da sua solução da AWS. Dessa maneira, é possível depurar uma falha de vários pontos com facilidade, caso ocorra. Antes de começar a monitorar o Amazon EKS, certifique-se de que seu plano de monitoramento aborde as seguintes questões.

- Quais são as suas metas? Você precisa de notificações em tempo real se os seus clusters forem escalados drasticamente?
- Que recursos precisam ser observados?
- Com que frequência esses recursos precisam ser observados? Sua empresa deseja responder rapidamente a riscos?
- Que ferramentas você pretende usar? Se você já executa o AWS Fargate como parte da sua inicialização, poderá utilizar o roteador de logs.
- Quem você pretende designar para realizar as tarefas de monitoramento?
- Para quem você deseja que notificações sejam enviadas quando algo der errado?

Monitoramento e registro em log no Amazon EKS

O Amazon EKS fornece ferramentas incorporadas para monitoramento e registro em log. Para versões com suporte, o painel de observabilidade oferece visibilidade à performance do seu cluster. Ele ajuda você a detectar, solucionar e remediar problemas rapidamente. Além dos atributos de monitoramento, ele inclui listas com base nos logs de auditoria do ambiente de gerenciamento. O

ambiente de gerenciamento do Kubernetes expõe várias métricas que também podem ser extraídas do console.

Os logs do ambiente de gerenciamento registram todas as chamadas de API para os clusters, as informações de auditoria que capturam quais usuários realizaram quais ações nesses clusters, bem como informações baseadas em perfil. Para obter mais informações, consulte Registro em log e monitoramento no Amazon EKS, no documento de Orientação prescritiva da AWS.

O registro em log do plano de controle do Amazon EKS fornece logs de auditoria e diagnóstico diretamente do plano de controle do Amazon EKS para o CloudWatch Logs em sua conta. Esses logs facilitam a proteção e a execução dos clusters. Você pode selecionar os tipos exatos de logs que precisa, e eles serão enviados como fluxos de log para cada cluster do Amazon EKS no CloudWatch. Para obter mais informações, consulte the section called “Logs do ambiente de gerenciamento”.

> **Note**
>
> Ao verificar os logs do autenticador do Amazon EKS no Amazon CloudWatch, são exibidas as entradas que contêm texto semelhante ao exemplo a seguir.
>
> ```
> level=info msg="mapping IAM role" groups="[]"
>  role="arn:aws:iam::111122223333:role/XXXXXXXXXXXXXXXXX-NodeManagerRole-
> XXXXXXXX" username="eks:node-manager"
> ```
>
> As entradas que contenham esse texto são esperadas. O `username` é uma função de serviço interna do Amazon EKS que executa operações específicas para grupos de nós gerenciados e o Fargate.
> Para registro em log personalizável de baixo nível, está disponível o registro em log do Kubernetes.

O Amazon EKS está integrado ao AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, um perfil ou um serviço da AWS no Amazon EKS. O CloudTrail captura todas as chamadas de API para o Amazon EKS como eventos. As chamadas capturadas incluem as chamadas do console do Amazon EKS e as chamadas de código para as operações da API do Amazon EKS. Para obter mais informações, consulte the section called “ AWS CloudTrail”.

O servidor de API do Kubernetes expões várias métricas úteis para monitoramento e análise. Para obter mais informações, consulte the section called “Métricas do Prometheus”.

Para configurar o Fluent Bit para logs personalizados do Amazon CloudWatch, consulte Configurar o Fluent Bit no Guia do usuário do Amazon CloudWatch.

Ferramentas do Amazon EKS de monitoramento e registro em log

A Amazon Web Services fornece várias ferramentas que você pode usar para monitorar o Amazon EKS. É possível configurar algumas ferramentas para definir o monitoramento automático, mas algumas exigem chamadas manuais. Convém automatizar as tarefas de monitoramento o tanto quanto permitido pelo seu ambiente e conjunto de ferramentas.

A tabela a seguir descreve várias opções de ferramentas de monitoramento.

Áreas	Ferramenta	Descrição	Configuração
Ambiente de gerenciamento	Painel de observabilidade	Para versões com suporte, o painel de observabilidade oferece visibilidade à performance do seu cluster. Ele ajuda você a detectar, solucionar e remediar problemas rapidamente.	Procedimento de configuração
Aplicações/ambiente de gerenciamento	Prometheus	O Prometheus pode ser usado para monitorar métricas e alertas de aplicações e ambiente de gerenciamento.	Procedimento de configuração
Aplicações	CloudWatch Container Insights	O CloudWatch Container Insights coleta, agrega e resume métricas e logs das suas aplicações e	Procedimento de configuração

Áreas	Ferramenta	Descrição	Configuração
		microsserviços conteinerizados.	
Aplicações	AWS Distro for OpenTelemetry (ADOT)	O ADOT coleta e envia métricas correlacionadas, dados de rastreamento e metadados para serviços de monitoramento da AWS ou a parceiros. Ele pode ser configurado por meio do CloudWatch Container Insights.	Procedimento de configuração
Aplicações	Amazon DevOps Guru	O Amazon DevOps Guru detecta a performance e a disponibilidade operacionais no nível do nó.	Procedimento de configuração

Áreas	Ferramenta	Descrição	Configuração
Aplicações	AWS X-Ray	O AWS X-Ray recebe dados de rastreamento sobre a sua aplicação. Esses dados de rastreamento incluem solicitações e metadados de entrada e de saída sobre as solicitações. Para o Amazon EKS, a implementação requer o complemento OpenTelemetry.	Procedimento de configuração
Aplicações	Amazon CloudWatch	O CloudWatch fornece algumas métricas básicas do Amazon EKS gratuitamente nas versões com suporte. É possível expandir essa funcionalidade com o CloudWatch Observability Operator para lidar com a coleta de métricas, logs e dados de rastreamento.	Procedimento de configuração

A tabela a seguir descreve várias opções de ferramentas de registro em log.

Áreas	Ferramenta	Descrição	Configuração
Ambiente de gerenciamento	Painel de observabilidade	Para versões com suporte, o painel de observabilidade mostra listas com base nos registros de auditoria do ambiente de gerenciamento. Ele também inclui links para controlar os registros do ambiente de gerenciamento no Amazon CloudWatch.	Procedimento de configuração
Aplicações	Amazon CloudWatch Container Insights	O Amazon CloudWatch Container Insights coleta, agrega e resume métricas e logs das suas aplicações e microsserviços em contêineres.	Procedimento de configuração
Ambiente de gerenciamento	Amazon CloudWatch Logs	É possível enviar logs de auditoria e diagnóstico diretamente do ambiente de gerenciamento do Amazon EKS para o CloudWatch Logs em sua conta.	Procedimento de configuração
Ambiente de gerenciamento	AWS CloudTrail	Ele registra chamadas de API por parte de um usuário, uma função ou um serviço.	Procedimento de configuração

Áreas	Ferramenta	Descrição	Configuração
Várias áreas para instâncias do AWS Fargate	AWS Roteador de logs do Fargate	Para as instâncias do AWS Fargate, o roteador de logs transmite os logs para os serviços da AWS ou para ferramentas de parceiros. Ele utiliza o AWS for Fluent Bit. Os logs podem ser transmitidos para outros serviços da AWS ou ferramentas de parceiros.	Procedimento de configuração

Monitore seu cluster com o painel de observabilidade

O console do Amazon EKS inclui um painel de observabilidade que dá visibilidade sobre a performance do seu cluster. As informações que ele fornece ajudam você a detectar, solucionar e remediar problemas rapidamente. É possível abrir a seção aplicável do painel de observabilidade escolhendo um item no Resumo de integridade e performance. Esse resumo está incluído em vários lugares, incluindo a guia Observabilidade.

O painel de observabilidade está dividido em várias guias.

Resumo

O Resumo de integridade e performance lista a quantidade de itens em várias categorias. Cada número atua como um hiperlink para um local no painel de observabilidade com uma lista para a categoria.

Integridade do cluster

A Integridade do cluster fornece notificações importantes de que você deve ter ciência, algumas das quais talvez você precise resolver o mais rápido possível. Com essa lista, é possível ver as

descrições e os recursos afetados. A integridade do cluster inclui duas tabelas: Problemas de integridade e Insights de configurações. Para atualizar o status dos Problemas de integridade, escolha o botão de atualização (↻). Os Insights de configurações são atualizados automaticamente uma vez a cada 24 horas e não podem ser atualizados manualmente.

Para obter mais informações sobre Problemas de integridade, consulte the section called “Perguntas frequentes sobre a integridade do cluster e códigos de erro com caminhos de resolução”. Para obter mais informações sobre Insights de configurações, consulte the section called “Insights de cluster”.

Monitoramento do ambiente de gerenciamento

A guia Monitoramento do ambiente de gerenciamento é dividida em três seções, cada uma das quais ajuda você a monitorar e solucionar problemas no ambiente de gerenciamento do cluster.

Métricas

Nos clusters com a versão `1.28` ou mais recente do Kubernetes, a seção Métricas apresenta grafos de diversas métricas coletadas de vários componentes do ambiente de gerenciamento.

É possível definir o período de tempo usado pelo eixo X de cada gráfico fazendo seleções na parte superior da seção. É possível atualizar os dados com o botão de atualização (↻). Para cada gráfico separado, o botão de reticências verticais (⋮) abre um menu com opções do CloudWatch.

Essas métricas e outras estão automaticamente disponíveis como métricas básicas de monitoramento no CloudWatch sob o namespace `AWS/EKS`. Para obter mais informações, consulte Monitoramento básico e monitoramento detalhado no Guia do usuário do Amazon CloudWatch. Para obter métricas, visualizações e insights mais detalhados, consulte Container Insights no Guia do usuário do Amazon CloudWatch. Ou, se você preferir o monitoramento baseado no Prometheus, consulte the section called “Métricas do Prometheus”.

A tabela a seguir descreve as métricas disponíveis.

Métrica	Descrição
Solicitações do APIServer	As solicitações por minuto feitas ao servidor de API.
Total de solicitações do APIserver 4XX	A contagem de solicitações ao servidor de API por minuto que receberam código de resposta HTTP 4XX (erro do lado do cliente).

Métrica	Descrição
Total de solicitações do APIserver 5XX	A contagem de solicitações ao servidor de API por minuto que receberam código de resposta HTTP 5XX (erro do lado do servidor).
Total de solicitações do APIserver 429	A contagem de solicitações ao servidor de API por minuto que receberam código de resposta HTTP 429 (excesso de solicitações).
Tamanho de armazenamento	O tamanho do banco de dados de armazenamento (`etcd`).
Tentativas do programador	O número de tentativas para programar pods por resultados "não programáveis", "erro" e "programados".
Pods pendentes	O número de pods pendentes por tipo de fila: "ativos", "suspensos", "não programáveis" e "fechados".
Latência da solicitação ao servidor de API	A latência das solicitações ao servidor de API.
Solicitações em trânsito atuais do servidor de API	As solicitações atuais em trânsito para o servidor de API.
Solicitações de webhook	As solicitações de webhook por minuto.
Rejeições de solicitações de webhook	A contagem de solicitações de webhook que foram rejeitadas.
Latência de solicitação de webhook P99	A latência do 99º percentil de solicitações de webhook de terceiros externos.

Insights do CloudWatch Logs

A seção Insights do CloudWatch Log mostra várias listas com base nos registros de auditoria do ambiente de gerenciamento. Os logs do ambiente de gerenciamento do Amazon EKS precisam estar

ativados para usar esse atributo, o que pode ser feito na seção Visualização de logs do ambiente de gerenciamento no CloudWatch.

Quando tiver passado tempo suficiente para coletar dados, será possível Executar todas as consultas ou escolher Executar consulta para uma única lista por vez. O CloudWatch terá um custo adicional sempre que você executar consultas. Escolha o período de resultados que você deseja visualizar na parte superior da seção. Se quiser um controle mais avançado para qualquer consulta, é possível escolher Exibir no CloudWatch. Isso permitirá que você atualize uma consulta no CloudWatch para atender às suas necessidades.

Para obter mais informações, consulte Análise de dados de log com o CloudWatch Logs Insights no Guia do usuário do Amazon CloudWatch Logs.

Visualize os logs do ambiente de gerenciamento no CloudWatch.

Escolha Gerenciar registro em log para atualizar os tipos de logs disponíveis. São necessários vários minutos para que os logs apareçam no CloudWatch Logs depois que você habilita o registro em log. Quando tiver passado tempo suficiente, escolha qualquer um dos links Visualizar nesta seção para navegar até o log aplicável.

Para obter mais informações, consulte the section called “Logs do ambiente de gerenciamento”.

Insights de um cluster

A tabela Insights de atualização revela problemas e recomenda ações corretivas, acelerando o processo de validação para a atualização para novas versões do Kubernetes. O Amazon EKS verifica os clusters automaticamente com relação a uma lista de possíveis problemas que afetam a atualização da versão do Kubernetes. A tabela Insights de atualização lista as verificações de insights realizadas pelo Amazon EKS em relação a esse cluster, junto com seus status associados.

O Amazon EKS mantém e atualiza periodicamente a lista de verificações de insights a serem realizadas com base nas avaliações das mudanças no projeto Kubernetes, bem como nas alterações do serviço do Amazon EKS vinculadas às novas versões. O console do Amazon EKS atualiza automaticamente o status de cada insight, o que pode ser visto na coluna de horário da última atualização.

Para obter mais informações, consulte the section called “Insights de cluster”.

Problemas de integridade de nós

O agente de monitoramento de nós do Amazon EKS lê automaticamente os logs dos nós para detectar problemas de integridade. Independentemente da configuração de reparo automático, todos os problemas de integridade do nó são relatados para que você possa investigar, conforme necessário. Se um tipo de problema estiver listado sem uma descrição, você poderá ler a descrição em seu elemento popover.

Quando você atualizar a página, todos os problemas resolvidos desaparecerão da lista. Se o reparo automático estiver habilitado, você poderá ver temporariamente alguns problemas de integridade que serão resolvidos sem uma ação sua. Problemas que não são compatíveis com o reparo automático podem exigir uma ação manual sua, dependendo do tipo.

Para que problemas de integridade do nó sejam relatados, o cluster deve usar o Modo Automático do Amazon EKS ou ter o complemento do agente de monitoramento de nós. Para obter mais informações, consulte the section called “Integridade de nós”.

Funcionalidades do EKS

A seção Funcionalidades mostra o status e a integridade dos seus recursos da funcionalidade do EKS no cluster. Você pode monitorar aqui as notificações de integridade e o status das funcionalidades e de seus respectivos recursos gerenciados pelo Kubernetes no cluster. Quando você atualizar a página, todos os problemas resolvidos desaparecerão da lista.

Para obter mais informações, consulte the section called “Como trabalhar com funcionalidades”.

Monitoramento do tráfego de workloads do Kubernetes com observabilidade da rede de contêineres

O Amazon EKS oferece recursos de observabilidade da rede aprimorados que proporcionam insights aprofundados sobre o seu ambiente de rede de contêineres. Essas funcionalidades auxiliam na melhor compreensão, monitoramento e solução de problemas no panorama de rede do Kubernetes na AWS. Com a observabilidade da rede de contêineres aprimorada, você pode aproveitar métricas de rede granulares para uma melhor detecção proativa de anomalias em todo o tráfego do cluster, fluxos entre zonas de disponibilidade e serviços da AWS. Com essas métricas, é possível mensurar a performance do sistema e visualizar as métricas subjacentes por meio da pilha de observabilidade de sua preferência.

Além disso, o Amazon EKS agora fornece visualizações para o monitoramento da rede no console da AWS, agilizando e aprimorando a precisão da solução de problemas para uma análise de causa raiz com mais rapidez. Você também pode aproveitar essas funcionalidades visuais para identificar os recursos que mais geram comunicação e os fluxos de rede que causam retransmissões e tempos limite de retransmissão, eliminando pontos cegos durante incidentes.

Essas funcionalidades são habilitadas pelo Amazon CloudWatch Network Flow Monitor.

Casos de uso

Mensurar a performance da rede para detecção de anomalias

Diversas equipes padronizam uma pilha de observabilidade que lhes permite mensurar a performance do sistema, visualizar métricas do sistema e receber alertas caso um limite específico seja violado. A observabilidade da rede de contêineres no EKS é compatível com esse modelo, fornecendo métricas essenciais do sistema para coleta, o que expande a observabilidade da performance da rede do sistema nos níveis de pods e de nós de processamento.

Aproveitar as visualizações do console para uma solução de problemas mais precisa

Caso ocorra um alarme em seu sistema de monitoramento, talvez você precise se concentrar no cluster e na workload em que o problema se originou. Para apoiar isso, você pode aproveitar as visualizações no console do EKS que restringem o escopo da investigação no nível do cluster e aceleram a identificação dos fluxos de rede responsáveis pela maioria das retransmissões, pelos tempos limite de retransmissão e pelo volume de dados transferidos.

Rastrear os recursos que mais geram comunicação no ambiente do Amazon EKS

Muitas equipes executam o EKS como a base de suas plataformas, tornando-o o ponto central da atividade da rede de um ambiente de aplicação. Usando as funcionalidades de monitoramento de rede deste recurso, é possível rastrear quais workloads são responsáveis pela maior parte do tráfego (com o valor mensurado pelo volume de dados) dentro do cluster, em todas as zonas de disponibilidade (AZs), bem como o tráfego para destinos externos dentro da AWS (como o DynamoDB e o S3) e além da Nuvem AWS (internet ou ambientes on-premises). Além disso, você pode monitorar a performance de cada um desses fluxos com base em retransmissões, tempos limite de retransmissão e dados transferidos.

Recursos

1. Métricas de performance: este recurso permite que você extraia métricas de sistema relativas à rede, tanto de pods quanto de nós de processamento, diretamente do Agente do Network Flow Monitor (NFM) que opera no seu cluster do EKS.
2. Mapa do serviço: este recurso oferece uma visualização dinâmica da comunicação interna entre as workloads do cluster, possibilitando a identificação rápida de métricas relevantes (retransmissões [RT], tempos limite de retransmissão [RTO] e dados transferidos [DT]) relativas aos fluxos de rede entre os pods em comunicação.
3. Tabela de fluxo: com esta tabela, é possível monitorar os recursos que mais geram comunicação entre as workloads do Kubernetes no cluster sob três perspectivas distintas: visualização de serviços da AWS, visualização do cluster e visualização externa. Para cada visualização, você consegue visualizar as retransmissões, os tempos limite de retransmissão e os dados transferidos entre o pod de origem e o destino.
   - Visualização de serviços da AWS: mostra os recursos que mais geram comunicação com os serviços da AWS (DynamoDB e S3)
   - Visualização do cluster: mostra os recursos que mais geram comunicação dentro do cluster (leste ← para → oeste)
   - Visualização externa: mostra os recursos que mais geram comunicação com destinos externos ao cluster fora da AWS

Conceitos básicos

Para começar a usar, habilite a observabilidade da rede de contêineres no console do EKS em um cluster novo ou já existente. Esta ação automatiza a criação das dependências do Network Flow Monitor (NFM), como os recursos de Escopo e Monitor. Além disso, você precisará instalar o complemento do Agente do Network Flow Monitor. Como alternativa, você pode instalar essas dependências usando a `AWS CLI`, APIs do EKS (para o complemento), APIs do NFM ou infraestrutura como código (como o Terraform). Assim que essas dependências estiverem configuradas, você poderá configurar a ferramenta de monitoramento de sua preferência para coletar métricas de performance da rede para pods e nós de processamento do agente do NFM. Para visualizar a atividade e a performance da rede para as workloads, basta acessar o console do EKS e selecionar a guia “Rede” no painel de observabilidade do cluster.

Ao usar o Network Flow Monitor no EKS, é possível manter o fluxo de trabalho de observabilidade e a pilha de tecnologia existentes, enquanto aproveita um conjunto de recursos adicionais que

permitem compreender e otimizar ainda mais a camada de rede do seu ambiente do EKS. Para saber mais sobre os preços do Network Flow Monitor, clique aqui.

Pré-requisitos e observações importantes

1. Conforme mencionado anteriormente, ao habilitar a observabilidade da rede de contêineres pelo console do EKS, as dependências de recursos subjacentes do NFM (nomeadamente Escopo e Monitor) serão criadas automaticamente para você, e o sistema guiará você durante a instalação do complemento do EKS para o NFM.
2. Se você desejar ativar este recurso usando infraestrutura como código (IaC), como o Terraform, precisará definir as seguintes dependências em sua IaC: Escopo do NFM, Monitor do NFM e o complemento do EKS para NFM. Além disso, você precisará conceder as permissões relevantes ao complemento do EKS usando a Identidade de Pods ou os perfis do IAM para contas de serviço (IRSA, na sigla em inglês).
3. Você deve estar utilizando, no mínimo, a versão 1.1.0 do complemento do EKS para o agente do NFM.
4. Você deve usar a versão 6.21.0 ou as versões posteriores do Terraform AWS Provider para obter suporte aos recursos do Network Flow Monitor.

Permissões obrigatórias do IAM

Complemento do EKS para o agente do NFM

É possível usar a política gerenciada pela AWS `CloudWatchNetworkFlowMonitorAgentPublishPolicy` com a Identidade de Pods. Esta política contém permissões para que o agente do NFM envie relatórios de telemetria (métricas) para um endpoint do Network Flow Monitor.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "networkflowmonitor:Publish"
      ],
      "Resource" : "*"
    }
  ]
```

```
}
```

Observabilidade da rede de contêineres no console do EKS

As permissões listadas abaixo são obrigatórias para habilitar o recurso e visualizar o mapa do serviço e a tabela de fluxo diretamente no console.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect": "Allow",
      "Action": [
        "networkflowmonitor:ListScopes",
        "networkflowmonitor:ListMonitors",
        "networkflowmonitor:GetScope",
        "networkflowmonitor:GetMonitor",
        "networkflowmonitor:CreateScope",
        "networkflowmonitor:CreateMonitor",
        "networkflowmonitor:TagResource",
        "networkflowmonitor:StartQueryMonitorTopContributors",
        "networkflowmonitor:StopQueryMonitorTopContributors",
        "networkflowmonitor:GetQueryStatusMonitorTopContributors",
        "networkflowmonitor:GetQueryResultsMonitorTopContributors"
      ],
      "Resource": "*"
    }
  ]
}
```

Uso da AWS CLI, da API do EKS e da API do NFM

```
#!/bin/bash

# Script to create required Network Flow Monitor resources
set -e

CLUSTER_NAME="my-eks-cluster"
CLUSTER_ARN="arn:aws:eks:{Region}:{Account}:cluster/{ClusterName}"
REGION="us-west-2"
AGENT_NAMESPACE="amazon-network-flow-monitor"
```

```
echo "Creating Network Flow Monitor resources..."

# Check if Network Flow Monitor agent is running in the cluster
echo "Checking for Network Flow Monitor agent in cluster..."
if kubectl get pods -n "$AGENT_NAMESPACE" --no-headers 2>/dev/null | grep -q "Running";
 then
    echo "Network Flow Monitor agent exists and is running in the cluster"
else
    echo "Network Flow Monitor agent not found. Installing as EKS addon..."
    aws eks create-addon \
        --cluster-name "$CLUSTER_NAME" \
        --addon-name "$AGENT_NAMESPACE" \
        --region "$REGION"
    echo "Network Flow Monitor addon installation initiated"
fi

# Get Account ID
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)

echo "Cluster ARN: $CLUSTER_ARN"
echo "Account ID: $ACCOUNT_ID"

# Check for existing scope
echo "Checking for existing Network Flow Monitor Scope..."
EXISTING_SCOPE=$(aws networkflowmonitor list-scopes --region $REGION --query
 'scopes[0].scopeArn' --output text 2>/dev/null || echo "None")

if [ "$EXISTING_SCOPE" != "None" ] && [ "$EXISTING_SCOPE" != "null" ]; then
    echo "Using existing scope: $EXISTING_SCOPE"
    SCOPE_ARN=$EXISTING_SCOPE
else
    echo "Creating new Network Flow Monitor Scope..."
    SCOPE_RESPONSE=$(aws networkflowmonitor create-scope \
        --targets "[{\"targetIdentifier\":{\"targetId\":{\"accountId\":
\"${ACCOUNT_ID}\"},\"targetType\":\"ACCOUNT\"},\"region\":\"${REGION}\"}]" \
        --region $REGION \
        --output json)

    SCOPE_ARN=$(echo $SCOPE_RESPONSE | jq -r '.scopeArn')
    echo "Scope created: $SCOPE_ARN"
fi

# Create Network Flow Monitor with EKS Cluster as local resource
echo "Creating Network Flow Monitor..."
```

```
MONITOR_RESPONSE=$(aws networkflowmonitor create-monitor \
    --monitor-name "${CLUSTER_NAME}-monitor" \
    --local-resources "type=AWS::EKS::Cluster,identifier=${CLUSTER_ARN}" \
    --scope-arn "$SCOPE_ARN" \
    --region $REGION \
    --output json)

MONITOR_ARN=$(echo $MONITOR_RESPONSE | jq -r '.monitorArn')

echo "Monitor created: $MONITOR_ARN"

echo "Network Flow Monitor setup complete!"
echo "Monitor ARN: $MONITOR_ARN"
echo "Scope ARN: $SCOPE_ARN"
echo "Local Resource: AWS::EKS::Cluster (${CLUSTER_ARN})"
```

Uso da infraestrutura como código (IaC)

Terraform

Se você estiver usando o Terraform para gerenciar a infraestrutura de nuvem da AWS, poderá incluir as configurações de recursos apresentadas a seguir para habilitar a observabilidade da rede de contêineres em seu cluster.

Escopo do NFM

```
data "aws_caller_identity" "current" {}

resource "aws_networkflowmonitor_scope" "example" {
  target {
    region = "us-east-1"
    target_identifier {
      target_type = "ACCOUNT"
      target_id {
        account_id = data.aws_caller_identity.current.account_id
      }
    }
  }

  tags = {
    Name = "example"
  }
```

```
}
```

Monitor do NFM

```
resource "aws_networkflowmonitor_monitor" "example" {
  monitor_name = "eks-cluster-name-monitor"
  scope_arn    = aws_networkflowmonitor_scope.example.scope_arn

  local_resource {
    type       = "AWS::EKS::Cluster"
    identifier = aws_eks_cluster.example.arn
  }

  remote_resource {
    type       = "AWS::Region"
    identifier = "us-east-1" # this must be the same region that the cluster is in
  }

  tags = {
    Name = "example"
  }
}
```

Complemento do EKS para o NFM

```
resource "aws_eks_addon" "example" {
  cluster_name                = aws_eks_cluster.example.name
  addon_name                  = "aws-network-flow-monitoring-agent"
}
```

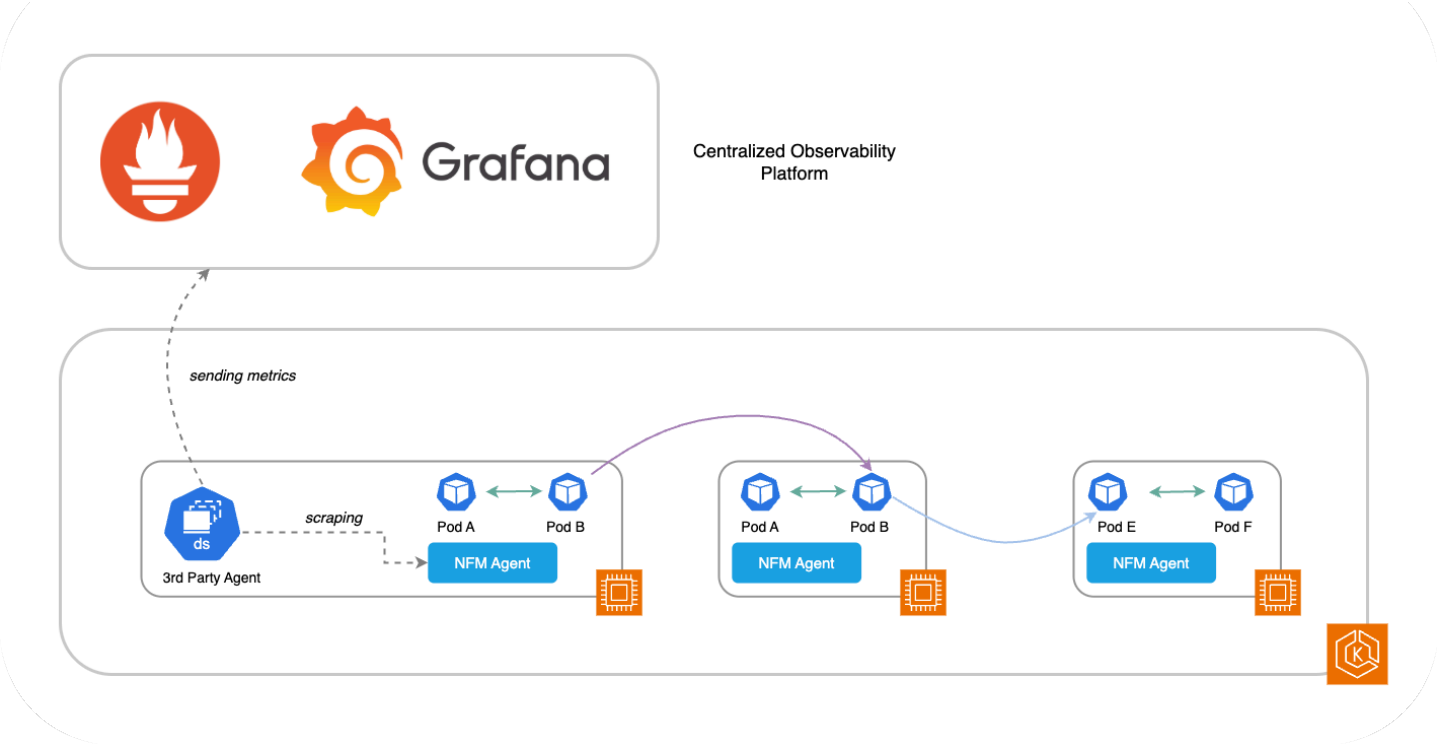
Como funciona?

Métricas de performance

Métricas do sistema

Caso você use ferramentas de terceiros (3P, na sigla em inglês) para o monitoramento do ambiente do EKS (como o Prometheus e o Grafana), é possível extrair as métricas de sistema compatíveis diretamente do agente do Network Flow Monitor. Essas métricas podem ser enviadas para a pilha de monitoramento para ampliar a medição da performance da rede do seu sistema no nível de pod

e de nó de processamento. Você encontra as métricas disponíveis na tabela na seção Métricas do sistema com suporte.



Para habilitar essas métricas, substitua as seguintes variáveis de ambiente usando as variáveis de configuração durante o processo de instalação (consulte: https://aws.amazon.com/blogs/containers/amazon-eks-add-ons-advanced-configuration/):

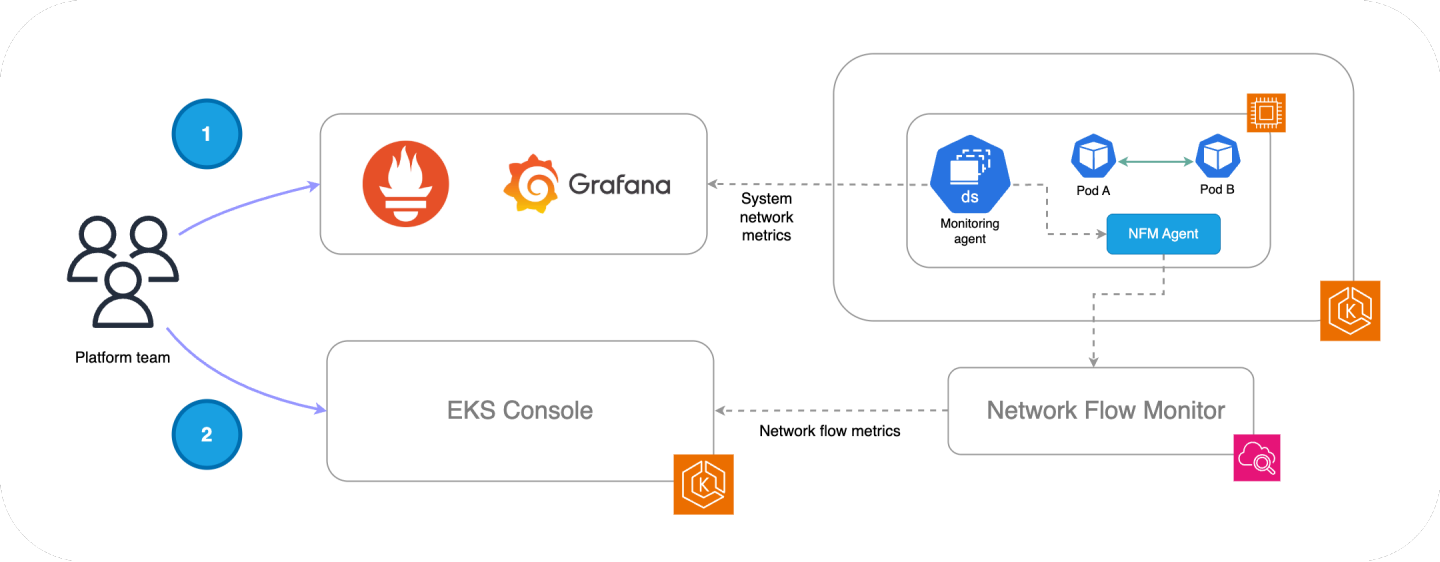
```
OPEN_METRICS:
    Enable or disable open metrics. Disabled if not supplied
    Type: String
    Values: [“on”, “off”]
OPEN_METRICS_ADDRESS:
    Listening IP address for open metrics endpoint. Defaults to 127.0.0.1 if not
 supplied
    Type: String
OPEN_METRICS_PORT:
    Listening port for open metrics endpoint. Defaults to 80 if not supplied
    Type: Integer
    Range: [0..65535]
```

Métricas de nível de fluxo

Além disso, o Network Flow Monitor captura dados de fluxo de rede juntamente com métricas de nível de fluxo: retransmissões, tempos limite de retransmissão e dados transferidos. Esses dados

são processados pelo Network Flow Monitor e exibidos no console do EKS para dar visibilidade ao tráfego do cluster e demonstrar a performance da rede com base nestas métricas de nível de fluxo.

O diagrama abaixo ilustra um fluxo de trabalho no qual ambos os tipos de métricas (de sistema e de nível de fluxo) podem ser aproveitados para obter maior inteligência operacional.



1. A equipe responsável pela plataforma pode coletar e visualizar as métricas do sistema na pilha de monitoramento. Com o sistema de alertas ativado, é possível detectar anomalias de rede ou problemas que impactem os pods ou os nós de processamento por meio das métricas de sistema do agente do NFM.
2. Como próximo passo, as equipes responsáveis pela plataforma podem aproveitar as visualizações nativas no console do EKS para restringir ainda mais o escopo da investigação e acelerar a solução de problemas com base em representações de fluxo e suas métricas associadas.

Observação importante: a coleta de métricas do sistema do agente do NFM e o processo de envio de métricas de nível de fluxo pelo agente do NFM ao backend do NFM são processos independentes.

Métricas do sistema com suporte

Observação importante: as métricas do sistema são exportadas no formato OpenMetrics.

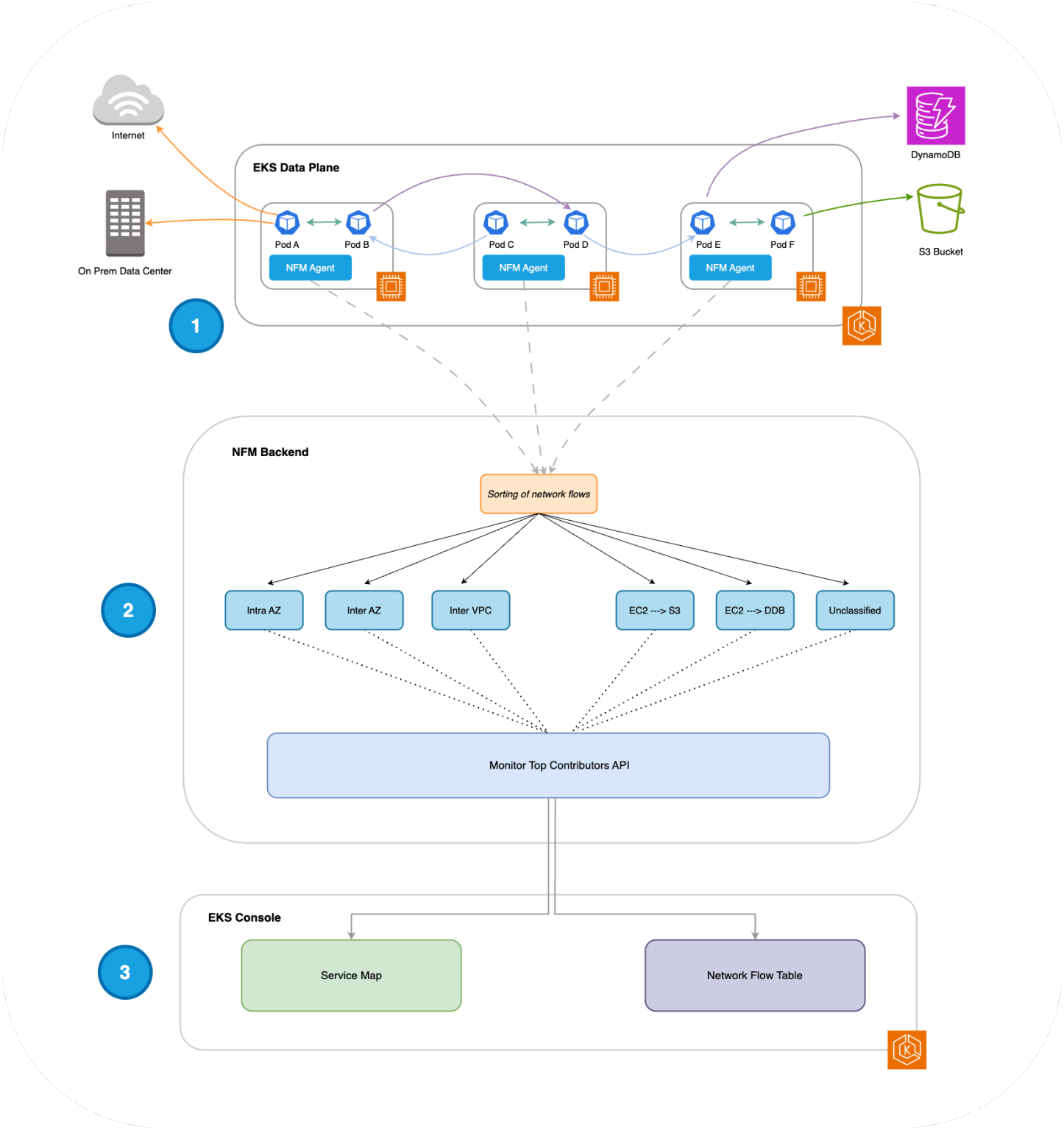
Nome da métrica	Tipo	Dimensões	Descrição
ingress_flow	Indicador	instance_id, iface, pod, namespace e node	Contagem de fluxos TCP de entrada (TcpPassiveOpens)
egress_flow	Indicador	instance_id, iface, pod, namespace e node	Contagem de fluxos TCP de saída (TcpActiveOpens)
ingress_packets	Indicador	instance_id, iface, pod, namespace e node	Contagem de pacotes de entrada (delta)
egress_packets	Indicador	instance_id, iface, pod, namespace e node	Contagem de pacotes de saída (delta)
ingress_bytes	Indicador	instance_id, iface, pod, namespace e node	Contagem de bytes de entrada (delta)
egress_bytes	Indicador	instance_id, iface, pod, namespace e node	Contagem de bytes de saída (delta)
bw_in_allowance_exceeded	Indicador	instance_id, eni e node	Pacotes enfileirados ou descartados devido ao limite de largura de banda de entrada
bw_out_allowance_exceeded	Indicador	instance_id, eni e node	Pacotes enfileirados ou descartados devido ao limite de largura de banda de saída

Nome da métrica	Tipo	Dimensões	Descrição
pps_allowance_exceeded	Indicador	instance_id, eni e node	Pacotes enfileirados ou descartados devido ao limite de PPS bidirecional
conntrack_allowance_exceeded	Indicador	instance_id, eni e node	Pacotes descartados devido ao limite de rastreamento de conexões
linklocal_allowance_exceeded	Indicador	instance_id, eni e node	Pacotes descartados devido ao limite de PPS do serviço de proxy local

Métricas de nível de fluxo com suporte

Nome da métrica	Tipo	Descrição
Retransmissões TCP	Contador	Número de vezes que um remetente reenvia um pacote que foi perdido ou corrompido durante a transmissão.
Tempos limite de retransmissão TCP	Contador	Número de vezes que um remetente iniciou um período de espera para determinar se um pacote foi perdido em trânsito.
Dados (bytes) transferidos	Contador	Volume de dados transferidos entre uma origem e um destino para um determinado fluxo.

Mapa do serviço e tabela de fluxo

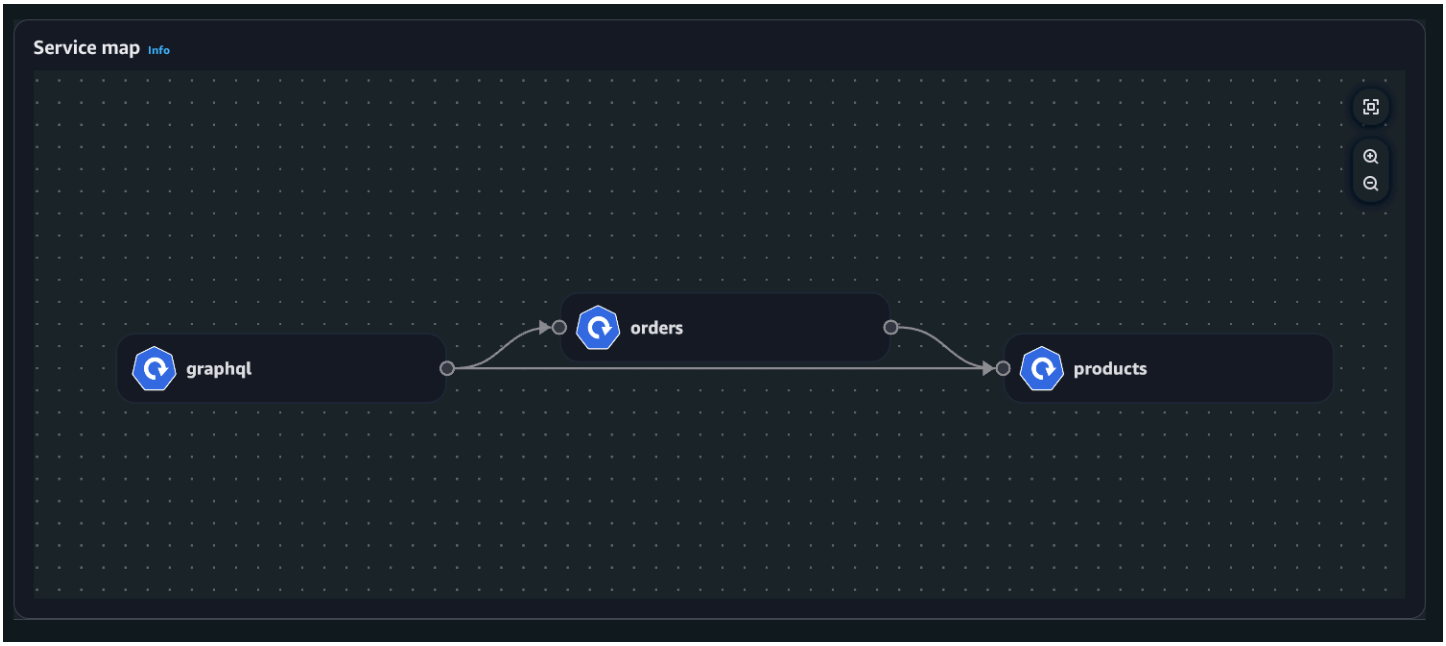


1. Quando instalado, o agente do Network Flow Monitor é executado como um DaemonSet em cada nó de processamento e coleta os 500 principais fluxos de rede (com base no volume de dados transferidos) a cada 30 segundos.
2. Esses fluxos de rede são classificados nas seguintes categorias: Intra-AZ, Inter-AZ, EC2 → S3, EC2 → DynamoDB (DDB) e Não classificados. Cada fluxo de rede é acompanhado por três métricas principais: retransmissões, tempos limite de retransmissão e volume de dados transferidos (em bytes).
   - Intra-AZ: fluxos de rede entre pods na mesma AZ
   - Inter-AZ: fluxos de rede entre pods em diferentes AZs
   - EC2 → S3: fluxos de rede dos pods para o S3
   - EC2 → DDB: fluxos de rede dos pods para o DDB
   - Não classificado: fluxos de rede dos pods para a internet ou para a infraestrutura on-premises
3. Os fluxos de rede provenientes da API “Principais colaboradores” do Network Flow Monitor são usados para potencializar as seguintes experiências no console do EKS:
   - Mapa do serviço: visualização dos fluxos de rede dentro do cluster (Intra-AZ e Inter-AZ).
   - Tabela de fluxo: apresentação em formato de tabela dos fluxos de rede dentro do cluster (Intra-AZ e Inter-AZ), de pods para serviços da AWS (EC2 → S3 e EC2 → DDB) e de pods para destinos externos (Não classificados).

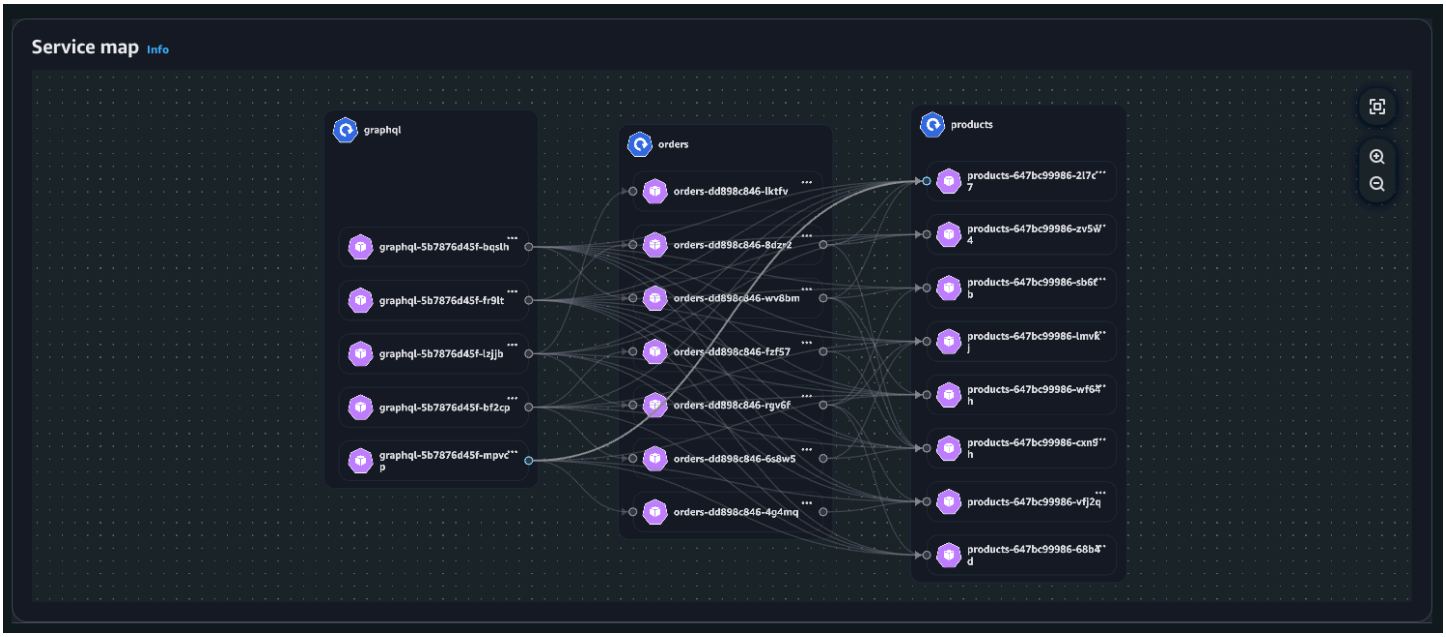
Os fluxos de rede extraídos da API “Principais colaboradores” são limitados a um intervalo de tempo de uma hora e podem incluir até 500 fluxos de cada categoria. Para o mapa do serviço, isso significa que até mil fluxos podem ser extraídos e apresentados com base nas categorias de fluxo Intra-AZ e Inter-AZ em um intervalo de tempo de uma hora. Para a tabela de fluxo, isso significa que até 3 mil fluxos de rede podem ser extraídos e apresentados com base em todas as seis categorias de fluxo de rede em um intervalo de tempo de duas horas.

Exemplo: mapa do serviço

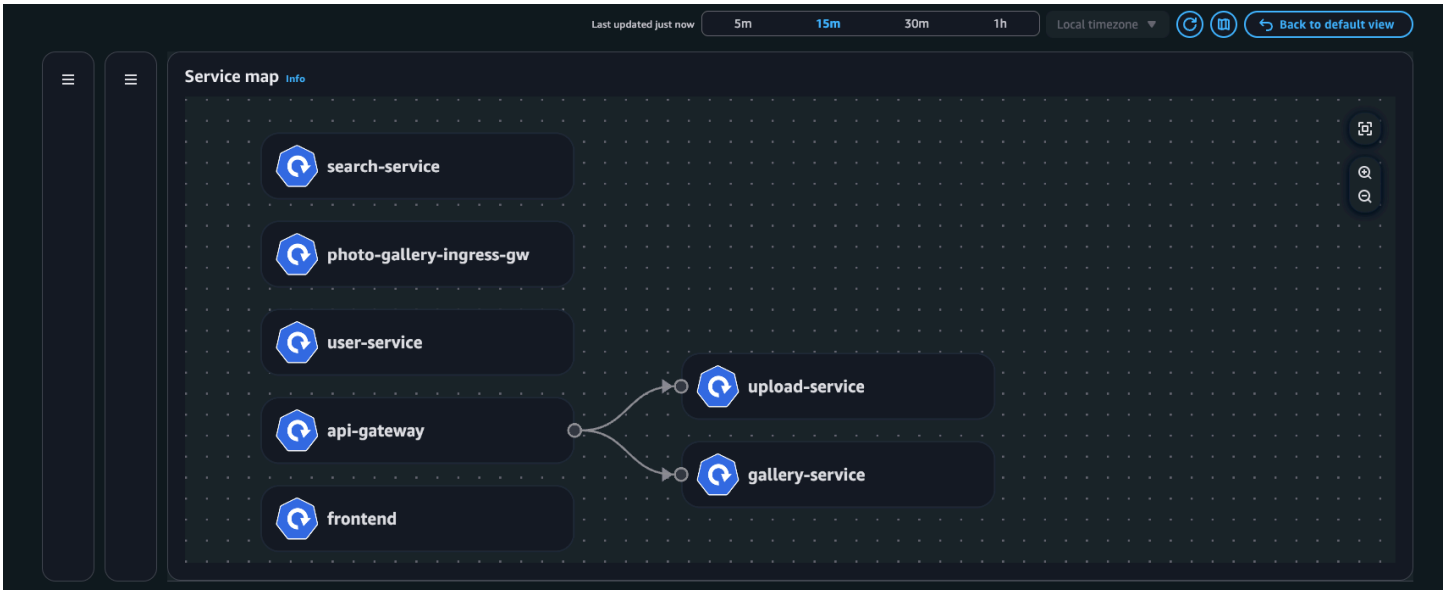
Visualização por implantação



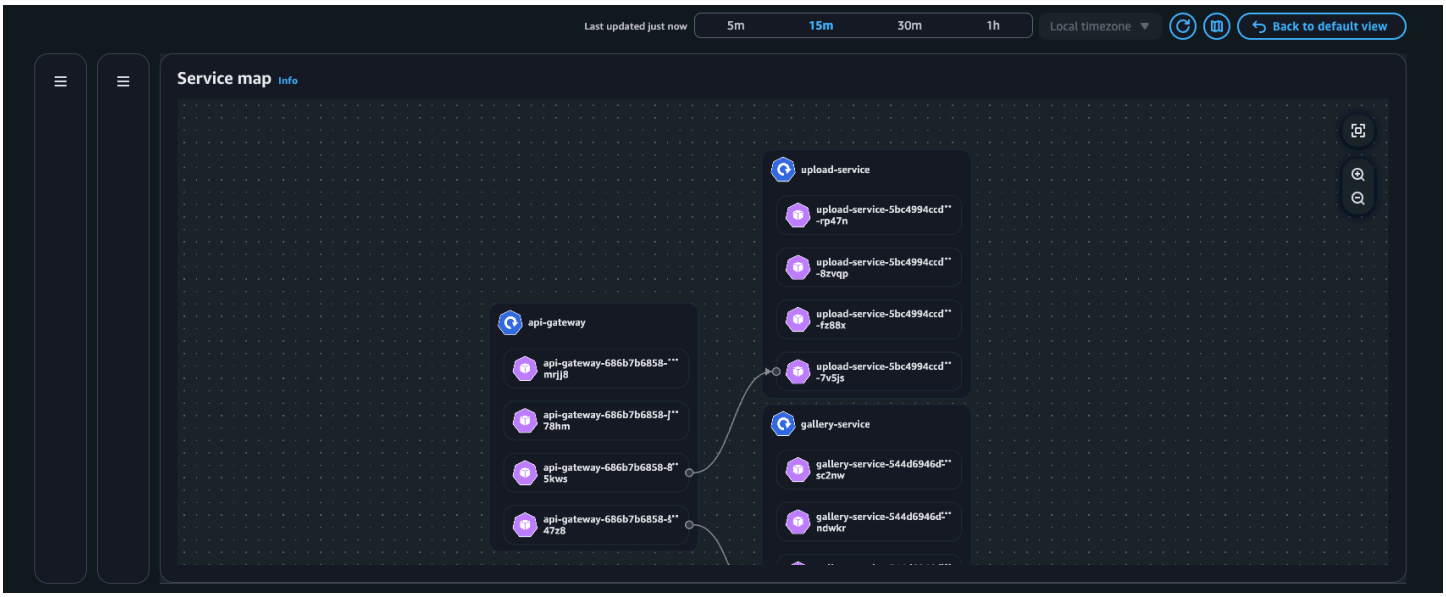
Visualização por pod



Visualização por implantação

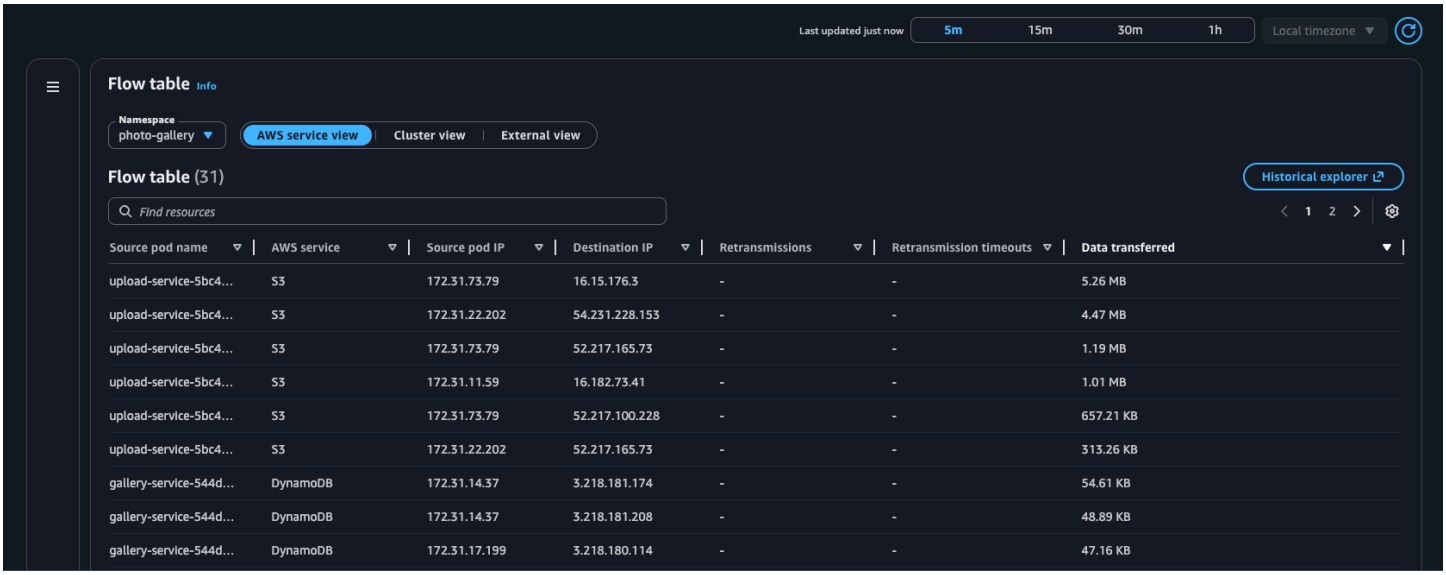


Visualização por pod

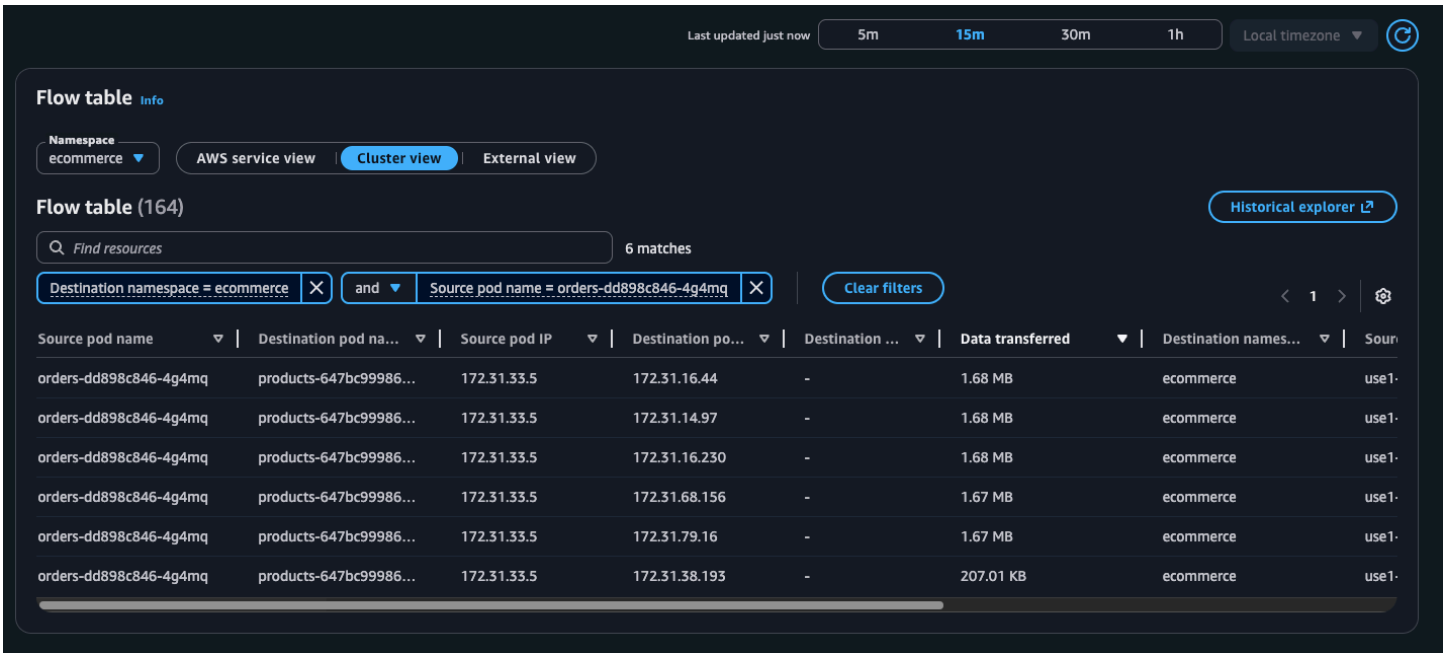


Exemplo: tabela de fluxo

Visualização por serviços da AWS



Visualização por cluster



Considerações e limitações

- A observabilidade da rede de contêineres no EKS está disponível apenas em regiões nas quais Network Flow Monitor tem suporte.
- As métricas de sistema compatíveis estão no formato OpenMetrics e podem ser coletadas diretamente do agente do Network Flow Monitor (NFM).

- Para habilitar a observabilidade da rede de contêineres no EKS usando Infraestrutura como Código (IaC), como o Terraform, é necessário ter estas dependências definidas e criadas em suas configurações: o escopo do NFM, o monitor do NFM e o agente do NFM.
- O Network Flow Monitor tem capacidade para processar aproximadamente cinco milhões de fluxos por minuto. Isso equivale a aproximadamente 5.000 instâncias do EC2 (nós de processamento do EKS) com o agente do Network Flow Monitor instalado. Instalar agentes em um número superior a 5.000 instâncias pode impactar a performance do monitoramento enquanto não houver capacidade adicional provisionada.
- Você deve estar utilizando, no mínimo, a versão 1.1.0 do complemento do EKS para o agente do NFM.
- Você deve usar a versão 6.21.0 ou as versões posteriores do Terraform AWS Provider para obter suporte aos recursos do Network Flow Monitor.
- Para enriquecer os fluxos de rede com metadados dos pods, os pods devem estar sendo executados em seus próprios namespaces de rede isolados, e não no namespace de rede do host.

Monitore as métricas do seu cluster com o Prometheus

O Prometheus é um banco de dados de monitoramento e de séries temporais que extrai endpoints. Ele permite consultar, agregar e armazenar dados coletados. Você também pode usá-lo para alertas e agregação de alertas. Este tópico explica como configurar o Prometheus como uma opção gerenciada ou de código aberto. Monitorar o ambiente de gerenciamento de métricas do Amazon EKS é um caso de uso comum.

O Amazon Managed Service for Prometheus é um serviço de monitoramento e alerta compatível com o Prometheus que facilita o monitoramento de aplicações em contêiner e de infraestrutura em escala. É um serviço totalmente gerenciado que dimensiona automaticamente a ingestão, o armazenamento, a consulta e o alerta de métricas. Também se integra aos serviços de segurança da AWS para permitir acesso rápido e seguro aos dados. É possível usar a linguagem de consulta PromQL de código aberto para consultar suas métricas e emitir alertas sobre elas. Você também pode usar o gerenciador de alertas no Amazon Managed Service for Prometheus para configurar regras de alertas para alertas críticos. Você pode então enviar esses alertas críticos para um tópico do Amazon SNS.

Há várias opções diferentes para usar o Prometheus com o Amazon EKS:

- Você pode ativar as métricas do Prometheus ao criar um cluster do Amazon EKS pela primeira vez, ou pode criar seu próprio extrator do Prometheus para clusters existentes. Essas duas opções são abordadas neste tópico.
- Você pode implantar o Prometheus usando o Helm. Para obter mais informações, consulte the section called “Implementar usando o Helm”.
- Você pode visualizar as métricas brutas do ambiente de gerenciamento no formato do Prometheus. Para obter mais informações, consulte the section called “Ambiente de gerenciamento”.

Etapa 1: ativar as métricas do Prometheus

> **Important**
>
> Os recursos do Amazon Managed Service for Prometheus estão fora do ciclo de vida do cluster e precisam ser mantidos separados do cluster. Ao excluir seu cluster, certifique-se de excluir também todos os extratores relevantes para interromper os custos aplicáveis. Para obter mais informações, consulte Encontrar e excluir extratores no Guia do usuário do Amazon Managed Service for Prometheus.

O Prometheus descobre e coleta métricas do cluster por meio de um modelo baseado em pull, denominado extração. Os extratores são configurados para coletar dados de sua infraestrutura de cluster e aplicações conteinerizadas. Quando você ativa a opção de enviar métricas do Prometheus, o Amazon Managed Service for Prometheus fornece um extrator sem agente totalmente gerenciado.

Se você ainda não tiver criado o cluster, poderá ativar a opção de enviar métricas para o Prometheus ao criar o cluster pela primeira vez. No console do Amazon EKS, essa opção está na etapa Configurar observabilidade da criação de um novo cluster. Para obter mais informações, consulte the section called “Criar um cluster”.

Se você já tiver um cluster existente, poderá criar seu próprio extrator do Prometheus. Para fazer isso no console do Amazon EKS, navegue até a guia Observabilidade do cluster e escolha o botão Adicionar scraper. Se preferir fazer isso com a API AWS ou a AWS CLI, consulte Create a scraper no Guia do usuário do Amazon Managed Service for Prometheus.

As seguintes opções estão disponíveis ao criar o scraper com o console do Amazon EKS.

Alias do extrator

(Opcional) Insira um alias exclusivo para o extrator.

Destination (Destino)

Escolha um espaço de trabalho do Amazon Managed Service for Prometheus. Um espaço de trabalho é um espaço lógico dedicado ao armazenamento e à consulta das métricas do Prometheus. Com esse espaço de trabalho, você poderá visualizar métricas do Prometheus em todas as contas que têm acesso a ele. A opção Criar novo espaço de trabalho instrui o Amazon EKS a criar um espaço de trabalho em seu nome usando o Alias de espaço de trabalho fornecido por você. Com a opção Selecionar espaço de trabalho existente, você pode selecionar um espaço de trabalho existente em uma lista suspensa. Para obter mais informações sobre espaços de trabalho, consulte Gerenciamento de espaços de trabalho no Guia do usuário do Amazon Managed Service for Prometheus.

Acesso ao serviço

Esta seção resume as permissões que você concede ao enviar métricas do Prometheus:

- Permita que o Amazon Managed Service for Prometheus descreva o cluster do Amazon EKS extraído
- Permitir gravação remota no espaço de trabalho do Amazon Managed Service for Prometheus

Se o `AmazonManagedScraperRole` já existir, o extrator o usará. Escolha o link `AmazonManagedScraperRole` para ver os Detalhes da permissão. Se ainda não houver um `AmazonManagedScraperRole`, escolha o link Visualizar detalhes da permissão para ver as permissões específicas que você está concedendo ao enviar métricas do Prometheus.

Sub-redes

Modifique as sub-redes que o raspador herdará conforme necessário. Se você precisar adicionar uma opção de sub-rede esmaecida, volte para a etapa Criar cluster Especificar rede.

Configuração do extrator

Modifique a configuração do extrator no formato YAML conforme necessário. Para isso, use o formulário ou faça upload de um arquivo YAML substituto. Para obter mais informações, consulte Configuração do extrator no Guia do usuário do Amazon Managed Service for Prometheus.

O Amazon Managed Service for Prometheus se refere ao extrator sem agente criado com o cluster como um coletor gerenciado pela AWS. Para obter mais informações sobre coletores gerenciados

AWS, consulte Ingerir métricas com coletores gerenciados AWS no Guia do usuário do Amazon Managed Service for Prometheus.

> **Important**
>
> - Se criar um extrator do Prometheus usando a AWS CLI ou a API da AWS, você precisará ajustar sua configuração para conceder permissões ao extrator no cluster. Para obter mais informações, consulte Configurar o cluster do Amazon EKS no Guia do usuário do Amazon Managed Service for Prometheus.
> - Se você tiver um extrator do Prometheus criado antes de 11 de novembro de 2024 que use o `ConfigMap` `aws-auth` em vez das entradas de acesso, você precisará atualizá-lo para acessar as métricas adicionais do ambiente de gerenciamento de cluster do Amazon EKS. Para obter a configuração atualizada, consulte Configuração manual do Amazon EKS para acesso de extrator no Guia do usuário do Amazon Managed Service for Prometheus.

Etapa 2: usar as métricas do Prometheus

Para obter mais informações sobre como usar as métricas do Prometheus depois de ativá-las no cluster, consulte o Guia do usuário do Amazon Managed Service for Prometheus.

Etapa 3: gerenciar extratores do Prometheus

Para gerenciar scrapers, escolha a guia Observabilidade no console do Amazon EKS. Uma tabela mostra uma lista de extratores para o cluster, incluindo informações como ID, alias, status e data de criação do extrator. É possível adicionar mais extratores, editar e excluir extratores ou visualizar mais informações sobre os extratores atuais.

Para ver mais detalhes sobre um scraper, selecione o link de ID do scraper. Por exemplo, você pode visualizar o ARN, o ambiente, o ID do espaço de trabalho, o perfil do IAM, a configuração e as informações da rede. É possível usar o ID do extrator como entrada para as operações de API do Amazon Managed Service for Prometheus como `DescribeScraper`, `UpdateScraper` e `DeleteScraper`. Para obter mais informações sobre como usar a API do Prometheus, consulte a Referência da API do Amazon Managed Service for Prometheus.

Implantar o Prometheus usando o Helm

Como alternativa ao uso do Amazon Managed Service for Prometheus, é possível implantar o Prometheus no cluster com o Helm. Se você já tiver o Helm instalado, poderá verificar sua versão com o comando `helm version`. O Helm é um gerenciador de pacotes para clusters do Kubernetes. Para obter mais informações sobre o Helm e como instalá-lo, consulte the section called “Implantar aplicações com o Helm”.

Após configurar o Helm para o cluster do Amazon EKS, você poderá usá-lo para implantar o Prometheus com as etapas a seguir.

1. Crie um namespace do Prometheus.

```
kubectl create namespace prometheus
```

2. Adicione o gráfico do repositório do `prometheus-community`.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

3. Implante o Prometheus.

```
helm upgrade -i prometheus prometheus-community/prometheus \
    --namespace prometheus \
    --set alertmanager.persistence.storageClass="gp2" \
    --set server.persistentVolume.storageClass="gp2"
```

> **Note**
>
> Se você receber o erro `Error: failed to download "stable/prometheus" (hint: running helm repo update may help)` ao executar este comando, execute `helm repo update prometheus-community` e, em seguida, tente executar o comando da etapa 2 novamente.

Se você receber o erro `Error: rendered manifests contain a resource that already exists` ao executar este comando, execute `helm uninstall` *`your-release-name`* `-n` *`namespace`* e, em seguida, tente executar o comando da etapa 3 novamente.

4. Verifique se todos os pods no namespace do `prometheus` estão no estado `READY`.

```
kubectl get pods -n prometheus
```

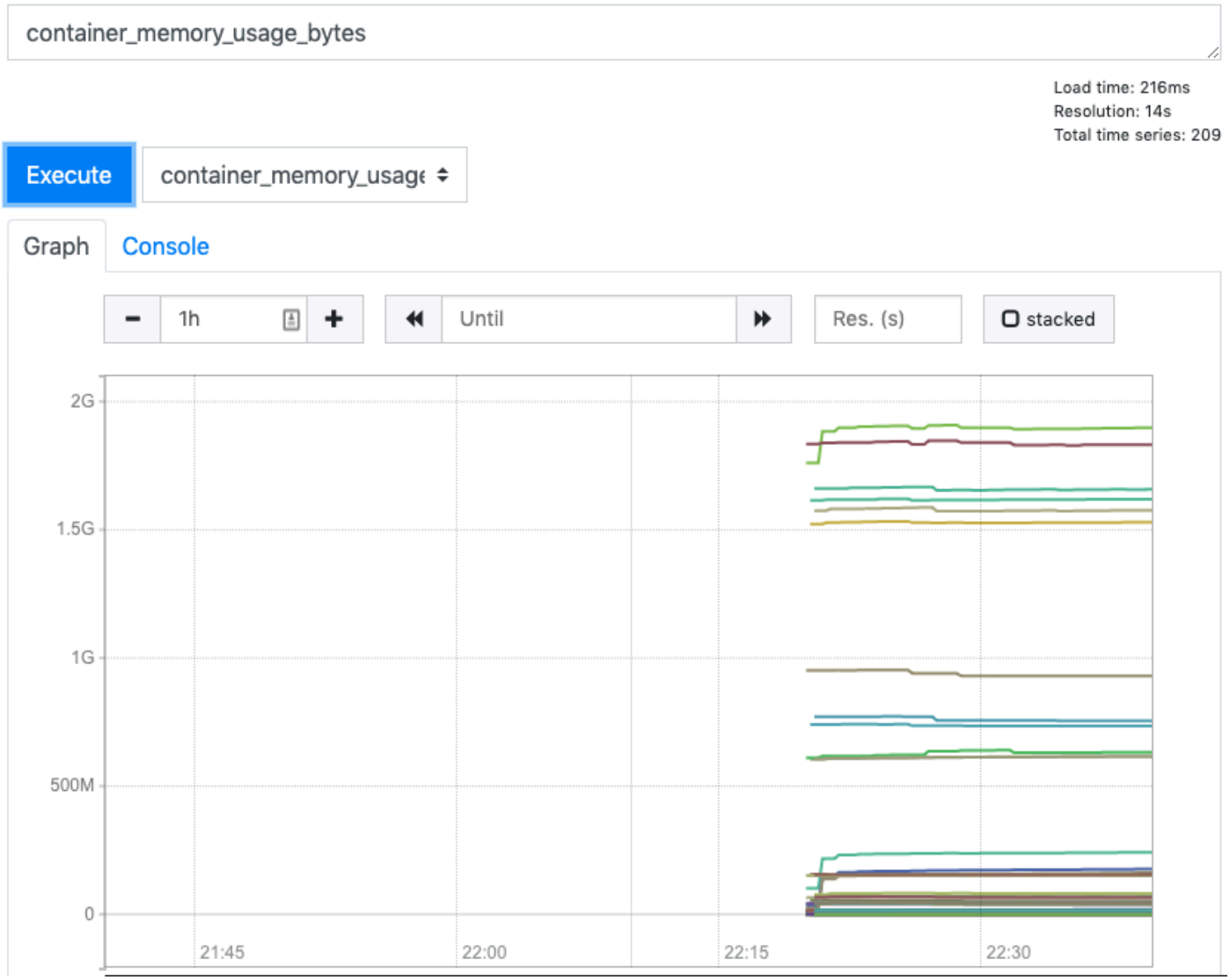
Veja um exemplo de saída abaixo.

```
NAME                                             READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-59b4c8c744-r7bgp         1/2     Running   0          48s
prometheus-kube-state-metrics-7cfd87cf99-jkz2f   1/1     Running   0          48s
prometheus-node-exporter-jcjqz                   1/1     Running   0          48s
prometheus-node-exporter-jxv2h                   1/1     Running   0          48s
prometheus-node-exporter-vbdks                   1/1     Running   0          48s
prometheus-pushgateway-76c444b68c-82tnw          1/1     Running   0          48s
prometheus-server-775957f748-mmht9               1/2     Running   0          48s
```

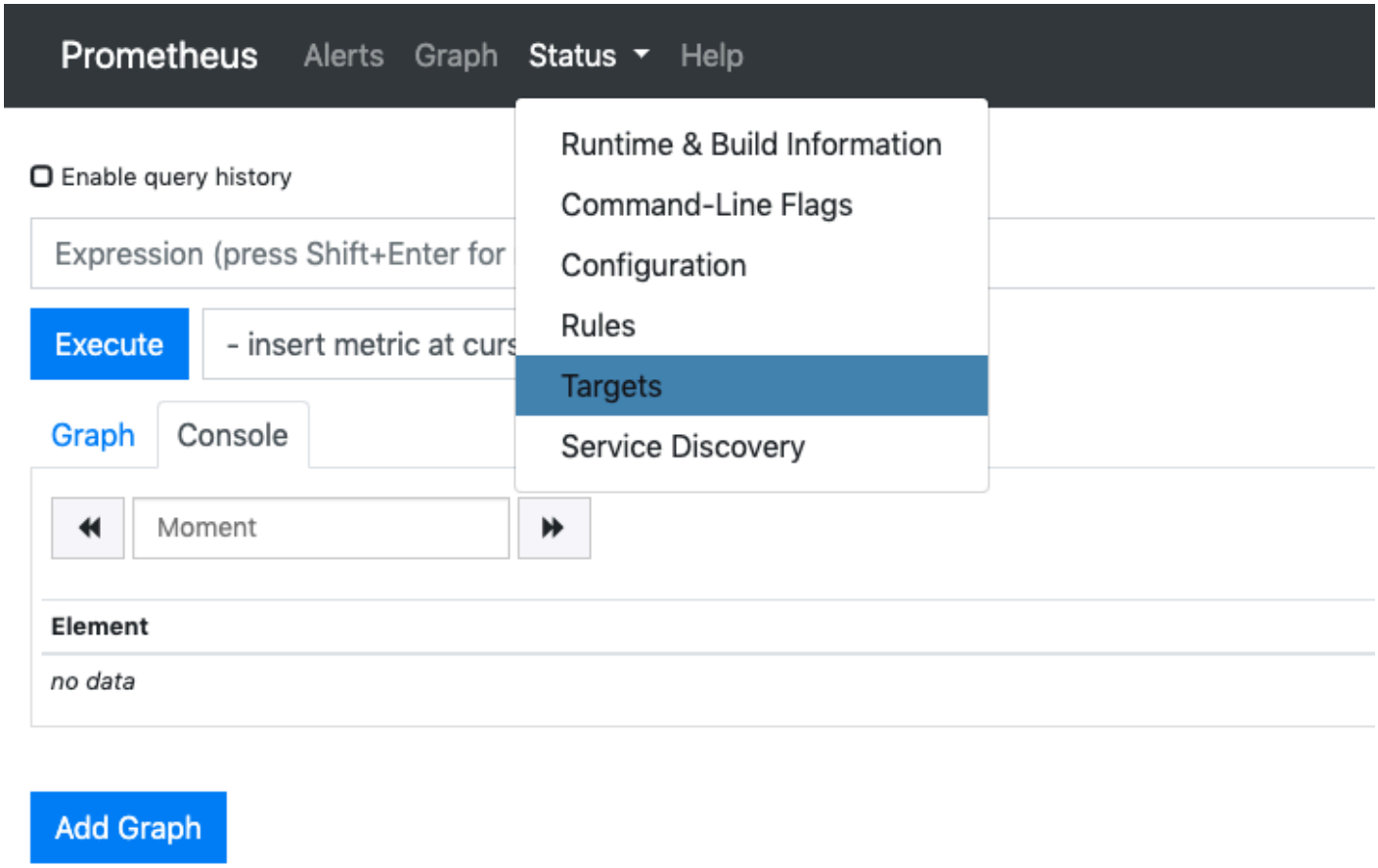
5. Use `kubectl` para fazer o redirecionamento de portas do console do Prometheus para sua máquina local.

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. Indique um navegador da web para `http://localhost:9090` para visualizar o console do Prometheus.
7. Escolha uma métrica do menu - insert metric at cursor (- inserir métrica no cursor) e selecione Execute (Executar). Selecione a guia Graph (Gráfico) para mostrar a métrica ao longo do tempo. A imagem a seguir mostra `container_memory_usage_bytes` ao longo do tempo.



8. Na barra de navegação superior, selecione Status e Targets (Destinos).



Todos os endpoints do Kubernetes que estão conectados ao Prometheus usando a descoberta de serviço são exibidos.

Obter as métricas brutas do ambiente de gerenciamento no formato do Prometheus

O ambiente de gerenciamento do Kubernetes expõe várias métricas que são representadas em um formato do Prometheus. Essas métricas são úteis para monitoramento e análise. Elas são expostas internamente por meio de endpoints de métricas e podem ser acessadas sem a implantação completa do Prometheus. No entanto, a implantação do Prometheus permite com mais facilidade a análise de métricas ao longo do tempo.

Para visualizar a saída de métricas brutas, substitua o `endpoint` e execute o comando a seguir.

```
kubectl get --raw endpoint
```

Esse comando permite passar qualquer caminho de endpoint e retorna a resposta bruta. A saída lista diferentes métricas linha por linha, com cada linha incluindo um nome de métrica, tags e um valor.

```
metric_name{tag="value"[,...]} value
```

Obter métricas do servidor de API

O endpoint do servidor de API geral é exposto no ambiente de gerenciamento do Amazon EKS. Esse endpoint é útil principalmente para observar uma métrica específica.

```
kubectl get --raw /metrics
```

Veja um exemplo de saída abaixo.

```
[...]
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
 method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

Essa saída bruta retorna textualmente o que o servidor de API expõe.

Obter métricas do ambiente de gerenciamento com **`metrics.eks.amazonaws.com`**

Para os clusters com a versão `1.28` do Kubernetes ou mais recente, o Amazon EKS também disponibiliza métricas no grupo de API `metrics.eks.amazonaws.com`. Essas métricas incluem componentes do ambiente de gerenciamento, como `kube-scheduler` e `kube-controller-manager`.

> **Note**
>
> Se você tiver uma configuração de webhook que possa bloquear a criação do novo recurso de `APIService` `v1.metrics.eks.amazonaws.com` no cluster, o recurso de endpoint de métricas talvez não esteja disponível. Você pode verificar isso no log `kube-apiserver` de auditoria pesquisando pela palavra-chave `v1.metrics.eks.amazonaws.com`.

Obter métricas do **`kube-scheduler`**

Para recuperar métricas do `kube-scheduler`, use o comando a seguir.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics"
```

Veja um exemplo de saída abaixo.

```
# TYPE scheduler_pending_pods gauge
scheduler_pending_pods{queue="active"} 0
scheduler_pending_pods{queue="backoff"} 0
scheduler_pending_pods{queue="gated"} 0
scheduler_pending_pods{queue="unschedulable"} 18
# HELP scheduler_pod_scheduling_attempts [STABLE] Number of attempts to successfully
 schedule a pod.
# TYPE scheduler_pod_scheduling_attempts histogram
scheduler_pod_scheduling_attempts_bucket{le="1"} 79
scheduler_pod_scheduling_attempts_bucket{le="2"} 79
scheduler_pod_scheduling_attempts_bucket{le="4"} 79
scheduler_pod_scheduling_attempts_bucket{le="8"} 79
scheduler_pod_scheduling_attempts_bucket{le="16"} 79
scheduler_pod_scheduling_attempts_bucket{le="+Inf"} 81
[...]
```

Obter métricas do **`kube-controller-manager`**

Para recuperar métricas do `kube-controller-manager`, use o comando a seguir.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics"
```

Veja um exemplo de saída abaixo.

```
[...]
```

```
workqueue_work_duration_seconds_sum{name="pvprotection"} 0
workqueue_work_duration_seconds_count{name="pvprotection"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-08"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-07"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-05"} 19
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.001"} 109
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.01"} 139
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.1"} 181
workqueue_work_duration_seconds_bucket{name="replicaset",le="1"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="10"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="+Inf"} 191
workqueue_work_duration_seconds_sum{name="replicaset"} 4.265655885000002
[...]
```

Noções básicas sobre as métricas do gerenciador do controlador e do programador

A tabela a seguir descreve as métricas do gerenciador do controlador e do agendador que são disponibilizadas para extração do estilo do Prometheus. Para obter mais informações sobre essas métricas, consulte Kubernetes Metrics Reference na documentação do Kubernetes.

Métrica	Componente do ambiente de gerenciamento	Descrição
scheduler_pending_pods	scheduler (programador)	O número de pods que estão aguardando para serem programados em um nó para execução.
scheduler_schedule_attempts_total	scheduler (programador)	O número de tentativas para programar pods.
scheduler_preemption_attempts_total	scheduler (programador)	O número de tentativas feitas pelo programador para programar pods de maior prioridade removendo os de menor prioridade.
scheduler_preemption_victims	scheduler (programador)	O número de pods que foram selecionados para remoção

Métrica	Componente do ambiente de gerenciamento	Descrição
		para abrir espaço para pods de maior prioridade.
scheduler_pod_scheduling_attempts	scheduler (programador)	O número de tentativas para programar com êxito um pod.
scheduler_scheduling_attempt_duration_seconds	scheduler (programador)	Indica com que rapidez ou lentidão o programador consegue encontrar um local adequado para a execução de um pod com base em vários fatores, como disponibilidade de recursos e regras de programação.
scheduler_pod_scheduling_sli_duration_seconds	scheduler (programador)	A latência de ponta a ponta de um pod que está sendo programado, a partir do momento em que o pod entra na fila de programação. Isso pode envolver várias tentativas de programação.
cronjob_controller_job_creation_skew_duration_seconds	gerenciador do controlador	O tempo entre o momento em que um cronjob está programado para ser executado e o momento em que o trabalho correspondente é criado.
workqueue_depth	gerenciador do controlador	A profundidade atual da fila.
workqueue_adds_total	gerenciador do controlador	O número total de adições tratadas pela fila de trabalho.

Métrica	Componente do ambiente de gerenciamento	Descrição
workqueue_queue_duration_seconds	gerenciador do controlador	O tempo em segundos em que um item permanece na fila de trabalho antes de ser solicitado.
workqueue_work_duration_seconds	gerenciador do controlador	O tempo em segundos que o processamento de um item da fila de trabalho leva.

Implantar um extrator do Prometheus para extrair métricas de forma consistente

Para implantar um extrator do Prometheus para extrair métricas de forma consistente, use a seguinte configuração:

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-conf
data:
  prometheus.yml: |-
    global:
      scrape_interval: 30s
    scrape_configs:
    # apiserver metrics
    - job_name: apiserver-metrics
      kubernetes_sd_configs:
      - role: endpoints
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels:
          [
            __meta_kubernetes_namespace,
```

```
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
          ]
        action: keep
        regex: default;kubernetes;https
    # Scheduler metrics
    - job_name: 'ksh-metrics'
      kubernetes_sd_configs:
      - role: endpoints
      metrics_path: /apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels:
          [
            __meta_kubernetes_namespace,
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
          ]
        action: keep
        regex: default;kubernetes;https
    # Controller Manager metrics
    - job_name: 'kcm-metrics'
      kubernetes_sd_configs:
      - role: endpoints
      metrics_path: /apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels:
          [
            __meta_kubernetes_namespace,
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
          ]
        action: keep
        regex: default;kubernetes;https
---
```

```
apiVersion: v1
kind: Pod
metadata:
  name: prom-pod
spec:
  containers:
  - name: prom-container
    image: prom/prometheus
    ports:
    - containerPort: 9090
    volumeMounts:
    - name: config-volume
      mountPath: /etc/prometheus/
  volumes:
  - name: config-volume
    configMap:
      name: prometheus-conf
```

A permissão a seguir é necessária para que o pod acesse o novo endpoint de métricas.

```
{
  "effect": "allow",
  "apiGroups": [
    "metrics.eks.amazonaws.com"
  ],
  "resources": [
    "kcm/metrics",
    "ksh/metrics"
  ],
  "verbs": [
    "get"
  ] },
```

Para aplicar patch no perfil em uso, use o comando a seguir.

```
kubectl patch clusterrole <role-name> --type=json -p='[
  {
    "op": "add",
    "path": "/rules/-",
    "value": {
      "verbs": ["get"],
      "apiGroups": ["metrics.eks.amazonaws.com"],
      "resources": ["kcm/metrics", "ksh/metrics"]
```

```
    }
  }
]'
```

Você pode então visualizar o painel do Prometheus fazendo proxy da porta do extrator do Prometheus para a sua porta local.

```
kubectl port-forward pods/prom-pod 9090:9090
```

Para seu cluster do Amazon EKS, as métricas principais do ambiente de gerenciamento do Kubernetes também são ingeridas no Amazon CloudWatch Metrics sob o namespace `AWS/EKS`. Para visualizá-las, abra o console do CloudWatch e selecione Todas as métricas no painel de navegação esquerdo. Na página de seleção de Métricas, escolha o namespace `AWS/EKS` e uma dimensão de métricas para seu cluster.

Monitorar dados de cluster com o Amazon CloudWatch

O Amazon CloudWatch é um serviço de monitoramento que coleta métricas e logs dos seus recursos na nuvem. O CloudWatch fornece algumas métricas básicas do Amazon EKS gratuitamente ao usar um novo cluster na versão `1.28` ou superior. No entanto, ao usar o CloudWatch Observability Operator como um complemento do Amazon EKS, você pode obter recursos aprimorados de observabilidade.

Métricas básicas no Amazon CloudWatch

Se o cluster estiver na versão `1.28` ou mais recente do Kubernetes, as métricas fornecidas do CloudWatch serão disponibilizadas gratuitamente no namespace do `AWS/EKS`. A tabela a seguir fornece uma lista das métricas básicas que estão disponíveis para as versões com suporte. Cada métrica listada tem um padrão de um minuto.

Nome da métrica	Descrição
`apiserver_flowcontrol_current_executing_seats`	O número de estações em uso, no momento, para a execução de solicitações de API. A alocação de estações é determinada pela configuração de priority_level e de flow_schema no recurso de prioridade e de equidade da API do Kubernetes.

Nome da métrica	Descrição
	Unidades: contagem Estatísticas válidas: número máximo
`scheduler_schedule_attempts_total`	O número total de tentativas do agendador para agendar pods no cluster para um determinado período. Essa métrica ajuda a monitorar a workload do agendador e pode indicar a pressão do agendamento ou possíveis problemas com o posicionamento do pod. Unidades: contagem Estatísticas válidas: soma
`scheduler_schedule_attempts_SCHEDULED`	O número de tentativas com êxito do agendador em agendar pods em nós no cluster por um determinado período. Unidades: contagem Estatísticas válidas: soma
`scheduler_schedule_attempts_UNSCHEDULABLE`	O número de tentativas de agendar pods que não puderam ser agendados por um determina do período devido a restrições válidas, como CPU ou memória insuficientes em um nó. Unidades: contagem Estatísticas válidas: soma

Nome da métrica	Descrição
scheduler_schedule_attempts_ERROR	O número de tentativas de agendar pods que falharam em um determinado período devido a um problema interno com o próprio agendador, como problemas de conectividade do servidor de API. Unidades: contagem Estatísticas válidas: soma
scheduler_pending_pods	O número total de pods pendentes a serem agendados pelo agendador no cluster em um determinado período. Unidades: contagem Estatísticas válidas: soma
scheduler_pending_pods_ACTIVEQ	O número de pods pendentes na activeQ, que estão aguardando para serem agendados no cluster por um determinado período. Unidades: contagem Estatísticas válidas: soma
scheduler_pending_pods_UNSCHEDULABLE	O número de pods pendentes que o agendador tentou agendar e falhou, e que estão mantidos em um estado não agendável para nova tentativa. Unidades: contagem Estatísticas válidas: soma

Nome da métrica	Descrição
scheduler_pending_pods_BACKOFF	O número de pods pendentes em backoffQ em um estado de backoff que aguardam a expiração do período de backoff. Unidades: contagem Estatísticas válidas: soma
scheduler_pending_pods_GATED	O número de pods pendentes que estão atualmente aguardando em um estado fechado, pois não podem ser agendados até que atendam às condições exigidas. Unidades: contagem Estatísticas válidas: soma
apiserver_request_total	O número de solicitações HTTP feitas em todos os servidores de API no cluster. Unidades: contagem Estatísticas válidas: soma
apiserver_request_total_4XX	O número de solicitações HTTP feitas a todos os servidores de API no cluster que resultaram em códigos de status 4XX (erro do cliente). Unidades: contagem Estatísticas válidas: soma

Nome da métrica	Descrição
apiserver_request_total_429	O número de solicitações HTTP feitas a todos os servidores de API no cluster que resultaram no código de status 429, que ocorre quando os clientes excedem os limites de taxa. Unidades: contagem Estatísticas válidas: soma
apiserver_request_total_5XX	O número de solicitações HTTP feitas a todos os servidores de API no cluster que resultaram em códigos de status 5XX (erro do servidor). Unidades: contagem Estatísticas válidas: soma
apiserver_request_total_LIST_PODS	O número de solicitações de pods LIST feitas a todos os servidores de API no cluster. Unidades: contagem Estatísticas válidas: soma
apiserver_request_duration_seconds_PUT_P99	O 99.º percentil de latência para solicitações PUT calculado com base em todas as solicitações em todos os servidores de API no cluster. Representa o tempo de resposta abaixo do qual 99% de todas as solicitações PUT são concluídas. Unidades: segundos Estatística válida: média

Nome da métrica	Descrição
apiserver_request_duration_seconds_PATCH_P99	O 99.º percentil de latência para solicitações PATCH calculado com base em todas as solicitações em todos os servidores de API no cluster. Representa o tempo de resposta abaixo do qual 99% de todas as solicitações PATCH são concluídas. Unidades: segundos Estatística válida: média
apiserver_request_duration_seconds_POST_P99	O 99.º percentil de latência para solicitações POST calculado com base em todas as solicitações em todos os servidores de API no cluster. Representa o tempo de resposta abaixo do qual 99% de todas as solicitações POST são concluídas. Unidades: segundos Estatística válida: média
apiserver_request_duration_seconds_GET_P99	O 99.º percentil de latência para solicitações GET calculado com base em todas as solicitações em todos os servidores de API no cluster. Representa o tempo de resposta abaixo do qual 99% de todas as solicitações GET são concluídas. Unidades: segundos Estatística válida: média

Nome da métrica	Descrição
apiserver_request_duration_seconds_LIST_P99	O 99.º percentil de latência para solicitações LIST calculado com base em todas as solicitações em todos os servidores de API no cluster. Representa o tempo de resposta abaixo do qual 99% de todas as solicitações LIST são concluídas. Unidades: segundos Estatística válida: média
apiserver_request_duration_seconds_DELETE_P99	O 99.º percentil de latência para solicitações DELETE calculado com base em todas as solicitações em todos os servidores de API no cluster. Representa o tempo de resposta abaixo do qual 99% de todas as solicitações DELETE são concluídas. Unidades: segundos Estatística válida: média
apiserver_current_inflight_requests_MUTATING	O número de solicitações mutantes (POST, PUT, DELETE, PATCH) atualmente sendo processadas em todos os servidores de API no cluster. Essa métrica representa solicitações que estão em andamento e ainda não concluíram o processamento. Unidades: contagem Estatísticas válidas: soma

Nome da métrica	Descrição
`apiserver_current_inflight_requests_READONLY`	O número de solicitações somente leitura (`GET`, `LIST`) que estão sendo processadas atualmente em todos os servidores de API no cluster. Essa métrica representa solicitações que estão em andamento e ainda não concluíram o processamento. Unidades: contagem Estatísticas válidas: soma
`apiserver_admission_webhook_request_total`	O número de solicitações de webhook de admissão feitas em todos os servidores de API no cluster. Unidades: contagem Estatísticas válidas: soma
`apiserver_admission_webhook_request_total_ADMIT`	O número de solicitações mutantes de webhook de admissão feitas em todos os servidores de API no cluster. Unidades: contagem Estatísticas válidas: soma
`apiserver_admission_webhook_request_total_VALIDATING`	O número de solicitações de validação de webhook de admissão feitas em todos os servidores de API no cluster. Unidades: contagem Estatísticas válidas: soma

Nome da métrica	Descrição
`apiserver_admission_webhook_rejection_count`	O número de solicitações de webhook de admissão feitas em todos os servidores de API no cluster que foram rejeitadas. Unidades: contagem Estatísticas válidas: soma
`apiserver_admission_webhook_rejection_count_ADMIT`	O número de solicitações mutantes de webhook de admissão feitas em todos os servidores de API no cluster que foram rejeitadas. Unidades: contagem Estatísticas válidas: soma
`apiserver_admission_webhook_rejection_count_VALIDATING`	O número de solicitações de validação de webhook de admissão feitas em todos os servidores de API no cluster que foram rejeitadas. Unidades: contagem Estatísticas válidas: soma
`apiserver_admission_webhook_admission_duration_seconds`	O 99.º percentil de latência para solicitações de webhook de admissão de terceiros calculado com base em todas as solicitações em todos os servidores de API no cluster. Representa o tempo de resposta abaixo do qual 99% de todas as solicitações de webhook de admissão de terceiros são concluídas. Unidades: segundos Estatística válida: média

Nome da métrica	Descrição
`apiserver_admission_webhook_admission_duration_seconds_ADMIT_P99`	O 99.º percentil de latência para solicitações mutantes de webhook de admissão de terceiros calculado com base em todas as solicitações em todos os servidores de API no cluster. Representa o tempo de resposta abaixo do qual 99% de todas as solicitações mutantes de webhook de admissão de terceiros são concluídas. Unidades: segundos Estatística válida: média
`apiserver_admission_webhook_admission_duration_seconds_VALIDATING_P99`	O 99.º percentil de latência para solicitações de validação de webhook de admissão de terceiros calculado com base em todas as solicitações em todos os servidores de API no cluster. Representa o tempo de resposta abaixo do qual 99% de todas as solicitações de validação de webhook de admissão de terceiros são concluídas. Unidades: segundos Estatística válida: média
`apiserver_storage_size_bytes`	O tamanho físico em bytes do arquivo de banco de dados de armazenamento etcd usado pelos servidores de API no cluster. Essa métrica representa o espaço em disco real alocado para o armazenamento. Unidades: bytes Estatísticas válidas: máximo

Amazon CloudWatch Observability Operator

O Amazon CloudWatch Observability coleta logs, métricas e dados de rastreamento em tempo real. Ele os envia para o Amazon CloudWatch e para o AWS X-Ray. É possível instalar este complemento para habilitar o CloudWatch Application Signals e o CloudWatch Container Insights com observabilidade aprimorada para o Amazon EKS. Isso ajuda você a monitorar a integridade e o desempenho da infraestrutura e de aplicações conteinerizadas. O Amazon CloudWatch Observability Operator foi desenvolvido para instalar e configurar os componentes necessários.

O Amazon EKS oferece suporte ao Amazon CloudWatch Observability Operator como um complemento do Amazon EKS. O complemento possibilita o uso do Container Insights em nós de processamento do Linux e do Windows no cluster. Para ativar o Container Insights no Windows, a versão do complemento do Amazon EKS deve ser `1.5.0` ou superior. No momento, o CloudWatch Application Signals não é compatível com o Windows para Amazon EKS.

Os tópicos abaixo descrevem como começar a utilizar o CloudWatch Observability Operator para o seu cluster do Amazon EKS.

- Para obter instruções sobre como instalar esse complemento, consulte Instalar o CloudWatch Agent com o complemento do EKS do Amazon CloudWatch Observability ou o chart do Help no Guia do usuário do Amazon CloudWatch.
- Para obter mais informações sobre os CloudWatch Application Signals, consulte Application Signals no Guia do usuário do Amazon CloudWatch.
- Para obter mais informações sobre o Container Insights, consulte Usar o Container Insights no Manual do usuário do Amazon CloudWatch.

Enviar logs do ambiente de gerenciamento para o CloudWatch Logs

O registro em log do plano de controle do Amazon EKS fornece logs de auditoria e diagnóstico diretamente do plano de controle do Amazon EKS para o CloudWatch Logs em sua conta. Esses logs facilitam a proteção e a execução dos clusters. Você pode selecionar os tipos exatos de logs que precisa, e eles serão enviados como fluxos de log para cada cluster do Amazon EKS no CloudWatch. Você pode usar filtros de assinatura do CloudWatch para fazer análises em tempo real nos logs ou encaminhá-los para outros serviços (os logs serão codificados em Base64 e compactados com o formato gzip). Para obter mais informações, consulte Registro em log do Amazon CloudWatch.

Você pode começar a usar o registro em log do ambiente de gerenciamento do Amazon EKS escolhendo os tipos de logs que deseja habilitar para cada cluster novo ou existente do Amazon EKS. Você pode habilitar ou desabilitar cada tipo de registro por cluster usando o Console de gerenciamento da AWS, a AWS CLI (versão `1.16.139` ou superior) ou por meio da API do Amazon EKS. Quando habilitados, os logs são enviados automaticamente do cluster do Amazon EKS para o CloudWatch Logs na mesma conta.

Ao usar o registro em log do ambiente de gerenciamento do Amazon EKS, você será cobrado pela definição de preço padrão do Amazon EKS para cada cluster executado. Você será cobrado pela ingestão de dados padrão do CloudWatch Logs e pelos custos de armazenamento de qualquer log enviado ao CloudWatch Logs pelos clusters. Você também será cobrado por qualquer recurso da AWS, como instâncias do Amazon EC2 ou volumes do Amazon EBS, que forem provisionados como parte do cluster.

Os seguintes tipos de log do ambiente de gerenciamento do cluster estão disponíveis. Cada tipo de log corresponde a um componente do plano de controle do Kubernetes. Para saber mais sobre esses componentes, consulte Componentes do Kubernetes na documentação do Kubernetes.

Servidor de API**`api`** ()

O servidor de API do cluster é o componente do ambiente de gerenciamento que expõe a API do Kubernetes. Se você ativar os logs do servidor de API ao iniciar o cluster, ou logo depois, os logs incluirão os sinalizadores do servidor de API que foram usados para iniciar o servidor de API. Para obter mais informações, consulte kube-apiserver e a política de auditoria, na documentação do Kubernetes.

Auditoria**`audit`** ()

Os logs de auditoria do Kubernetes fornecem um registro dos componentes do sistema, usuários ou administradores individuais que afetaram o cluster. Para obter mais informações, consulte Auditorias na documentação do Kubernetes.

Autenticador**`authenticator`** ()

Os logs do autenticador são exclusivos do Amazon EKS. Esses logs representam o componente do plano de controle que o Amazon EKS usa para a autenticação do Controle de acesso com base em função (RBAC – Role Based Access Control) usando credenciais do IAM. Para ter mais informações, consulte Gerenciamento de clusters.

Gerenciador do controlador**`controllerManager`** ()

O gerenciador do controlador gerencia os principais loops de controle que são fornecidos com o Kubernetes. Para obter mais informações, consulte kube-controller-manager na documentação do Kubernetes.

Agendador**`scheduler`** ()

O componente agendador gerencia quando e onde os pods são executados no cluster. Para obter mais informações, consulte kube-scheduler na documentação do Kubernetes.

Habilitar ou desabilitar os logs do ambiente de gerenciamento

Por padrão, os logs do ambiente de gerenciamento de cluster não são enviados ao CloudWatch Logs. É necessário habilitar cada tipo de log individualmente para enviar logs ao cluster. As taxas de ingestão, armazenamento de arquivamento e verificação de dados do CloudWatch Logs se aplicam aos logs do ambiente de gerenciamento habilitados. Para obter mais informações, consulte Preço do CloudWatch.

Para atualizar a configuração de registro em log do ambiente de gerenciamento, o Amazon EKS requer até cinco endereços IP disponíveis em cada sub-rede. Quando você habilita um tipo de log, os logs são enviados com um nível de verbosidade de log de 2.

É possível habilitar ou desabilitar os logs do ambiente de gerenciamento com o Console de gerenciamento da AWS ou a AWS CLI.

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Selecione o nome do cluster para exibir as informações dele.
3. Escolha a guia Observabilidade.
4. Na seção Log do ambiente de gerenciamento, escolha Gerenciar registro em log.
5. Para cada tipo de log individual, escolha se o tipo de log deve ser ativado ou desativado. Por padrão, o cada tipo de log está desativado.
6. Escolha Save changes (Salvar alterações) para terminar.

AWS CLI

1. Verifique a versão da AWS CLI com o seguinte comando.

```
aws --version
```

   Se a versão da AWS CLI for anterior a `1.16.139`, você deverá primeiro atualizar para a versão mais recente. Para instalar ou atualizar a AWS CLI, consulte Instalar AWS Command Line Interface no Guia do usuário da AWS Command Line Interface.

2. Atualize a configuração de exportação de log do ambiente de gerenciamento do cluster com o seguinte comando da CLI AWS. Substitua *my-cluster* pelo nome do cluster e especifique os valores desejados para acesso ao endpoint.

> **Note**
>
> O comando a seguir envia todos os tipos de log disponíveis ao CloudWatch Logs.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

   Veja um exemplo de saída abaixo.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
```

```
        "errors": []
    }
}
```

3. Monitore o status da atualização da configuração do log com o comando a seguir, usando o nome do cluster e o ID da atualização que foram retornados pelo comando anterior. Sua atualização estará concluída quando o status aparecer como `Successful`.

```
aws eks describe-update \
    --region region-code\
    --name my-cluster \
    --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9
```

Veja um exemplo de saída abaixo.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "Successful",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

Visualizar os logs do ambiente de gerenciamento do cluster

Depois que tiver habilitado qualquer um dos tipos de log do ambiente de gerenciamento para o cluster do Amazon EKS, você poderá visualizá-los no console do CloudWatch.

Para saber mais sobre como visualizar, analisar e gerenciar logs no CloudWatch, consulte o Manual do usuário do Amazon CloudWatch Logs.

1. Abra o console do CloudWatch. Esse link abre o console, exibe os grupos de log disponíveis atualmente e os filtra com o prefixo `/aws/eks`.
2. Escolha o cluster para o qual deseja visualizar os logs. O formato do nome do grupo de logs é `/aws/eks/my-cluster/cluster`.
3. Escolha o fluxo de logs que deseja visualizar. A lista a seguir descreve o formato do nome do fluxo de logs para cada tipo de log.

   > **Note**
   >
   > À medida que os dados do fluxo de logs aumentam, os nomes dos fluxos de log são alternados. Quando existem vários fluxos de logs para um determinado tipo de log, você pode visualizar o fluxo de logs mais recente procurando o nome do fluxo de logs com a hora do último evento mais recente.

   - Kubernetes API server component logs (**`api`**) – `kube-apiserver-1234567890abcdef01234567890abcde` (Logs do componente do servidor de API do Kubernetes)
   - Auditoria (**`audit`**) – `kube-apiserver-audit-1234567890abcdef01234567890abcde`
   - Autenticador (**`authenticator`**) – `authenticator-1234567890abcdef01234567890abcde`
   - Controller manager (**`controllerManager`**) (Gerenciador do controlador) – `kube-controller-manager-1234567890abcdef01234567890abcde`
   - Scheduler (**`scheduler`**) – `kube-scheduler-1234567890abcdef01234567890abcde` (Agendador)
4. Revise os eventos do fluxo de logs.

   Por exemplo, você deve ver os sinalizadores iniciais do servidor de API para o cluster ao visualizar a parte superior de `kube-apiserver-1234567890abcdef01234567890abcde` .

   > **Note**
   >
   > Caso não veja os logs do servidor de API no começo do fluxo de log, provavelmente o arquivo de log do servidor de API foi alternado no servidor antes de você ativar o registro em log do servidor de API no servidor. Qualquer arquivo de log que for alternado antes

que o registro em log do servidor de API seja ativado não poderá ser exportado para o CloudWatch.

No entanto, é possível criar outro cluster com a mesma versão do Kubernetes e ativar o registro em log do servidor de API ao criar o cluster. Os clusters com a mesma versão de plataforma têm os mesmos sinalizadores ativados, portanto, os sinalizadores devem corresponder aos sinalizadores do novo cluster. Ao terminar de visualizar os sinalizadores do novo cluster no CloudWatch, você poderá excluir o novo cluster.

Registre chamadas de API como eventos do AWS CloudTrail

O Amazon EKS está integrado ao AWS CloudTrail. O CloudTrail é um serviço que fornece um registro das ações realizadas por um usuário, um perfil ou um serviço da AWS no Amazon EKS. O CloudTrail captura todas as chamadas de API para o Amazon EKS como eventos. Isso inclui chamadas do console do Amazon EKS e as chamadas de código para as operações de API do Amazon EKS.

Se você criar uma trilha, poderá habilitar a entrega contínua de eventos do CloudTrail para um bucket do Amazon S3. Isso inclui eventos para o Amazon EKS. Se você não configurar uma trilha, ainda poderá visualizar os eventos mais recentes no console do CloudTrail em Histórico de eventos. Utilizando as informações coletadas pelo CloudTrail, é possível determinar vários detalhes sobre uma solicitação. Por exemplo, é possível determinar quando a solicitação foi feita ao Amazon EKS, o endereço IP de onde ela foi feita e quem a fez.

Para saber mais sobre o CloudTrail, consulte o Guia do Usuário doAWS CloudTrail.

Tópicos

- Veja referências úteis para o AWS CloudTrail
- Analise as entradas do arquivo de log do AWS CloudTrail
- Visualizar métricas para grupos do Amazon EC2 Auto Scaling

Veja referências úteis para o AWS CloudTrail

Quando você cria sua conta da AWS, o CloudTrail também está habilitado na conta da AWS. Quando ocorre qualquer atividade no Amazon EKS, essa atividade é registrada em um evento do

CloudTrail junto com outros eventos de serviços da AWS no Event history (Histórico de eventos). É possível visualizar, pesquisar e baixar eventos recentes em sua AWS conta. Para obter mais informações, consulte Viewing events with CloudTrail event history.

Para obter um registro de eventos em andamento na sua conta da AWS, incluindo eventos do Amazon EKS, crie uma trilha. Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. Por padrão, quando uma trilha é criada no console, a mesma é aplicada a todas as regiões da AWS. A trilha registra em log eventos de todas as regiões da AWS na partição da AWS e entrega os arquivos de log para o bucket do Amazon S3 especificado por você. Além disso, é possível configurar outros serviços da AWS para analisar mais ainda mais e agir com base nos dados de eventos coletados nos logs do CloudTrail. Para obter mais informações, consulte os recursos a seguir.

- Visão geral da criação de uma trilha
- Serviços e integrações compatíveis com o CloudTrail
- Configurar notificações do Amazon SNS para o CloudTrail
- Receber arquivos de log do CloudTrail de várias regiões e Receber arquivos de log do CloudTrail de várias contas

Todas as ações do Amazon EKS são registradas pelo CloudTrail e documentadas na Referência da API do Amazon EKS. Por exemplo, as chamadas para as seções CreateCluster, ListClusters e DeleteCluster geram entradas nos arquivos de log do CloudTrail.

Cada evento ou entrada de log contém informações sobre o tipo de identidade do IAM que fez a solicitação e quais credenciais foram usadas. Se forem usadas credenciais temporárias, a entrada mostrará como as credenciais foram obtidas.

Para obter mais informações, consulte Elemento userIdentity do CloudTrail.

Analise as entradas do arquivo de log do AWS CloudTrail

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log a um bucket do Amazon S3 especificado. Os arquivos de log do CloudTrail contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer origem e inclui informações sobre a ação solicitada. Isso inclui informações como a data e a hora da ação, bem como os parâmetros de solicitação que foram utilizados. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API públicas. Portanto, não são exibidos em qualquer ordem específica.

O exemplo a seguir mostra uma entrada de log do CloudTrail que demonstra a ação `CreateCluster`.


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/username",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "username"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "region-code",
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/6.4.0",
  "requestParameters": {
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-a670c2df",
        "subnet-4f8c5004"
      ]
    },
    "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
    "clusterName": "test"
  },
  "responseElements": {
    "cluster": {
      "clusterName": "test",
      "status": "CREATING",
      "createdAt": 1527535003.208,
      "certificateAuthority": {},
      "arn": "arn:aws:eks:region-code:111122223333:cluster/test",
      "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
      "version": "1.10",
      "resourcesVpcConfig": {
        "securityGroupIds": [],
        "vpcId": "vpc-21277358",
```

```
        "subnetIds": [
          "subnet-a670c2df",
          "subnet-4f8c5004"
        ]
      }
    }
  },
  "requestID": "a7a0735d-62ab-11e8-9f79-81ce5b2b7d37",
  "eventID": "eab22523-174a-499c-9dd6-91e7be3ff8e3",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

Entradas de log para funções vinculadas ao serviço do Amazon EKS

As funções vinculadas ao serviço do Amazon EKS fazem chamadas de API para recursos da AWS. Entradas de log do CloudTrail com `username: AWSServiceRoleForAmazonEKS` e `username: AWSServiceRoleForAmazonEKSNodegroup` aparecerão para chamadas feitas pelas funções vinculadas ao serviço do Amazon EKS. Para obter mais informações sobre o Amazon EKS e as funções vinculadas ao serviço, consulte the section called “Perfis vinculados a serviço”.

O exemplo a seguir mostra uma entrada de log do CloudTrail que demonstra uma ação `DeleteInstanceProfile` executada pelo perfil vinculado ao serviço `AWSServiceRoleForAmazonEKSNodegroup`, anotada no `sessionContext`.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3WHGPEZ7SJ2CW55C5:EKS",
        "arn": "arn:aws:sts::111122223333:assumed-role/
AWSServiceRoleForAmazonEKSNodegroup/EKS",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA3WHGPEZ7SJ2CW55C5",
                "arn": "arn:aws:iam::111122223333:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "accountId": "111122223333",
```

```
                "userName": "AWSServiceRoleForAmazonEKSNodegroup"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-02-26T00:56:33Z"
            }
        },
        "invokedBy": "eks-nodegroup.amazonaws.com"
    },
    "eventTime": "2020-02-26T00:56:34Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "DeleteInstanceProfile",
    "awsRegion": "region-code",
    "sourceIPAddress": "eks-nodegroup.amazonaws.com",
    "userAgent": "eks-nodegroup.amazonaws.com",
    "requestParameters": {
        "instanceProfileName": "eks-11111111-2222-3333-4444-abcdef123456"
    },
    "responseElements": null,
    "requestID": "11111111-2222-3333-4444-abcdef123456",
    "eventID": "11111111-2222-3333-4444-abcdef123456",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

Visualizar métricas para grupos do Amazon EC2 Auto Scaling

Os grupos de nós gerenciados pelo Amazon EKS têm as métricas de grupo do Amazon EC2 Auto Scaling habilitadas por padrão, sem custo adicional. O grupo do Auto Scaling envia dados de amostragem ao Amazon CloudWatch a cada minuto. Essas métricas podem ser refinadas pelo nome dos grupos do Auto Scaling. Eles fornecem visibilidade contínua do histórico do grupo do Auto Scaling que alimenta seus grupos de nós gerenciados, como alterações no tamanho do grupo ao longo do tempo. As métricas do grupo do Auto Scaling estão disponíveis no console do Auto Scaling ou no console do Amazon CloudWatch. Para obter mais informações, consulte Monitorar métricas do CloudWatch para seus grupos e instâncias do Auto Scaling.

Com a coleta de métricas de grupos do Auto Scaling, você é capaz de monitorar a escalação de grupos de nós gerenciados. As métricas do grupo do Auto Scaling informam o tamanho mínimo, máximo e desejado de um grupo do Auto Scaling. Você pode criar um alarme se o número de nós em um grupo de nós ficar abaixo do tamanho mínimo, o que indicaria um grupo de nós não íntegro.

O rastreamento do tamanho do grupo de nós também é útil para ajustar o número máximo para a capacidade do plano de dados não se esgote.

Se você preferir que essas métricas não sejam coletadas, poderá optar por desabilitar todas ou apenas algumas delas. Por exemplo, você pode fazer isso para evitar ruídos nos seus painéis do CloudWatch. Para obter mais informações, consulte Métricas de uso do Amazon CloudWatch para Amazon EC2 Auto Scaling.

Enviar dados de métricas e rastreamento com o ADOT Operator

O Amazon EKS suporta o uso do Console de gerenciamento da AWS, AWS CLI e Amazon EKS API para instalar e gerenciar o AWS Distro for OpenTelemetry (ADOT) Operator. Isso torna mais fácil habilitar suas aplicações em execução no Amazon EKS para enviar dados de métrica e rastreamento para várias opções de serviços de monitoramento, como Amazon CloudWatch, Prometheus e X-Ray.

Para obter mais informações, consulte Getting Started with AWS Distro for OpenTelemetry using EKS Add-Ons na documentação do AWS Distro para OpenTelemetry.

Visualizar dados agregados sobre recursos de cluster com o painel do EKS

image::images/eks-dashboard.png[screenshot of account level cluster metrics]

O que é o painel do Amazon EKS?

O painel do Amazon EKS fornece visibilidade consolidada de seus clusters do Kubernetes em várias regiões da AWS e contas da AWS. Com esse painel, você pode:

- Rastrear clusters programados para atualizações automáticas de fim de suporte nos próximos 90 dias.
- Projetar custos do ambiente de gerenciamento do EKS para clusters em suporte estendido.
- Revisar clusters com insights que precisam de atenção antes da atualização.
- Identificar grupos de nós gerenciados que executam versões específicas da AMI.
- Monitorar a distribuição do tipo de suporte ao cluster (suporte padrão comparado com suporte estendido).

O painel do EKS se integra ao EKS Cluster Insights para revelar problemas com os clusters, como o uso de APIs obsoletas do Kubernetes. Para obter mais informações, consulte the section called "Insights de cluster".

> **Note**
>
> O painel do EKS não é em tempo real e é atualizado a cada 12 horas. Para monitoramento de clusters em tempo real, consulte Monitorar clusters

Como o painel usa o AWS Organizations?

O painel do Amazon EKS requer a integração com o AWS Organizations para operacionalidade. Ele aproveita o AWS Organizations para coletar com segurança as informações de clusters em todas as contas. Essa integração fornece gerenciamento e governança centralizados à medida que sua infraestrutura da AWS escala.

Se o AWS Organizations não estiver habilitado para sua infraestrutura, consulte o Guia do usuário do AWS Organizations para obter instruções de configuração.

Acesso entre contas e entre regiões

O painel do EKS pode ver os recursos do cluster em qualquer conta que seja membro da organização da AWS. Para gerar uma lista de contas da AWS em sua organização, consulte Export details for all accounts in an organization.

A região us-east-1 da AWS gera o painel. Você deve fazer login nessa região para ver o painel. O painel agrega dados em regiões da AWS, mas isso não inclui as regiões `GovCloud` e da China.

Principais termos

- Organização da AWS: uma estrutura de gerenciamento unificada para várias contas da AWS.
- Conta gerencial: a conta principal que controla a organização da AWS.
- Conta de membro: qualquer conta dentro da organização, exceto a conta gerencial.
- Administrador delegado: uma conta de membro à qual são concedidas permissões administrativas específicas entre contas. Na conta gerencial, você pode selecionar uma conta de administrador delegado por serviço da AWS.

- Acesso confiável: autorização para que o painel do EKS acesse as informações de clusters em contas organizacionais.
- Perfil vinculado ao serviço (SLRs): é um tipo exclusivo de perfil do IAM diretamente vinculado a um serviço da AWS. O painel do EKS usa um SLR para ler informações sobre suas contas e organizações.
- Para obter mais informações, consulte Terminology and concepts no Guia do usuário do AWS Organizations.

Visão geral

1. Acesse a conta gerencial da sua organização da AWS.
   - As etapas para acessar a conta gerencial dependem de como você configurou sua organização da AWS. Por exemplo, você pode acessar a conta gerencial pelo Centro de Identidade da AWS ou pelo Okta.
2. Habilite o acesso confiável no console do EKS.
3. Atribua um administrador delegado usando o ID da conta da AWS.
4. Mude para a conta de administrador delegado.
5. Acesse o console aprimorado do EKS com visibilidade em toda a organização.

Habilitar o painel do EKS usando o Console da AWS

> **Important**
>
> Você deve fazer login na conta gerencial da sua organização da AWS para habilitar o painel do EKS.

Acessar as configurações do painel do EKS

1. Confirme o seguinte:
   a. Você tem o AWS Organizations habilitado e configurado.
   b. Você fez login na conta gerencial da organização.
   c. Você está visualizando o Console de Gerenciamento da AWS na região us-east-1.
2. Navegue até o console do EKS.

3. Na barra lateral esquerda, abra as configurações do painel.

Configurar o acesso ao painel do Amazon EKS

1. Localize o ID da conta da AWS da conta da AWS que você deseja permitir que visualize o painel do EKS.
   a. Esta etapa é opcional, mas sugerida. Se você não fizer isso, só poderá acessar o painel na conta gerencial. Como prática recomendada, você deve limitar o acesso à conta gerencial.
2. Clique em Habilitar acesso confiável.
   a. Você agora pode visualizar o painel na conta gerencial.
3. Clique em Registrar administrador delegado e insira o ID da conta da conta da AWS que você usará para visualizar o painel.
   a. Você agora pode visualizar o painel na conta de administrador delegado ou na conta gerencial.

Para obter informações sobre as permissões necessárias para habilitar o painel, consulte Minimum IAM policies required.

Visualizar o painel do EKS

1. Faça login na conta de administrador delegado (sugerido) ou na conta gerencial.
2. Faça login na região us-east-1.
3. Vá para o serviço do EKS e selecione Painel na barra lateral esquerda.

> **Note**
>
> Analise as permissões do IAM necessárias para visualizar o painel do EKS.

Configurar o painel

Você pode configurar a visualização do painel e filtrar recursos.

Recursos disponíveis

- Clusters: visualize as informações agregadas sobre o status e a localização dos clusters do EKS.

  - Clusters com problemas de integridade.
  - Clusters no suporte estendido do EKS.
  - Detalhamento dos clusters por versão do Kubernetes.
- Grupos de nós gerenciados: analise os grupos de nós gerenciados e as instâncias do EC2.
  - Grupos de nós por tipo de AMI, como Amazon Linux ou Bottlerocket.
  - Problemas de integridade do grupo de nós.
  - Distribuição de tipos de instâncias.
- Complementos: saiba mais sobre quais complementos do Amazon EKS você instalou e o status deles.
  - Número de instalações por complemento.
  - Complementos com problemas de integridade.
  - Distribuição de versões por complemento.

Visualizações disponíveis

- Visualização em grafo
  - Uma visualização de widget personalizável exibindo grafos e visualizações do recurso selecionado.
  - As alterações na visualização em grafo, como a remoção de um widget, são visíveis para todos os usuários do painel do EKS.
- Visualização de recurso
  - Uma visualização em lista do recurso selecionado, com filtros.
- Visualização em mapa
  - Visualize a distribuição geográfica do recurso selecionado.

Filtrar o painel do EKS

Você pode filtrar o painel do EKS por:

- AWSConta da
- Unidade organizacional, definida pelo AWS Organizations
- Região da AWS

Desabilitar o painel do EKS usando o Console da AWS

1. Confirme o seguinte:
   a. Você tem o AWS Organizations habilitado e configurado.
   b. Você fez login na conta gerencial da organização.
   c. Você está visualizando o Console de Gerenciamento da AWS na região us-east-1.
2. Navegue até o console do EKS.
3. Na barra lateral esquerda, abra as configurações do painel.
4. Clique em Desabilitar acesso confiável.

Solucionar problemas do painel do EKS

Problema ao habilitar o painel do EKS

- Você deve fazer login na conta gerencial de uma organização da AWS.
  - Se você ainda não tem uma organização da AWS, crie uma. Saiba como criar e configurar uma organização.
  - Se sua conta da AWS já for membro de uma organização da AWS, identifique o administrador da organização.
- Você deve fazer login na conta da AWS com permissões suficientes do IAM para criar e atualizar recursos do AWS Organizations.

Problema ao visualizar o painel do EKS

- Você deve fazer login em uma das seguintes contas da AWS:
  - A conta gerencial da organização da AWS
  - Uma conta de administrador delegado, identificada nas configurações do painel do EKS da conta gerencial.
- Se você habilitou recentemente o painel do EKS, observe que o preenchimento inicial de dados pode levar até 12 horas.
- Tente reabilitar o painel usando a CLI, incluindo a criação do perfil vinculado ao serviço.

Widgets do painel se movem inesperadamente

- O painel do EKS salva a visualização configurável do widget no nível da conta da AWS. Se você alterar a visualização do widget, outras pessoas usando a mesma conta da AWS verão as alterações.

Configurar a integração do painel do EKS com o AWS Organizations

Esta seção fornece instruções passo a passo para configurar a integração do painel do EKS com o AWS Organizations. Você aprenderá como habilitar e desabilitar o acesso confiável entre serviços, bem como registrar e cancelar o registro de contas de administrador delegado. Cada tarefa de configuração pode ser executada usando o Console da AWS ou a AWS CLI.

Habilitar acesso confiável

O acesso confiável autoriza que o painel do EKS acesse com segurança as informações de clusters em todas as contas da sua organização.

Usar o console de AWS

1. Faça login na conta gerencial da sua organização da AWS.
2. Navegue até o console do EKS na região us-east-1.
3. Na barra lateral esquerda, selecione Configurações do painel.
4. Clique em Habilitar acesso confiável.

> **Note**
>
> Quando você habilita o acesso confiável pelo console do EKS, o sistema cria automaticamente o perfil `AWSServiceRoleForAmazonEKSDashboard` vinculado ao serviço. Essa criação automática não ocorrerá se você habilitar o acesso confiável usando a AWS CLI ou o console do AWS Organizations.

Usar a CLI do AWS

1. Faça login na conta gerencial da sua organização da AWS.
2. Execute os seguintes comandos:

```
aws iam create-service-linked-role --aws-service-name dashboard.eks.amazonaws.com
aws organizations enable-aws-service-access --service-principal eks.amazonaws.com
```

Desabilitar o acesso confiável

Desabilitar o acesso confiável revoga a permissão do painel do EKS para acessar informações do cluster nas contas da sua organização.

Usar o console de AWS

1. Faça login na conta gerencial da sua organização da AWS.
2. Navegue até o console do EKS na região us-east-1.
3. Na barra lateral esquerda, selecione Configurações do painel.
4. Clique em Desabilitar acesso confiável.

Usar a CLI do AWS

1. Faça login na conta gerencial da sua organização da AWS.
2. Execute o seguinte comando:

```
aws organizations disable-aws-service-access --service-principal eks.amazonaws.com
```

Habilitar uma conta de administrador delegado

Um administrador delegado é uma conta de membro que tem permissão para acessar o painel do EKS.

Usar o console de AWS

1. Faça login na conta gerencial da sua organização da AWS.
2. Navegue até o console do EKS na região us-east-1.
3. Na barra lateral esquerda, selecione Configurações do painel.
4. Clique em Registrar administrador delegado.
5. Insira o ID da conta da conta da AWS que deseja escolher como administrador delegado.

6. Confirme o registro.

Usar a CLI do AWS

1. Faça login na conta gerencial da sua organização da AWS.
2. Execute o comando abaixo, substituindo `123456789012` pelo ID da sua conta:

```
aws organizations register-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

Desabilitar uma conta de administrador delegado

Desabilitar um administrador delegado remove a permissão da conta para acessar o painel do EKS.

Usar o console de AWS

1. Faça login na conta gerencial da sua organização da AWS.
2. Navegue até o console do EKS na região us-east-1.
3. Na barra lateral esquerda, selecione Configurações do painel.
4. Localize o administrador delegado na lista.
5. Clique em Cancelar registro ao lado da conta que deseja remover como administrador delegado.

Usar a CLI do AWS

1. Faça login na conta gerencial da sua organização da AWS.
2. Execute o comando a seguir, substituindo `123456789012` pelo ID da conta do administrador delegado:

```
aws organizations deregister-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

Políticas mínimas do IAM necessárias

Esta seção descreve as políticas mínimas do IAM necessárias para habilitar o acesso confiável e delegar um administrador para a integração do painel do EKS com o AWS Organizations.

Política para habilitar o acesso confiável

Para habilitar o acesso confiável entre o painel do EKS e o AWS Organizations, você precisa das seguintes permissões:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:DescribeOrganization",
                "organizations:ListAWSServiceAccessForOrganization"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
dashboard.eks.amazonaws.com/AWSServiceRoleForAmazonEKSDashboard"
        }
    ]
}
```

Política para delegar um administrador

Para registrar ou cancelar o registro de um administrador delegado para o painel do EKS, você precisa das seguintes permissões:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "organizations:RegisterDelegatedAdministrator",
                "organizations:DeregisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Política para visualizar o painel do EKS

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonEKSDashboardReadOnly",
            "Effect": "Allow",
            "Action": [
                "eks:ListDashboardData",
                "eks:ListDashboardResources",
                "eks:DescribeClusterVersions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonOrganizationsReadOnly",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListRoots",
                "organizations:ListAccountsForParent",
                "organizations:ListOrganizationalUnitsForParent"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonOrganizationsDelegatedAdmin",
            "Effect": "Allow",
            "Action": [
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "organizations:ServicePrincipal": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Note**
>
> Essas políticas devem ser anexadas à entidade principal do IAM (usuário ou perfil) na conta gerencial da sua organização da AWS. As contas de membro não podem habilitar o acesso confiável nem delegar administradores.

Melhorar o EKS com serviços integrados da AWS

Além dos serviços abordados em outras seções, o Amazon EKS trabalha com mais serviços da AWS para fornecer soluções adicionais. Este tópico identifica alguns dos outros serviços que usam o Amazon EKS para adicionar funcionalidades ou serviços que o Amazon EKS usa para executar tarefas.

Tópicos

- Backup dos clusters do EKS com o AWS Backup
- Crie recursos do Amazon EKS com o AWS CloudFormation
- Estabelecimento de conexão com repositórios do Git usando o AWS CodeConnections
- Analisar eventos de segurança no EKS com o Amazon Detective
- Detectar ameaças com o Amazon GuardDuty
- Executar clusters do EKS de baixa latência com zonas locais da AWS
- Avalie a resiliência do cluster do EKS com o AWS Resilience Hub
- Gerenciamento de segredos de aplicações com o AWS Secrets Manager
- Centralizar e analisar os dados de segurança do EKS com o Security Lake
- Habilitar a conectividade segura entre clusters com o Amazon VPC Lattice

Backup dos clusters do EKS com o AWS Backup

O AWS Backup oferece suporte a backups de clusters do Amazon EKS, incluindo o estado do cluster do Kubernetes e o armazenamento persistente anexado ao cluster do EKS por meio de uma solicitação de volumes persistentes, que conta com volumes do EBS, sistemas de arquivos do EFS e buckets do S3. Um backup do Amazon EKS criará um ponto de recuperação composto, no qual haverá um ponto de recuperação secundário para cada recurso incluído no backup.

- Como fazer backup de recursos: Conceitos básicos do AWS Backup
- Criação de backup por tipo de recurso: Backups do Amazon EKS
- Como restaurar clusters do Amazon EKS: Restauração de backups do Amazon EKS

Para começar a usar pelo console do EKS, pelo console do AWS Backup ou pela CLI, certifique-se de que o seu perfil do IAM conta com as seguintes permissões:

- Estas permissões podem ser encontradas na política gerenciada do AWS Backup, AWSBackupServiceRolePolicyForBackup. Ela contém as permissões necessárias para fazer backup do cluster do Amazon EKS e do armazenamento persistente do EBS e do EFS.
- Se o cluster do EKS contiver um bucket do S3, será necessário garantir que as seguintes políticas e pré-requisitos para o bucket do S3 estejam adicionados e habilitados, conforme documentado:
- AWSBackupServiceRolePolicyForS3Backup
- Pré-requisitos para backups do S3

Certifique-se de que os clusters do EKS tenham as seguintes configurações:

- O modo de autorização do cluster do EKS deve estar definido como API ou API_AND_CONFIG_MAP para que o AWS Backup possa criar entradas de acesso para acessar o cluster do EKS.

Crie recursos do Amazon EKS com o AWS CloudFormation

O Amazon EKS está integrado ao AWS CloudFormation, um serviço que o ajuda a modelar e configurar seus recursos do AWS para que você possa gastar menos tempo criando e gerenciando seus recursos e sua infraestrutura. Você cria um modelo que descreve todos os recursos do AWS que deseja, por exemplo, um cluster do Amazon EKS, e o AWS CloudFormation se encarrega de provisionar e configurar esses recursos para você.

Ao usar o AWS CloudFormation, você pode reutilizar o modelo para configurar os recursos do Amazon EKS de forma consistente e repetida. Basta descrever seus recursos uma vez e, depois, provisionar os mesmos recursos repetidamente em várias contas e regiões da AWS.

Modelos do Amazon EKS e do AWS CloudFormation

Para provisionar e configurar recursos para o Amazon EKS e serviços relacionados, você deve entender modelos do AWS CloudFormation. Os modelos são arquivos de texto formatados em JSON ou YAML. Esses modelos descrevem os recursos que você deseja provisionar nas pilhas do AWS CloudFormation. Se você não estiver familiarizado com JSON ou YAML, poderá usar o AWS CloudFormation Designer para ajudá-lo a começar a usar os modelos do AWS CloudFormation. Para obter mais informações, consulte O que é o AWS CloudFormation Designer? no Guia do Usuário do AWS CloudFormation.

O Amazon EKS oferece suporte à criação de clusters e grupos de nós em AWS CloudFormation. Para obter mais informações, incluindo exemplos de modelos JSON e YAML para seus recursos do Amazon EKS, consulte a referência de tipo de recurso do Amazon EKS no Guia do usuário do AWS CloudFormation.

Saiba mais sobre o AWS CloudFormation

Para saber mais sobre o AWS CloudFormation, consulte os seguintes recursos:

- AWS CloudFormation
- AWS Guia do usuário do CloudFormation
- AWS Guia do usuário da interface de linha de comando do CloudFormation

Estabelecimento de conexão com repositórios do Git usando o AWS CodeConnections

O AWS CodeConnections fornece uma forma segura para o estabelecimento de conexão entre os serviços da AWS e os repositórios de código-fonte de terceiros. O AWS CodeConnections é compatível com GitHub, GitLab, Bitbucket e outros provedores. Para saber mais informações e começar a usar, consulte Como trabalhar com conexões.

Uso do AWS CodeConnections com o Argo CD

Ao empregar a funcionalidade do EKS no Argo CD, você pode optar por usar o AWS CodeConnections para habilitar autenticação segura em repositórios do Git, sem a necessidade de gerenciar credenciais de longa duração ou tokens de acesso pessoal. O AWS CodeConnections gerencia o fluxo de autenticação OAuth e realiza a manutenção da conexão com o provedor Git, oferecendo uma abordagem segura e gerenciável para acessar os repositórios de GitOps e manifestos de aplicações armazenados em provedores Git de terceiros.

Pré-requisitos

- Um cluster do Amazon EKS com a funcionalidade para o Argo CD criada
- Uma conexão criada no AWS CodeConnections para o provedor Git
- Permissões do IAM configuradas para que o Argo CD use a conexão

Como configurar o CodeConnections para acesso a repositórios do Argo CD

1. Crie uma conexão no console do CodeConnections:
   a. Abra o console do CodeConnections.
   b. Escolha Criar conexão.
   c. Selecione seu provedor (GitHub, GitLab ou Bitbucket) e siga o fluxo de autenticação.
   d. Anote o ARN da conexão para uso na configuração do Argo CD.
2. Garanta que o perfil da funcionalidade para o Argo CD tenha permissões para usar a conexão com uma política baseada em recursos:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection",
        "codeconnections:GetConnection"
      ],
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-
id"
    }
  ]
}
```

3. Configure o Argo CD para referenciar o recurso CodeConnections ao adicionar um repositório, usando o endpoint do recurso CodeConnections como o URL do repositório. A funcionalidade do Argo CD usa a conexão para realizar a autenticação no seu provedor Git sem precisar de credenciais de longa duração.

Considerações para usar o CodeConnections com o Argo CD

Ao usar o AWS CodeConnections com a funcionalidade do EKS para o Argo CD, lembre-se dos seguintes pontos:

- A conexão do CodeConnections deve estar na mesma região da AWS que o cluster do EKS
- O perfil da funcionalidade para o Argo CD deve ter as permissões `codeconnections:UseConnection` e `codeconnections:GetConnection`
- O CodeConnections gerencia automaticamente o fluxo OAuth e o ciclo de vida da credencial

Para obter mais informações sobre como configurar o acesso a repositórios com o Argo CD, consulte the section called “Configuração dos repositórios”.

Analisar eventos de segurança no EKS com o Amazon Detective

O Amazon Detective ajuda a analisar, investigar e identificar rapidamente a causa raiz de descobertas de segurança ou atividades suspeitas. O Detective coleta automaticamente dados de log dos seus recursos da AWS. Em seguida, ele usa machine learning, análises estatísticas e a teoria de grafos para gerar visualizações que ajudam a realizar investigações de segurança eficazes com maior rapidez. O Detective faz a pré-construção de agregações de dados, resumos e contexto predefinidos que podem ajudar você a analisar e determinar a natureza e a extensão de possíveis problemas de segurança. Para obter mais informações, consulte o Guia do usuário do Amazon Detective.

O Detective organiza dados da AWS e do Kubernetes em descobertas como:

- Detalhes do cluster do Amazon EKS, incluindo a identidade IAM que criou o cluster e o perfil de serviço do cluster. Você pode investigar a atividade de API do Kubernetes e da AWS dessas identidades do IAM com o Detective.
- Detalhes do contêiner, como a imagem e o contexto de segurança. Também é possível revisar detalhes de pods encerrados.
- A atividade de API do Kubernetes, incluindo tendências gerais na atividade de API e detalhes sobre chamadas de API específicas. Por exemplo, você pode mostrar o número de chamadas de API do Kubernetes com êxito e com falha que foram emitidas durante um intervalo de tempo selecionado. Além disso, a seção sobre chamadas de API recém-observadas pode ser útil para identificar atividades suspeitas.

Logs de auditoria do Amazon EKS são um pacote de origem de dados opcional que pode ser adicionado ao seu gráfico de comportamento do Detective. Você pode visualizar os pacotes de origem opcionais disponíveis e seu status na sua conta. Para obter mais informações, consulte Logs de auditoria do Amazon EKS para Detective e o Guia do usuário do Amazon Detective.

Usar o Amazon Detective com o Amazon EKS

Antes de poder analisar as descobertas, o Detective deve estar ativado por pelo menos 48 horas na mesma região AWS em que o cluster está. Para obter mais informações, consulte Configurar o Amazon Detective no Guia do usuário do Amazon Detective.

1. Abra o console do Detective em https://console.aws.amazon.com/detective/.
2. No painel de navegação à esquerda, selecione Search (Pesquisar).
3. Selecione Choose type (Escolher tipo) e, em seguida, selecione EKS cluster.
4. Insira o nome do cluster ou ARN e escolha Search (Pesquisar).
5. Nos resultados da pesquisa, escolha o nome do cluster cujas atividades deseja visualizar. Para obter mais informações sobre o que você pode visualizar, consulte Atividade geral da API do Kubernetes envolvendo um cluster do Amazon EKS no Guia do usuário do Amazon Detective.

Detectar ameaças com o Amazon GuardDuty

O Amazon GuardDuty é um serviço de detecção de ameaças que ajuda a proteger contas, contêineres, workloads e dados no ambiente da AWS. Usando modelos de machine learning (ML) e recursos de detecção de anomalias e ameaças, o GuardDuty monitora continuamente diferentes fontes de log e atividades de runtime para identificar e priorizar possíveis riscos de segurança e atividades maliciosas no seu ambiente.

Entre outros recursos, o GuardDuty oferece os dois recursos a seguir que detectam possíveis ameaças aos seus clusters do EKS: Proteção do EKS e Monitoramento de runtime.

> **Note**
>
> Novo: o Modo Automático do Amazon EKS se integra ao GuardDuty.

Proteção do EKS

Esse recurso fornece cobertura de detecção de ameaças para ajudar você a proteger clusters do Amazon EKS por meio do monitoramento dos logs de auditoria associados do Kubernetes. Os logs de auditoria do Kubernetes capturam ações sequenciais no cluster, incluindo atividades de usuários e aplicações usando a API do Kubernetes e o ambiente de gerenciamento. Por exemplo, o GuardDuty pode identificar que APIs chamadas para potencialmente adulterar recursos em um cluster do Kubernetes foram invocadas por um usuário não autenticado.

Quando a Proteção do EKS estiver habilitada, o GuardDuty poderá acessar seus logs de auditoria do Amazon EKS somente para detecção contínua de ameaças. Se o GuardDuty identificar uma possível ameaça ao cluster, ele vai gerar uma descoberta de log de auditoria associado do

Kubernetes de um tipo específico. Para obter mais informações sobre os tipos de descobertas disponíveis nos logs de auditoria do Kubernetes, consulte Kubernetes audit logs finding types no Guia do usuário do Amazon GuardDuty.

Para obter mais informações, consulte Proteção do EKS no Guia do usuário do Amazon GuardDuty.

Monitoramento de runtime

Esse recurso monitora e analisa eventos em nível de sistema operacional, rede e arquivos para ajudar você a detectar possíveis ameaças em workloads do AWS específicas do seu ambiente.

Quando você habilita o Monitoramento de runtime e instala o agente do GuardDuty em seus clusters do Amazon EKS, o GuardDuty começa a monitorar os eventos de runtime associados a esse cluster. Observe que o agente do GuardDuty e o Monitoramento de runtime não estão disponíveis para o Amazon EKS Hybrid Nodes, portanto, o Monitoramento de runtime não está disponível para eventos de runtime que ocorrem nos nós híbridos. Se o GuardDuty identificar uma possível ameaça ao seu cluster, ele gerará uma descoberta de Monitoramento de runtime. Por exemplo, uma ameaça pode começar comprometendo um único contêiner que executa uma aplicação Web vulnerável. Essa aplicação Web pode ter permissões de acesso aos contêineres e workloads subjacentes. Nesse cenário, credenciais configuradas incorretamente podem levar a um acesso mais amplo à conta e aos dados nela armazenados.

Para configurar o Monitoramento de runtime, instale o agente do GuardDuty no seu cluster como um complemento do Amazon EKS. Para obter mais informações sobre o complemento, consulte the section called “ Complementos da AWS”.

Para obter mais informações, consulte Monitoramento de runtime no Guia do usuário do Amazon GuardDuty.

Executar clusters do EKS de baixa latência com zonas locais da AWS

Uma Zona local da AWS é uma extensão de uma região da AWS na proximidade geográfica de seus usuários. As zonas locais têm suas próprias conexões com a Internet e oferecem suporte ao AWS Direct Connect. Os recursos criados em uma zona local podem atender usuários locais com comunicações de latência baixa. Para obter mais informações, consulte o Guia do usuário de zonas locais da AWS e Zonas locais no Guia do usuário do Amazon EC2.

O Amazon EKS é compatível com determinados recursos em zonas locais. Isso inclui grupos de nós gerenciados, nós do Amazon EC2 autogerenciados, volumes do Amazon EBS e Application Load Balancers (ALBs). Recomendamos que você considere o seguinte quando usar zonas locais como parte do cluster do Amazon EKS:

- Você não pode criar grupos de nós do Fargate em zonas locais com o Amazon EKS.
- O plano de controle do Kubernetes gerenciado pelo Amazon EKS é sempre executado na região da AWS. O ambiente de gerenciamento do Kubernetes gerenciado pelo Amazon EKS não pode ser executado na zona local. Como o Local Zones aparece como uma sub-rede na VPC, o Kubernetes considera os recursos da zona local como parte dessa sub-rede.
- O cluster do Amazon EKS Kubernetes se comunica com as instâncias do Amazon EC2 que você executa na região ou zona local da AWS e usando as interfaces de rede elástica gerenciadas pelo Amazon EKS. Para saber mais sobre a arquitetura de rede do Amazon EKS, consulte Configurar redes.
- Ao contrário das sub-redes regionais, o Amazon EKS não pode colocar interfaces de rede nas sub-redes da zona local. Isso significa que você não deve especificar sub-redes da zona local ao criar o cluster. No entanto, você pode ter nós de processamento em diversas zonas locais conectados ao mesmo cluster.

Avalie a resiliência do cluster do EKS com o AWS Resilience Hub

O Hub de resiliência da AWS Resilience Hub avalia a resiliência de um cluster do Amazon EKS analisando sua infraestrutura. AWS O Resilience Hub usa a configuração de controle de acesso baseado em perfil (RBAC) do Kubernetes para avaliar as workloads do Kubernetes implementadas no cluster. Para obter mais informações, consulte Habilitação do acesso do AWS Resilience Hub ao seu cluster do Amazon EKS no Guia do Usuário do AWS Resilience Hub.

Gerenciamento de segredos de aplicações com o AWS Secrets Manager

O AWS Secrets Manager permite gerenciar, acessar e rotacionar credenciais, chaves de API e outros segredos durante todo o ciclo de vida. Com o Secrets Manager, é possível proteger e gerenciar os segredos usados para acessar recursos na AWS, em serviços de terceiros e no ambiente on-premises. Para obter mais informações, consulte o Guia do usuário do AWS Secrets Manager.

Ao usar a funcionalidade do EKS para o Argo CD, o Secrets Manager fornece uma forma segura de armazenar e recuperar credenciais de repositórios do Git, sem a inclusão de dados sensíveis diretamente na configuração e nos recursos do Argo CD. Essa integração é particularmente vantajosa para gerenciar tokens de acesso a repositórios privados e chaves SSH que o Argo CD usa para sincronizar aplicações de repositórios do Git.

Uso do AWS Secrets Manager com o Argo CD

Ao usar a funcionalidade do EKS para o Argo CD, é possível armazenar credenciais de repositórios do Git no Secrets Manager e configurar o Argo CD para recuperá-las. Essa abordagem é mais segura do que armazenar credenciais diretamente na configuração do Argo CD ou usar tokens de acesso pessoal de longa duração.

Pré-requisitos

- Um cluster do Amazon EKS com a funcionalidade para o Argo CD habilitada
- Credenciais de repositórios do Git armazenadas no AWS Secrets Manager
- Permissões do IAM configuradas para que o Argo CD acesse o Secrets Manager

Como configurar o Argo CD para usar o Secrets Manager para credenciais do repositório

1. Armazene suas credenciais do Git no Secrets Manager. Por exemplo, para armazenar um token de acesso pessoal do GitHub:

```
aws secretsmanager create-secret \
  --name argocd/github-token \
  --secret-string '{"username":"git","password":"ghp_xxxxxxxxxxxx"}'
```

2. Certifique-se de que o perfil da funcionalidade para o Argo CD tenha permissões para recuperar o segredo:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
```

```
      "Resource": "arn:aws:secretsmanager:region:account-id:secret:argocd/github-
token*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:region:account-id:key/*",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:SecretARN": "arn:aws:secretsmanager:region:account-
id:secret:argocd/*",
          "kms:ViaService": "secretsmanager.*.amazonaws.com"
        }
      }
    }
  ]
}
```

> **Note**
>
> A permissão de descriptografia do KMS é necessária porque o Secrets Manager criptografa todos os segredos usando o AWS KMS. A condição restringe a descriptografia apenas a segredos com o prefixo `argocd/`. Se você usar a chave gerenciada pela AWS padrão para o Secrets Manager, essa permissão será suficiente. Para chaves do KMS gerenciadas pelo cliente, atualize o campo `Resource` com o ARN correspondente.

3. Configure o Argo CD para usar as credenciais do Secrets Manager. Para obter informações sobre a sincronização de segredos do Secrets Manager para os segredos do Kubernetes que o Argo CD pode referenciar, consulte Secret Management na documentação do Argo CD.
4. Crie uma configuração de repositório do Argo CD que faça referência ao ARN do segredo:

```
apiVersion: v1
kind: Secret
metadata:
  name: private-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
```

```
  type: git
  url: https://github.com/org/repo
  secretArn: arn:aws:secretsmanager:region:account-id:secret:argocd/github-token
```

Para obter mais informações sobre como configurar o acesso a repositórios com o Argo CD, consulte the section called “Configuração dos repositórios”.

Centralizar e analisar os dados de segurança do EKS com o Security Lake

O Amazon Security Lake é um serviço de data lake de segurança totalmente gerenciado que permite centralizar dados de segurança de várias fontes, incluindo o Amazon EKS. Ao integrar o Amazon EKS com o Security Lake, você pode obter insights mais profundos sobre as atividades realizadas em seus recursos do Kubernetes e aprimorar a postura de segurança de seus clusters do Amazon EKS.

> **Note**
>
> Para obter mais informações sobre como usar o Security Lake com o Amazon EKS e configurar fontes de dados, consulte a documentação do Amazon Security Lake.

Como usar o Security Lake com o Amazon EKS

Dados de segurança centralizados: o Security Lake coleta e centraliza automaticamente os dados de segurança dos clusters do Amazon EKS, juntamente com dados de outros serviços do AWS, provedores de SaaS, fontes on-premises e fontes de terceiros. Isso fornece uma visão abrangente de sua postura de segurança em toda a sua organização.

Formato de dados padronizado- o Security Lake converte os dados coletados no formato OCSF (Open Cybersecurity Schema Framework), que é um esquema padrão de código aberto. Essa normalização permite uma análise mais fácil e a integração com outras ferramentas e serviços de segurança.

Detecção aprimorada de ameaças- Ao analisar os dados de segurança centralizados, incluindo os logs do ambiente de gerenciamento do Amazon EKS, é possível detectar atividades potencialmente suspeitas nos clusters do Amazon EKS com mais eficiência. Isso ajuda a identificar e responder prontamente aos incidentes de segurança.

Gerenciamento de dados simplificado- O Security Lake gerencia o ciclo de vida dos seus dados de segurança com configurações personalizáveis de retenção e replicação. Isso simplifica as tarefas de gerenciamento de dados e garante que você retenha os dados necessários para fins de conformidade e auditoria.

Habilitar o Security Lake para Amazon EKS

1. Habilite o ambiente de gerenciamento do Amazon EKS para seus clusters do EKS. Consulte Habilitar e desabilitar os logs do ambiente de gerenciamento para obter instruções detalhadas.
2. Adicione os logs de auditoria do Amazon EKS como fonte no Security Lake. Em seguida, o Security Lake começará a coletar informações detalhadas sobre as atividades realizadas nos recursos do Kubernetes em execução em seus clusters do EKS.
3. Configure as opções de retenção e replicação para seus dados de segurança no Security Lake com base em seus requisitos.
4. Use os dados de OCSF normalizados armazenados no Security Lake para resposta a incidentes, analytics de segurança e integração com outros serviços da AWS ou ferramentas de terceiros. Por exemplo, você pode Gerar insights de segurança a partir dos dados do Amazon Security Lake usando o Amazon OpenSearch Ingestion.

Analisar logs do EKS no Security Lake

O Security Lake normaliza os eventos de log do EKS para o formato OCSF, facilitando a análise e a correlação dos dados com outros eventos de segurança. Você pode usar várias ferramentas e serviços, como Amazon Athena, Amazon QuickSight ou ferramentas de analytics de segurança de terceiros, para consultar e visualizar os dados normalizados.

Para mais informações sobre o mapeamento OCSF para eventos de log do EKS, consulte https://github.com/ocsf/examples/tree/main/mappings/markdown/ AWS/v1.1.0/EKS Audit Logs[referência de mapeamento] no repositório OCSF do GitHub.

Habilitar a conectividade segura entre clusters com o Amazon VPC Lattice

O Amazon VPC Lattice é um serviço de rede de aplicações totalmente gerenciado, criado diretamente na infraestrutura de rede da AWS, que você pode usar para conectar, proteger e monitorar seus serviços em várias contas e nuvens privadas virtuais (VPCs). Com o Amazon EKS, é

possível utilizar o Amazon VPC Lattice usando o AWS Gateway API Controller, uma implementação da Kubernetes Gateway API. Com o Amazon VPC Lattice, você configura a conectividade entre clusters com a semântica padrão do Kubernetes de maneira simples e consistente. Para começar a usar o Amazon VPC Lattice com o Amazon EKS, consulte o Guia do usuário do AWS Gateway API Controller.

Conecte um cluster Kubernetes a um console de gerenciamento do Amazon EKS com o Amazon EKS Connector

Você pode usar o Amazon EKS Connector para registrar e conectar à AWS qualquer cluster do Kubernetes compatível e visualizá-lo no console do Amazon EKS. Quando um cluster é conectado, você pode ver o status, a configuração e as workloads desse cluster no console do Amazon EKS. É possível usar esse recurso para visualizar clusters conectados no console do Amazon EKS, mas não é possível gerenciá-los. O Amazon EKS Connector também é um projeto de código aberto no Github. Para obter conteúdo técnico adicional, incluindo perguntas frequentes e solução de problemas, consulte the section called “Solucionar problemas do EKS Connector”.

O Amazon EKS Connector pode conectar os seguintes tipos de clusters do Kubernetes ao Amazon EKS.

- Clusters do Kubernetes on-premises
- Clusters autogerenciados que estão em execução no Amazon EC2
- Clusters gerenciados de outros provedores de nuvem

Considerações sobre o Amazon EKS Connector

Antes de usar o Amazon EKS Connector, entenda:

- Você deve ter privilégios administrativos no cluster do Kubernetes para conectar o cluster ao Amazon EKS.
- O cluster do Kubernetes deve ter nós de processamento do Linux 64 bits (x86) presentes antes de se conectar. Os nós de processamento do ARM não são compatíveis.
- Você deve ter nós de processamento no cluster do Kubernetes que tenham acesso de saída a endpoints `ssm.` e `ssmmessages.` do Systems Manager. Para obter mais informações, consulte os endpoints do Systems Manager no AWS General Reference.
- Por padrão, é possível conectar até dez clusters em uma região. Você pode solicitar um aumento pelo console da cota de serviço. Consulte Solicitar um aumento de cota para obter mais informações.

- Somente as APIs `RegisterCluster`, `ListClusters`, `DescribeCluster` e `DeregisterCluster` do Amazon EKS são compatível com clusters externos do Kubernetes.
- É necessário ter as seguintes permissões para registrar um cluster:
  - eks:RegisterCluster
  - ssm:CreateActivation
  - ssm:DeleteActivation
  - iam:PassRole
- É necessário ter as seguintes permissões para cancelar o registro de um cluster:
  - eks:DeregisterCluster
  - ssm:DeleteActivation
  - ssm:DeregisterManagedInstance

Funções do IAM necessárias para o Amazon EKS Connector

O uso do Amazon EKS Connector requer estas duas funções do IAM:

- O perfil vinculado ao serviço do Amazon EKS Connector é criado quando você registra o cluster.
- Você deve criar o perfil do IAM de agente do Amazon EKS Connector. Para mais detalhes, consulte the section called “Função do IAM do conector”.

Para habilitar a permissão de visualização de cluster e workload para as entidades principais do IAM, é necessário aplicar `eks-connector` e as funções de cluster do Amazon EKS Connector ao cluster. Siga as etapas em Conceder acesso para visualizar os recursos do cluster do Kubernetes em um console do Amazon EKS.

Conecte um cluster Kubernetes externo ao console de gerenciamento do Amazon EKS

Você pode conectar um cluster externo do Kubernetes ao Amazon EKS usando vários métodos no processo descrito a seguir. Esse processo envolve duas etapas: registrar o cluster no Amazon EKS e instalar o agente do `eks-connector` no cluster.

> **Important**
>
> Você deve concluir a segunda etapa em até 3 dias após a conclusão da primeira etapa, antes que o registro expire.

Considerações

Você pode usar manifestos YAML ao instalar o agente. Como alternativa, você pode usar o Helm se registrar o cluster com o Console de gerenciamento da AWS ou a AWS Command Line Interface. No entanto, não é possível usar o Helm para instalar o agente caso registre o cluster com `eksctl`.

Pré-requisitos

- Certifique-se de que o perfil do agente do Amazon EKS Connector foi criado. Siga as etapas em Criar o perfil de agente do conector do Amazon EKS.
- É necessário ter as seguintes permissões para registrar um cluster:
  - `eks:RegisterCluster`
  - `ssm:CreateActivation`
  - `ssm:DeleteActivation`
  - `iam:PassRole`

Etapa 1: Registrar o cluster

Para registrar um cluster no conector do Amazon EKS, você pode usar uma destas ferramentas:

- the section called “AWS CLI”
- the section called “Console de gerenciamento da AWS”
- the section called “`eksctl`”

AWS CLI

1. AWS A CLI deve estar instalada. Para instalá-lo ou atualizá-lo, consulte Instalar a AWS CLI.
2. Para a configuração do conector, especifique a função do IAM do agente do Amazon EKS Connector. Para ter mais informações, consulte the section called “Funções do IAM necessárias para o Amazon EKS Connector”.

```
aws eks register-cluster \
     --name my-first-registered-cluster \
     --connector-config roleArn=arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole,provider="OTHER" \
     --region aws-region
```

Veja um exemplo de saída abaixo.

```
{
    "cluster": {
        "name": "my-first-registered-cluster",
        "arn": "arn:aws:eks:region:111122223333:cluster/my-first-registered-cluster",
        "createdAt": 1627669203.531,
        "ConnectorConfig": {
            "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
            "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
            "activationExpiry": 1627672543.0,
            "provider": "OTHER",
            "roleArn": "arn:aws:iam::111122223333:role/AmazonEKSConnectorAgentRole"
        },
        "status": "CREATING"
    }
}
```

Você usará os valores de `aws-region`, `activationId` e `activationCode` na próxima etapa.

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Escolha Add cluster (Adicionar cluster) e selecione Register (Registrar) para abrir a página de configuração.
3. Na seção Configure cluster (Configurar cluster), preencha os seguintes campos:
   - Name (Nome) – um nome exclusivo para o cluster.
   - Provider (Fornecedor): clique para exibir a lista suspensa de provedores de cluster do Kubernetes. Se não souber qual é o provedor específico, selecione Outro.
   - EKS Connector role (Função do EKS Connector): Selecione a função a ser usada para conectar o cluster.

4. Selecione Register cluster (Registrar cluster).
5. A página de visão geral do cluster é exibida. Se você quiser usar o chart do Helm, copie o comando `helm install` e prossiga na próxima etapa. Se desejar usar o manifesto YAML, escolha Baixar arquivo YAML para baixar o arquivo manifesto na unidade local.

   > **Important**
   >
   > Esta é sua única oportunidade de copiar o comando `helm install` ou baixar esse arquivo. Não saia dessa página, pois o link não estará acessível, e você precisará cancelar o registro do cluster e iniciar as etapas desde o início.

   O comando ou o arquivo do manifesto só podem ser usados uma vez para o cluster registrado. Se você excluir recursos do cluster do Kubernetes, deve registrar novamente o cluster e obter um novo arquivo manifesto.

Prossiga para a próxima etapa para aplicar o arquivo manifesto ao cluster do Kubernetes.

`eksctl`

1. O `eksctl` versão `0.68` ou posterior deve estar instalado. Para obter informações sobre instalação ou upgrade, consulte the section called “Criar cluster (eksctl)”.
2. Registre o cluster informando um nome, um provedor e uma região.

   ```
   eksctl register cluster --name my-cluster --provider my-provider --region region-code
   ```

   Resultado do exemplo:

   ```
   2021-08-19 13:47:26 [#]  creating IAM role "eksctl-20210819194112186040"
   2021-08-19 13:47:26 [#]  registered cluster "<name>" successfully
   2021-08-19 13:47:26 [#]  wrote file eks-connector.yaml to <current directory>
   2021-08-19 13:47:26 [#]  wrote file eks-connector-clusterrole.yaml to <current
    directory>
   2021-08-19 13:47:26 [#]  wrote file eks-connector-console-dashboard-full-access-
   group.yaml to <current directory>
   2021-08-19 13:47:26 [!]  note: "eks-connector-clusterrole.yaml" and "eks-connector-
   console-dashboard-full-access-group.yaml" give full EKS Console access to IAM
    identity "<aws-arn>", edit if required; read https://eksctl.io/usage/eks-connector
    for more info
   ```

```
2021-08-19 13:47:26 [#]  run `kubectl apply -f eks-connector.yaml,eks-connector-
clusterrole.yaml,eks-connector-console-dashboard-full-access-group.yaml` before
 expiry> to connect the cluster
```

Isso cria arquivos no seu computador local. Esses arquivos devem ser aplicados ao cluster externo em até três dias, ou o registro expira.

3. Em um terminal que possa acessar o cluster, aplique o arquivo `eks-connector-binding.yaml`:

```
kubectl apply -f eks-connector-binding.yaml
```

Etapa 2: instalar o agente do **`eks-connector`**

Para instalar o agente do `eks-connector`, use uma das seguintes ferramentas:

- the section called “Helm”
- the section called “yaml”

Helm

> **Note**
>
> Caso tenha registrado o cluster com `eksctl`, use o método de manifesto YAML em vez do chart do Helm.

1. Se você usou a CLI AWS na etapa anterior, substitua `ACTIVATION_CODE` e `ACTIVATION_ID` no comando a seguir pelos valores `activationId` e `activationCode`, respectivamente. Substitua o `aws-region` pela região da AWS que você usou na etapa anterior. Execute o seguinte comando para instalar o agente do `eks-connector` no cluster sendo resgistrado:

```
$ helm install eks-connector \
  --namespace eks-connector \
  oci://public.ecr.aws/eks-connector/eks-connector-chart \
  --set eks.activationCode=ACTIVATION_CODE \
  --set eks.activationId=ACTIVATION_ID \
  --set eks.agentRegion=aws-region
```

Se você usou o Console de gerenciamento da AWS na etapa anterior, use o comando que você copiou da etapa anterior que tem esses valores preenchidos.

2. Verifique a integridade da implantação do `eks-connector` instalado e aguarde que o status do cluster registrado no Amazon EKS seja `ACTIVE`.

yaml

Conclua a conexão aplicando o arquivo manifesto do Amazon EKS Connector ao cluster do Kubernetes. Para isso, você deve usar os métodos descritos anteriormente. Se o manifesto não for aplicado em até três dias, a validade do registro do Amazon EKS Connector expirará. Se a conexão do cluster expirar, o registro do cluster deve ser cancelado antes de ser conectado novamente.

1. Baixe o arquivo YAML do Amazon EKS Connector.

```
curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/eks-connector/latest/eks-connector.yaml
```

2. Edite o arquivo YAML do Amazon EKS Connector para substituir todas as referências de `%AWS_REGION%`, `%EKS_ACTIVATION_ID%` e `%EKS_ACTIVATION_CODE%` por `aws-region`, `activationId` e `activationCode` da saída da etapa anterior.

   O exemplo de comando a seguir pode substituir esses valores.

```
sed -i "s~%AWS_REGION%~$aws-region~g; s~%EKS_ACTIVATION_ID%~$EKS_ACTIVATION_ID~g; s~%EKS_ACTIVATION_CODE%~$(echo -n $EKS_ACTIVATION_CODE | base64)~g" eks-connector.yaml
```

   > **Important**
   >
   > Certifique-se de que seu código de ativação esteja no formato de base64.

3. Em um terminal que possa acessar o cluster, você pode aplicar o arquivo de manifesto atualizado, executando o seguinte comando:

```
kubectl apply -f eks-connector.yaml
```

4. Depois que os arquivos YAML de associação de função e manifesto do Amazon EKS Connector forem aplicados ao cluster do Kubernetes, confirme se o cluster está conectado.

```
aws eks describe-cluster \
     --name "my-first-registered-cluster" \
     --region AWS_REGION
```

A saída deve incluir `status=ACTIVE`.

5. (Opcional) Adicione etiquetas ao seu cluster. Para ter mais informações, consulte the section called “Marcando seus Recursos”.

Próximas etapas

Se você tiver algum problema com essas etapas, consulte the section called “Solucionar problemas do EKS Connector”.

Para conceder acesso adicional para as entidades principais do IAM ao console do Amazon EKS para visualizar recursos do Kubernetes em um cluster conectado, consulte the section called “Conceder acesso aos clusters”.

Conceder acesso para visualizar os recursos do cluster Kubernetes em um console do Amazon EKS

Conceda às entidades principais do IAM acesso ao console do Amazon EKS para visualizar informações sobre os recursos do Kubernetes em execução no cluster conectado.

Pré-requisitos

A entidade principal do IAM que você usa para acessar o Console de gerenciamento da AWS deve atender aos seguintes requisitos:

- Você deve ter a permissão `eks:AccessKubernetesApi` do IAM.
- A conta do Amazon EKS Connector Service pode representar o IAM ou a entidade principal no cluster. Isso permite que o Amazon EKS Connector mapeie a entidade principal do IAM para um usuário do Kubernetes.

Para criar e aplicar a função de cluster do Amazon EKS Connector

1. Baixe o modelo de função do cluster do `eks-connector`.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-clusterrole.yaml
```

2. Edite o arquivo YAML do template de função de cluster. Substitua as referências de `%IAM_ARN%` pelo nome do recurso da Amazon (ARN) da entidade principal do IAM.
3. Aplique o YAML da função do cluster do Amazon EKS Connector ao cluster do Kubernetes.

```
kubectl apply -f eks-connector-clusterrole.yaml
```

Para que uma entidade principal do IAM visualize recursos do Kubernetes no console do Amazon EKS, ela deve estar associada a um `role` ou `clusterrole` do Kubernetes com as permissões necessárias para ler os recursos. Para mais informações, consulte Usando a autorização de RBAC na documentação do Kubernetes.

Para configurar um entidade principal do IAM para acessar o cluster conectado

1. Você pode baixar um desses dois exemplos de arquivo de manifesto para criar um `clusterrole` e `clusterrolebinding` ou um `role` e `rolebinding`, respectivamente:

   Visualizar recursos do Kubernetes em todos os namespaces

   - O perfil do cluster `eks-connector-console-dashboard-full-access-clusterrole` dá acesso a todos os namespaces e recursos que podem ser visualizados no console. Você pode alterar o nome da `role`, `clusterrole` e sua respectiva vinculação antes de aplicar ao cluster. Use o seguinte comando para baixar um exemplo de arquivo.

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-full-access-group.yaml
     ```

   Visualizar recursos do Kubernetes em um namespace específico

   - O namespace desse arquivo é `default`. Então, se quiser especificar um namespace diferente, edite o arquivo antes de aplicá-lo ao cluster. Use o seguinte comando para baixar um exemplo de arquivo.

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-restricted-access-group.yaml
     ```

2. Edite o arquivo YAML de acesso total ou restrito para substituir as referências de `%IAM_ARN%` pelo nome do recurso da Amazon (ARN) da entidade principal do IAM.
3. Aplique os arquivos YAML de acesso total ou de acesso restrito ao cluster do Kubernetes. Substitua os valores do arquivo YAML por seus próprios valores.

```
kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
```

Para ver recursos do Kubernetes no cluster conectado, consulte the section called “Acessar recursos do cluster”. Dados para alguns tipos de recursos na guia Recursos não estão disponível para clusters conectados.

Cancelar o registro de um cluster do Kubernetes no console do Amazon EKS

Se você terminou de usar um cluster conectado, pode cancelar seu registro. Depois que o registro for cancelado, o cluster não ficará mais visível no console do Amazon EKS.

É necessário ter estas permissões para chamar a API deregisterCluster:

- `eks:DeregisterCluster`
- `ssm:DeleteActivation`
- `ssm:DeregisterManagedInstance`

Esse processo envolve duas etapas: cancelar o registro do cluster no Amazon EKS e desinstalar o agente do eks-connector do cluster.

Cancelar o registro de clusters do Kubernetes

Para cancelar o registro de um cluster do conector Amazon EKS, é possível usar uma destas ferramentas:

- the section called “AWS CLI”
- the section called “Console de gerenciamento da AWS”
- the section called “`eksctl`”

AWS CLI

1. AWS A CLI deve estar instalada. Para instalá-lo ou atualizá-lo, consulte Instalar a AWS CLI.
2. Certifique-se de que o perfil do agente do Amazon EKS Connector foi criado.
3. Cancele o registro do cluster conectado.

```
aws eks deregister-cluster \
    --name my-cluster \
    --region region-code
```

Console de gerenciamento da AWS

1. Abra o console do Amazon EKS.
2. Escolha Clusters.
3. Na página Clusters, selecione o cluster conectado e selecione Deregister (Cancelar registro).
4. Confirme que você deseja cancelar o registro do cluster.

`eksctl`

1. Instale a versão `eksctl 0.68` ou posterior. Para obter informações sobre instalação ou upgrade, consulte the section called “Criar cluster (eksctl)”.
2. Certifique-se de que o perfil do agente do Amazon EKS Connector foi criado.
3. Cancele o registro do cluster conectado:

```
eksctl deregister cluster --name my-cluster
```

Limpar os recursos no cluster do Kubernetes

Para desinstalar o agente `eks-connector`, use uma das seguintes ferramentas:

- the section called “helm”
- the section called “yaml”

helm

Execute o comando a seguir para desinstalar o agente.

```
helm -n eks-connector uninstall eks-connector
```

yaml

1. Exclua o arquivo YAML do Amazon EKS Connector do cluster do Kubernetes.

   ```
   kubectl delete -f eks-connector.yaml
   ```

2. Se você criou um `clusterrole` ou `clusterrolebindings` para outras entidades principais do IAM visando fornecer acesso ao cluster, exclua-o do cluster do Kubernetes.

Solução de problemas do Amazon EKS Connector

Este tópico aborda alguns dos erros comuns que você pode encontrar ao usar o Amazon EKS Connector, incluindo instruções sobre como solucioná-los e soluções de contorno.

Solução básica de problemas

Esta seção descreve as etapas para diagnosticar problemas no Amazon EKS Connector.

Verifique o status do Amazon EKS Connector

Para verificar o status do Amazon EKS Connector, digite:

```
kubectl get pods -n eks-connector
```

Inspecione logs do Amazon EKS Connector

O pod do Amazon EKS Connector consiste em três contêineres. Para recuperar logs completos de todos esses contêineres para que possa inspecioná-los, execute os seguintes comandos:

- `connector-init`

  ```
  kubectl logs eks-connector-0 --container connector-init -n eks-connector
  kubectl logs eks-connector-1 --container connector-init -n eks-connector
  ```

- `connector-proxy`

```
kubectl logs eks-connector-0 --container connector-proxy -n eks-connector
kubectl logs eks-connector-1 --container connector-proxy -n eks-connector
```

- `connector-agent`

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
kubectl exec eks-connector-1 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
```

Obtenha o nome efetivo do cluster

Os clusters do Amazon EKS são identificados exclusivamente como `clusterName` em uma única conta da AWS e região da AWS. Se você tiver vários clusters conectados no Amazon EKS, será possível confirmar em qual cluster do Amazon EKS o cluster atual do Kubernetes está registrado. Para fazê-lo, insira o seguinte para descobrir o `clusterName` do cluster atual.

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
kubectl exec eks-connector-1 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
```

Comandos diversos

Os comandos a seguir são úteis para recuperar informações necessárias para solucionar problemas.

- Use o comando a seguir para coletar imagens usadas por pods no Amazon EKS Connector.

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}"
 | tr -s '[[:space:]]' '\n'
```

- Use o comando a seguir para coletar os nomes dos nós nos quais o Amazon EKS Connector está sendo executado.

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" | tr -s
 '[[:space:]]' '\n'
```

- Execute o comando a seguir para obter as suas versões de cliente e servidor do Kubernetes.

```
kubectl version
```

- Execute o comando a seguir para obter informações sobre os nós.

```
kubectl get nodes -o wide --show-labels
```

Problema do helm: 403 proibido

Se você recebeu o seguinte erro ao executar os comandos de instalação do helm:

```
Error: INSTALLATION FAILED: unexpected status from HEAD request to https://
public.ecr.aws/v2/eks-connector/eks-connector-chart/manifests/0.0.6: 403 Forbidden
```

Você pode executar a seguinte linha para corrigi-lo:

```
docker logout public.ecr.aws
```

Erro do console: o cluster está paralisado no estado Pending (Pendente)

Se o cluster ficar paralisado no estado `Pending` no console do Amazon EKS depois que você o registrou, isso pode indicar que o Amazon EKS Connector ainda não conectou o cluster à AWS com êxito. Em um cluster registrado, o estado `Pending` significa que a conexão não foi estabelecida com êxito. Para solucionar esse problema, certifique-se de que aplicou o manifesto ao cluster do Kubernetes de destino. Se você o aplicou ao cluster, mas o cluster permanece no estado `Pending`, pode ser que o statefulset `eks-connector` não esteja íntegro. Para solucionar esse problema, consulte the section called “Os pods do Amazon EKS Connector estão apresentando loops de falhas” neste tópico.

Erro no console: O usuário system:serviceaccount:eks-connector:eks-connector não pode representar usuários de recursos no grupo API no escopo do cluster

O Amazon EKS Connector usa a representação de usuário do Kubernetes para atuar em nome das entidades principais do IAM no Console de gerenciamento da AWS. Toda entidade principal que acessa a API do Kubernetes na conta de serviço do AWS `eks-connector` precisa receber

uma permissão para representar o usuário do Kubernetes correspondente usando um ARN do IAM como o nome de usuário do Kubernetes. Nos exemplos a seguir, o ARN do IAM é mapeado para um usuário do Kubernetes.

- O usuário do IAM *john*, da conta *111122223333* da AWS, é mapeado para um usuário do Kubernetes. As melhores práticas do IAM recomendam que você conceda permissões para perfis e não para usuários.

  ```
  arn:aws:iam::111122223333:user/john
  ```

- O perfil do IAM *admin*, da conta da *111122223333* da AWS, é mapeado para um usuário do Kubernetes:

  ```
  arn:aws:iam::111122223333:role/admin
  ```

  O resultado é um ARN de perfil do IAM, em vez do ARN da sessão do AWS STS.

Para obter instruções sobre como configurar o `ClusterRole` e o `ClusterRoleBinding` para conceder à conta de serviço do `eks-connector` o privilégio de representar o usuário mapeado, consulte the section called “Conceder acesso aos clusters”. Substitua `%IAM_ARN%` no modelo pelo ARN do IAM do Console de gerenciamento da AWS da entidade principal do IAM.

Erro no console: [...] é proibido: O usuário [...] não pode listar o recurso [...] no grupo API no escopo do cluster

Considere o seguinte problema. O Amazon EKS Connector representou com êxito a entidade principal do IAM da solicitação no Console de gerenciamento da AWS no cluster de destino do Kubernetes. No entanto, a entidade principal representada não tem a permissão RBAC para as operações de API do Kubernetes.

Para resolver esse problema, existem dois métodos de conceder permissões a usuários adicionais. Se você instalou anteriormente o eks-connector por meio do chart do Helm, poderá conceder acesso aos usuários facilmente executando o comando a seguir. Substitua `userARN1` e `userARN2` por uma lista dos ARNs dos perfis do IAM para conceder acesso à visualização dos recursos do Kubernetes:

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
```

```
    --reuse-values \
    --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

Ou, como administrador do cluster, conceda aos usuários individuais do Kubernetes o nível apropriado de privilégios de RBAC. Para ter mais informações e exemplos, consulte the section called “Conceder acesso aos clusters”.

Erro no console: o Amazon EKS não pode se comunicar com o servidor de API do cluster do Kubernetes. O cluster deve estar em um estado ATIVO para estabelecer uma conexão bem-sucedida. Tente novamente em alguns minutos.

Se o serviço Amazon EKS não conseguir se comunicar com o Amazon EKS Connector no cluster de destino, pode ser por um dos seguintes motivos:

- O Amazon EKS Connector no cluster de destino não está íntegro.
- Baixa conectividade ou conexão interrompida entre o cluster de destino e a região da AWS.

Para resolver esse problema, verifique os logs do Amazon EKS Connector. Se você não encontrar nenhum erro para o Amazon EKS Connector, tente fazer a conexão novamente após alguns minutos. Se você costuma apresentar alta latência ou conectividade intermitente para o cluster de destino, considere a possibilidade de registrar novamente o cluster em uma região do AWS que esteja localizada mais perto de você.

Os pods do Amazon EKS Connector estão apresentando loops de falhas

Há vários motivos que podem fazer com que o pod do Amazon EKS Connector entre no status `CrashLoopBackOff`. É provável que esse problema envolva o contêiner `connector-init`. Verifique o status do pod do Amazon EKS Connector.

```
kubectl get pods -n eks-connector
```

Veja um exemplo de saída abaixo.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:CrashLoopBackOff   1          7s
```

Se a saída for semelhante à saída anterior, consulte the section called “Inspecione logs do Amazon EKS Connector” para solucionar o problema.

Falha ao iniciar eks-connector: InvalidActivation

Quando você inicia o Amazon EKS Connector pela primeira vez, ele registra um `activationId` e um `activationCode` na Amazon Web Services. O registro pode falhar, o que pode fazer com que o contêiner `connector-init` falhe com um erro similar ao erro a seguir.

```
F1116 20:30:47.261469       1 init.go:43] failed to initiate eks-connector:
 InvalidActivation:
```

Para solucionar esse problema, considere as seguintes causas e correções recomendadas:

- O registro pode ter falhado porque o `activationId` e o `activationCode` não estavam no arquivo de manifesto. Se for esse o caso, verifique se os valores corretos retornados na operação da API `RegisterCluster` estavam corretos e se o `activationCode` está no arquivo de manifesto. O `activationCode` é adicionado aos segredos do Kubernetes, portanto, deve ser codificado em `base64`. Para ter mais informações, consulte the section called “Etapa 1: Registrar o cluster”.
- O registro pode ter falhado porque sua ativação expirou. Isso ocorre porque, por motivos de segurança, você deve ativar o Amazon EKS Connector em até três dias após registrar o cluster. Para solucionar esse problema, certifique-se de que o manifesto do Amazon EKS Connector seja aplicado ao cluster de destino do Kubernetes antes da data e hora de expiração. Para confirmar a data de expiração da ativação, chame a operação da API `DescribeCluster`.

  ```
  aws eks describe-cluster --name my-cluster
  ```

  No exemplo de resposta a seguir, a data e a hora de expiração estão registradas como `2021-11-12T22:28:51.101000-08:00`.

  ```
  {
      "cluster": {
          "name": "my-cluster",
          "arn": "arn:aws:eks:region:111122223333:cluster/my-cluster",
          "createdAt": "2021-11-09T22:28:51.449000-08:00",
          "status": "FAILED",
          "tags": {
          },
  ```

```
        "connectorConfig": {
            "activationId": "00000000-0000-0000-0000-000000000000",
            "activationExpiry": "2021-11-12T22:28:51.101000-08:00",
            "provider": "OTHER",
            "roleArn": "arn:aws:iam::111122223333:role/my-connector-role"
        }
    }
}
```

Se a `activationExpiry` já tiver passado, cancele o registro do cluster e registre-o novamente. Isso gera uma nova ativação.

O nó do cluster está sem conectividade de saída

Para funcionar corretamente, o Amazon EKS Connector requer conectividade de saída para vários endpoints da AWS. Não é possível conectar um cluster privado sem conectividade de saída a uma região de destino AWS. Para solucionar esse problema, você deve adicionar a conectividade de saída necessária. Para obter mais informações sobre requisitos de conector, consulte the section called “Considerações sobre o Amazon EKS Connector”.

Os pods do Amazon EKS Connector estão no estado **`ImagePullBackOff`**

Se você executar o comando `get pods` e os pods estiverem no estado `ImagePullBackOff`, eles poderão não funcionar corretamente. Se os pods do Amazon EKS Connector estiverem no estado `ImagePullBackOff`, eles poderão não funcionar corretamente. Verifique o status dos pods do Amazon EKS Connector.

```
kubectl get pods -n eks-connector
```

Veja um exemplo de saída abaixo.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:ImagePullBackOff   0          4s
```

O arquivo de manifesto padrão do Amazon EKS Connector referencia imagens da Amazon ECR Public Gallery. É possível que o cluster de destino do Kubernetes não possa extrair imagens da Galeria Pública do Amazon ECR. Solucione o problema de extração de imagens da Amazon ECR Public Gallery ou considere espelhar as imagens no registro do contêiner privado de sua escolha.

Perguntas frequentes sobre o AWS Connector

P: Como funciona a tecnologia subjacente por trás do Amazon EKS Connector?

R: O Amazon EKS Connector é baseado no agente AWS Systems Manager (Gerenciador de sistemas). O Amazon EKS Connector é executado como um `StatefulSet` no cluster do Kubernetes. Ele estabelece uma conexão e configura o proxy da comunicação entre o servidor da API do cluster e a Amazon Web Services. Ele faz isso para exibir dados de cluster no console do Amazon EKS até que você desconecte o cluster da AWS. O agente do Systems Manager é um projeto de código aberto. Para obter mais informações sobre o projeto, consulte a página do projeto do GitHub.

P: Tenho um cluster on-premises do Kubernetes que quero conectar. Preciso abrir portas de firewall para conectá-lo?

R: Não, não é necessário abrir nenhuma porta de firewall. O cluster do Kubernetes exige apenas uma conexão de saída com as regiões da AWS. Os serviços da AWS nunca acessam os recursos na sua rede on-premises. O Amazon EKS Connector é executado no seu cluster e inicia a conexão com a AWS. Quando o registro do cluster for concluído, a AWS só emitirá comandos ao Amazon EKS Connector depois que você iniciar uma ação no console do Amazon EKS que exija informações do servidor da API do Kubernetes em seu cluster.

P: Quais dados são enviados do meu cluster para a AWS pelo Amazon EKS Connector?

R: O Amazon EKS Connector envia informações técnicas necessárias para o registro do seu cluster na AWS. Ele também envia metadados de clusters e workloads para os recursos do console do Amazon EKS que os clientes solicitam. O Amazon EKS Connector só coletará ou enviará esses dados se você iniciar uma ação no console ou na API do Amazon EKS que exija que os dados sejam enviados para a AWS. Além do número da versão do Kubernetes, a AWS não armazena dados por padrão. Ele só armazenará os dados se você autorizar.

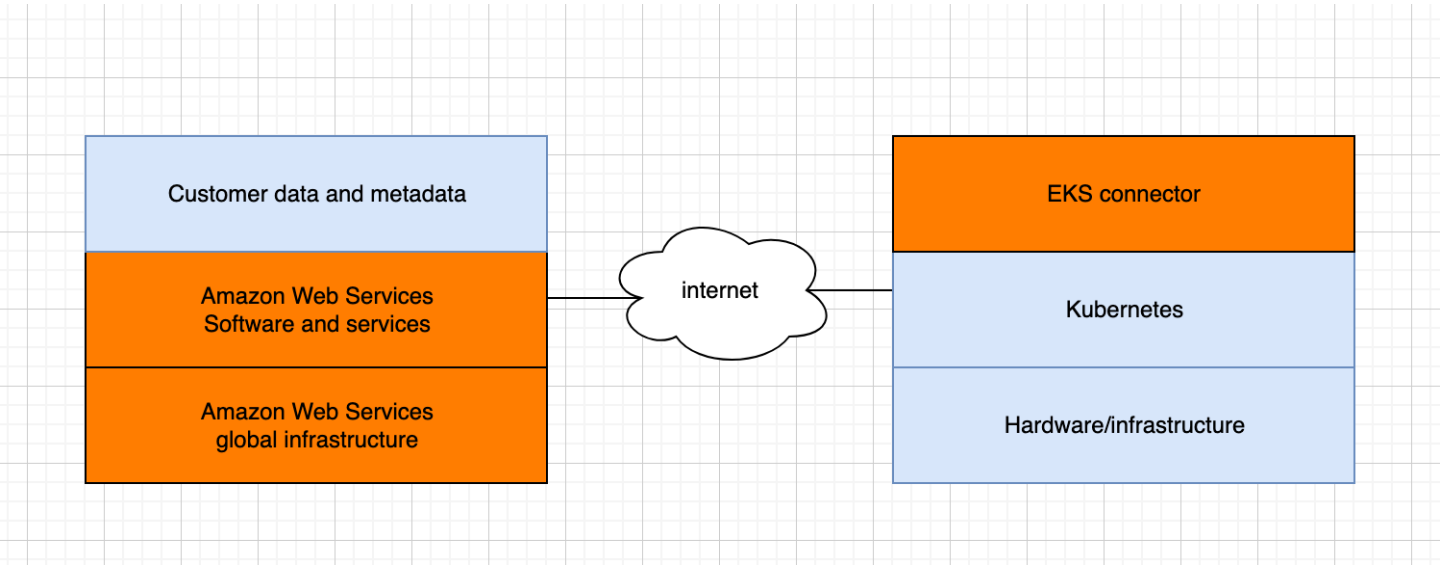
P: Posso conectar um cluster fora de uma região AWS?

R: Sim, você pode conectar um cluster de qualquer local ao Amazon EKS. Além disso, seu serviço Amazon EKS pode estar localizado em qualquer região AWS comercial pública AWS. Isso funciona com uma conexão de rede válida do seu cluster para a região da AWS de destino. Recomendamos que você escolha uma AWS Region que esteja mais próxima do local do cluster para otimizar o desempenho da interface do usuário. Por exemplo, se você tiver um cluster em execução em Tóquio, conecte seu cluster à região AWS em Tóquio (ou seja, a região `ap-northeast-1` AWS) para obter

baixa latência. Você pode conectar um cluster de qualquer local ao Amazon EKS em qualquer uma das regiões comerciais públicas AWS, exceto nas regiões China ou GovCloud AWS.

Entenda a segurança no Amazon EKS Connector

O Amazon EKS Connector é um componente de código aberto que é executado no seu cluster do Kubernetes. Esse cluster pode estar localizado fora do ambiente AWS. Isso gera considerações adicionais para responsabilidades de segurança. Essa configuração pode ser ilustrada pelo diagrama a seguir. A cor laranja representa responsabilidades da AWS, enquanto a cor azul representa as responsabilidades do cliente:



Este tópico descreve as diferenças no modelo de responsabilidade quando o cluster conectado está fora da AWS.

Responsabilidades da AWS

- Manutenção, criação e entrega do Amazon EKS Connector, um componente de código aberto executado no cluster do Kubernetes de um cliente e que se comunica com a AWS.
- Manutenção da segurança da comunicação da camada de transporte e aplicação entre o cluster do Kubernetes conectado e os serviços da AWS.

Responsabilidades do cliente

- Segurança específica do cluster do Kubernetes, especificamente nas seguintes linhas:
  - Os segredos do Kubernetes devem ser criptografados e protegidos adequadamente.

  - Bloqueio do acesso ao namespace `eks-connector`.
- Configuração de permissões para controle de acesso baseado em funções (RBAC) a fim de gerenciar o acesso das entidades principais do IAM da AWS. Para obter instruções, consulte the section called “Conceder acesso aos clusters”.
- Instalação e upgrade do Amazon EKS Connector.
- Manutenção do hardware, do software e da infraestrutura que oferece suporte ao cluster do Kubernetes conectado.
- Proteção de suas contas da AWS (por exemplo, protegendo as credenciais de usuário raiz).

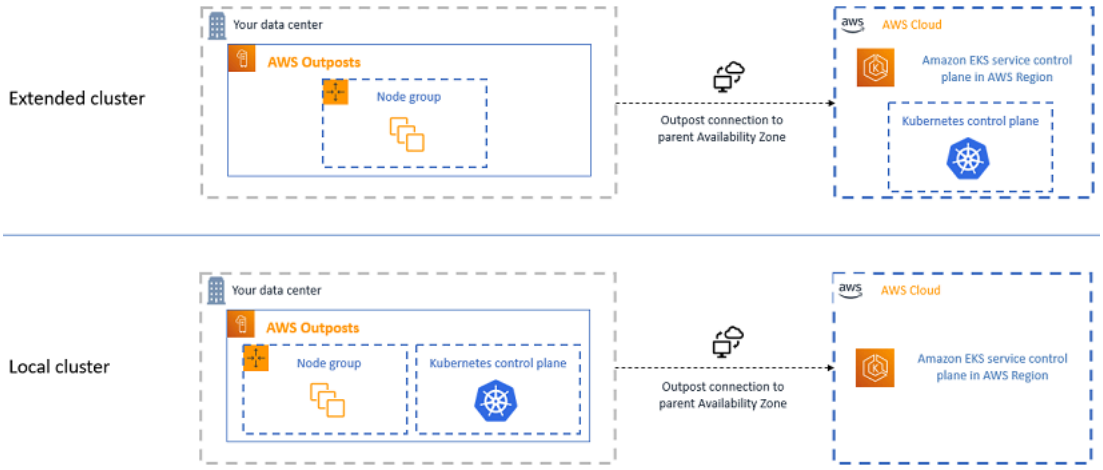
Implantar o Amazon EKS on-premises com o AWS Outposts

Você pode usar o Amazon EKS para executar aplicações on-premises do Kubernetes no AWS Outposts. É possível implantar o Amazon EKS no Outposts das seguintes maneiras:

- Clusters estendidos: execute o ambiente de gerenciamento do Kubernetes em uma região da AWS e em nós no Outpost.
- Clusters locais: execute os nós e o ambiente de gerenciamento do Kubernetes no Outpost.

Para ambas as opções de implantação, o ambiente de gerenciamento do Kubernetes é totalmente gerenciado pela AWS. É possível utilizar as mesmas APIs, ferramentas e console do Amazon EKS utilizados na nuvem para criar e executar o Amazon EKS no Outposts.

O diagrama a seguir mostra essas opções de implantação.



Quando usar cada opção de implantação

Os clusters locais e estendidos são opções de implantação de uso geral e podem ser usados para uma variedade de aplicações.

Os clusters locais permitem que você execute todo o cluster do Amazon EKS localmente no Outposts. Isso ajuda a reduzir o risco de tempo de inatividade das aplicações que pode resultar de perdas temporárias de conexão da rede com a nuvem. Essas desconexões da rede podem ser causadas por cortes de fibra ou eventos climáticos. Como todo o cluster do Amazon EKS é executado localmente no Outposts, as aplicações permanecem disponíveis. Você pode executar operações de cluster durante as desconexões da rede com a nuvem. Para obter mais informações,

consulte the section called “Preparar para desconexões”. Se a qualidade da conexão de rede entre os postos avançados e a região da AWS principal for uma preocupação e você precisar de alta disponibilidade por meio de desconexões de rede, use a opção de implantação de cluster local.

Com clusters estendidos, você pode economizar capacidade no Outpost porque o ambiente de gerenciamento do Kubernetes é executado na região da principal da AWS. Essa opção é adequada se você puder investir em conectividade de rede confiável e redundante do seu Outpost para a região da AWS. A qualidade da conexão de rede é fundamental para essa opção. A forma como o Kubernetes lida com as desconexões da rede entre os nós e o ambiente de gerenciamento do Kubernetes pode resultar em tempo de inatividade das aplicações. Para obter mais informações sobre o comportamento do Kubernetes, consulte Programação, preempção e remoção na documentação do Kubernetes.

Comparar as opções de implantação

A tabela a seguir compara as diferenças entre as duas opções.

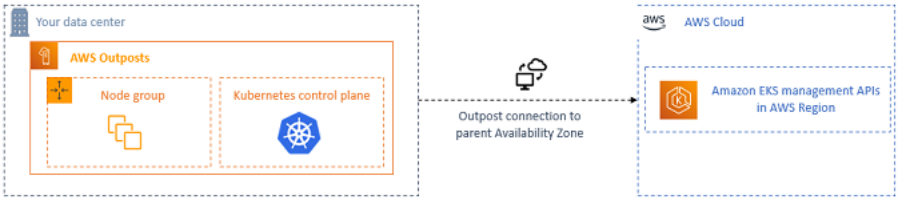
Recurso	Cluster estendido	Cluster local
Localização do ambiente de gerenciamento do Kubernetes	Região da AWS	Outpost
Conta do ambiente de gerenciamento do Kubernetes	Conta da AWS	Sua conta
Disponibilidade regional	Consulte Endpoints de serviço	Leste dos EUA (Ohio), Leste dos EUA (Norte da Virgínia) , Oeste dos EUA (N. da Califórnia), Oeste dos EUA (Oregon), Ásia-Pacífico (Seul), Ásia-Pacífico (Singapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio) Canadá (Central), Europa (Frankfur t), Europa (Irlanda), Europa (Londres), Oriente Médio (Bahrein) e América do Sul (São Paulo)

Recurso	Cluster estendido	Cluster local
Versões secundárias do Kubernetes	eks/latest/userguide/kubernetes-versions.html [Versões com suporte no Amazon EKS, type="documentation"].	eks/latest/userguide/kubernetes-versions.html [Versões com suporte no Amazon EKS, type="documentation"].
Versões da plataforma	Consulte Versões da plataforma do EKS	Consulte the section called “Versões da plataforma do EKS”
Fatores de formato do Outpost	Racks do Outpost	Racks do Outpost
Interfaces do usuário	Console de gerenciamento da AWS, AWS CLI, API do Amazon EKS, `eksctl`, AWS CloudFormation e Terraform	Console de gerenciamento da AWS, AWS CLI, API do Amazon EKS, `eksctl`, AWS CloudFormation e Terraform
Políticas gerenciadas	AmazonEKSClusterPolicy e the section called “Política gerenciada pela AWS: AmazonEKSServiceRolePolicy”	AmazonEKSLocalOutpostClusterPolicy e the section called “Política gerenciada pela AWS: AmazonEKSLocalOutpostServiceRolePolicy”
VPC e sub-redes de clusters	Consulte the section called “Requisitos para sub-rede e VPC”	Consulte the section called “Criar uma VPC e sub-redes”
Acesso ao endpoint do cluster	Público ou privado ou ambos	Privado apenas
Autenticação do servidor de API do Kubernetes	AWS Identity and Access Management (IAM) e OIDC	IAM e certificados `x.509`
Tipos de nó	Somente autogerenciado	Somente autogerenciado
Tipos de computação de nós	Amazon EC2 sob demanda	Amazon EC2 sob demanda

Recurso	Cluster estendido	Cluster local
Tipos de armazenamento de nós	Amazon EBS gp2 e SSD de NVMe local	Amazon EBS gp2 e SSD de NVMe local
AMIs otimizadas para o Amazon EKS	Amazon Linux, Windows e Bottlerocket	Somente para Amazon Linux
Versões de IP	Somente `IPv4`	Somente `IPv4`
Complementos	Complementos do Amazon EKS ou complementos autogerenciados	Somente complementos autogerenciados
Container Network Interface padrão	Plug-in CNI da Amazon VPC para Kubernetes	Plug-in CNI da Amazon VPC para Kubernetes
Logs do ambiente de gerenciamento do Kubernetes	Amazon CloudWatch Logs	Amazon CloudWatch Logs
Balanceamento de carga	Use o AWS Load Balancer Control ler para provisionar somente os balanceadores de carga de aplicações (sem balanceadores de carga de rede)	Use o AWS Load Balancer Control ler para provisionar somente os balanceadores de carga de aplicações (sem balanceadores de carga de rede)
Criptografia envelopada de segredos	Consulte the section called “Habilitar a criptografia de segredos”	Não compatível
Perfis do IAM para contas de serviço	Consulte the section called “Credenciais com IRSA”	Não compatível
Solução de problemas	Consulte Solução de problemas	Consulte the section called “Solução de problemas de clusters”

Crie clusters locais do Amazon EKS em AWS Outposts para alta disponibilidade

Você pode usar clusters locais para executar todo o seu cluster do Amazon EKS localmente em AWS Outposts. Isso ajuda a reduzir o risco de tempo de inatividade da aplicação, que pode resultar de desconexões temporárias da rede com a nuvem. Essas desconexões podem ser causadas por cortes de fibra ou eventos climáticos. Como o cluster inteiro do Kubernetes é executado localmente no Outposts, as aplicações permanecem disponíveis. Você pode executar operações de cluster durante as desconexões da rede com a nuvem. Para ter mais informações, consulte the section called “Preparar para desconexões”. O diagrama a seguir mostra a implantação de um cluster local.



Clusters locais geralmente estão disponíveis para uso com racks do Outposts.

Regiões do AWS compatíveis

Você pode criar clusters locais nas seguintes regiões AWS: Leste dos EUA (Ohio), Leste dos EUA (N. Virgínia), Oeste dos EUA (N. Califórnia), Oeste dos EUA (Oregon), Ásia-Pacífico (Seul), Ásia-Pacífico (Cingapura), Ásia-Pacífico (Sydney), Ásia-Pacífico (Tóquio), Canadá (Central), Europa (Frankfurt), Europa (Irlanda), Europa (Londres), Oriente Médio (Bahrein) e América do Sul (São Paulo). Para obter informações sobre os recursos compatíveis, consulte the section called “Comparar as opções de implantação”.

Implantar um cluster do Amazon EKS em AWS Outposts

Este tópico fornece uma visão geral do que deve ser considerado ao executar um cluster local em um Outpost. O tópico também fornece instruções sobre como implantar um cluster local em um Outpost.

Important

- Essas considerações não são replicadas na documentação relacionada do Amazon EKS. Se outros tópicos da documentação do Amazon EKS entrarem em conflito com as considerações aqui apresentadas, siga estas considerações.

- Essas considerações estão sujeitas a alterações, que podem ocorrer com frequência. Portanto, recomendamos que você revise este tópico periodicamente.
- Muitas das considerações são diferentes das considerações para a criação de um cluster no AWS Cloud.

- Os clusters locais somente são compatíveis com racks do Outpost. Um único cluster local pode ser executado em vários racks físicos do Outpost que incluem um único Outpost lógico. Um único cluster local não pode ser executado em vários Outposts lógicos. Cada Outpost lógico tem um único ARN de Outpost.
- Clusters locais executam e gerenciam o ambiente de gerenciamento do Kubernetes na sua conta no Outpost. Você não pode executar workloads nas instâncias do ambiente de gerenciamento do Kubernetes nem modificar os componentes dele. Esses nós são gerenciados pelo serviço do Amazon EKS. As alterações no ambiente de gerenciamento do Kubernetes não persistem por meio de ações automáticas de gerenciamento do Amazon EKS, como aplicação de patches.
- Os clusters locais são compatíveis com complementos autogerenciados e grupos de nós autogerenciados do Amazon Linux. O plug-in Amazon VPC CNI para Kubernetes, o kube-proxy e os complementos CoreDNS são instalados automaticamente nos clusters locais.
- Os clusters locais exigem o uso do Amazon EBS no Outposts. O Outpost deve ter o Amazon EBS disponível para o armazenamento do ambiente de gerenciamento do Kubernetes. O Outposts somente é compatível com volumes `gp2` do Amazon EBS.
- Os `PersistentVolumes` do Kubernetes baseados no Amazon EBS são compatíveis com o uso do driver CSI do Amazon EBS.
- As instâncias do ambiente de gerenciamento dos clusters locais são configuradas em uma topologia empilhada altamente disponível. Duas das três instâncias do ambiente de gerenciamento devem estar sempre íntegras para manter o quórum. Se o quórum for perdido, entre em contato com o suporte da AWS, pois algumas ações do lado do serviço serão necessárias para habilitar as novas instâncias gerenciadas.

Pré-requisitos

- Familiaridade com as opções de implementação do Outposts, Selecione tipos de instância e grupos de posicionamento para clusters do Amazon EKS no AWS Outposts com base em considerações de capacidade e requisitos e considerações de VPC.
- Um Outpost existente. Para obter mais informações, consulte O que é AWS Outposts.

- A ferramenta da linha de comando `kubectl` está instalada no seu computador ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- Uma entidade principal do IAM (usuário ou perfil) com permissões para `create` e `describe` um cluster do Amazon EKS. Para obter mais informações, consulte the section called “Criar um cluster local do Kubernetes em um Outpost” e the section called “Listar ou descrever todos os clusters”.

Quando um cluster local do Amazon EKS é criado, a entidade principal do IAM que cria o cluster é adicionada permanentemente. A entidade principal é adicionada especificamente à tabela de autorização RBAC do Kubernetes como administradora. Esta entidade tem permissões `system:masters`. A identidade desta entidade não fica visível na configuração do seu cluster. Por isso, é importante observar a entidade que criou o cluster e garantir que ela nunca tenha sido excluída. Inicialmente, somente a entidade principal que criou o servidor poderá fazer chamadas para o servidor de API do Kubernetes usando o `kubectl`. Se usar o console para criar o cluster, você deverá se certificar de que as mesmas credenciais do IAM estejam na cadeia de credenciais do AWS SDK quando executar os comandos `kubectl` no cluster. Depois de criar o cluster, você poderá conceder acesso a ele para outras entidades do IAM.

Criar um cluster local do Amazon EKS

Você pode criar um cluster local com as seguintes ferramentas descritas nesta página:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”

Além disso, é possível usar a AWS CLI, a API do Amazon EKS, os AWS SDKs, o AWS CloudFormation ou o Terraform para a criação de clusters no Outposts.

`eksctl`

Para criar um cluster local com **`eksctl`**

1. Instale a versão `0.215.0` ou posterior da ferramenta de linha de comando `eksctl` em seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
2. Copie o conteúdo a seguir no seu dispositivo. Substitua os valores a seguir e execute o comando modificado para criar o arquivo `outpost-control-plane.yaml`:
   - Substitua *`region-code`* pela região AWS suportada na qual você deseja criar o cluster.
   - Substitua *`my-cluster`* por um nome para seu cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
   - Substitua *`vpc-ExampleID1`* e *`subnet-ExampleID1`* pelos IDs da sua VPC e da sub-rede existentes. A VPC e a sub-rede devem atender aos requisitos em Criar uma VPC e sub-redes para clusters do Amazon EKS no AWS Outposts.
   - Substitua *`uniqueid`* pelo ID do seu Outpost.
   - Substitua *`m5.large`* por um tipo de instância disponível no seu Outpost. Antes de escolher um tipo de instância, consulte the section called “Considerações sobre a capacidade”. Três instâncias do ambiente de gerenciamento são implantadas. Não é possível alterar esse número.

```
cat >outpost-control-plane.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "1.33"

vpc:
  clusterEndpoints:
    privateAccess: true
  id: "vpc-vpc-ExampleID1"
```

```
  subnets:
    private:
      outpost-subnet-1:
        id: "subnet-subnet-ExampleID1"

outpost:
  controlPlaneOutpostARN: arn:aws:outposts:region-code:111122223333:outpost/op-
uniqueid
  controlPlaneInstanceType: m5.large
EOF
```

Para obter uma lista completa de todas as opções e padrões disponíveis, consulte AWS Outposts Support e Config file schema na documentação `eksctl`.

3. Crie o cluster usando o arquivo de configuração que você criou na etapa anterior. O `eksctl` cria uma VPC e uma sub-rede no seu Outpost para implantar o cluster.

```
eksctl create cluster -f outpost-control-plane.yaml
```

O provisionamento de cluster leva alguns minutos. Durante a criação do cluster, várias linhas de saída são exibidas. A última linha do resultado é semelhante ao exemplo de linha a seguir.

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> Para ver a maioria das opções que podem ser especificadas ao criar um cluster com `eksctl`, use o comando `eksctl create cluster --help`. Para ver todas as opções disponíveis, é possível utilizar um arquivo `config`. Para obter mais informações, consulte Uso dos arquivos de configuração e o esquema de arquivo de configuração na documentação do `eksctl`. Você pode encontrar exemplos de arquivos de configuração no GitHub.

O comando `eksctl` criou automaticamente uma entrada de acesso para a entidade principal do IAM (usuário ou perfil) que criou o cluster, e concedeu à entidade principal do IAM permissões de administrador para objetos do Kubernetes no cluster. Se não quiser que o criador do cluster tenha acesso de administrador aos objetos do Kubernetes no cluster, adicione o seguinte texto ao arquivo de configuração anterior: `bootstrapClusterCreatorAdminPermissions: false`

(no mesmo nível que `metadata`, `vpc` e `outpost`). Caso tenha adicionado a opção, então, depois da criação do cluster, você precisará criar uma entrada de acesso para pelo menos uma entidade principal do IAM, ou nenhuma entidade principal do IAM terá acesso aos objetos do Kubernetes no cluster.

Console de gerenciamento da AWS

Para criar o cluster com a Console de gerenciamento da AWS

1. Você precisa de uma VPC e uma sub-rede existentes que atendam aos requisitos do Amazon EKS. Para obter mais informações, consulte the section called “Criar uma VPC e sub-redes”.
2. Se você já tiver um perfil do IAM do cluster local ou se for criar seu cluster com `eksctl`, poderá ignorar essa etapa. Por padrão, `eksctl` cria um perfil para você.
   a. Execute o comando a seguir para criar um arquivo JSON de política de confiança do IAM:

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

   b. Crie o perfil do IAM do cluster do Amazon EKS. Para criar um perfil do IAM, a entidade principal do IAM que estiver criando o perfil deverá ser atribuída à seguinte ação `iam:CreateRole` (permissão):

```
aws iam create-role --role-name myAmazonEKSLocalClusterRole --assume-role-policy-
document file://"eks-local-cluster-role-trust-policy.json"
```

   c. Anexe a política gerenciada do Amazon EKS denominada AmazonEKSLocalOutpostClusterPolicy ao perfil. Para anexar uma política do IAM a uma entidade principal do IAM a entidade principal do IAM que está anexando a política deve

receber uma das seguintes ações do IAM (permissões): `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

3. Abra o console do Amazon EKS.
4. Na parte superior da tela do console, certifique-se de ter selecionado uma região compatível com AWS.
5. Escolha Add cluster (Adicionar cluster) e, em seguida, Create (Criar).
6. Na página Configure cluster (Configurar cluster), insira ou selecione valores para os seguintes campos:
   - Localização do ambiente de gerenciamento do Kubernetes: escolha AWS Outposts.
   - Outpost ID (ID do Outpost): escolha o ID do Outpost no qual você deseja criar seu ambiente de gerenciamento.
   - Em Instance type (Tipo de instância): selecione um tipo de instância. Somente os tipos de instância disponíveis no seu Outpost são exibidos. Na lista suspensa, cada tipo de instância descreve para quantos nós o tipo de instância é recomendado. Antes de escolher um tipo de instância, consulte the section called “Considerações sobre a capacidade”. Todas as réplicas são implantadas com o uso do mesmo tipo de instância. Você não poderá alterar o tipo de instância depois que seu cluster for criado. Três instâncias do ambiente de gerenciamento são implantadas. Não é possível alterar esse número.
   - Name (Nome): um nome exclusivo para o cluster. Ele deve ser exclusivo na sua conta da AWS. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
   - Versão do Kubernetes: escolha a versão do Kubernetes que você deseja usar para o cluster. A menos que precise usar uma versão anterior, recomendamos que você selecione a versão mais recente.
   - Perfil de serviço do cluster: escolha o perfil do IAM do cluster do Amazon EKS criado em uma etapa anterior para permitir que o ambiente de gerenciamento do Kubernetes gerencie os recursos da AWS.

- Acesso de administrador do cluster do Kubernetes: se quiser que a entidade principal do IAM (perfil ou usuário) que está criando o cluster tenha acesso de administrador aos objetos Kubernetes no cluster, aceite o padrão (permitir). O Amazon EKS cria uma entrada de acesso para a entidade principal do IAM e concede permissões de administrador do cluster à entrada de acesso. Para obter mais informações sobre as entradas de acesso, consulte the section called “Entradas de acesso”.

  Se você quiser que uma entidade principal do IAM diferente daquela que está criando o cluster tenha acesso de administrador aos objetos do Kubernetes do cluster, escolha a opção de não permitir. Após a criação do cluster, qualquer entidade principal do IAM que tenha permissões do IAM para criar entradas de acesso poderá adicionar uma entrada de acesso para qualquer entidade principal do IAM que precise acessar os objetos do Kubernetes do cluster. Para obter mais informações sobre as permissões necessárias do IAM, consulte Ações definidas pelo Amazon Elastic Kubernetes Service na Referência de autorização de serviço. Se você escolher a opção de não permitir e não criar nenhuma entrada de acesso, nenhuma entidade principal do IAM terá acesso aos objetos do Kubernetes no cluster.
- Tags (Etiquetas) – (opcional) adicione etiquetas ao cluster. Para obter mais informações, consulte the section called “Marcando seus Recursos”. Após terminar com essa página, escolha Próximo.

7. Na página Specify networking (Especificar redes), selecione valores para os seguintes campos:
   - VPC: escolha uma VPC existente. A VPC deve ter uma quantidade suficiente de endereços IP disponíveis para o cluster, todos os nós e outros recursos Kubernetes que você deseja criar. Sua VPC deve atender aos requisitos em Requisitos e considerações da VPC.
   - Subnets (Sub-redes): por padrão, as sub-redes disponíveis na VPC especificada no campo anterior são pré-selecionadas. As sub-redes que você escolher devem atender aos requisitos em Requisitos e considerações da sub-rede.
   - Security groups (Grupos de segurança) (Opcional): especifique um ou mais grupos de segurança que você deseja que o Amazon EKS associe às interfaces de rede que ele cria. O Amazon EKS cria automaticamente um grupo de segurança que permite a comunicação entre o cluster e a VPC. O Amazon EKS associa esse grupo de segurança, e qualquer um que você escolher, às interfaces de rede que ele cria. Para saber mais sobre o grupo de segurança de cluster criado pelo Amazon EKS, consulte the section called “Requisitos para grupos de segurança”. É possível modificar as regras no grupo de segurança do cluster criado pelo Amazon EKS. Se optar por adicionar seus próprios grupos de segurança, você não poderá alterar os escolhidos após a criação desse cluster. Para que os hosts on-premises se comuniquem com o endpoint do cluster, você deve permitir tráfego de entrada do grupo de

segurança do cluster. Para clusters que não têm uma conexão de internet de entrada e saída (também conhecidos como clusters privados), você deve fazer o seguinte:

- Adicione o grupo de segurança associado aos endpoints da VPC necessários. Para obter mais informações sobre os endpoints necessários, consulte the section called “Usar VPC endpoints da interface do” em Acesso da sub-rede a serviços da AWS.
- Modifique o grupo de segurança que o Amazon EKS criou para permitir tráfego do grupo de segurança associado aos endpoints da VPC. Após terminar com essa página, escolha Próximo.

8. Na página Configurar observabilidade, é possível escolher opcionalmente quais opções de Métricas e Log do ambiente de gerenciamento você deseja ativar. Por padrão, o cada tipo de log está desativado.
   - Para obter mais informações sobre a opção de métricas do Prometheus, consulte the section called “Etapa 1: ativar as métricas do Prometheus”.
   - Para obter mais informações sobre as opções do Log do ambiente de gerenciamento, consulte the section called “Logs do ambiente de gerenciamento”. Após terminar com essa página, escolha Próximo.
9. Na página Review and create (Revisar e criar), revise as informações que você inseriu ou selecionou nas páginas anteriores. Se precisar fazer alterações, escolha Edit (Editar). Quando estiver satisfeito, escolha Criar. O campo Status mostra o campo CREATING (Criando) enquanto o cluster é provisionado.

   O provisionamento de cluster leva alguns minutos.

Visualizar seu cluster local do Amazon EKS

1. Depois que o cluster for criado, você poderá visualizar as instâncias do ambiente de gerenciamento do Amazon EC2 que foram criadas.

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?
Key==`Name`]|[0].Value}' | grep my-cluster-control-plane
```

   Veja um exemplo de saída abaixo.

```
"Name": "my-cluster-control-plane-id1"
"Name": "my-cluster-control-plane-id2"
"Name": "my-cluster-control-plane-id3"
```

Cada instância recebeu taints com o `node-role.eks-local.amazonaws.com/control-plane` para que nenhuma workload seja programada nas instâncias do ambiente de gerenciamento. Para obter mais informações sobre taints, consulte Taints e Tolerâncias na documentação do Kubernetes. O Amazon EKS monitora continuamente o estado dos clusters locais. Executamos ações automáticas de gerenciamento, como patches de segurança e reparo de instâncias não íntegras. Quando os clusters locais são desconectados da nuvem, executamos ações para garantir que o cluster seja reparado para um estado íntegro após a reconexão.

2. Se você criou seu cluster usando `eksctl`, pode ignorar esta etapa. O `eksctl` completa esta etapa para você. Habilite o `kubectl` para se comunicar com o cluster adicionando um novo contexto ao arquivo `kubectl config`. Para obter instruções sobre como criar e atualizar o arquivo, consulte the section called “Acessar meu cluster com o kubectl”.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Veja abaixo um exemplo de saída.

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

3. Para se conectar ao servidor de API do Kubernetes do cluster local, você deve ter acesso ao gateway local da sub-rede ou conectar-se de dentro da VPC. Para obter mais informações sobre como conectar um rack de Outpost à sua rede on-premises, consulte Como funcionam gateways locais para racks no Guia do usuário do AWS Outposts. Se você usar roteamento direto da VPC e a sub-rede do Outpost tiver uma rota para o gateway local, os endereços IP privados das instâncias do ambiente de gerenciamento do Kubernetes serão transmitidos automaticamente pela rede local. O endpoint do servidor de API do Kubernetes do cluster local está hospedado no Amazon Route 53 (Route 53). O endpoint do serviço de API pode ser resolvido por servidores DNS públicos para os endereços IP privados dos servidores de API do Kubernetes.

   As instâncias do ambiente de gerenciamento do Kubernetes dos clusters locais são configuradas com interfaces de rede elásticas estáticas com endereços IP privados fixos que não mudam durante o ciclo de vida do cluster. As máquinas que interagem com o servidor de API do Kubernetes podem não ter conectividade com o Route 53 durante as desconexões da rede. Se for esse o caso, recomendamos configurar `/etc/hosts` com os endereços IP privados estáticos para operações contínuas. Também recomendamos configurar servidores DNS locais e conectá-los ao Outpost. Para obter mais informações, consulte a documentação doAWS Outposts. Execute o comando a seguir para confirmar que a comunicação foi estabelecida com o cluster.

```
kubectl get svc
```

Veja abaixo um exemplo de saída.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

4. (Opcional) Teste a autenticação em seu cluster local quando ele estiver em um estado desconectado do AWS Cloud. Para instruções, consulte the section called “Preparar para desconexões”.

Recursos internos

O Amazon EKS cria os seguintes recursos no seu cluster. Os recursos são para uso interno do Amazon EKS. Para o funcionamento adequado do cluster, não edite nem modifique esses recursos.

- Os seguintes pods de espelho:
  - `aws-iam-authenticator-`*`node-hostname`*
  - `eks-certificates-controller-`*`node-hostname`*
  - `etcd-`*`node-hostname`*
  - `kube-apiserver-`*`node-hostname`*
  - `kube-controller-manager-`*`node-hostname`*
  - `kube-scheduler-`*`node-hostname`*
- Os seguintes complementos autogerenciados:
  - `kube-system/coredns`
  - `kube-system/ kube-proxy` (só será criado quando você adicionar seu primeiro nó)
  - `kube-system/aws-node` (só será criado quando você adicionar seu primeiro nó). Clusters locais usam o plug-in CNI da Amazon VPC para o plug-in do Kubernetes de redes de clusters. Não altere a configuração das instâncias do ambiente de gerenciamento (pods denominados `aws-node-controlplane-*`). Há variáveis de configuração que você pode usar para alterar o valor padrão para quando o plug-in criar novas interfaces de rede. Para obter mais informações, consulte a documentação sobre o GitHub.
- Os seguintes serviços:
  - `default/kubernetes`

  - `kube-system/kube-dns`
- Um `PodSecurityPolicy` denominado `eks.system`
- Um `ClusterRole` denominado `eks:system:podsecuritypolicy`
- Um `ClusterRoleBinding` denominado `eks:system`
- Além do grupo de segurança do cluster, o Amazon EKS cria um grupo de segurança em sua conta AWS chamado `eks-local-internal-do-not-use-or-edit-`*`cluster-name`*`-`*`uniqueid`*. Esse grupo de segurança permite que o tráfego flua livremente entre os componentes do Kubernetes em execução nas instâncias do ambiente de gerenciamento.

Próximas etapas recomendadas:

- Conceda à entidade principal do IAM que criou o cluster as permissões necessárias para visualizar os recursos do Kubernetes no Console de gerenciamento da AWS
- Conceda acesso para entidades do IAM ao cluster. Se quiser que as entidades visualizem os recursos do Kubernetes no console do Amazon EKS, conceda as permissões necessárias às entidades.
- Configurar o registro em log para o seu cluster
- Familiarize-se com o que acontece durante desconexões de rede.
- Adicionar nós ao seu cluster
- Pense em configurar um plano de backup para seu `etcd`. O Amazon EKS não é compatível com backup e restauração automatizadas do `etcd` para clusters locais. Para obter mais informações, consulte Backing up an etcd cluster na documentação do Kubernetes. As duas principais opções estão usando o `etcdctl` para automatizar a captura de snapshots ou o uso do backup de volume de armazenamento do Amazon EBS.

Saiba mais sobre as versões das plataformas do Kubernetes e do Amazon EKS para AWS Outposts

As versões da plataforma do cluster local representam os recursos do cluster Amazon EKS no AWS Outposts. As versões incluem os componentes que são executados no ambiente de gerenciamento do Kubernetes e quais sinalizadores do servidor de API do Kubernetes estão habilitados. Também incluem a versão atual do patch do Kubernetes. Cada versão secundária do Kubernetes tem uma ou mais versões de plataforma do associadas. As versões da plataforma para diferentes versões

secundárias do Kubernetes são independentes. As versões de plataforma para clusters locais e clusters do Amazon EKS na nuvem são independentes.

Quando uma nova versão secundária do Kubernetes está disponível para clusters locais, como `1.31`, a versão inicial da plataforma para essa versão secundária do Kubernetes começa em `eks-local-outposts.1`. No entanto, o Amazon EKS lança novas versões de plataforma periodicamente para habilitar novas configurações do ambiente de gerenciamento do Kubernetes e para fornecer correções de segurança.

Quando novas versões da plataforma do cluster local são disponibilizadas para uma versão secundária:

- O número da versão da plataforma é incrementado (`eks-local-outposts.n+1`).
- O Amazon EKS atualiza automaticamente todos os clusters locais existentes para a versão mais recente da plataforma disponível para a versão secundária correspondente do Kubernetes. Atualizações automáticas de versões de plataforma existentes do são implementadas de forma incremental. O processo de implantação consiste na substituição das instâncias gerenciadas do ambiente de gerenciamento do Kubernetes em execução no Outpost, uma por vez, até que todas as três instâncias sejam substituídas por novas.
- O processo de substituição da instância do ambiente de gerenciamento do Kubernetes interromperá o andamento se houver risco de interrupção do serviço. O Amazon EKS só tentará substituir uma instância caso as outras duas instâncias do ambiente de gerenciamento do Kubernetes estejam íntegras e aprovadas quanto a todas as condições de prontidão como um nó de cluster.
- A implantação de uma versão da plataforma normalmente leva menos de 30 minutos para ser concluída. Se um cluster permanecer no estado `UPDATING` por um longo período de tempo, consulte the section called “Solução de problemas de clusters” e obtenha ajuda do AWS Support. Nunca encerre manualmente as instâncias do ambiente de gerenciamento do Kubernetes, a menos que seja instruído pelo AWS Support.
- O Amazon EKS pode publicar uma nova AMI do nó com uma versão de patch correspondente. Todas as versões de patch são compatíveis entre o ambiente de gerenciamento do Kubernetes e as AMIs dos nós para uma única versão secundária do Kubernetes.

Novas versões de plataforma não apresentam alterações que podem causar interrupções nem causam interrupções de serviço.

Os clusters locais são sempre criados com a versão da plataforma mais recente disponível (`eks-local-outposts.n`) para a versão especificada do Kubernetes.

As versões recentes e atuais da plataforma estão descritas nas tabelas a seguir.

Para receber notificações de todas as alterações do arquivo de origem nesta página específica da documentação, você pode se inscrever no seguinte URL com um leitor de RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/outposts/eks-outposts-platform-versions.adoc.atom
```

Versão do Kubernetes **`1.31`**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.31`: `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `ClusterTrustBundleAttest`, `DefaultIngressClass`, `DefaultStorageClass`, `DefaultTolerationSeconds`, `ExtendedResourceToleration`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `PersistentVolumeClaimResize`, `PodSecurity`, `Priority`, `ResourceQuota`, `RuntimeClass`, `ServiceAccount`, `StorageObjectInUseProtection`, `TaintNodesByCondition`, `ValidatingAdmissionPolicy` e `ValidatingAdmissionWebhook`.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
`1.31.12`	`eks-local-outposts.5`	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para `v1.31.10`. AWS IAM Authentic ator atualizado para `v0.7.4`. Plug-in da CNI da Amazon VPC para Kubernete	3 de outubro de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
		s atualizada para `v1.20.2`. Versão do Bottlerocket atualizada para `v1.47.0`.	
`1.31.9`	`eks-local-outposts.4`	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para `v1.31.9`. AWS IAM Authenticator atualizado para `v0.7.2`. Plug-in da CNI da Amazon VPC para Kubernetes atualizado para a versão `v1.20.0`. Versão do Bottlerocket atualizada para a versão `v1.43.0`.	9 de agosto de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
1.31.7	eks-local-outposts.3	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.31.9. AWS IAM Authenticator atualizado para v0.7.1. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para v1.19.5. Versão do Bottlerocket atualizada para v1.40.0.	19 de junho de 2025
1.31.6	eks-local-outposts.2	Nova versão da plataforma com correções de segurança e melhorias. Versão do Bottlerocket atualizada para v1.36.0.	24 de abril de 2025
1.31.6	eks-local-outposts.1	Lançamento inicial do Kubernetes versão v1.31 para clusters locais do Amazon EKS no Outposts	9 de abril de 2025

Versão do Kubernetes **1.30**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.30`: `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `ClusterTrustBundleAttest`, `DefaultIngressClass`, `DefaultStorageClass`, `DefaultTolerationSeconds`, `ExtendedResourceToleration`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `PersistentVolumeClaimResize`, `PodSecurity`, `Priority`, `ResourceQuota`, `RuntimeClass`, `ServiceAccount`, `StorageObjectInUseProtection`, `TaintNodesByCondition`, `ValidatingAdmissionPolicy` e `ValidatingAdmissionWebhook`.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
`1.30.14`	`eks-local-outposts.7`	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para `v1.30.14`. AWS IAM Authentic ator atualizado para `v0.7.4`. Plug-in da CNI da Amazon VPC para Kubernete s atualizada para `v1.20.2`. Versão do Bottlerocket atualizad a para `v1.47.0`.	3 de outubro de 2025
`1.30.13`	`eks-local-outposts.6`	Nova versão da plataforma com correções de segurança e melhorias. kube-	09 de agosto de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
		proxy atualizado para `v1.30.13`. AWS IAM Authentic ator atualizado para `v0.7.2`. Plug-in da CNI da Amazon VPC para Kubernete s atualizada para `v1.20.0`. Versão do Bottlerocket atualizad a para `v1.43.0`.	
`1.30.11`	`eks-local-outposts.5`	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para `v1.30.11`. AWS IAM Authentic ator atualizado para `v0.7.1`. Plug-in da CNI da Amazon VPC para Kubernete s atualizado para a versão `v1.19.5`. Versão do Bottleroc ket atualizada para a versão `v1.40.0`.	19 de junho de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
1.30.10	eks-local-outposts.4	Nova versão da plataforma com correções de segurança e melhorias. Versão do Bottlerocket atualizada para v1.36.0.	24 de abril de 2025
1.30.10	eks-local-outposts.3	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.30.10. AWS IAM Authenticator atualizado para v0.6.29. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para v1.19.2. CoreDNS atualizado para v1.11.4. AWS Cloud Controller Manager atualizado para v1.30.8. Versão do Bottlerocket atualizada para v1.34.0.	27 de março de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
1.30.7	eks-local-outposts.2	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.30.7. AWS IAM Authenticator atualizado para v0.6.28. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para v1.19.0. Versão do Bottlerocket atualizada para v1.29.0.	10 de janeiro de 2025
1.30.5	eks-local-outposts.1	Lançamento inicial do Kubernetes versão v1.30 para clusters locais do Amazon EKS no Outposts	13 de novembro de 2024

Versão do Kubernetes **1.29**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.29: CertificateApproval, CertificateSigning, CertificateSubjectRestriction, ClusterTrustBundleAttest, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, PodSecurity, Priority, ResourceQuota, RuntimeClass, ServiceAccount,

`StorageObjectInUseProtection`, `TaintNodesByCondition`, `ValidatingAdmissionPolicy` e `ValidatingAdmissionWebhook`.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
`v1.29.15`	`eks-local-outposts.10`	Nova versão da plataforma com correções de segurança e melhorias. AWS IAM Authenticator atualizado para `v0.7.4`. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para `v1.20.2`. Versão do Bottlerocket atualizada para `v1.47.0`.	3 de outubro de 2025
`v1.29.15`	`eks-local-outposts.9`	Nova versão da plataforma com correções de segurança e melhorias. AWS IAM Authenticator atualizado para `v0.7.2`. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para `v1.20.0`. Versão do Bottlerocket atualizada para `v1.43.0`.	9 de agosto de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
v1.29.15	eks-local-outposts.8	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.29.15. AWS IAM Authentic ator atualizado para v0.7.1. Plug-in da CNI da Amazon VPC para Kubernete s atualizada para v1.19.5. Versão do Bottlerocket atualizad a para v1.40.0.	19 de junho de 2025
v1.29.14	eks-local-outposts.7	Nova versão da plataforma com correções de segurança e melhorias. Versão do Bottlerocket atualizad a para v1.36.0.	24 de março de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
v1.29.14	eks-local-outposts.6	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.29.14. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para v1.19.2. CoreDNS atualizado para v1.11.4. AWS Cloud Controller Manager atualizado para v1.29.8. Versão do Bottlerocket atualizada para v1.34.0.	27 de março de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
v1.29.11	eks-local-outposts.5	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.29.11. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para v1.19.0. A imagem do CoreDNS foi atualizada para a versão v1.11.3. Versão do Bottlerocket atualizada para v1.29.0.	10 de janeiro de 2025
1.29.9	eks-local-outposts.4	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.29.9. AWS IAM Authenticator atualizado para v0.6.26. Versão do Bottlerocket atualizada para v1.26.0.	8 de novembro de 2024

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
1.29.6	eks-local-outposts.3	Nova versão da plataforma com correções de segurança e melhorias. Versão do Bottlerocket atualizada para v1.22.0.	22 de outubro de 2024
1.29.6	eks-local-outposts.2	Nova versão da plataforma com correções de segurança e melhorias. Versão do Bottlerocket atualizada para v1.21.0.	27 de agosto de 2024
1.29.6	eks-local-outposts.1	Lançamento inicial do Kubernetes versão v1.29 para clusters locais do Amazon EKS no Outposts	20 de agosto de 2024

Versão do Kubernetes **1.28**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.28: CertificateApproval, CertificateSigning, CertificateSubjectRestriction, ClusterTrustBundleAttest, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, PodSecurity, Priority, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionPolicy e ValidatingAdmissionWebhook.

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
1.28.15	eks-local-outposts.17	Nova versão da plataforma com correções de segurança e melhorias. AWS IAM Authenticator atualizado para v0.7.4. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para v1.20.2. Versão do Bottlerocket atualizada para v1.47.0.	3 de outubro de 2025
1.28.15	eks-local-outposts.16	Nova versão da plataforma com correções de segurança e melhorias. AWS IAM Authenticator atualizado para v0.7.2. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para v1.20.0. Versão do Bottlerocket atualizada para v1.43.0.	9 de agosto de 2025
1.28.15	eks-local-outposts.15	Nova versão da plataforma com correções de segurança e	19 de junho de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
		melhorias. AWS IAM Authenticator atualizado para `v0.7.1`. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para `v1.19.5`. CoreDNS atualizado para `v1.11.4`. Versão do Bottlerocket atualizada para `v1.40.0`.	
`1.28.15`	`eks-local-outposts.14`	Nova versão da plataforma com correções de segurança e melhorias. Versão do Bottlerocket atualizada para `v1.36.0`.	24 de abril de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
1.28.15	eks-local-outposts.13	Nova versão da plataforma com correções de segurança e melhorias. Compilação v1.28.15 do kube-proxy atualizada. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para v1.19.0. AWS Cloud Controller Manager atualizado para v1.28.11.	27 de março de 2025
1.28.15	eks-local-outposts.12	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.28.15. Plug-in da CNI da Amazon VPC para Kubernetes atualizada para v1.19.0. Versão do Bottlerocket atualizada para v1.29.0.	10 de janeiro de 2025

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
1.28.14	eks-local-outposts.11	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.28.14. AWS IAM Authentic ator atualizado para v0.6.26. A imagem do CoreDNS foi atualizada para a versão v1.11.1. Versão do Bottleroc ket atualizada para v1.26.0.	8 de novembro de 2024
1.28.10	eks-local-outposts.10	Nova versão da plataforma com correções de segurança e melhorias. Versão do Bottlerocket atualizad a para v1.22.0.	22 de outubro de 2024
1.28.10	eks-local-outposts.9	Nova versão da plataforma com correções de segurança e melhorias. Versão do Bottlerocket atualizad a para v1.21.0.	27 de agosto de 2024

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
1.28.10	eks-local-outposts.8	Nova versão da plataforma com correções de segurança e melhorias.	30 de julho de 2024
1.28.10	eks-local-outposts.6	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.28.10. AWS IAM Authentic ator atualizado para v0.6.20. Versão do Bottlerocket atualizad a para v1.20.2.	19 de junho de 2024
1.28.6	eks-local-outposts.5	Versão atualizada do Bottlerocket para v1.19.3 contendo as correções de erros mais recentes para oferecer suporte à inicialização local no Outposts.	18 de abril de 2024

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
1.28.6	eks-local-outposts.4	Nova versão da plataforma com correções de segurança e melhorias. Suporte restaurado ou inicialização local no Outposts. Downgrade da versão do Bottlerocket para v1.15.1 por questões de compatibilidade.	2 de abril de 2024
1.28.6	eks-local-outposts.3	Nova versão da plataforma com correções de segurança e melhorias.	22 de março de 2024

Versão do Kubernetes	Versão da plataforma Amazon EKS	Notas da versão	Data de lançamento
1.28.6	eks-local-outposts.2	Nova versão da plataforma com correções de segurança e melhorias. kube-proxy atualizado para v1.28.6. AWS IAM Authenticator atualizado para v0.6.17. Plug-in CNI da Amazon VPC para Kubernetes foi rebaixado para v1.13.2 por questões de compatibilidade. Versão do Bottlerocket atualizada para v1.19.2.	8 de março de 2024
1.28.1	eks-local-outposts.1	Lançamento inicial do Kubernetes versão v1.28 para clusters locais do Amazon EKS no Outposts	4 de outubro de 2023

Criar um VPC e sub-redes para clusters do Amazon EKS em AWS Outposts

Ao criar um cluster local, você especifica uma VPC e pelo menos uma sub-rede privada que é executada no Outposts. Este tópico fornece uma visão geral dos requisitos e considerações sobre a VPC e as sub-redes para o cluster local.

Requisitos e considerações para VPCs

Quando você cria um cluster local, a VPC especificada deve atender aos requisitos e considerações a seguir:

- Certifique-se de que a VPC tenha endereços IP suficientes para o cluster local, para todos os nós e para outros recursos do Kubernetes que você queira criar. Se a VPC que você deseja usar não tiver endereços IP suficientes, tente aumentar a quantidade de endereços IP disponíveis. É possível fazer isso associando blocos de Encaminhamento Entre Domínios Sem Classificação (CIDR) com sua VPC. É possível associar blocos CIDR privados (RFC 1918) e públicos (não RFC 1918) à VPC antes ou depois de criar o cluster. Um cluster pode levar até cinco horas para reconhecer um bloco CIDR que você associou a uma VPC.
- A VPV não pode ter prefixos IP ou blocos CIDR IPv6 atribuídos. Devido a essas restrições, as informações abordadas em Atribuir mais endereços IP aos nós do Amazon EKS com prefixos e the section called “IPv6” não se aplicam à sua VPC.
- A VPC tem um nome de host DNS e a resolução de DNS habilitados. Sem esses recursos, haverá falha na criação do cluster e será necessário habilitar os recursos e recriar o cluster. Para mais informações, consulte Atributos de DNS para sua VPC, no Guia do usuário da Amazon VPC.
- Para acessar o cluster local pela rede local, a VPC deve estar associada à tabela de rotas do gateway local do Outpost. Para obter mais informações, consulte Associações de VPC no Guia do Usuário do AWS Outposts.

Requisitos e considerações para sub-redes

Quando você criar o cluster, especifique pelo menos uma sub-rede privada. Se você especificar mais de uma sub-rede, as instâncias do ambiente de gerenciamento do Kubernetes serão distribuídas uniformemente pelas sub-redes. Se mais de uma sub-rede for especificada, as sub-redes deverão estar no mesmo Outpost. Além disso, as sub-redes também devem ter rotas adequadas e permissões de grupo de segurança para se comunicarem entre si. As sub-redes que você especificar ao criar um cluster local devem atender aos requisitos a seguir:

- As sub-redes devem estar todas no mesmo Outpost lógico.
- Juntas, as sub-redes devem ter pelo menos três endereços IP disponíveis para as instâncias do ambiente de gerenciamento do Kubernetes. Se três sub-redes forem especificadas, cada uma delas deverá ter pelo menos um endereço IP disponível. Se duas sub-redes forem especificadas, cada uma delas deverá ter pelo menos dois endereços IP disponíveis. Se uma sub-rede for especificada, ela deverá ter pelo menos três endereços IP disponíveis.

- As sub-redes têm uma rota para o gateway local do rack do Outpost para acessar o servidor de API do Kubernetes pela rede local. Se as sub-redes não tiverem uma rota para o gateway local do rack do Outpost, você deverá se comunicar com o servidor de API do Kubernetes de dentro da VPC.
- As sub-redes devem utilizar uma nomenclatura baseada em endereço IP. A nomenclatura baseada em recursos do Amazon EC2 não é compatível com o Amazon EKS.

Acesso à sub-rede a serviços da AWS

As sub-redes privadas do cluster local nos postos avançados devem ser capazes de se comunicar com os serviços regionais do AWS. É possível conseguir isso ao usar um gateway NAT para acesso de saída à Internet ou, caso deseje manter todo o tráfego privado em sua VPC, ao usar endpoints da VPC de interface.

Como usar um gateway NAT

As sub-redes privadas do cluster local no Outposts devem ter uma tabela de rotas associada que tenha uma rota para um gateway NAT em uma sub-rede pública que esteja na zona de disponibilidade principal do Outpost. A sub-rede pública deve ter uma rota para um gateway da Internet. O gateway NAT permite acesso de saída à Internet e impede conexões de entrada não solicitadas da Internet para instâncias no Outpost.

Usar VPC endpoints da interface do

Se as sub-redes privadas do cluster local no Outposts não tiverem uma conexão de saída com a Internet ou se você desejar manter todo o tráfego privado em sua VPC, crie os endpoints da VPC de interface e o endpoint do gateway a seguir em uma sub-rede regional antes de criar seu cluster.

Endpoint	Tipo de endpoint
`com.amazonaws.` *`region-code`* `.ssm`	Interface
`com.amazonaws.` *`region-code`* `.ssmmessages`	Interface
`com.amazonaws.` *`region-code`* `.ec2messages`	Interface
`com.amazonaws.` *`region-code`* `.ec2`	Interface

Endpoint	Tipo de endpoint
com.amazonaws. *region-code* .secretsmanager	Interface
com.amazonaws. *region-code* .logs	Interface
com.amazonaws. *region-code* .sts	Interface
com.amazonaws. *region-code* .ecr.api	Interface
com.amazonaws. *region-code* .ecr.dkr	Interface
com.amazonaws. *region-code* .s3	Gateway

Os endpoints devem atender aos seguintes requisitos:

- Ter sido criado em uma sub-rede privada localizada na zona de disponibilidade principal do Outpost.
- Ter nomes DNS privados habilitados.
- Ter um grupo de segurança anexado que permita o tráfego HTTPS de entrada do intervalo CIDR da sub-rede privada do Outpost.

A criação de endpoints gera cobranças. Para obter mais informações, consulte Preços do AWS PrivateLink. Se o pod precisar de acesso a outros serviços da AWS, será necessário criar endpoints adicionais. Para obter uma lista abrangente de endpoints, consulte AWS services that integrate with AWS PrivateLink.

Crie uma VPC

Você pode criar uma VPC que atenda aos requisitos anteriores usando um dos seguintes modelos do AWS CloudFormation:

- Modelo 1: esse modelo cria uma VPC com uma sub-rede privada no Outpost e uma sub-rede pública na região da AWS. A sub-rede privada tem uma rota para a Internet por meio de um gateway NAT que reside na sub-rede pública na região AWS. Esse modelo pode ser usado para criar um cluster local em uma sub-rede com acesso de egresso à Internet.

- Modelo 2: esse modelo cria uma VPC com uma sub-rede privada no Outpost e o conjunto mínimo de endpoints de VPC necessários para criar um cluster local em uma sub-rede que não tenha acesso à Internet de entrada ou saída (também chamada de sub-rede privada).

Prepare clusters locais do Amazon EKS em AWS Outposts para desconexões de rede

Se a sua rede local tiver perdido a conectividade com o AWS Cloud, você poderá continuar a usar o cluster local do Amazon EKS em um Outpost. Este tópico aborda como você pode preparar o cluster local para desconexões de rede e as considerações relacionadas.

- Clusters locais permitem estabilidade e operações contínuas durante desconexões temporárias e não planejadas da rede. AWS O Outposts continua sendo uma oferta totalmente conectada que atua como uma extensão da Nuvem AWS no seu data center. Em caso de desconexão da rede entre o Outpost e o AWS Cloud, recomendamos tentar restaurar a conexão. Para obter instruções, consulte a lista de verificação de solução de problemas da rede do rack do AWS Outposts no Guia do usuário do AWS Outposts. Para obter informações sobre como solucionar problemas de clusters locais, consulte the section called “Solução de problemas de clusters”.
- Os Outposts emitem uma métrica `ConnectedStatus` que você pode usar para monitorar o estado da conectividade do Outpost. Para obter mais informações, consulte Outposts Metrics no Guia do usuário do AWS Outposts.
- Os clusters locais usam o IAM como mecanismo de autenticação padrão usando o autenticador do AWS Identity and Access Management para Kubernetes. O IAM não está disponível durante desconexões de rede. Assim, os clusters locais são compatíveis com um mecanismo de autenticação alternativo usando certificados `x.509` que você pode usar para se conectar ao cluster durante desconexões de rede. Para obter informações sobre como obter e usar um certificado `x.509` para o cluster, consulte the section called “Autenticar no cluster local durante uma desconexão de rede”.
- Caso não você consiga acessar o Route 53 durante desconexões de rede, considere o uso de servidores DNS locais no ambiente on-premises. As instâncias do ambiente de gerenciamento do Kubernetes usam endereços IP estáticos. Você pode configurar os hosts que usa para se conectar ao cluster com o nome do host do endpoint e os endereços IP como uma alternativa ao uso de servidores DNS locais. Para obter mais informações, consulte DNS no Guia do usuário do AWS Outposts.

- Se você esperar aumentos no tráfego das aplicações durante desconexões de rede, pode provisionar capacidade computacional adicional no cluster quando estiver conectado à nuvem. As instâncias do Amazon EC2 estão incluídas no preço do AWS Outposts. Portanto, a execução de instâncias adicionais não afeta o custo de uso da AWS.
- Durante desconexões de rede para permitir operações de criação, atualização e escalação de workloads, as imagens de contêiner da aplicação devem estar acessíveis pela rede local e o cluster deve ter capacidade suficiente. Os clusters locais não hospedam um registro de contêiner para você. Se os pods foram executados anteriormente nesses nós, as imagens de contêineres estão armazenadas em cache nos nós. Se você normalmente extrair imagens de contêiner da aplicação do Amazon ECR na nuvem, considere executar um cache ou registro local. Um cache ou registro local será útil se você precisar de operações de criação, atualização e escalação para recursos de workload durante desconexões de rede.
- Os clusters locais usam o Amazon EBS como a classe de armazenamento padrão para volumes persistentes e o driver da CSI do Amazon EBS para gerenciar o ciclo de vida dos volumes persistentes do Amazon EBS. Durante as desconexões de rede, os pods que são baseados no Amazon EBS não podem ser criados, atualizados nem escalados. Isso porque essas operações exigem chamadas para a API do Amazon EBS na nuvem. Se você estiver implantando workloads com estado em clusters locais e precisar de operações de criação, atualização ou escalabilidade durante desconexões de rede, considere usar um mecanismo de armazenamento alternativo.
- Os snapshots do Amazon EBS não poderão ser criados ou excluídos se o AWS Outposts não puder acessar as APIs relevantes da região do AWS (como as APIs do Amazon EBS ou do Amazon S3).
- Ao integrar o ALB (Ingress) com o AWS Certificate Manager (ACM), os certificados são enviados e armazenados na memória da instância do AWS Outposts ALB Compute. A terminação TLS atual continuará a funcionar no caso de uma desconexão da região AWS. As operações de mutação nesse contexto falharão (como novas definições de entrada, novas operações de API de certificados baseados em ACM, escala de computação do ALB ou rotação de certificados). Para obter mais informações, consulte Solução de problemas de renovação de certificados gerenciados no Guia do usuário do AWS Certificate Manager.
- Os logs do ambiente de gerenciamento do Amazon EKS são armazenados em cache localmente nas instâncias do ambiente de gerenciamento do Kubernetes durante as desconexões de rede. Após a reconexão, os logs são enviados para o CloudWatch Logs na região da AWS principal. Você pode usar o Prometheus, o Grafana ou as soluções de parceiros do Amazon EKS para monitorar o cluster localmente usando o endpoint de métricas do servidor de API do Kubernetes ou usando o Fluent Bit para logs.

- Se você estiver usando o AWS Load Balancer Controller no Outposts para tráfego das aplicações, os pods existentes comandados pelo AWS Load Balancer Controller continuarão a receber tráfego durante as desconexões de rede. Os novos pods criados durante as desconexões de rede não recebem tráfego até que o Outpost seja reconectado à Nuvem AWS. Considere a possibilidade de definir a contagem de réplicas para suas aplicações enquanto estiver conectado ao AWS Cloud para acomodar suas necessidades de dimensionamento durante as desconexões de rede.
- O plug-in CNI da Amazon VPC para Kubernetes usa como padrão o modo IP secundário. Ele é configurado com `WARM_ENI_TARGET = 1`, o que permite que o plug-in mantenha disponível "uma interface de rede totalmente elástica" de endereços IP. Considere mudar os valores `WARM_ENI_TARGET`,`WARM_IP_TARGET` e `MINIMUM_IP_TARGET` de acordo com suas necessidades de escalonamento durante um estado desconectado. Para obter mais informações, consulte o arquivo readme do plug-in no GitHub. Para obter uma lista do número máximo de pods que é compatível com cada tipo de instância, consulte o arquivo eni-max-pods.txt no GitHub.

Autenticar no cluster local durante uma desconexão de rede

O AWS Identity and Access Management (IAM) não está disponível durante desconexões de rede. Você não pode autenticar no cluster local usando credenciais do IAM enquanto está desconectado. Porém, você pode se conectar ao cluster pela rede local usando certificados `x509` quando estiver desconectado. Você precisa baixar e armazenar o certificado `X509` de um cliente para usar durante as desconexões. Neste tópico, você aprenderá a criar e usar o certificado para autenticação no cluster quando ele está em um estado desconectado.

1. Crie uma solicitação de assinatura do certificado.
   a. Gere uma solicitação de assinatura do certificado.

```
openssl req -new -newkey rsa:4096 -nodes -days 365 \
    -keyout admin.key -out admin.csr -subj "/CN=admin"
```

   b. Crie uma solicitação de assinatura do certificado no Kubernetes.

```
BASE64_CSR=$(cat admin.csr | base64 -w 0)
cat << EOF > admin-csr.yaml
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: admin-csr
spec:
```

```
  signerName: kubernetes.io/kube-apiserver-client
  request: ${BASE64_CSR}
  usages:
  - client auth
EOF
```

2. Crie uma solicitação de assinatura do certificado usando o `kubectl`.

```
kubectl create -f admin-csr.yaml
```

3. Verifique o status da solicitação de assinatura do certificado.

```
kubectl get csr admin-csr
```

   Veja um exemplo de saída abaixo.

```
NAME       AGE   REQUESTOR                      CONDITION
admin-csr  11m   kubernetes-admin               Pending
```

   O Kubernetes criou a solicitação de assinatura do certificado.

4. Aprove a solicitação de assinatura do certificado.

```
kubectl certificate approve admin-csr
```

5. Verifique novamente o status da solicitação de assinatura do certificado para aprovação.

```
kubectl get csr admin-csr
```

   Veja um exemplo de saída abaixo.

```
NAME       AGE   REQUESTOR                    CONDITION
admin-csr  11m   kubernetes-admin             Approved
```

6. Recupere e verifique o certificado.

   a. Recupere o certificado.

```
kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode >
 admin.crt
```

   b. Verifique o certificado.

```
cat admin.crt
```

7. Crie uma vinculação de função de cluster para um usuário `admin`.

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
    --user=admin --group=system:masters
```

8. Gere um kubeconfig com escopo de usuário para um estado desconectado.

   Você pode gerar um arquivo `kubeconfig` usando os certificados `admin` baixados. Substitua *`my-cluster`* e *`apiserver-endpoint`* nos comandos a seguir.

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.certificateAuthority" \
    --output text | base64 --decode > ca.crt
```

```
kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
    --certificate-authority=ca.crt --server apiserver-endpoint --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
    --client-certificate=admin.crt --client-key=admin.key --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
    --cluster my-cluster --user admin
```

```
kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
```

9. Visualize o arquivo `kubeconfig`.

```
kubectl get nodes --kubeconfig admin.kubeconfig
```

10. Se você já tiver serviços em produção no Outpost, pule esta etapa. Se o Amazon EKS for o único serviço em execução no Outpost e o Outpost não estiver em produção no momento, você poderá simular uma desconexão de rede. Antes de entrar em produção com o cluster local, simule uma desconexão para ter certeza de que pode acessar o cluster mesmo quando ele está no estado desconectado.

a. Aplique regras de firewall nos dispositivos de rede que conectam seu Outpost à região da AWS. Com isso, o link de serviço do Outpost é desconectado. Você não pode criar novas instâncias. As instâncias em execução no momento perdem a conectividade com a região AWS e a Internet.

b. Você pode testar a conexão com o cluster local enquanto estiver desconectado usando o certificado `x509`. Certifique-se de alterar o `kubeconfig` para o `admin.kubeconfig` que você criou em uma etapa anterior. Substitua *`my-cluster`* pelo nome do cluster local.

```
kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
```

Se você notar algum problema com os clusters locais enquanto eles estiverem no estado desconectado, será recomendável abrir um tíquete de suporte.

Selecione tipos de instância e grupos de posicionamento para clusters do Amazon EKS em AWS Outposts com base em considerações de capacidade

Este tópico fornece orientação para a seleção de tipo de instância do ambiente de gerenciamento do Kubernetes e (opcionalmente) o uso de grupos de posicionamento para atender aos requisitos de alta disponibilidade do cluster local do Amazon EKS em um Outpost.

Antes de selecionar um tipo de instância (como `m5`, `c5` ou `r5`) para usar no ambiente de gerenciamento do Kubernetes do cluster local no Outposts, confirme os tipos de instâncias que estão disponíveis na configuração do Outpost. Depois que você identificar os tipos de instância disponíveis, selecione o tamanho da instância (por exemplo, `large`, `xlarge` ou `2xlarge`) com base no número de nós que as workloads exigem. A tabela a seguir fornece recomendações para a escolha do tamanho de uma instância.

> **Note**
>
> Os tamanhos das instâncias devem ser alocados nos Outposts. Certifique-se de ter capacidade suficiente para três instâncias do tamanho disponível no Outposts durante a vida útil do cluster local. Para obter uma lista dos tipos de instância do Amazon EC2 disponíveis, consulte as seções de computação e armazenamento nos recursos de rack do AWS Outposts.

Número de nós	Tamanho da instância do ambiente de gerenciamento do Kubernetes
1–20	`large`
21–100	`xlarge`
101–250	`2xlarge`
251–500	`4xlarge`

O armazenamento para o ambiente de gerenciamento do Kubernetes requer 246 GB de armazenamento do Amazon EBS para cada cluster local para atender ao IOPS necessário para `etcd`. Na criação do cluster local, os volumes do Amazon EBS são provisionados automaticamente.

Posicionamento do ambiente de gerenciamento

Quando você não especifica um grupo de posicionamento com a propriedade `OutpostConfig.ControlPlanePlacement.GroupName`, as instâncias do Amazon EC2 provisionadas para o ambiente de gerenciamento do Kubernetes não recebem nenhuma imposição específica de posicionamento de hardware em toda a capacidade subjacente disponível no Outpost.

Você pode usar grupos de posicionamento para atender aos requisitos de alta disponibilidade do cluster local do Amazon EKS em um Outpost. Ao especificar um grupo de posicionamento durante a criação do cluster, você influencia o posicionamento das instâncias do ambiente de gerenciamento do Kubernetes. As instâncias são distribuídas pelo hardware subjacente independente (racks ou hosts), minimizando o impacto das instância correlacionadas no caso de falhas de hardware.

O tipo de distribuição que você pode configurar depende do número de racks do Outpost que você tem na implantação.

- Implantações com um ou dois racks físicos em um único Outpost lógico: é necessário ter pelo menos três hosts configurados com o tipo de instância que você escolher para as suas instâncias do ambiente de gerenciamento do Kubernetes. Um grupo com posicionamento distribuído usando a distribuição em nível de host garante que todas as instâncias do ambiente de gerenciamento do Kubernetes sejam executadas em hosts distintos, em todos os racks subjacentes disponíveis na implantação do Outpost.

- Implantações com três ou mais racks físicos em um único Outpost lógico: é necessário ter, pelo menos, três hosts configurados com o tipo de instância que você escolher para as suas instâncias do ambiente de gerenciamento do Kubernetes. Um grupo com posicionamento distribuído usando a distribuição em nível de rack garante que todas as instâncias do ambiente de gerenciamento do Kubernetes sejam executadas em racks distintos na implantação do Outpost. Ou então, você pode usar o grupo de posicionamento de distribuição no nível dos hosts, conforme descrito na opção anterior.

Você é responsável por criar o grupo de posicionamento desejado. Você especifica o grupo de posicionamento ao chamar a API `CreateCluster`. Para obter mais informações sobre grupos de posicionamento e como criá-los, consulte Grupos de posicionamento no Guia do usuário do Amazon EC2.

- Quando um grupo de posicionamento é especificado, deve haver capacidade disponível alocada no Outpost para a criação bem-sucedida de um cluster local do Amazon EKS. A capacidade varia dependendo do tipo de distribuição que você usa: distribuição em hosts ou em racks. Se não houver capacidade suficiente, o cluster permanecerá no estado `Creating`. Você pode verificar o endereço `Insufficient Capacity Error` no campo health da resposta da API DescribeCluster. Você deve liberar capacidade para que o processo de criação prossiga.
- Durante as atualizações de versão e plataforma do cluster local do Amazon EKS, as instâncias do ambiente de gerenciamento do Kubernetes do cluster são substituídas por novas instâncias usando uma estratégia de atualização contínua. Durante esse processo de substituição, cada instância do ambiente de gerenciamento é encerrada, liberando o slot correspondente. Uma nova instância atualizada é provisionada em seu lugar. A instância atualizada pode ser posicionada no slot que foi liberado. Se o slot for consumido por outra instância não relacionada e não restar mais capacidade para atender ao requisito de topologia de distribuição, o cluster permanecerá no estado `Updating`. Você pode ver o respectivo `Insufficient Capacity Error` no campo health da resposta da API DescribeCluster. Você deve liberar capacidade para que o processo de atualização possa prosseguir e restabelecer os altos níveis de disponibilidade anteriores.
- É possível criar até 500 grupos de posicionamento por conta em cada região da AWS. Para obter mais informações, consulte Regras gerais e limitações no Guia do usuário do Amazon EC2.

Solucionar problemas de clusters locais do Amazon EKS em AWS Outposts

Este tópico aborda alguns erros comuns que você pode encontrar ao usar clusters locais e como solucionar esses problemas. Os clusters locais são semelhantes aos clusters do Amazon EKS na nuvem, mas existem algumas diferenças na forma como eles são gerenciados pelo Amazon EKS.

> **Important**
>
> Nunca encerre nenhuma instância do ambiente de gerenciamento do `Kubernetes` do cluster local gerenciado do EKS em execução no Outpost, a menos que seja explicitamente instruído pelo AWS Support. O encerramento dessas instâncias representa um risco à disponibilidade do serviço de cluster local, incluindo a perda do cluster local caso várias instâncias sejam encerradas simultaneamente. As instâncias do ambiente de gerenciamento do `Kubernetes` de cluster local do EKS são identificadas pela tag `eks-local:controlplane-name` no console de instâncias do EC2.

Comportamento da API

Os clusters locais são criados por meio da API do Amazon EKS, mas são executados de maneira assíncrona. Isso significa que as solicitações à API do Amazon EKS retornam imediatamente para os clusters locais. Porém, essas solicitações podem ser bem-sucedidas, antecipar-se à falha devido a erros de validação de entrada ou falhar e ter erros de validação descritivos. Esse comportamento é semelhante ao da API do Kubernetes.

Os clusters locais não fazem a transição para um status `FAILED`. O Amazon EKS tenta continuamente reconciliar o estado do cluster com o estado desejado solicitado pelo usuário. Como resultado, um cluster local pode permanecer no estado `CREATING` por um período prolongado até que o problema subjacente seja resolvido.

Descrever o campo de integridade do cluster

Os problemas do cluster local podem ser descobertos usando o comando describe-cluster do Amazon EKS AWS CLI. Problemas de cluster local são revelados pelo campo `cluster.health` da resposta do comando `describe-cluster`. A mensagem contida nesse campo inclui um código de erro, uma mensagem descritiva e IDs de recursos relacionados. Essas informações estão disponíveis somente por meio da API do Amazon EKS e da AWS CLI. No exemplo a seguir, substitua *`my-cluster`* pelo nome do cluster local.

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

Veja um exemplo de saída abaixo.

```
{
    "issues": [
        {
            "code": "ConfigurationConflict",
            "message": "The instance type 'm5.large' is not supported in Outpost 'my-
outpost-arn'.",
            "resourceIds": [
                "my-cluster-arn"
            ]
        }
    ]
}
```

Se o problema não puder ser reparado, talvez seja necessário excluir o cluster local e criar um novo. Por exemplo, tentar provisionar um cluster com um tipo de instância que não está disponível no Outpost. A tabela a seguir inclui erros comuns relacionados à integridade.

Cenário de erro	Código	Mensagem	ResourceIds
Não foi possível encontrar as sub-redes fornecidas.	`ResourceNotFound`	`The subnet ID` *`subnet-id`* `does not exist`	Todos os IDs de sub-rede fornecidos
As sub-redes fornecidas não pertencem à mesma VPC.	`ConfigurationConflict`	`Subnets specified must belong to the same VPC`	Todos os IDs de sub-rede fornecidos
Algumas sub-redes fornecidas não pertencem ao Outpost especificado.	`ConfigurationConflict`	`Subnet` *`subnet-id`* `expected to be in` *`outpost-arn`* `, but is in` *`other-outpost-arn`*	ID de sub-rede problemática

Cenário de erro	Código	Mensagem	ResourceIds
Algumas sub-redes fornecidas não pertencem a nenhum Outpost.	`ConfigurationConflict`	`Subnet subnet-id is not part of any Outpost`	ID de sub-rede problemática
Algumas sub-redes fornecidas não têm endereços livres suficientes para a criação de interfaces de rede elásticas para as instâncias do ambiente de gerenciamento.	`ResourceLimitExceeded`	`The specified subnet does not have enough free addresses to satisfy the request.`	ID de sub-rede problemática
O tipo de instância do ambiente de gerenciamento especificado não é compatível com o Outpost.	`ConfigurationConflict`	`The instance type type is not supported in Outpost outpost-arn`	ARN do cluster

Cenário de erro	Código	Mensagem	ResourceIds
Você encerrou uma instância do Amazon EC2 do ambiente de gerenciamento ou `run-instance` teve êxito, mas o estado sofreu alterações para `Terminated`. Isso pode acontecer por um período após a reconexão do Outpost e os erros internos do Amazon EBS causarem uma falha no fluxo de trabalho interno do Amazon EC2.	`InternalFailure`	`EC2 instance state "Terminated" is unexpected`	ARN do cluster
Você não tem capacidade suficiente no Outpost. Isso também pode acontecer quando um cluster estiver sendo criado se um Outpost for desconectado da região da AWS.	`ResourceLimitExceeded`	`There is not enough capacity on the Outpost to launch or start the instance.`	ARN do cluster
A conta excedeu a cota de grupo de segurança.	`ResourceLimitExceeded`	Mensagem de erro retornada pela API do Amazon EC2	ID da VPC de destino

Cenário de erro	Código	Mensagem	ResourceIds
A conta excedeu a cota de interface de rede elástica.	`ResourceLimitExceeded`	Mensagem de erro retornada pela API do Amazon EC2	ID da sub-rede de destino
As instâncias do ambiente de gerenciamento não eram acessíveis por meio do AWS Systems Manager. Para saber a resolução, consulte the section called “As instâncias do ambiente de gerenciamento não podem ser acessadas por meio do AWS Systems Manager”.	`ClusterUnreachable`	As instâncias do ambiente de gerenciamento do Amazon EKS não podem ser acessadas por meio do SSM. Verifique a configuração do SSM e da rede e consulte a documentação de solução de problemas do EKS no Outposts.	IDs de instâncias do Amazon EC2
Ocorreu um erro ao obter detalhes de um grupo de segurança gerenciado ou de uma interface de rede elástica.	Com base no código de erro do cliente Amazon EC2.	Mensagem de erro retornada pela API do Amazon EC2	Todos os IDs de grupos de segurança gerenciados

Cenário de erro	Código	Mensagem	ResourceIds
Ocorreu um erro ao autorizar ou revogar as regras de ingresso de grupo de segurança. Isso se aplica aos grupos de segurança do cluster e do ambiente de gerenciamento.	Com base no código de erro do cliente Amazon EC2.	Mensagem de erro retornada pela API do Amazon EC2	ID de grupo de segurança problemático
Ocorreu um erro ao excluir uma interface de rede elástica de uma instância do ambiente de gerenciamento.	Com base no código de erro do cliente Amazon EC2.	Mensagem de erro retornada pela API do Amazon EC2	ID de interface de rede elástica problemática

A tabela a seguir lista os erros de outros serviços do AWS que são apresentados no campo health da resposta do `describe-cluster`.

Código de erro do Amazon EC2	Código de problema de integridade do cluster	Descrição
`AuthFailure`	`AccessDenied`	Esse erro pode ocorrer por vários motivos. O motivo mais comum é que você acidentalmente removeu do serviço do ambiente de gerenciamento uma tag que o serviço usa para reduzir o escopo da política de perfil vinculada. Se isso ocorrer, o Amazon EKS não poderá mais gerenciar e

Código de erro do Amazon EC2	Código de problema de integridade do cluster	Descrição
		monitorar esses recursos da AWS.
`UnauthorizedOperation`	`AccessDenied`	Esse erro pode ocorrer por vários motivos. O motivo mais comum é que você acidentalmente removeu do serviço do ambiente de gerenciamento uma tag que o serviço usa para reduzir o escopo da política de perfil vinculada. Se isso ocorrer, o Amazon EKS não poderá mais gerenciar e monitorar esses recursos da AWS.
`InvalidSubnetID.NotFound`	`ResourceNotFound`	Esse erro ocorre quando o ID da sub-rede para as regras de ingresso de um grupo de segurança não pode ser encontrado.
`InvalidPermission.NotFound`	`ResourceNotFound`	Esse erro ocorre quando as permissões para as regras de ingresso de um grupo de segurança não estão corretas.
`InvalidGroup.NotFound`	`ResourceNotFound`	Esse erro ocorre quando o grupo das regras de ingresso de um grupo de segurança não pode ser encontrado.

Código de erro do Amazon EC2	Código de problema de integridade do cluster	Descrição
`InvalidNetworkInterfaceID.NotFound`	`ResourceNotFound`	Esse erro ocorre quando o ID da interface de rede para as regras de ingresso de um grupo de segurança não pode ser encontrado.
`InsufficientFreeAddressesInSubnet`	`ResourceLimitExceeded`	Esse erro ocorre quando a cota de recursos da sub-rede é excedida.
`InsufficientCapacityOnOutpost`	`ResourceLimitExceeded`	Esse erro ocorre quando a cota de capacidade do outpost é excedida.
`NetworkInterfaceLimitExceeded`	`ResourceLimitExceeded`	Esse erro ocorre quando a cota de interface de rede elástica é excedida.
`SecurityGroupLimitExceeded`	`ResourceLimitExceeded`	Esse erro ocorre quando a cota de capacidade do grupo de segurança é excedida.

Código de erro do Amazon EC2	Código de problema de integridade do cluster	Descrição
`VcpuLimitExceeded`	`ResourceLimitExceeded`	Isso é observado na criação de uma instância do Amazon EC2 em uma nova conta. O procedimento pode ser semelhante ao seguinte: `"You have requested more vCPU capacity than your current vCPU limit of 32 allows for the instance bucket that the specified instance type belongs to. Please visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to this limit."`
`InvalidParameterValue`	`ConfigurationConflict`	O Amazon EC2 retornará esse código de erro se não houver suporte no Outpost para o tipo de instância especificado.
Todas as outras falhas	`InternalFailure`	Nenhum

Impossível criar ou modificar clusters

Os clusters locais exigem permissões e políticas diferentes das exigidas pelos clusters do Amazon EKS hospedados na nuvem. Quando a criação do cluster falhar e gera um erro `InvalidPermissions`, verifique novamente se o perfil do cluster que você está usando tem a política gerenciada AmazonEKSLocalOutpostClusterPolicy anexada a ele. Todas as outras chamadas de API exigem o mesmo conjunto de permissões que os clusters do Amazon EKS na nuvem.

O cluster está travado no estado **`CREATING`**

O tempo necessário para criar um cluster local varia dependendo de vários fatores. Esses fatores incluem a configuração da rede, a configuração do Outpost e a configuração do cluster. Em geral, um cluster local é criado e passa para o status `ACTIVE` dentro de 15 a 20 minutos. Se um cluster local permanecer no estado `CREATING`, você poderá chamar `describe-cluster` para obter informações sobre a causa no campo de saída `cluster.health`.

Os problemas mais comuns são os seguintes:

- Seu cluster não pode se conectar à instância do ambiente de gerenciamento da região AWS em que o Systems Manager está. Você pode verificar isso chamando `aws ssm start-session --target` *`instance-id`* em um bastion host na região. Se esse comando não funcionar, verifique se o Systems Manager está sendo executado na instância do ambiente de gerenciamento. Outra solução de contorno é excluir o cluster e depois recriá-lo.
- As instâncias do ambiente de gerenciamento falham na criação devido às permissões da chave do KMS para volumes do EBS. Com chaves do KMS gerenciadas pelo usuário para volumes criptografados do EBS, as instâncias do ambiente de gerenciamento serão encerradas se a chave não estiver acessível. Se as instâncias forem encerradas, alterne para uma chave do KMS gerenciada pela AWS ou garanta que sua política de chaves gerenciadas pelo usuário conceda as permissões necessárias para o perfil do cluster.
- As instâncias do ambiente de gerenciamento do Systems Manager podem não ter acesso à Internet. Verifique se a sub-rede que você forneceu ao criar o cluster tem um gateway NAT e uma VPC com um gateway da internet. Use o analisador de acessibilidade da VPC para verificar se a instância do ambiente de gerenciamento pode acessar o gateway da Internet. Para obter mais informações, consulte Getting started with VPC Reachability Analyzer (Conceitos básicos do VPC Reachability Analyzer).
- O ARN de perfil fornecido não tem políticas. Verifique se a política gerenciada da AWS: AmazonEKSLocalOutpostClusterPolicy foi removida do perfil. Isso também pode ocorrer se uma pilha do AWS CloudFormation estiver mal configurada.
- Todas as sub-redes fornecidas devem estar associadas ao mesmo Outpost e devem alcançar umas às outras. Quando várias sub-redes são especificadas durante a criação do cluster, o Amazon EKS tenta distribuir as instâncias do ambiente de gerenciamento entre várias sub-redes.
- Os grupos de segurança gerenciados pelo Amazon EKS são aplicados à interface de rede elástica. No entanto, outros elementos da configuração, como regras de firewall NACL, podem entrar em conflito com as regras da interface de rede elástica.

A VPC e o DNS da sub-rede estão mal configurados ou falta a configuração

Revisão Crie um VPC e sub-redes para clusters do Amazon EKS em AWS Outposts.

O cluster está travado no estado **`UPDATING`**

O Amazon EKS atualiza automaticamente todos os clusters locais existentes para a versão mais recente da plataforma para a versão secundária correspondente do Kubernetes. Para obter mais informações sobre versões da plataforma, consulte the section called “Versões da plataforma do EKS”.

Durante a implantação automática da versão da plataforma, o status do cluster muda para `UPDATING`. O processo de atualização consiste na substituição de todas as instâncias do ambiente de gerenciamento do Kubernetes por novas contendo os patches de segurança e correções de bugs mais recentes lançados para a respectiva versão secundária do Kubernetes. Em geral, um processo de atualização da plataforma do cluster local é concluído em menos de trinta minutos, e o cluster volta ao status `ACTIVE`. Se um cluster local permanecer no estado `UPDATING` por um período prolongado, você poderá chamar `describe-cluster` para verificar as informações sobre a causa no campo de saída `cluster.health`.

O Amazon EKS garante que pelo menos duas em cada três instâncias do ambiente de gerenciamento do Kubernetes sejam nós de cluster íntegros e operacionais para manter a disponibilidade do cluster local e evitar a interrupção do serviço. Se um cluster local estiver paralisado no estado `UPDATING`, geralmente é porque há algum problema de infraestrutura ou configuração que impede que a disponibilidade mínima de duas instâncias seja garantida caso o processo continue. Portanto, o processo de atualização para de avançar para proteger o cluster local contra a interrupção do serviço.

É importante solucionar o problema de um cluster local preso no status `UPDATING` e tratar a causa raiz para que o processo de atualização possa ser concluído, e restaurar o cluster local de volta ao status `ACTIVE` com a alta disponibilidade de três instâncias do ambiente de gerenciamento do Kubernetes.

Não encerre nenhuma instância do `Kubernetes` gerenciada do cluster local do EKS no Outposts, a menos que seja explicitamente instruído pelo AWS Support. Isso é especialmente importante para clusters locais presos no estado `UPDATING`, pois há uma grande probabilidade de que outros nós do ambiente de gerenciamento não estejam completamente íntegros, e o encerramento da instância errada pode causar interrupção do serviço e risco de perda de dados do cluster local.

Os problemas mais comuns são os seguintes:

- Uma ou mais instâncias do ambiente de gerenciamento não conseguem se conectar ao Systems Manager devido a uma alteração na configuração de rede desde que o cluster local foi criado pela primeira vez. Você pode verificar isso chamando `aws ssm start-session --target instance-id` em um bastion host na região. Se esse comando não funcionar, verifique se o Systems Manager está sendo executado na instância do ambiente de gerenciamento.
- As novas instâncias do ambiente de gerenciamento falham ao serem criadas devido às permissões da chave do KMS para volumes do EBS. Com chaves do KMS gerenciadas pelo usuário para volumes criptografados do EBS, as instâncias do ambiente de gerenciamento serão encerradas se a chave não estiver acessível. Se as instâncias forem encerradas, alterne para uma chave do KMS gerenciada pela AWS ou garanta que sua política de chaves gerenciadas pelo usuário conceda as permissões necessárias para o perfil do cluster.
- As instâncias do ambiente de gerenciamento do Systems Manager podem não ter perdido o acesso à internet. Verifique se a sub-rede que você forneceu ao criar o cluster tem um gateway NAT e uma VPC com um gateway da internet. Use o analisador de acessibilidade da VPC para verificar se a instância do ambiente de gerenciamento pode acessar o gateway da Internet. Para obter mais informações, consulte Getting started with VPC Reachability Analyzer (Conceitos básicos do VPC Reachability Analyzer). Se as redes privadas não tiverem conexão de saída com a internet, certifique-se de que todos os endpoints da VPC e endpoints de gateway necessários ainda estejam presentes na sub-rede regional do cluster (consulte the section called “Acesso à sub-rede a serviços da AWS”).
- O ARN de perfil fornecido não tem políticas. Verifique se a política gerenciada da AWS: AmazonEKSLocalOutpostClusterPolicy foi removida do perfil.
- Uma das novas instâncias do ambiente de gerenciamento do Kubernetes pode ter sofrido uma falha inesperada de inicialização. Crie um tíquete no AWS Support Center para obter mais orientações sobre solução de problemas e a coleta de logs neste caso excepcional.

Não é possível integrar nós a um cluster

- Problemas de AMI:
  - Você está usando uma AMI incompatível. Somente AMIs do Amazon Linux 2 otimizadas para o Amazon EKS são compatíveis (`amazon-linux-2`, `amazon-linux-2-gpu` e `amazon-linux-2-arm64`). Se tentar integrar nós do AL2023 a LocalClusters do EKS no AWS Outposts, os nós falharão ao ingressar no cluster. Para mais informações, consulte Criar nós Amazon Linux no AWS Outposts.

  - Se você usou um modelo do AWS CloudFormation para criar seus nós, verifique se ele não estava usando uma AMI sem suporte.
- O `ConfigMap` do autenticador do AWS IAM está ausente: nesse caso, é necessário criá-lo. Para ter mais informações, consulte the section called “Como aplicar o `ConfigMap aws-auth` ao seu cluster” .
- O grupo de segurança errado é usado: certifique-se de usar `eks-cluster-sg-`*`cluster-name`*`-`*`uniqueid`* para o grupo de segurança dos nós de processamento. O grupo de segurança selecionado é alterado pelo AWS CloudFormation para permitir um novo grupo de segurança sempre que a pilha for usada.
- Seguir etapas inesperadas de uma VPC de link privado: são especificados dados de CA (`--b64-cluster-ca`) ou endpoint de API (`--apiserver-endpoint`) incorretos.

Coletar logs

Quando um Outpost é desconectado da região da AWS à qual está associado, o cluster do Kubernetes provavelmente continuará funcionando normalmente. No entanto, se o cluster não funcionar corretamente, siga as etapas de solução de problemas em Preparar clusters locais do Amazon EKS no AWS Outposts para desconexões de rede. Se encontrar outros problemas, entre em contato com o suporte AWS. AWS O suporte pode orientar você sobre como baixar e executar uma ferramenta de coleta de logs. Dessa forma, você pode coletar logs das instâncias do ambiente de gerenciamento do cluster do Kubernetes e enviá-los para o AWS Support para uma investigação adicional.

As instâncias do ambiente de gerenciamento não podem ser acessadas por meio do AWS Systems Manager

Quando as instâncias do ambiente de gerenciamento do Amazon EKS não podem ser acessadas pelo AWS Systems Manager (Systems Manager), o Amazon EKS exibe o seguinte erro para o seu cluster.

```
Amazon EKS control plane instances are not reachable through SSM. Please verify your
 SSM and network configuration, and reference the EKS on Outposts troubleshooting
 documentation.
```

Para resolver esse problema, verifique se a VPC e as sub-redes atendem aos requisitos em Criar uma VPC e sub-redes para clusters do Amazon EKS em AWS Outposts e se você concluiu as etapas em Configurar o Session Manager no Guia do usuário do AWS Systems Manager.

Criar nós Amazon Linux no AWS Outposts

Important

Os clusters locais do Amazon EKS no Outposts fornecem suporte apenas nós criados com base nas seguintes AMIs do Amazon Linux 2023 otimizadas para o Amazon EKS:

- Amazon Linux 2023 padrão (`amazon-linux-2023/x86_64/standard`)
- Amazon Linux 2023 com aceleração para a NVIDIA (`amazon-linux-2023/x86_64/nvidia`)
- Amazon Linux 2023 com aceleração para o Neuron (`amazon-linux-2023/x86_64/neuron`)

A AWS encerrou o suporte para as AMIs otimizadas e aceleradas para o AL2 do EKS, com efeito a partir de 26 de novembro de 2025. Embora você possa continuar usando as AMIs do AL2 do EKS após a data de fim do suporte (EOS) (26 de novembro de 2025), o EKS não lançará mais nenhuma nova versão do Kubernetes ou atualizações das AMIs do AL2, incluindo versões menores, patches e correções de bugs após essa data. Consulte aqui para obter mais informações sobre a descontinuação do AL2.

Este tópico descreve como iniciar um grupo do Auto Scaling de nós do Amazon Linux em um Outpost que são registrados no cluster do Amazon EKS. O cluster pode estar na Nuvem AWS ou em um Outpost.

- Um Outpost existente. Para obter mais informações, consulte O que é AWS Outposts.
- Um cluster existente do Amazon EKS. Para implementar um cluster no AWS Cloud, consulte the section called “Criar um cluster”. Para implantar um cluster em um Outpost, consulte the section called “Executar clusters locais”.
- Suponha que você esteja criando seus nós em um cluster na nuvem AWS e que tenha sub-redes na região AWS, na qual você tem AWS Outposts, AWS Wavelength ou AWS Local Zones ativados. Assim sendo, essas sub-redes não devem ter sido transmitidas quando você criou o cluster. Se você estiver criando nós em um cluster em um Outpost, deverá ter transmitido uma sub-rede do Outpost ao criar o cluster.
- (Recomendado para clusters na Nuvem AWS) O plug-in CNI da Amazon VPC para o complemento do Kubernetes configurado com seu próprio perfil do IAM que tem a política do IAM necessária

anexada a ele. Para obter mais informações, consulte the section called “Configuração para IRSA”. Clusters locais não são compatíveis com perfis do IAM para contas de serviço.

Você pode criar um grupo de nós do Amazon Linux autogerenciado com `eksctl` ou Console de gerenciamento da AWS (com um modelo AWS CloudFormation). Você também pode usar o Terraforma.

Você pode criar um grupo de nós autogerenciados para o cluster local com as seguintes ferramentas descritas nesta página:

- the section called “`eksctl`”
- the section called “Console de gerenciamento da AWS”

⚠ Important

- Um grupo de nós autogerenciados inclui instâncias do Amazon EC2 em sua conta. Essas instâncias não são atualizadas automaticamente quando você ou o Amazon EKS atualizam a versão do seu ambiente de gerenciamento em seu nome. Um grupo de nós autogerenciado não tem nenhuma indicação no console de que ele precisa ser atualizado. Você pode visualizar a versão do `kubelet` instalada em um nó selecionando o nó na lista Nodes (Nós) da guia Overview (Visão geral) do cluster para determinar quais nós precisam ser atualizados. Você deve atualizar manualmente os nós. Para obter mais informações, consulte the section called “Métodos de atualização”.
- Os certificados usados pelo kubelet em seus nós autogerenciados são emitidos com um ano de expiração. Por padrão, a rotação de certificados não está habilitada (consulte: https://kubernetes.io/docs/reference/config-api/kubelet-config.v1beta1/#kubelet-config-k8s-io-v1beta1-KubeletConfiguration). Isso significa que se você tiver um nó autogerenciado em execução por mais de um ano, ele não poderá mais se autenticar na API do Kubernetes.
- Como prática recomendada, recomendamos que os clientes atualizem regularmente seus grupos de nós autogerenciados para receber CVEs e patches de segurança da AMI otimizada para Amazon EKS mais recente. A atualização da AMI usada em grupos de nós autogerenciados também aciona a recriação de nós e garante que eles não tenham problemas devido a certificados kubelet expirados.

- Como alternativa, você também pode habilitar a rotação de certificados do cliente (consulte: https://kubernetes.io/docs/tasks/tls/certificate-rotation/) ao criar os grupos de nós autogerenciados para garantir que os certificados do kubelet sejam renovados à medida que o certificado atual se aproxima da expiração.

`eksctl`

Para iniciar nós autogerenciados do Linux usando **`eksctl`**

1. Instale a versão `0.215.0` ou posterior da ferramenta de linha de comando da `eksctl` instalada no seu dispositivo ou AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
2. Se o cluster estiver na Nuvem AWS e a política do IAM gerenciada AmazonEKS_CNI_Policy estiver anexada ao perfil do IAM do nó do Amazon EKS, recomendamos atribuí-la a um perfil do IAM que você associa à conta de serviço `aws-node` do Kubernetes. Para obter mais informações, consulte the section called “Configuração para IRSA”. Se o cluster estiver no Outpost, a política deverá estar anexada à função do nó.
3. O comando a seguir cria um grupo de nós em um cluster existente. O cluster dove ter sido criado usando `eksctl`. Substitua *`al-nodes`* por um nome para o grupo de nós. O nome do grupo de nós não pode exceder 63 caracteres. Deve começar com uma letra ou um dígito, mas pode incluir hifens e sublinhados para os demais caracteres. Substitua *`my-cluster`* pelo nome do cluster. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster. Se o cluster estiver em um Outpost, substitua *`id`* pelo ID de uma sub-rede do Outpost. Se o seu cluster existir no AWS Cloud, substitua *`id`* pelo ID de uma sub-rede que você não especificou ao criar o cluster. Substitua os valores de exemplo restantes por seus próprios valores. Os nós são criados com a mesma versão do Kubernetes que o plano de controle, por padrão.

   Substitua *`instance-type`* por um tipo de instância disponível no Outpost.

   Substitua *`my-key`* pelo nome do seu par de chaves ou chave pública do Amazon EC2. Essa chave é usada para SSH em seus nós depois que eles forem iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no Console de gerenciamento da AWS. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2.

Crie seu grupo de nós com o comando a seguir.

```
eksctl create nodegroup --cluster my-cluster --name al-nodes --node-type instance-
type \
    --nodes 3 --nodes-min 1 --nodes-max 4 --managed=false \
    --node-volume-type gp2 --subnet-ids subnet-id \
    --node-ami-family AmazonLinux2023
```

Se o seu cluster estiver implementado no AWS Cloud:

- O grupo de nós que você implantar poderá atribuir endereços `IPv4` aos pods de um bloco CIDR diferente do bloco da instância. Para obter mais informações, consulte the section called “Redes personalizadas”.
- O grupo de nós que você implantar não exigirá acesso de saída à Internet. Para obter mais informações, consulte the section called “Clusters privados”.

Para obter uma lista completa de todas as opções e padrões disponíveis, consulte AWS Outposts Support na documentação `eksctl`.

- Se os nós não conseguirem ingressar no cluster, consulte the section called “Falha nos nós ao ingressar no cluster” em Solucionar problemas com clusters e nós do Amazon EKS e the section called “Não é possível integrar nós a um cluster” em Solucionar problemas de clusters locais do Amazon EKS no AWS Outposts.
- Veja abaixo um exemplo de saída. Várias linhas são emitidas enquanto os nós são criados. Uma das últimas linha de saída é o exemplo de linha a seguir.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

4. (Opcional) Implante uma aplicação de exemplo para testar o cluster e os nós do Linux.

Console de gerenciamento da AWS

Etapa 1: iniciar nós autogerenciados do Linux usando Console de gerenciamento da AWS

1. Baixe a versão mais recente do modelo do AWS CloudFormation.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2025-11-24/
amazon-eks-outpost-nodegroup.yaml
```

2. Abra o console do AWS CloudFormation.
3. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos, padrão).
4. Para Specify template (Especificar modelo), selecione Upload a template file (Fazer upload de um arquivo de modelo) e depois Choose file (Escolher arquivo). Selecione o arquivo `amazon-eks-nodegroup.yaml` que você baixou em uma etapa anterior e selecione Next (Próximo).
5. Na página Specify stack details (Especificar detalhes da pilha), insira os parâmetros de acordo e escolha Next (Próximo):
   - Nome da pilha: escolha um nome para a pilha do AWS CloudFormation. Por exemplo, você pode chamá-lo de *`al-nodes`*. O nome só pode conter caracteres alfanuméricos (sensíveis a maiúsculas e minúsculas) e hifens. Ele deve começar com um caractere alfanumérico e não pode ter mais de 100 caracteres. O nome deve ser exclusivo na região da AWS e na conta da AWS em que você está criando o cluster.
   - ApiServerEndpoint: informe o endpoint do servidor de APIs do Kubernetes, que pode ser visualizado no console do EKS ou por meio da API DescribeCluster.
   - ClusterName: insira o nome do cluster. Se esse nome não corresponder ao nome do cluster, os nós não poderão se integrar ao cluster.
   - ClusterId: informe o ID designado ao cluster pelo serviço EKS. Visível por meio da API DescribeCluster. Se este ID não corresponder ao ID do seu cluster, os nós não poderão ingressar no cluster.
   - CertificateAuthority: informe a string codificada em base64 da Autoridade de Certificação do Kubernetes. Visível no console do EKS ou por meio da API DescribeCluster.
   - ServiceCidr: informe o CIDR de serviços do Kubernetes. Visível no console do EKS ou por meio da API DescribeCluster.
   - ClusterControlPlaneSecurityGroup: Escolha o valor SecurityGroups da saída do AWS CloudFormation que você gerou ao criar sua VPC.

     As etapas a seguir mostram uma operação para recuperar o grupo aplicável.

     i. Abra o console do Amazon EKS.

     ii. Escolha o nome do cluster.

     iii. Escolha a guia Redes.

     iv. Use o valor de Additional security group (Grupos de segurança adicionais) como referência ao selecionar na lista suspensa ClusterControlPlaneSecurityGroup.

- NodeGroupName: insira um nome para o grupo de nós. Esse nome poderá ser usado superiormente para identificar o grupo de nós do Auto Scaling que for criado para os nós.
- NodeAutoScalingGroupMinSize: digite o número mínimo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeAutoScalingGroupDesiredCapacity: insira o número desejado de nós para o qual dimensionar quando a pilha for criada.
- NodeAutoScalingGroupMaxSize: digite o número máximo de nós para o qual o grupo Auto Scaling de nós pode ser escalado.
- NodeInstanceType: escolha um tipo de instância para os nós. Se o seu cluster estiver sendo executado no AWS Cloud, para obter mais informações, consulte the section called “Tipos de instância do Amazon EC2”. Se o cluster estiver sendo executado em um Outpost, você só poderá selecionar um tipo de instância que esteja disponível no Outpost.
- NodeImageIdSSMParam: pré-preenchido com o parâmetro do Amazon EC2 Systems Manager de uma AMI recente otimizada para Amazon EKS, para uma versão variável do Kubernetes. Para usar uma versão secundária diferente do Kubernetes compatível com o Amazon EKS, substitua *1.XX* por uma versão compatível diferente. Recomendamos especificar a mesma versão do Kubernetes que seu cluster.

  Para usar uma AMI acelerada e otimizada para o Amazon EKS, atualize o valor de *NodeImageIdSSMParam* para o parâmetro do SSM desejado. Confira como recuperar IDs de AMIs do EKS usando o SSM aqui.

  > **Note**
  >
  > As AMIs dos nós do Amazon EKS são baseadas no Amazon Linux. É possível acompanhar os eventos de segurança ou privacidade do Amazon Linux no centro de segurança do Amazon Linux, selecionando a guia da versão desejada. Você também pode assinar o feed RSS aplicável. Os eventos de segurança e privacidade incluem uma visão geral do problema, quais pacotes são afetadas e como atualizar suas instâncias para corrigir o problema.

- NodeImageId: (opcional) se estiver usando sua própria AMI personalizada (em vez da AMI otimizada do Amazon EKS), insira um ID de AMI de nó para a região da AWS. Se você especificar um valor aqui, ele substituirá todos os valores no campo NodeImageIdSSMParam.
- NodeVolumeSize: especifique um tamanho de volume raiz para os nós, em GiB.
- NodeVolumeType: especifique um tipo de volume raiz para os nós.

- KeyName: insira o nome de um par de chaves SSH do Amazon EC2; que você pode usar para conectar-se usando SSH em seus nós, depois de iniciados. Se ainda não tiver um par de chaves do Amazon EC2, você poderá criar um no Console de gerenciamento da AWS. Para obter mais informações, consulte Pares de chaves do Amazon EC2, no Guia do usuário do Amazon EC2.

  > **Note**
  >
  > Se você não fornecer um par de chaves aqui, a criação da pilha do AWS CloudFormation falhará.

- DisableIMDSv1: por padrão, cada nó oferece suporte ao serviço de metadados de instância versão 1 (IMDSv1) e IMDSv2. Você pode desabilitar IMDSv1. Para evitar que futuros nós e pods do grupo de nós usem IMDSv1, defina DisableIMDSv1 como true. Para obter mais informações, consulte Configurar o serviço de metadados da instância. Para obter mais informações sobre como restringir o acesso a ele em seus nós, consulte Restringir o acesso ao perfil da instância atribuído ao nó de processamento.
- VpcId: insira o ID da VPC que você criou. Antes de escolher uma VPC, analise os requisitos e as considerações da VPC.
- Sub-redes:se o cluster estiver em um Outpost, escolha pelo menos uma sub-rede privada na VPC. Antes de escolher as sub-redes, analise os Requisitos e considerações para sub-redes. É possível ver quais sub-redes são privadas abrindo cada link de sub-rede na guia Networking (Redes) do cluster.

6. Selecione as opções desejadas na página Configure stack options (Configurar opções de pilha) e depois escolha Next (Próximo).
7. Marque a caixa de seleção à esqu erda de I acknowledge that AWS CloudFormation might create IAM resources. (Eu reconheço que o CloudFormation pode criar recursos de IAM) e, em seguida, selecione Create stack (Criar pilha).
8. Quando a criação da pilha for concluída, selecione-a no console e escolha Outputs (Saídas).
9. Registre o valor de NodeInstanceRole para o grupo de nós criado. Você precisará dele ao configurar os nós do Amazon EKS.

Etapa 2: habilite os nós para participar do cluster

1. Verifique se você já tem um `aws-auth` `ConfigMap`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se você receber um `aws-auth ConfigMap`, atualize-o conforme necessário.

   a. Abra o `ConfigMap` para edição.

   ```
   kubectl edit -n kube-system configmap/aws-auth
   ```

   b. Adicione uma nova `mapRoles` entrada conforme necessário. Defina o valor `rolearn` para o valor NodeInstanceRole que você registrou no procedimento anterior.

   ```
   [...]
   data:
     mapRoles: |
       - rolearn: <ARN of instance role (not instance profile)>
         username: system:node:{{EC2PrivateDNSName}}
         groups:
           - system:bootstrappers
           - system:nodes
   [...]
   ```

   c. Salve o arquivo e saia do seu editor de texto.

3. Se você recebeu um erro informando "`Error from server (NotFound): configmaps "aws-auth" not found`, aplique o estoque`ConfigMap`.

   a. Faça download do mapa de configuração.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
   aws-auth-cm.yaml
   ```

   b. No arquivo `aws-auth-cm.yaml`, configure `rolearn` para o valor NodeInstanceRole que você registrou no procedimento anterior. Você pode fazer isso com um editor de texto ou substituindo *`my-node-instance-role`* e executando o seguinte comando:

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
   role|' aws-auth-cm.yaml
   ```

   c. Aplique a configuração. Esse comando pode demorar alguns minutos para ser concluído.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

4. Observe o status de seus nós e aguarde até que eles atinjam o status `Ready`.

```
kubectl get nodes --watch
```

   Insira `Ctrl+C` para retornar a um prompt de shell.

> **Note**
>
> Se você receber qualquer erro de autorização ou de tipo de recurso, consulte the section called “Acesso negado ou não autorizado (`kubectl`)” no tópico de solução de problemas.

   Se os nós não conseguirem ingressar no cluster, consulte the section called “Falha nos nós ao ingressar no cluster” em Solucionar problemas com clusters e nós do Amazon EKS e the section called “Não é possível integrar nós a um cluster” em Solucionar problemas de clusters locais do Amazon EKS no AWS Outposts.

5. Instale o driver da CSI do Amazon EBS. Para obter mais informações, consulte Instalação no GitHub. Na seção Set up driver permission (Configurar permissão do driver), certifique-se de seguir as instruções para a opção Using IAM instance profile (Usar o perfil de instância do IAM). Você deve usar a classe de armazenamento `gp2`. A classe de armazenamento `gp3` não é compatível.

   Para criar uma classe de armazenamento `gp2` no cluster, conclua as etapas a seguir.

   a. Execute o seguinte comando para criar o arquivo `gp2-storage-class.yaml`.

```
cat >gp2-storage-class.yaml <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp2
  encrypted: "true"
allowVolumeExpansion: true
EOF
```

b. Aplique o manifesto ao cluster.

```
kubectl apply -f gp2-storage-class.yaml
```

6. (Somente nós de GPU) Caso escolha um tipo de instância de GPU e a AMI acelerada otimizada do Amazon EKS, você deverá aplicar o plug-in do dispositivo NVIDIA para Kubernetes como um DaemonSet no cluster. Substitua *vX.X.X* pela versão desejada do NVIDIA/k8s-device-plugin antes de executar o seguinte comando.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Etapa 3: Ações adicionais

1. (Opcional) Implante uma aplicação de exemplo para testar o cluster e os nós do Linux.
2. Se o cluster estiver implantado em um Outpost, pule esta etapa. Se o seu cluster estiver implantado no AWS Cloud, as informações a seguir são opcionais. Se a política do IAM gerenciada AmazonEKS_CNI_Policy estiver anexada ao perfil do IAM do nó do Amazon EKS, recomendamos atribuí-la a um perfil do IAM que você associa à conta de serviço `aws-node` do Kubernetes. Para obter mais informações, consulte the section called “Configuração para IRSA”.

Visão geral da Inteligência Artificial (IA) e do Machine Learning (ML) no Amazon EKS

O Amazon Elastic Kubernetes Service (EKS) é uma plataforma gerenciada do Kubernetes que capacita as organizações a implantar, gerenciar e escalar workloads de IA e machine learning (ML) com flexibilidade e controle incomparáveis. Desenvolvido no ecossistema de código aberto do Kubernetes, o EKS permite que você aproveite sua experiência atual no Kubernetes e, ao mesmo tempo, se integre perfeitamente aos serviços da AWS e às ferramentas de código aberto.

Esteja você treinando modelos de grande escala, executando inferência on-line em tempo real ou implantando aplicações de IA generativa, o EKS oferece a performance, escalabilidade e eficiência de custos que seus projetos de IA e ML exigem.

Por que escolher o EKS para IA e ML?

O EKS é uma plataforma gerenciada do Kubernetes que ajuda você a implantar e gerenciar workloads complexas de IA e ML. Desenvolvido no ecossistema do Kubernetes de código aberto, ele se integra aos serviços da AWS, fornecendo o controle e a escalabilidade necessários para projetos avançados. Para equipes novas em implantações de IA e ML, as habilidades existentes do Kubernetes são transferidas diretamente, permitindo a orquestração eficiente de várias workloads.

O EKS é compatível com tudo, desde personalizações de sistemas operacionais até escalabilidade computacional, e sua base de código aberto promove flexibilidade tecnológica, preservando a escolha para futuras decisões de infraestrutura. A plataforma fornece as opções de performance e ajuste que as workloads de IA e ML exigem, oferecendo suporte a recursos como:

- Controle total do cluster para ajustar custos e configurações sem abstrações ocultas
- Latência de menos de um segundo para workloads de inferência em tempo real na produção
- Personalizações avançadas, como GPUs de várias instâncias, estratégias de multinuvem e ajuste no nível do sistema operacional
- Capacidade de centralizar workloads usando o EKS como um orquestrador unificado em todos os pipelines de IA e ML

Casos de uso principais

O Amazon EKS fornece uma plataforma robusta para uma ampla variedade de workloads de IA e ML, compatível com várias tecnologias e padrões de implantação:

- Inferência em tempo real (on-line): o EKS possibilita previsões imediatas sobre dados recebidos, como detecção de fraudes, com latência de menos de um segundo usando ferramentas como TorchServe, Triton Inference Server e KServe nas instâncias Inf1 e Inf2 do Amazon EC2. Essas workloads se beneficiam da escalabilidade dinâmica com o Karpenter e o KEDA, ao mesmo tempo em que utilizam o Amazon EFS para fragmentação de modelos em pods. O Pull Through Cache (PTC) do Amazon ECR acelera as atualizações do modelo, e os volumes de dados do Bottlerocket com volumes otimizados para Amazon EBS garantem o acesso rápido aos dados.
- Treinamento de modelos em geral: as organizações empregam o EKS para treinar modelos complexos em grandes conjuntos de dados por longos períodos, usando o Kubeflow Training Operator, o Ray Serve e o Torch Distributed Elastic nas instâncias P4d do Amazon EC2 e Trn1 do Amazon EC2. Essas workloads são compatíveis com o agendamento em lote com ferramentas como Volcano, Yunikorn e Kueue. O Amazon EFS permite o compartilhamento de pontos de verificação de modelos, e o Amazon S3 processa a importação e exportação de modelos com políticas de ciclo de vida para gerenciamento de versões.
- Pipelines de geração aumentada via recuperação (RAG): o EKS gerencia chatbots de suporte ao cliente e aplicações similares integrando processos de recuperação e geração. Essas workloads geralmente usam ferramentas como Argo Workflows e Kubeflow para orquestração, bancos de dados vetoriais como Pinecone, Weaviate ou Amazon OpenSearch, e expõem aplicações para os usuários por meio do Application Load Balancer Controller (LBC). O NVIDIA NIM otimiza a utilização da GPU, enquanto o Prometheus e o Grafana monitoram o uso de recursos.
- Implantação do modelo de IA generativa: as empresas implantam serviços de criação de conteúdo em tempo real no EKS, como geração de texto ou imagem, usando o Ray Serve, o vLLM e o Triton Inference Server nos aceleradores EC2 G5 e Inferentia da Amazon. Essas implantações otimizam a performance e a utilização da memória para modelos de grande escala. O JupyterHub permite o desenvolvimento iterativo, o Gradio fornece interfaces simples da web e o driver CSI do Mountpoint para S3 permite montar buckets do S3 como sistemas de arquivos para acessar grandes arquivos de modelo.
- Inferência em lote (offline): as organizações processam grandes conjuntos de dados de forma eficiente por meio de trabalhos programados com o AWS Batch ou o Volcano. Essas workloads geralmente usam as instâncias Inf1 e Inf2 do EC2 para chips do AWS Inferentia, instâncias G4dn do Amazon EC2 para GPUs NVIDIA T4 ou instâncias c5 e c6i da CPU, maximizando a utilização

de recursos fora do horário de pico para tarefas de analytics. O SDK do AWS Neuron e os drivers de GPU NVIDIA otimizam a performance, enquanto o MIG/TS permite o compartilhamento de GPU. As soluções de armazenamento incluem o Amazon S3 e o Amazon EFS e FSx para Lustre, com drivers CSI para várias classes de armazenamento. O gerenciamento de modelos usa ferramentas como o Kubeflow Pipelines, Argo Workflows e Ray Cluster, enquanto o monitoramento é realizado pelo Prometheus, pelo Grafana e por ferramentas personalizadas de monitoramento de modelos.

Estudos de caso

Os clientes escolhem o Amazon EKS por diversos motivos, como otimizar o uso da GPU ou executar workloads de inferência em tempo real com latência inferior a um segundo, conforme demonstrado nos estudos de caso a seguir. Para obter uma lista de todos os estudos de caso do Amazon EKS, consulte Histórias de sucesso clientes da AWS.

- A Unitary processa 26 milhões de vídeos diariamente usando IA para moderação de conteúdo, exigindo inferência de alto throughput e baixa latência, e alcançou uma redução de 80% nos tempos de inicialização de contêineres, garantindo uma resposta rápida aos eventos de escalabilidade à medida que o tráfego flutua.
- A Miro, a plataforma de colaboração visual que atende 70 milhões de usuários em todo o mundo, relatou uma redução de 80% nos custos de computação em comparação com seus clusters autogerenciados do Kubernetes anteriores.
- A Synthesia, que oferece criação de vídeos com IA generativa como um serviço para os clientes criarem vídeos realistas com base em prompts de texto, obteve uma melhoria de 30 vezes no throughput do treinamento de modelo de ML.
- A Harri, que fornece tecnologia de RH para o setor de hospitalidade, alcançou uma escalabilidade 90% mais rápida em resposta aos picos de demanda e reduziu seus custos de computação em 30% ao migrar para os processadores AWS Graviton.
- A Ada Support, uma empresa de automação de atendimento ao cliente baseada em IA, alcançou uma redução de 15% nos custos de computação, juntamente com um aumento de 30% na eficiência computacional.
- A Snorkel AI, que prepara as empresas para criar e adaptar modelos de base e grandes modelos de linguagem, obteve mais de 40% de economia de custos implementando mecanismos inteligentes de escalabilidade para seus recursos de GPU.

Comece a usar machine learning no EKS

Para começar a planejar e usar plataformas e workloads de machine learning no EKS na nuvem da AWS, vá para a seção the section called “Recursos”.

Execução de workloads de inferência online em tempo real no Amazon EKS

Esta seção foi projetada para ajudar você a implantar e operar workloads de inferência online em tempo real no Amazon Elastic Kubernetes Service (EKS). Você encontrará orientações sobre como criar clusters otimizados com nós acelerados por GPU, integrar serviços da AWS para armazenamento e ajuste de escala automático, implantar modelos de amostra para validação e considerações arquitetônicas importantes, como desacoplar tarefas de CPU e GPU, selecionar AMIs e tipos de instância apropriados e garantir a exposição de baixa latência dos endpoints de inferência.

Tópicos

- Guia de configuração de cluster do melhores práticas para inferência em tempo real no Amazon EKS
- Início rápido: inferência de LLM de alto throughput usando vLLM no Amazon EKS

Guia de configuração de cluster do melhores práticas para inferência em tempo real no Amazon EKS

Introdução

Este guia oferece um passo a passo prático para configurar um cluster do Amazon Elastic Kubernetes Service (EKS) otimizado para workloads de inferência online em tempo real, incorporando as melhores práticas organizadas por especialistas da AWS em todo o processo. Ele usa uma arquitetura opinativa de início rápido do EKS: um conjunto selecionado de drivers, tipos de instância e configurações alinhados às melhores práticas da AWS para modelos, aceleradores e escalabilidade. Essa abordagem ajuda você a ignorar a tarefa de selecionar as configurações do cluster, permitindo que você coloque um cluster funcional e pré-configurado em funcionamento rapidamente. Ao longo do caminho, implantaremos amostras de workloads para validar sua configuração, explicar os principais conceitos arquitetônicos (como desacoplar tarefas vinculadas à CPU de cálculos com uso intensivo de GPU) e abordar questões comuns (por exemplo, por que

escolher o Bottlerocket AMI em vez do AL2023?) e descrever as próximas etapas para ampliar os recursos do seu cluster.

Projetado especificamente para engenheiros de Machine Learning (ML) e Inteligência Artificial (IA), administradores de plataformas, operadores e especialistas em dados/IA que são novos no ecossistema do EKS e da AWS, este guia pressupõe familiaridade com o Kubernetes, mas nenhuma experiência anterior com o EKS. Ele foi projetado para ajudar você a entender as etapas e os processos necessários para colocar as workloads de inferência online em tempo real em funcionamento. O guia mostra os fundamentos da criação de um cluster do inferência de nó único, incluindo o provisionamento de recursos de GPU, a integração do armazenamento para artefatos de modelo, a habilitação do acesso seguro aos serviços da AWS e a exposição de endpoints de inferência. Ele enfatiza o design resiliente e de baixa latência para aplicações voltadas para o usuário, como detecção de fraudes, chatbots em tempo real e análise de sentimentos em sistemas de feedback de clientes.

Neste guia, nos concentramos exclusivamente na configuração de um ponto de partida básico e prescritivo usando instâncias G5 do EC2. Se você está procurando configurações de cluster específicas do AWS Inferentia ou fluxos de trabalho de ponta a ponta, consulte the section called “Configureação de clusters de inferência com o Inferentia” ou nossos workshops em the section called “Recursos”.

Antes de começar

Antes de começar, certifique-se de ter realizado as seguintes tarefas:

- Configuração do seu ambiente para o Amazon EKS
- Instalação da versão mais recente do eksctl
- Instale o Helm
- (Opcional) Instalação do Docker
- (Opcional) Instalação da CLI do NGC

Arquitetura

A inferência online em tempo real se refere ao processo de usar um modelo de machine learning treinado para gerar previsões ou resultados em fluxos de dados recebidos com latência mínima. Por exemplo, ele permite a detecção de fraudes em tempo real, a classificação de imagens ou a geração de gráficos de conhecimento em resposta às entradas do usuário. A arquitetura de um

sistema de inferência online em tempo real fornece previsões de machine learning de baixa latência em aplicações voltadas para o usuário ao desacoplar o tratamento do tráfego da Web vinculado à CPU dos cálculos de IA com uso intensivo de GPU. Esse processo normalmente reside em um ecossistema de aplicações maior e geralmente inclui serviços de backend, frontend, vetor e modelos, com foco em componentes especializados para permitir escalabilidade independente, desenvolvimento paralelo e resiliência contra falhas. O isolamento de tarefas de inferência em hardware de GPU dedicado e o aproveitamento de interfaces como APIs e WebSockets garante alta simultaneidade, processamento rápido de modelos como transformadores e interações com o usuário por meio do frontend. Observe que, embora os bancos de dados de vetores e os pipelines de geração aumentada via recuperação (RAG) geralmente desempenhem um papel importante nos sistemas de inferência em tempo real, esses componentes não são abordados neste guia. No mínimo, uma arquitetura típica geralmente inclui:

- Serviço de frontend: serve como interface voltada para o usuário, manipulando a lógica do lado do cliente, renderizando conteúdo dinâmico e facilitando interações em tempo real. Ele se comunica com o serviço de backend para iniciar solicitações de inferência e exibir resultados, geralmente iniciando solicitações para o serviço de backend que usa WebSockets para atualizações de streaming ou APIs para troca estruturada de dados. Esse serviço normalmente não requer um balanceador de carga dedicado, pois pode ser hospedado em redes de entrega de conteúdo (CDN) como o AWS CloudFront para ativos estáticos ou servido diretamente de servidores Web, com escalabilidade gerenciada por meio de grupos de ajuste de escala automático, se necessário, para conteúdo dinâmico.
- Serviço de backend: atua como orquestrador da aplicação, gerenciando a lógica de negócios, como autenticação de usuários, validação de dados e coordenação de serviços (por exemplo, por meio de APIs para endpoints RESTful ou WebSockets para conexões persistentes). Ele se comunica com o serviço de inferência, escala de forma independente em CPUs e RAM de vários núcleos para lidar com o alto tráfego da Web sem depender de GPUs e geralmente requer um balanceador de carga (como o AWS Application Load Balancer ou Network Load Balancer) para distribuir as solicitações recebidas em várias instâncias, especialmente em cenários de alta simultaneidade. Um controlador de entrada pode gerenciar ainda mais o acesso externo e as regras de roteamento para melhorar a segurança e a modelagem do tráfego.
- Serviço de inferência: serve como núcleo para cálculos de IA, sendo executado em GPUs com VRAM suficiente (por exemplo, 8-12 GB para modelos como DistilBERT) para realizar incorporações de vetor, extração de conhecimento e inferência do modelo (por exemplo, expostos por meio de APIs para solicitações em lote ou WebSockets para streaming em tempo real) usando modelos personalizados ou de código aberto. Esse isolamento evita conflitos de dependência,

permite atualizações do modelo sem tempo de inatividade e possibilita a escalabilidade horizontal com balanceamento de carga para várias solicitações simultâneas. Para expor o serviço de modelo de forma eficaz, ele normalmente fica atrás de um balanceador de carga para distribuir workloads vinculadas à GPU entre instâncias replicadas, enquanto um recurso ou controlador de entrada (como o ALB Ingress Controller na AWS) lida com roteamento externo, terminação de SSL e encaminhamento baseado em caminho para garantir acesso seguro e eficiente sem sobrecarregar GPUs individuais.

Visão geral da solução

Os sistemas de inferência online em tempo real exigem uma arquitetura resiliente e de alta performance que possa oferecer latência ultrabaixa e, ao mesmo tempo, lidar com picos de tráfego imprevisíveis e de alto volume. Essa visão geral da solução explica como os componentes da AWS a seguir funcionam juntos no cluster do Amazon EKS que criaremos para garantir que nosso cluster seja capaz de hospedar e gerenciar modelos de machine learning que forneçam previsões imediatas sobre dados ativos com o mínimo de atraso para os usuários finais.

- Instâncias Amazon G5 do EC2: para tarefas de inferência com uso intensivo de GPU, estamos usando os tipos de instância G5.xlarge e g5.2xlarge G5 do EC2, que apresentam uma única (1) GPU NVIDIA A10G com 24 GB de memória (por exemplo, 8 bilhões de parâmetros no FP16). Com base na arquitetura NVIDIA Ampere, essas GPUs são equipadas com GPUs NVIDIA A10G Tensor Core e processadores AMD EPYC de 2ª geração, suportam de 4 a 8 vCPUs, largura de banda da rede de até 10 Gbps e 250 a 450 GB de armazenamento SSD NVMe local, garantindo rápida movimentação de dados e potência computacional para modelos complexos, tornando-as ideais para tarefas de inferência de baixa latência e alto throughput. A escolha de um tipo de instância do EC2 é específica da aplicação, depende do seu modelo (por exemplo, imagem, vídeo, modelo de texto) e dos requisitos de latência e throughput. Por exemplo, se estiver usando um modelo de imagem e/ou vídeo, talvez você queira usar instâncias P5 do EC2 para obter uma latência ideal em tempo real. Recomendamos começar com as instâncias G5 do EC2, pois elas fornecem um bom ponto de partida para começar a trabalhar rapidamente e, em seguida, avaliar se elas são adequadas para suas workloads por meio de testes de benchmark de performance. Para casos mais avançados, considere as instâncias G6 do EC2.
- Instâncias M7g do Amazon EC2: para tarefas intensivas de CPU, como pré-processamento de dados, tratamento de solicitações de API, hospedagem do controlador Karpenter, complementos e outros componentes do sistema, estamos usando o tipo de instância m5.xlarge M7g do EC2. As instâncias M7g são baseadas em ARM que apresentam 4 vCPUs, 16 GB de memória, largura de banda da rede de até 12,5 Gbps e são alimentadas por processadores AWS Graviton3. A

escolha de um tipo de instância do EC2 é específica da aplicação e depende dos requisitos de computação, memória e escalabilidade da sua workload. Para workloads otimizadas para computação, é possível considerar as instâncias C7g do EC2, que também usam processadores Graviton3, mas são otimizadas para maior performance computacional do que as instâncias M7g para determinados casos de uso. Como alternativa, as instâncias C8g do EC2 mais novas (quando disponíveis) oferecem performance computacional até 30% melhor do que as instâncias C7g. Recomendamos começar com as instâncias M7g do EC2 por sua eficiência de custos e compatibilidade com uma ampla variedade de workloads (por exemplo, servidores de aplicações, microsserviços, servidores de jogos, armazenamentos de dados de médio porte) e, em seguida, avaliar se é a opção certa para suas workloads por meio de testes de benchmark de performance.

- Driver CSI Mountpoint do Amazon S3: para workloads em instâncias de GPU única em que vários pods compartilham uma GPU (por exemplo, vários pods programados no mesmo nó para utilizar seus recursos de GPU), estamos usando o driver CSI Mountpoint do S3 para otimizar o uso da memória, essencial para tarefas como inferência de modelos grandes em configurações econômicas e de baixa complexidade. Ele expõe os buckets do Amazon S3 como um sistema de arquivos semelhante ao POSIX disponível para o cluster do Kubernetes, o que permite que os pods de inferência leiam artefatos do modelo (por exemplo, pesos do modelo) diretamente na memória sem precisar baixá-los primeiro e inserir conjuntos de dados usando operações de arquivo padrão. Além disso, o S3 tem capacidade de armazenamento praticamente ilimitada e acelera as workloads de inferência com uso intenso de dados. A escolha de um driver CSI de armazenamento é específica da aplicação e depende dos requisitos de throughput e latência da workload. Embora o driver CSI FSx para OpenZFS ofereça latência inferior a um milissegundo para E/S aleatória ou volumes persistentes compartilhados totalmente compatíveis com POSIX entre os nós, recomendamos começar com o driver CSI Mountpoint S3 devido à sua escalabilidade, custos mais baixos para grandes conjuntos de dados e integração integrada com armazenamento de objetos gerenciado pelo S3 para padrões de inferência com muita leitura (por exemplo, entradas do modelo de streaming) e, em seguida, avaliar se é o correto adequado às suas workloads por meio de testes de benchmark de performance.
- Agente de Identidade de Pods do EKS: para permitir o acesso aos serviços da AWS, estamos usando o Agente de Identidade de Pods do EKS, que usa uma única entidade principal de serviço e facilita associações de perfis do IAM em nível de pod no cluster do Amazon EKS. A Identidade de Pods do EKS oferece uma alternativa simplificada à abordagem tradicional de perfis do IAM para contas de serviço (IRSA), utilizando uma única entidade principal de serviço (pods.eks.amazonaws.com) em vez de depender de provedores OIDC individuais para cada cluster, o que facilita a atribuição de permissões. Além disso, ele permite que os perfis sejam

reutilizados em vários clusters e oferece suporte a recursos avançados, como tags de sessão de perfil do IAM e perfis do IAM de destino.

- Agente de monitoramento de nós do EKS: para garantir a disponibilidade e a confiabilidade contínuas dos serviços de inferência, estamos usando o agente de monitoramento de nós do EKS com reparo automático, que detecta e substitui automaticamente os nós não saudáveis, minimizando o tempo de inatividade. Ele monitora continuamente os nós em busca de problemas de hardware, kernel, rede e armazenamento usando verificações de integridade aprimoradas (por exemplo, KernelReady, NetworkingReady). Para nós de GPU, ele detecta falhas específicas do acelerador, iniciando uma correção normal com o isolamento de nós não íntegros, aguardando 10 minutos para que problemas transitórios de GPU sejam resolvidos e substituindo os nós após 30 minutos por falhas persistentes.
- AMI do Bottlerocket: para fornecer uma base segura para nosso cluster do EKS, estamos usando a AMI do Bottlerocket, que inclui apenas os componentes essenciais necessários para executar contêineres e oferece tempos mínimos de inicialização para escalabilidade rápida. A escolha de uma AMI de nó é específica da aplicação e depende dos requisitos de personalização, segurança e escalabilidade de sua workload. Embora a AMI AL2023 ofereça maior flexibilidade para instalações e personalizações em nível de host (por exemplo, especificar um diretório de cache dedicado em um PV/PVC sem nenhuma configuração de nó adicional), recomendamos começar com a AMI do Bottlerocket por seu tamanho menor e otimização integrada para workloads em contêineres (por exemplo, microsserviços, servidores de inferência, APIs escaláveis) e, em seguida, avaliar se ela é a opção certa para suas workloads por meio de testes de benchmark de performance.
- AWS Load Balancer Controller (LBC): para expor endpoints de inferência em tempo real, estamos usando o AWS Load Balancer Controller, que provisiona e gerencia automaticamente Application Load Balancers (ALBs) para tráfego HTTP/HTTPS e Network Load Balancers (NLBs) para tráfego TCP/UDP com base nos recursos de entrada e serviço do Kubernetes, permitindo a integração de modelos de inferência com clientes externos. Além disso, ele oferece suporte a recursos como roteamento baseado em caminhos para distribuir solicitações de inferência em vários pods ou nós, garantindo escalabilidade durante picos de tráfego e minimizando a latência por meio de otimizações nativas da AWS, como multiplexação de conexão e verificações de integridade.

1. Criação do seu cluster do EKS

Nesta etapa, criamos um cluster com nós de CPU e um grupo de nós gerenciados usando um modelo eksctl ClusterConfig baseado no AWS CloudFormation. Inicializar o cluster apenas com nós de CPU nos permite usar o Karpenter exclusivamente para gerenciar nós de GPU e com uso

intensivo de CPU para alocação otimizada de recursos usando Karpenter NodePools, que criaremos em etapas posteriores. Para oferecer suporte às nossas workloads de inferência em tempo real, provisionamos o cluster com a AMI do Bottlerocket do EKS, o Agente de monitoramento de nós do EKS, o Agente de Identidade de Pods do EKS, o driver CSI Mountpoint S3, o AWS Load Balancer Controller (LBC) e os drivers kube-proxy, vpc-cni e coredns. As instâncias m7g.xlarge serão usadas para tarefas do sistema de CPU, incluindo hospedagem do controlador Karpenter, complementos e outros componentes do sistema.

Por padrão, o `eksctl` criará uma VPC dedicada para o cluster com um bloco CIDR `192.168.0.0/16`. A VPC inclui três sub-redes públicas e três sub-redes privadas, cada uma distribuída em três zonas de disponibilidade diferentes (ou duas AZs na região `us-east-1`), o que é o método ideal para implantar workloads do Kubernetes. O modelo também implanta um gateway da Internet, fornecendo acesso à Internet às sub-redes públicas por meio de rotas padrão em suas tabelas de rotas e um único gateway de NAT em uma das sub-redes públicas, com rotas padrão nas tabelas de rotas das sub-redes privadas direcionando o tráfego de saída por meio do gateway de NAT para acesso à Internet. Para saber mais sobre essa configuração, consulte Implantação de nós em sub-redes privadas.

Verificação das suas credenciais

Verifique se suas credenciais da AWS CLI são válidas e podem ser autenticadas com serviços da AWS:

```
aws sts get-caller-identity
```

Se tiver êxito, a CLI retornará detalhes sobre sua identidade da AWS (ID do usuário, conta e ARN).

Verificação de disponibilidade de instâncias

Os tipos de instâncias G5 não estão disponíveis em todas as regiões. Verifique sua região mais próxima. Por exemplo:

```
aws ec2 describe-instance-types --instance-types g5.xlarge g5.2xlarge --region us-
east-1
```

Se tiver êxito, o tipo de instância G5 estará disponível na região que você especificou.

A AMI do Bottlerocket não está disponível em todas as regiões. Verifique recuperando um ID de AMI do Bottlerocket para sua região mais próxima. Por exemplo:

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.33/arm64/latest/
image_id \
    --region us-east-1 --query "Parameter.Value" --output text
```

Se tiver êxito, a AMI do Bottlerocket estará disponível na região que você especificou.

Preparação do seu ambiente

Primeiro, defina as seguintes variáveis de ambiente em uma nova janela do terminal. Observação: certifique-se de substituir os espaços reservados do exemplo por seus valores exclusivos, incluindo o nome do cluster, a região desejada, a versão de lançamento do Karpenter e a versão do Kubernetes.

> **Tip**
>
> Algumas variáveis (como `${AWS_REGION}` e `${K8S_VERSION}`) são definidas no início do bloco e depois referenciadas em comandos posteriores para fins de consistência e para evitar repetições. Certifique-se de executar os comandos em sequência para que esses valores sejam exportados adequadamente e estejam disponíveis para uso em definições subsequentes.

```
export TEMPOUT="$(mktemp)"
export K8S_VERSION=1.33
export KARPENTER_VERSION="1.5.0"
export AWS_REGION="us-east-1"
export EKS_CLUSTER_NAME="eks-rt-inference-${AWS_REGION}"
export S3_BUCKET_NAME="eks-rt-inference-models-${AWS_REGION}-$(date +%s)"
export NVIDIA_BOTTLEROCKET_AMI="$(aws ssm get-parameter --name /aws/service/
bottlerocket/aws-k8s-${K8S_VERSION}-nvidia/x86_64/latest/image_id --query
 Parameter.Value --output text)"
export STANDARD_BOTTLEROCKET_AMI="$(aws ssm get-parameter --name /aws/service/
bottlerocket/aws-k8s-${K8S_VERSION}/arm64/latest/image_id --query Parameter.Value --
output text)"
export AWS_ACCOUNT_ID="$(aws sts get-caller-identity --query Account --output text)"
export ALIAS_VERSION="$(aws ssm get-parameter --name "/aws/service/eks/optimized-
ami/${K8S_VERSION}/amazon-linux-2023/x86_64/standard/recommended/image_id" --query
 Parameter.Value | xargs aws ec2 describe-images --query 'Images[0].Name' --image-ids |
 sed -r 's/^.*(v[[:digit:]]+).*$/\1/')"
```

Criação dos perfis e das políticas necessárias

O Karpenter precisa de políticas e perfis do IAM específicos (por exemplo, perfil do IAM do controlador do Karpenter, perfil de instância e políticas) para gerenciar instâncias do EC2 como nós de processamento do Kubernetes. Ele usa esses perfis para realizar ações como iniciar e encerrar instâncias do EC2, marcar recursos e interagir com outros serviços da AWS. Crie os perfis e as políticas do Karpenter usando o cloudformation.yaml do Karpenter:

```
curl -fsSL https://raw.githubusercontent.com/aws/karpenter-provider-aws/v
${KARPENTER_VERSION}/website/content/en/preview/getting-started/getting-started-with-
karpenter/cloudformation.yaml > "${TEMPOUT}" \
&& aws cloudformation deploy \
  --stack-name "Karpenter-${EKS_CLUSTER_NAME}" \
  --template-file "${TEMPOUT}" \
  --capabilities CAPABILITY_NAMED_IAM \
  --parameter-overrides "ClusterName=${EKS_CLUSTER_NAME}"
```

O AWS LBC precisa de permissão para provisionar e gerenciar balanceadores de carga da AWS, como criar ALBs para recursos do Ingress ou NLBs para serviços do tipo `LoadBalancer`. Especificaremos essa política de permissões durante a criação do cluster. Durante a criação do cluster, criaremos a conta de serviço com o eksctl no ClusterConfig. Criação da política do IAM da LBC:

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerIAMPolicy \
  --policy-document "$(curl -fsSL https://raw.githubusercontent.com/kubernetes-sigs/
aws-load-balancer-controller/v2.14.1/docs/install/iam_policy.json)"
```

Quando o driver CSI Mountpoint S3 é instalado, seus pods DaemonSet são configurados para usar uma conta de serviço para execução. O driver CSI Mountpoint para Mountpoint S3 precisa de permissão para interagir com o bucket do Amazon S3 que você criará posteriormente neste guia. Especificaremos essa política de permissões durante a criação do cluster. Durante a criação do cluster, criaremos a conta de serviço com o eksctl no ClusterConfig. Criação da política do IAM do S3:

```
aws iam create-policy \
    --policy-name S3CSIDriverPolicy \
    --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\":
 \"Allow\", \"Action\": [\"s3:GetObject\", \"s3:PutObject\", \"s3:AbortMultipartUpload
```

```
\", \"s3:DeleteObject\", \"s3:ListBucket\"], \"Resource\": [\"arn:aws:s3:::
${S3_BUCKET_NAME}\", \"arn:aws:s3:::${S3_BUCKET_NAME}/*\"]}]}"
```

Observação: se já existir um perfil com esse nome, dê ao perfil um nome diferente. O perfil que criamos nesta etapa é específico para seu cluster e seu bucket do S3.

Criar um cluster

Neste modelo, o eksctl cria automaticamente uma conta de serviço do Kubernetes para a Identidade de Pods do EKS, o Agente de monitoramento de nós, o CoreDNS, o Kubeproxy e o plug-in CNI da VPC. Atualmente, o driver CSI Mountpoint S3 não está disponível para a Identidade de Pods do EKS, então criamos Perfis do IAM para contas de serviço (IRSA) e um endpoint do OIDC. Além disso, criamos uma conta de serviço para o AWS Load Balancer Controller (LBC). Para acessar os nós do Bottlerocket, o eksctl anexa automaticamente o AmazonSSMManagedInstanceCore para Bottlerocket para permitir sessões de shell seguras via SSM.

No mesmo terminal em que você define suas variáveis de ambiente, execute o bloco de comando a seguir para criar o cluster:

```
eksctl create cluster -f - <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: ${EKS_CLUSTER_NAME}
  region: ${AWS_REGION}
  version: "${K8S_VERSION}"
  tags:
    karpenter.sh/discovery: ${EKS_CLUSTER_NAME}
    # Add more tags if needed for billing
iam:
  # Creates an OIDC endpoint and IRSA service account for the Mountpoint S3 CSI Driver
  # Uses the S3 CSI Driver policy for permissions
  withOIDC: true
  podIdentityAssociations:
  # Creates the pod identity association and service account
  # Uses the Karpenter controller IAM policy for permissions
  - namespace: "kube-system"
    serviceAccountName: karpenter
    roleName: ${EKS_CLUSTER_NAME}-karpenter
    permissionPolicyARNs:
```

```
    - arn:aws:iam::${AWS_ACCOUNT_ID}:policy/KarpenterControllerPolicy-
${EKS_CLUSTER_NAME}
  # Creates the pod identity association and service account
  # Uses the {aws} LBC policy for permissions
  - namespace: kube-system
    serviceAccountName: aws-load-balancer-controller
    createServiceAccount: true
    roleName: AmazonEKSLoadBalancerControllerRole
    permissionPolicyARNs:
    - arn:aws:iam::${AWS_ACCOUNT_ID}:policy/AWSLoadBalancerControllerIAMPolicy
iamIdentityMappings:
- arn: "arn:aws:iam::${AWS_ACCOUNT_ID}:role/KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  username: system:node:{{EC2PrivateDNSName}}
  groups:
  - system:bootstrappers
  - system:nodes
managedNodeGroups:
  # Creates 2 CPU nodes for lightweight system tasks
  - name: ${EKS_CLUSTER_NAME}-m7-cpu
    instanceType: m7g.xlarge
    amiFamily: Bottlerocket
    desiredCapacity: 2
    minSize: 1
    maxSize: 10
    labels:
      role: cpu-worker
# Enable automatic Pod Identity associations for VPC CNI Driver, coreDNS, kube-proxy
addonsConfig:
  autoApplyPodIdentityAssociations: true
addons:
  # Installs the S3 CSI Driver addon and creates IAM role
  # Uses the S3 CSI Driver policy for IRSA permissions
  - name: aws-mountpoint-s3-csi-driver
    attachPolicyARNs:
      - "arn:aws:iam::${AWS_ACCOUNT_ID}:policy/S3CSIDriverPolicy"
  - name: eks-pod-identity-agent
  - name: eks-node-monitoring-agent
  - name: coredns
  - name: kube-proxy
  - name: vpc-cni
EOF
```

Esse processo leva vários minutos para ser concluído. Se quiser monitorar o status, consulte o console do AWS CloudFormation.

2. Verificação da integridade do nó do cluster e do pod

Vamos realizar algumas verificações de integridade para garantir que o cluster esteja pronto. Quando o comando anterior for concluído, veja os tipos de instâncias e verifique se os nós de sistema da CPU atingiram o estado `Ready` com o comando a seguir:

```
kubectl get nodes -L node.kubernetes.io/instance-type
```

A saída esperada deve ser semelhante a essa:

```
NAME                              STATUS   ROLES    AGE     VERSION
 INSTANCE-TYPE
ip-192-168-35-103.ec2.internal   Ready    <none>   12m     v1.33.0-eks-802817d
 m7g.xlarge
ip-192-168-7-15.ec2.internal     Ready    <none>   12m     v1.33.0-eks-802817d
 m7g.xlarge
```

Verifique todas as associações de Identidade de Pods e como elas mapeiam um perfil para uma conta de serviço em um namespace no cluster com o comando a seguir:

```
eksctl get podidentityassociation --cluster ${EKS_CLUSTER_NAME} --region ${AWS_REGION}
```

A saída deve mostrar os perfis do IAM para Karpenter ("karpenter") e o AWS LBC ("aws-load-balancer-controller").

Verifique se os DaemonSets estão disponíveis:

```
kubectl get daemonsets -n kube-system
```

A saída esperada deve ser semelhante a essa:

```
NAME                         DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR
          AGE
aws-node                     3       3       3     3          3         <none>
          12m
dcgm-server                  0       0       0     0          0
 kubernetes.io/os=linux 12m
```

```
eks-node-monitoring-agent      3       3       3     3          3
 kubernetes.io/os=linux 12m
eks-pod-identity-agent         3       3       3     3          3         <none>
          12m
kube-proxy                     3       3       3     3          3         <none>
          12m
s3-csi-node                    2       2       2     2          2
 kubernetes.io/os=linux 12m
```

Verifique se todos os complementos estão instalados no cluster:

```
eksctl get addons --cluster ${EKS_CLUSTER_NAME} --region ${AWS_REGION}
```

A saída esperada deve ser semelhante a essa:

```
NAME                           VERSION              STATUS    ISSUES    IAMROLE
                                      UPDATE AVAILABLE    CONFIGURATION VALUES    POD
 IDENTITY ASSOCIATION ROLES
aws-mountpoint-s3-csi-driver   v1.15.0-eksbuild.1   ACTIVE    0
 arn:aws:iam::143095308808:role/eksctl-eks-rt-inference-us-east-1-addon-aws-m-Role1-
RAUjk4sJnc0L
coredns                        v1.12.1-eksbuild.2   ACTIVE    0
eks-node-monitoring-agent      v1.3.0-eksbuild.2    ACTIVE    0
eks-pod-identity-agent         v1.3.7-eksbuild.2    ACTIVE    0
kube-proxy                     v1.33.0-eksbuild.2   ACTIVE    0
metrics-server                 v0.7.2-eksbuild.3    ACTIVE    0
vpc-cni                        v1.19.5-eksbuild.1   ACTIVE    0
```

3. Instalação do Karpenter

Instale o controlador Karpenter nos nós de processamento da CPU (`cpu-worker`) para otimizar custos e conservar os recursos da GPU. Vamos instalá-lo no namespace "kube-system" e especificar a conta de serviço "karpenter" que definimos durante a criação do cluster. Além disso, esse comando configura o nome do cluster e uma fila de interrupção de instância spot para os nós da CPU. O Karpenter usará IRSA para assumir esse perfil do IAM.

```
# Logout of helm registry before pulling from public ECR
helm registry logout public.ecr.aws

# Install Karpenter
helm upgrade --install karpenter oci://public.ecr.aws/karpenter/karpenter --version
 "${KARPENTER_VERSION}" --namespace "kube-system" --create-namespace  \
```

```
  --set "settings.clusterName=${EKS_CLUSTER_NAME}" \
  --set "settings.interruptionQueue=${EKS_CLUSTER_NAME}" \
  --set controller.resources.requests.cpu=1 \
  --set controller.resources.requests.memory=1Gi \
  --set controller.resources.limits.cpu=1 \
  --set controller.resources.limits.memory=1Gi \
  --set serviceAccount.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::
${AWS_ACCOUNT_ID}:role/${EKS_CLUSTER_NAME}-karpenter" \
  --wait
```

A saída esperada deve ser semelhante a essa:

```
Release "karpenter" does not exist. Installing it now.
Pulled: public.ecr.aws/karpenter/karpenter:1.5.0
Digest: sha256:9a155c7831fbff070669e58500f68d7ccdcf3f7c808dcb4c21d3885aa20c0a1c
NAME: karpenter
LAST DEPLOYED: Thu Jun 19 09:57:06 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Verifique se o Karpenter está em execução:

```
kubectl get pods -n kube-system -l app.kubernetes.io/name=karpenter
```

A saída esperada deve ser semelhante a essa:

```
NAME                         READY   STATUS    RESTARTS   AGE
karpenter-555895dc-865bc   1/1     Running   0          5m58s
karpenter-555895dc-j7tk9   1/1     Running   0          5m58s
```

4. Configuração dos NodePools do Karpenter

Nesta etapa, configuraremos os NodePools do Karpenter de CPU e GPU mutuamente exclusivos. O campo `limits` na especificação do NodePool restringe o total máximo de recursos (por exemplo, CPU, memória, GPUs) que cada NodePool pode consumir em todos os nós provisionados, impedindo o provisionamento adicional de nós se esses limites forem excedidos. Embora o NodePools ofereça suporte a categorias amplas de instâncias (por exemplo, `c`, `g`), a especificação de tipos de instância, tipos de capacidade e limites de recursos específicos ajuda você a estimar

com mais facilidade os custos de suas workloads sob demanda. Nesses NodePools, usamos um conjunto diversificado de tipos de instância dentro da família de instâncias G5. Isso permite que o Karpenter selecione automaticamente o tipo de instância mais apropriado com base nas solicitações de recursos do pod, otimizando a utilização dos recursos e respeitando os limites totais do NodePool. Para saber mais, consulte Criação de NodePools.

Configuração do NodePool de GPU

Neste NodePool, definimos limites de recursos para gerenciar o provisionamento de nós com recursos de GPU. Esses limites foram projetados para limitar o total de recursos em todos os nós do pool, permitindo até 10 instâncias no total. Cada instância pode ser g5.xlarge (4 vCPUs, 16 GiB de memória, 1 GPU) ou g5.2xlarge (8 vCPUs, 32 GiB de memória, 1 GPU), desde que o total de vCPUs não exceda 80, a memória total não exceda 320 GiB e o total de GPUs não exceda 10. Por exemplo, o grupo pode provisionar 10 instâncias g5.2xlarge (80 vCPUs, 320 GiB, 10 GPUs) ou 10 instâncias g5.xlarge (40 vCPUs, 160 GiB, 10 GPUs) ou uma combinação, como 5 g5.xlarge e 5 g5.2xlarge (60 vCPUs, 240 GiB, 10 PUs), garantindo flexibilidade com base nas demandas de workload, respeitando as restrições de recursos.

Além disso, especificamos o ID da variante Nvidia da AMI do Bottlerocket. Por fim, definimos uma política de interrupção para remover nós vazios após 30 minutos (`consolidateAfter: 30m`) e definimos uma vida útil máxima de nó de 30 dias (`expireAfter: 720h`) para otimizar os custos e manter a integridade dos nós para tarefas que exijam muita GPU. Para saber mais, consulte Desabilitação da consolidação do Karpenter para workloads sensíveis à interrupção e Uso de TTLSecondAfterFinished para lipeza automática de tarefas do Kubernetes.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu-a10g-inference-g5
spec:
  template:
    metadata:
      labels:
        role: gpu-worker
        gpu-type: nvidia-a10g
    spec:
      requirements:
        - key: node.kubernetes.io/instance-type
          operator: In
          values: ["g5.xlarge", "g5.2xlarge"]
```

```
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
      nodeClassRef:
        name: gpu-a10g-inference-ec2
        group: karpenter.k8s.aws
        kind: EC2NodeClass
      expireAfter: 720h
  limits:
    cpu: "80"
    memory: "320Gi"
    nvidia.com/gpu: "10"
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: 30m
---
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: gpu-a10g-inference-ec2
spec:
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - id: ${NVIDIA_BOTTLEROCKET_AMI}
  role: "KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  tags:
    nvidia.com/gpu: "true"
EOF
```

A saída esperada deve ser semelhante a essa:

```
nodepool.karpenter.sh/gpu-a10g-inference-g5 created
ec2nodeclass.karpenter.k8s.aws/gpu-a10g-inference-ec2 created
```

Verifique se o NodePool foi criado e está íntegro:

```
kubectl get nodepool gpu-a10g-inference-g5 -o yaml
```

Procure por `status.conditions` como `ValidationSucceeded: True`, `NodeClassReady: True` e `Ready: True` para confirmar se o NodePool está íntegro.

Configuração do NodePool de CPU

Neste NodePool, definimos limites para oferecer suporte a aproximadamente 50 instâncias, alinhando-os a uma workload de CPU moderada (por exemplo, 100-200 pods) e cotas típicas de vCPU da AWS (por exemplo, 128-1152). Os limites são calculados supondo que o NodePool deva aumentar a escala verticalmente até 50 instâncias m7.xlarge: CPU (4 vCPUs por instância × 50 instâncias = 200 vCPUs) e memória (16 GiB por instância × 50 instâncias = 800 GiB). Esses limites foram projetados para limitar o total de recursos em todos os nós do grupo, permitindo instâncias de até 50 m7g.xlarge (cada uma com 4 vCPUs e 16 GiB de memória), desde que o total de vCPUs não exceda 200 e a memória total não exceda 800 GiB.

Além disso, especificamos o ID da variante padrão da AMI do Bottlerocket. Por fim, definimos uma política de interrupção para remover nós vazios após 60 minutos (`consolidateAfter: 60m`) e definimos uma vida útil máxima de nó de 30 dias (`expireAfter: 720h`) para otimizar os custos e manter a integridade dos nós para tarefas que exijam muita GPU. Para saber mais, consulte Desabilitação da consolidação do Karpenter para workloads sensíveis à interrupção e Uso de TTLSecondAfterFinished para lipeza automática de tarefas do Kubernetes.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: cpu-inference-m7gxlarge
spec:
  template:
    metadata:
      labels:
        role: cpu-worker
    spec:
      requirements:
        - key: node.kubernetes.io/instance-type
          operator: In
          values: ["m7g.xlarge"]
```

```
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["on-demand"]
      taints:
        - key: role
          value: cpu-intensive
          effect: NoSchedule
      nodeClassRef:
        name: cpu-inference-m7gxlarge-ec2
        group: karpenter.k8s.aws
        kind: EC2NodeClass
      expireAfter: 720h
  limits:
    cpu: "200"
    memory: "800Gi"
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: 60m
---
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: cpu-inference-m7gxlarge-ec2
spec:
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - id: ${STANDARD_BOTTLEROCKET_AMI}
  role: "KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
EOF
```

A saída esperada deve ser semelhante a essa:

```
nodepool.karpenter.sh/cpu-inference-m7gxlarge created
ec2nodeclass.karpenter.k8s.aws/cpu-inference-m7gxlarge-ec2 created
```

Verifique se o NodePool foi criado e está íntegro:

```
kubectl get nodepool cpu-inference-m7gxlarge -o yaml
```

Procure por `status.conditions` como `ValidationSucceeded: True`, `NodeClassReady: True` e `Ready: True` para confirmar se o NodePool está íntegro.

5. Implantação de um pod de GPU para expor uma GPU

Você precisa do plug-in para dispositivos Nvidia para permitir que o Kubernetes exponha dispositivos de GPU aos clusters Kubernetes. Normalmente, você precisaria implantar o plug-in como DaemonSet; no entanto, a AMI do Bottlerocket pré-instala o plug-in como parte da AMI. Isso significa que, ao usar as AMIs do Bottlerocket, não há necessidade de implantar o DaemonSet do plug-in para dispositivo Nvidia. Para saber mais, consulte Plug-in para dispositivos Kubernetes para expor GPUs.

Implantação de um pod de exemplo

O Karpenter age dinamicamente: ele provisiona os nós da GPU quando uma workload (pod) solicita recursos da GPU. Para verificar se os pods são capazes de solicitar e usar GPUs, implante um pod que solicite o recurso `nvidia.com/gpu` dentro de seus limites (por exemplo, `nvidia.com/gpu: 1`). Para saber mais sobre esses rótulos, consulte Programação de workloads com requisitos de GPU usando rótulos conhecidos.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: gpu-nvidia-smi
spec:
  restartPolicy: OnFailure
  tolerations:
  - key: "nvidia.com/gpu"
    operator: "Exists"
    effect: "NoSchedule"
  nodeSelector:
    role: gpu-worker  # Matches GPU NodePool's label
  containers:
  - name: cuda-container
    image: nvidia/cuda:12.9.1-base-ubuntu20.04
    command: ["nvidia-smi"]
    resources:
      limits:
        nvidia.com/gpu: 1
```

```
      requests:
        nvidia.com/gpu: 1
EOF
```

A saída esperada deve ser semelhante a essa:

```
pod/gpu-ndivia-smi created
```

Espere um minuto e verifique se o pod tem o status "Pendente", "CriandoContêiner", "Em execução", e depois "Concluído":

```
kubectl get pod gpu-nvidia-smi -w
```

Verifique se o nó do pod pertence ao NodePool da GPU:

```
kubectl get node $(kubectl get pod gpu-nvidia-smi -o jsonpath='{.spec.nodeName}') -o
 custom-columns="Name:.metadata.name,Nodepool:.metadata.labels.karpenter\.sh/nodepool"
```

A saída esperada deve ser semelhante a essa:

```
Name                            Nodepool
ip-192-168-83-245.ec2.internal   gpu-a10g-inference-g5
```

Verifique os logs do pod:

```
kubectl logs gpu-nvidia-smi
```

A saída esperada deve ser semelhante a essa:

```
Thu Jul 17 04:31:33 2025
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI 570.148.08                Driver Version: 570.148.08         CUDA
 Version: 12.9 |
|-----------------------------------------+----------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
 |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M.
 |
```

```
|                                         |                      |               MIG M.
 |
|=========================================+======================
+======================|
|   0  NVIDIA A10G                    On  | 00000000:00:1E.0 Off |                    0
 |
|  0%   30C    P8               9W / 300W |      0MiB / 23028MiB |      0%      Default
 |
|                                         |                      |                  N/A
 |
+-----------------------------------------------------------------------------------------
+

+-----------------------------------------------------------------------------------------
+
| Processes:
 |
|  GPU        GI     CI        PID   Type   Process name                  GPU Memory
 |
|                     ID        ID                                         Usage
 |
|
=========================================================================================|
|  No running processes found
 |
+-----------------------------------------------------------------------------------------
+
```

6. (Opcional) Preparação e transferência de artefatos de modelo para implantação

Nesta etapa, você implantará um serviço de modelo para classificação de imagens em tempo real, começando com a transferência dos pesos do modelo para um bucket do Amazon S3. Para demonstração, estamos usando o modelo de visão GPUnet-0 de código aberto, parte da GPUnet da NVIDIA, que suporta inferência de baixa latência em imagens usando GPUs NVIDIA e TensorRT. Esse modelo é pré-treinado no ImageNet, e permite classificar objetos em fotos ou streams de vídeo em tempo real e é considerado um modelo pequeno com 11,9 milhões de parâmetros.

Configure o ambiente.

Para baixar os pesos do modelo GPUnet-0 nesta etapa, você precisará acessar o catálogo NGC da NVIDIA e ter o Docker instalado em sua máquina local. Siga estas etapas para criar uma conta gratuita e configurar a CLI do NGC:

- Inscreva-se em uma conta gratuita do NGC e gere uma chave de API no painel do NGC (Ícone do usuário > Configuração > Gerar chave de API > Gerar chave pessoal > Catálogo NGC).
- Baixe e instale a CLI do NGC (Linux/macOS/Windows) e configure a CLI usando: `ngc config set`. Insira sua chave de API quando solicitada; defina org como `nvidia` e pressione Enter para aceitar padrões para outras pessoas. Se tiver êxito, você deverá ver algo parecido com: `Successfully saved NGC configuration to /Users/your-username/.ngc/config`.

Verificação de permissões de conta de serviço

Antes de começarmos, verifique as permissões da conta de serviço do Kubernetes:

```
kubectl get serviceaccount s3-csi-driver-sa -n kube-system -o yaml
```

Durante a criação do cluster, anexamos a S3CSIDriverPolicy a um perfil do IAM e anotamos a conta de serviço ("s3-csi-driver-sa"). Os pods do driver CSI Mountpoint S3 herdam as permissões do perfil do IAM ao interagir com o S3. A saída esperada deve ser semelhante a essa:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::143095308808:role/eksctl-eks-rt-inference-
us-east-1-addon-aws-m-Role1-fpXXjRYdKN8r
  creationTimestamp: "2025-07-17T03:55:29Z"
  labels:
    app.kubernetes.io/component: csi-driver
    app.kubernetes.io/instance: aws-mountpoint-s3-csi-driver
    app.kubernetes.io/managed-by: EKS
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: s3-csi-driver-sa
  namespace: kube-system
  resourceVersion: "2278"
  uid: 50b36272-6716-4c68-bdc3-c4054df1177c
```

Adição de uma tolerância

O driver CSI S3 é executado como um DaemonSet em todos os nós. Os pods usam o driver CSI nesses nós para montar volumes do S3. Para permitir que ele seja programado em nossos nós de GPU que tenham taint, adicione uma tolerância ao DaemonSet:

```
kubectl patch daemonset s3-csi-node -n kube-system --type='json' -p='[{"op": "add",
 "path": "/spec/template/spec/tolerations/-", "value": {"key": "nvidia.com/gpu",
 "operator": "Exists", "effect": "NoSchedule"}}]'
```

A saída esperada deve ser semelhante a essa:

```
daemonset.apps/s3-csi-node patched
```

Transferência dos pesos do modelo para o S3

Nesta etapa, você criará um bucket do Amazon S3, baixará os pesos do modelo GPUnet-0 da NVIDIA GPU Cloud (NGC) e os transferirá para o bucket. Esses pesos serão acessados por nossa aplicação no runtime para inferência.

Crie seu bucket do Amazon S3:

```
aws s3 mb s3://${S3_BUCKET_NAME} --region ${AWS_REGION}
```

Habilite o versionamento do S3 para o bucket, para evitar que exclusões e substituições acidentais causem perda imediata e permanente de dados:

```
aws s3api put-bucket-versioning --bucket ${S3_BUCKET_NAME} --versioning-configuration
 Status=Enabled
```

Aplique uma regra de ciclo de vida ao bucket para remover versões de objetos substituídas ou excluídas 14 dias depois de se tornarem inatuais, remover marcadores de exclusão expirados e remover transferências incompletas de várias partes após 7 dias. Para saber mais, consulte Exemplos de configuração do Ciclo de Vida do S3.

```
aws s3api put-bucket-lifecycle-configuration --bucket $S3_BUCKET_NAME --lifecycle-
configuration '{"Rules":[{"ID":"LifecycleRule","Status":"Enabled","Filter":
{},"Expiration":{"ExpiredObjectDeleteMarker":true},"NoncurrentVersionExpiration":
{"NoncurrentDays":14},"AbortIncompleteMultipartUpload":{"DaysAfterInitiation":7}}]}'
```

Baixe os pesos do modelo GPUnet-0 do NGC. Por exemplo, no macOS:

```
ngc registry model download-version nvidia/dle/gpunet_0_pyt_ckpt:21.12.0_amp --dest ~/
downloads
```

> **Note**
>
> Talvez seja necessário ajustar esse comando de download para seu sistema operacional. Para que esse comando funcione em um sistema Linux, você provavelmente precisará criar o diretório como parte do comando (por exemplo, `mkdir ~/downloads`).

A saída esperada deve ser semelhante a essa:

```
{
  "download_end": "2025-07-18 08:22:39",
  "download_start": "2025-07-18 08:22:33",
  "download_time": "6s",
  "files_downloaded": 1,
  "local_path": "/Users/your-username/downloads/gpunet_0_pyt_ckpt_v21.12.0_amp",
  "size_downloaded": "181.85 MB",
  "status": "Completed",
  "transfer_id": "gpunet_0_pyt_ckpt[version=21.12.0_amp]"
}
```

Renomeie o arquivo do ponto de verificação para corresponder à nomenclatura esperada no código da nossa aplicação nas etapas posteriores (nenhuma extração é necessária, pois é um ponto de verificação padrão do PyTorch *.pth.tar contendo o dicionário de estado do modelo):

```
mv ~/downloads/gpunet_0_pyt_ckpt_v21.12.0_amp/0.65ms.pth.tar gpunet-0.pth
```

Habilite o AWS Common Runtime na CLI da AWS para otimizar o throughput do S3:

```
aws configure set s3.preferred_transfer_client crt
```

Transfira os pesos do modelo no bucket do S3:

```
aws s3 cp gpunet-0.pth s3://${S3_BUCKET_NAME}/gpunet-0.pth
```

A saída esperada deve ser semelhante a essa:

```
upload: ./gpunet-0.pth to s3://eks-rt-inference-models-us-east-1-1752722786/
gpunet-0.pth
```

Criação do serviço de modelo

Nesta etapa, você configurará uma aplicação Web FastAPI para classificação de imagens acelerada por GPU usando o modelo de visão GPUnet-0. A aplicação baixa os pesos do modelo do Amazon S3 no runtime, busca a arquitetura do modelo no repositório da NVIDIA para armazenamento em cache e baixa os rótulos da classe ImageNet via HTTP. A aplicação inclui transformações de pré-processamento de imagens e expõe dois endpoints: um GET raiz para verificação de status e um endpoint POST `/predict` que aceita uma URL de imagem.

Servimos o modelo usando FastAPI com PyTorch, carregando pesos do Amazon S3 no runtime em uma configuração em contêineres para prototipagem rápida e implantação do Kubernetes. Para outros métodos, como lotes otimizados ou mecanismos de alto throughput, consulte Fornecimento de modelos de ML.

Criar a aplicação

Crie um diretório para os arquivos da sua aplicação `model-testing`, como, em seguida, altere os diretórios nele e adicione o código a seguir a um novo arquivo chamado `app.py`:

```
import os
import torch
import json
import requests
from fastapi import FastAPI, HTTPException
from PIL import Image
from io import BytesIO, StringIO
import torchvision.transforms as transforms
from torch.nn.functional import softmax
import warnings
from contextlib import redirect_stdout, redirect_stderr
import argparse
import boto3
app = FastAPI()

# Suppress specific warnings from the model code (quantization is optional and unused
 here)
warnings.simplefilter("ignore", UserWarning)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load model code from cache (if present)
# Use backed cache directory
```

```
torch.hub.set_dir('/cache/torch/hub')

# Allowlist for secure deserialization (handles potential issues in older checkpoints)
torch.serialization.add_safe_globals([argparse.Namespace])
# Load the model architecture only on container startup (changed to pretrained=False)
# Precision (FP32 for full accuracy, could be 'fp16' for speed on Ampere+ GPUs)
with redirect_stdout(StringIO()), redirect_stderr(StringIO()):
    gpunet = torch.hub.load('NVIDIA/DeepLearningExamples:torchhub', 'nvidia_gpunet',
 pretrained=False, model_type='GPUNet-0', model_math='fp32')

# Download weights from S3 if not present, then load them
model_path = os.getenv('MODEL_PATH', '/cache/torch/hub/checkpoints/gpunet-0.pth')
os.makedirs(os.path.dirname(model_path), exist_ok=True)  # Ensure checkpoints dir
 exists
if not os.path.exists(model_path):
    s3 = boto3.client('s3')
    s3.download_file(os.getenv('S3_BUCKET_NAME'), 'gpunet-0.pth', model_path)
checkpoint = torch.load(model_path, map_location=device, weights_only=True)
gpunet.load_state_dict(checkpoint['state_dict'])
# Move to GPU/CPU
gpunet.to(device)
gpunet.eval()

# Preprocessing
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])

# Load ImageNet labels
labels_url = "https://s3.amazonaws.com/deep-learning-models/image-models/
imagenet_class_index.json"
response = requests.get(labels_url)
json_data = json.loads(response.text)
labels = [json_data[str(i)][1].replace('_', ' ') for i in range(1000)]

# Required, FastAPI root
@app.get("/")
async def hello():
    return {"status": "hello"}

# Serve model requests
```

```
@app.post("/predict")
async def predict(image_url: str):
    try:
        response = requests.get(image_url)
        response.raise_for_status()
        img = Image.open(BytesIO(response.content)).convert("RGB")
        input_tensor = preprocess(img).unsqueeze(0).to(device)

        with torch.no_grad():
            output = gpunet(input_tensor)

        probs = softmax(output, dim=1)[0]
        top5_idx = probs.topk(5).indices.cpu().numpy()
        top5_probs = probs.topk(5).values.cpu().numpy()

        results = [{ "label": labels[idx], "probability": float(prob) } for idx, prob
 in zip(top5_idx, top5_probs)]

        return {"predictions": results}
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))
```

Crie o Dockerfile

O Dockerfile a seguir cria uma imagem de contêiner para nossa aplicação utilizando o modelo GPUnet do repositório NVIDIA Deep Learning Examples for Tensor Cores do GitHub.

Reduzimos o tamanho da imagem do contêiner usando uma base PyTorch somente no runtime, instalando somente pacotes essenciais com limpeza de cache, pré-armazenamento em cache do código do modelo e evitando o "baking" de pesos na imagem do contêiner para permitir extrações e atualizações mais rápidas. Para saber mais, consulte Redução de tamanhos de imagens de contêineres.

No mesmo diretório do `app.py`, crie o `Dockerfile`:

```
FROM pytorch/pytorch:2.4.0-cuda12.4-cudnn9-runtime

# Install required system packages required for git cloning
RUN apt-get update && apt-get install -y git && rm -rf /var/lib/apt/lists/*

# Install application dependencies
RUN pip install --no-cache-dir fastapi uvicorn requests pillow boto3 timm==0.5.4
```

```
# Pre-cache the GPUNet code from Torch Hub (without weights)
# Clone the repository containing the GPUNet code
RUN mkdir -p /cache/torch/hub && \
    cd /cache/torch/hub && \
    git clone --branch torchhub --depth 1 https://github.com/NVIDIA/
DeepLearningExamples NVIDIA_DeepLearningExamples_torchhub

COPY app.py /app/app.py

WORKDIR /app

CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "80"]
```

Teste a aplicação

No mesmo diretório do seu `app.py` e `Dockerfile`, crie a imagem do contêiner para a aplicação de inferência, visando a arquitetura AMD64:

```
docker build --platform linux/amd64 -t gpunet-inference-app .
```

Defina variáveis de ambiente para suas credenciais da AWS e, opcionalmente, um token de sessão da AWS. Por exemplo:

```
export AWS_REGION="us-east-1"
export AWS_ACCESS_KEY_ID=ABCEXAMPLESCUJFEIELSMUHHAZ
export AWS_SECRET_ACCESS_KEY=123EXAMPLEMZREoQXr8XkiicsOgWDQ5TpUsq0/Z
```

Execute o contêiner localmente, injetando credenciais da AWS como variáveis de ambiente para acesso ao S3. Por exemplo:

```
docker run --platform linux/amd64 -p 8080:80 \
  -e S3_BUCKET_NAME=${S3_BUCKET_NAME} \
  -e AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID} \
  -e AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY} \
  -e AWS_DEFAULT_REGION=${AWS_REGION} \
  gpunet-inference-app
```

A saída esperada deve ser semelhante a essa:

```
INFO:     Started server process [1]
```

```
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)
```

Em uma nova janela de terminal, teste o endpoint de inferência enviando uma amostra de solicitação POST com uma URL de imagem pública como parâmetro de consulta:

```
curl -X POST "http://localhost:8080/predict?image_url=http://images.cocodataset.org/
test-stuff2017/000000024309.jpg"
```

A saída esperada deve ser uma resposta JSON com as 5 principais previsões, semelhante a esta (rótulos e probabilidades reais podem variar um pouco com base na precisão da imagem e do modelo):

```
{"predictions":[{"label":"desk","probability":0.28885871171951294},
{"label":"laptop","probability":0.24679335951805115},
{"label":"notebook","probability":0.08539070934057236},
{"label":"library","probability":0.030645888298749924},
{"label":"monitor","probability":0.02989606373012066}]}
```

Saia da aplicação usando "Ctrl + C".

Envio do contêiner para o Amazon ECR por push

Nesta etapa, carregamos a imagem do contêiner do serviço de modelo GPUnet-0 no Amazon Elastic Container Registry (ECR), disponibilizando-a para implantação no Amazon EKS. Esse processo envolve a criação de um novo repositório do ECR para armazenar a imagem, a autenticação com o ECR e, em seguida, a marcação e o envio da imagem do contêiner para o nosso registro.

Primeiro, volte para o diretório em que você definiu suas variáveis de ambiente no início deste guia. Por exemplo:

```
cd ..
```

Crie um repositório no Amazon ECR:

```
aws ecr create-repository --repository-name gpunet-inference-app --region ${AWS_REGION}
```

Faça login no Amazon ECR:

```
aws ecr get-login-password --region ${AWS_REGION} | docker login --username AWS --
password-stdin ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com
```

A saída esperada deve ser semelhante a essa:

```
Login Succeeded
```

Aplique tags na imagem:

```
docker tag gpunet-inference-app:latest ${AWS_ACCOUNT_ID}.dkr.ecr.
${AWS_REGION}.amazonaws.com/gpunet-inference-app:latest
```

Envie a imagem por push para o seu repositório do Amazon ECR:

```
docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/gpunet-inference-
app:latest
```

Demorará vários minutos para que essa etapa seja concluída.

7. (Opcional) Exposição do serviço de modelo

Nesta etapa, você exporá seu serviço de modelo de inferência em tempo real externamente no Amazon EKS usando o AWS Load Balancer Controller (LBC). Isso envolve configurar o LBC, montar pesos do modelo do Amazon S3 como um volume persistente usando o driver CSI Mountpoint S3, implantar um pod de aplicações acelerado por GPU, criar um serviço e uma entrada para provisionar um Application Load Balancer (ALB) e testar o endpoint.

Primeiro, verifique a associação de Identidade de Pods com o AWS LBC, confirmando se a conta de serviço está vinculada corretamente ao perfil do IAM necessário:

```
eksctl get podidentityassociation --cluster ${EKS_CLUSTER_NAME} --namespace kube-system
 --service-account-name aws-load-balancer-controller
```

A saída esperada deve ser semelhante a essa:

```
ASSOCIATION ARN                                                NAMESPACE    SERVICE
 ACCOUNT NAME        IAM ROLE ARN    OWNER ARN
arn:aws:eks:us-east-1:143095308808:podidentityassociation/eks-rt-inference-us-
east-1/a-buavluu2wp1jropya    kube-system     aws-load-balancer-controller
 arn:aws:iam::143095308808:role/AmazonEKSLoadBalancerControllerRole
```

Aplicação de tags no seu grupo de segurança de cluster

O AWS Load Balancer Controller oferece suporte apenas um único grupo de segurança com a chave de tag `karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"` para a seleção do grupo de segurança do Karpenter. Ao criar um cluster com o eksctl, o grupo de segurança padrão do cluster (que tem a tag `"kubernetes.io/cluster/<cluster-name>: owned"`) não é automaticamente marcado com tags `karpenter.sh/discovery`. Essa tag é essencial para que o Karpenter descubra e anexe esse grupo de segurança aos nós que ele provisiona. A associação desse grupo de segurança garante a compatibilidade com o AWS Load Balancer Controller (LBC), permitindo que ele gerencie automaticamente as regras de tráfego de entrada para serviços expostos via Ingress, como o serviço modelo nessas etapas.

Exporte o ID da VPC para o seu cluster:

```
CLUSTER_VPC_ID="$(aws eks describe-cluster --name ${EKS_CLUSTER_NAME} --query
 cluster.resourcesVpcConfig.vpcId --output text)"
```

Exporte o grupo de segurança padrão para seu cluster:

```
CLUSTER_SG_ID="$(aws ec2 describe-security-groups --filters Name=vpc-id,Values=
$CLUSTER_VPC_ID Name=tag-key,Values=kubernetes.io/cluster/${EKS_CLUSTER_NAME} --query
 'SecurityGroups[].[GroupId]' --output text)"
```

Adicione a tag `karpenter.sh/discovery` ao grupo de segurança padrão do cluster. Isso permitirá que nossos seletores EC2NodeClass de CPU e GPU o usem:

```
aws ec2 create-tags --resources ${CLUSTER_SG_ID} --tags Key=karpenter.sh/
discovery,Value=${EKS_CLUSTER_NAME}
```

Verifique se a tag foi adicionada:

```
aws ec2 describe-security-groups --group-ids ${CLUSTER_SG_ID} --query
 "SecurityGroups[].Tags"
```

Entre os resultados, você deve ver a saída a seguir com a tag e o nome do seu cluster. Por exemplo:

```
{
  "Key": "karpenter.sh/discovery",
  "Value": "eks-rt-inference-us-east-1"
```

```
}
```

Configuração do AWS Load Balancer Controller (LBC)

O AWS LBC é essencial para gerenciar o tráfego de entrada para workloads de IA/ML no Amazon EKS, garantindo acesso a endpoints de inferência ou pipelines de processamento de dados. Ao se integrar com AWS Application Load Balancers (ALB) e Network Load Balancers (NLB), o LBC roteia dinamicamente o tráfego para aplicações em contêineres, como aqueles que executam grandes modelos de linguagem, modelos de visão computacional ou serviços de inferência em tempo real. Como já criamos a conta de serviço e a associação de Identidade de Pods durante a criação do cluster, definimos `serviceAccount.name` para corresponder ao que está definido na configuração do nosso cluster (`aws-load-balancer-controller`).

Adicione o repositório de chart do Helm eks-charts de propriedade de AWS:

```
helm repo add eks https://aws.github.io/eks-charts
```

Atualize seus repositórios locais do Helm com os charts mais recentes:

```
helm repo update eks
```

Implante o AWS LBC usando o Helm, especificando o nome do cluster do EKS e referenciando a conta de serviço pré-criada:

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=${EKS_CLUSTER_NAME} \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

A saída esperada deve ser semelhante a essa:

```
NAME: aws-load-balancer-controller
LAST DEPLOYED: Wed Jul 9 15:03:31 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```

Monte o modelo em um volume persistente

Nesta etapa, você montará pesos de modelo a partir do seu bucket do Amazon S3 usando um PersistentVolume (PV) apoiado pelo driver CSI Mountpoint para Amazon S3. Isso permite que os pods do Kubernetes acessem objetos do S3 como arquivos locais, eliminando downloads que consumam muitos recursos para armazenamento temporário de pods ou contêineres iniciais, ideais para pesos de modelos grandes de vários gigabytes.

O PV monta toda a raiz do bucket (nenhum caminho especificado em `volumeAttributes`), oferece suporte ao acesso simultâneo somente para leitura por vários pods e expõe arquivos como os pesos do modelo (`/models/gpunet-0.pth`) dentro do contêiner para inferência. Isso garante que o "download" alternativo em nossa aplicação (`app.py`) não seja acionado porque o arquivo existe por meio da montagem. Ao desacoplar o modelo da imagem do contêiner, isso permite acesso compartilhado e atualizações independentes da versão do modelo sem reconstruções de imagens.

Criação do PersistentVolume (PV)

Crie um recurso PersistentVolume (PV) para montar o bucket do S3 contendo os pesos do seu modelo, permitindo o acesso somente de leitura a vários pods sem baixar arquivos no runtime:

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: PersistentVolume
metadata:
  name: s3-model-pv
spec:
  capacity:
    storage: 5Gi  # Ignored by the driver; can be any value
  accessModes:
    - ReadOnlyMany  # Read only
  persistentVolumeReclaimPolicy: Retain
  storageClassName: ""  # Required for static provisioning
  claimRef:
    namespace: default  # Adjust if you prefer a different namespace
    name: s3-model-pvc
  mountOptions:
    - allow-other  # Enables multi-user access (useful for non-root pods)
    - region ${AWS_REGION} # Optional, include if your bucket is in a different region
 than the cluster
  csi:
    driver: s3.csi.aws.com
    volumeHandle: gpunet-model-volume  # Must be unique across all PVs
```

```
    volumeAttributes:
      bucketName: ${S3_BUCKET_NAME}
EOF
```

Criação do PersistentVolumeClaim (PVC)

Crie um PersistentVolumeClaim (PVC) para vincular ao PV, solicitando acesso somente de leitura aos dados do modelo S3 montado:

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: s3-model-pvc
spec:
  accessModes:
    - ReadOnlyMany
  storageClassName: ""  # Required for static provisioning
  resources:
    requests:
      storage: 5Gi  # Ignored, match PV capacity
  volumeName: s3-model-pv  # Bind to the PV created above
EOF
```

Implantar a aplicação

Implante a aplicação de inferência como uma implantação do Kubernetes, montando o volume persistente baseado em S3 para acesso ao modelo, aplicando seletores e tolerâncias de nós da GPU e definindo variáveis de ambiente para o caminho do modelo. Essa implantação define o caminho do modelo (var env de "`/models/gpunet-0.pth`"), então nossa aplicação (em `app.py`) usará esse caminho por padrão. Com a montagem do volume de implantação em `/models` (somente leitura), o download do modelo não será acionado se o arquivo já estiver presente por meio do PVC.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpunet-inference-app
spec:
  replicas: 1
  selector:
```

```
    matchLabels:
      app: gpunet-inference-app
  template:
    metadata:
      labels:
        app: gpunet-inference-app
    spec:
      tolerations:
      - key: "nvidia.com/gpu"
        operator: "Exists"
        effect: "NoSchedule"
      nodeSelector:
        role: gpu-worker
      containers:
      - name: inference
        image: ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/gpunet-inference-
app:latest
        ports:
        - containerPort: 80
        env:
        - name: MODEL_PATH
          value: "/models/gpunet-0.pth"
        resources:
          limits:
            nvidia.com/gpu: 1
          requests:
            nvidia.com/gpu: 1
        volumeMounts:
        - name: model-volume
          mountPath: /models
          readOnly: true
      volumes:
      - name: model-volume
        persistentVolumeClaim:
          claimName: s3-model-pvc
EOF
```

Serão necessários alguns minutos para que o Karpenter provisione um nó de GPU, caso ainda não haja um disponível. Verifique se o pod de inferência está em um estado "Em execução":

```
kubectl get pods -l app=gpunet-inference-app
```

A saída esperada deve ser semelhante a essa:

```
NAME                                  READY   STATUS    RESTARTS   AGE
gpunet-inference-app-5d4b6c7f8-abcde        1/1     Running   0          2m
```

Exposição do serviço com Ingress e balanceador de carga

Crie um serviço ClusteriP para expor a implantação de inferência internamente no cluster do EKS, visando a porta da aplicação:

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: gpunet-model-service
spec:
  type: ClusterIP
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: gpunet-inference-app
EOF
```

Crie um recurso do Ingress para provisionar um Application Load Balancer (ALB) voltado para a Internet por meio do AWS LBC, roteando o tráfego externo para o serviço de inferência:

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: gpunet-model-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
```

```
            name: gpunet-model-service
            port:
              number: 80
EOF
```

Espere alguns minutos para que o Application Load Balancer (ALB) conclua o provisionamento. Monitore o status do recurso de entrada para confirmar se o ALB foi provisionado:

```
kubectl get ingress gpunet-model-ingress
```

A saída esperada deve ter a seguinte aparência (com o campo ENDEREÇO preenchido):

```
NAME                   CLASS   HOSTS   ADDRESS
 PORTS   AGE
gpunet-model-ingress   alb     *       k8s-default-gpunetmo-183de3f819-516310036.us-
east-1.elb.amazonaws.com   80      6m58s
```

Extraia e exporte o nome do host de ALB do status do Ingress para uso em testes subsequentes:

```
export ALB_HOSTNAME=$(kubectl get ingress gpunet-model-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

Teste do serviço de modelo

Valide o endpoint de inferência exposto enviando uma solicitação POST com um exemplo de URL de imagem (por exemplo, do conjunto de dados COCO), simulando a previsão em tempo real:

```
curl -X POST "http://${ALB_HOSTNAME}/predict?image_url=http://images.cocodataset.org/
test-stuff2017/000000024309.jpg"
```

A saída esperada deve ser uma resposta JSON com as 5 principais previsões, semelhante a esta (rótulos e probabilidades reais podem variar um pouco com base na precisão da imagem e do modelo):

```
{"predictions":[{"label":"desk","probability":0.2888975441455841},
{"label":"laptop","probability":0.2464350312948227},
{"label":"notebook","probability":0.08554483205080032},
{"label":"library","probability":0.030612602829933167},
{"label":"monitor","probability":0.02989672815084457}]}
```

Opcionalmente, é possível continuar testando outras imagens em uma nova solicitação POST. Por exemplo:

```
http://images.cocodataset.org/test-stuff2017/000000024309.jpg
http://images.cocodataset.org/test-stuff2017/000000028117.jpg
http://images.cocodataset.org/test-stuff2017/000000006149.jpg
http://images.cocodataset.org/test-stuff2017/000000004954.jpg
```

Conclusão

Neste guia, você configurou um cluster do Amazon EKS otimizado para workload de inferência em tempo real aceleradas por GPU. Você provisionou um cluster com instâncias G5 do EC2, instalou o driver CSI Mountpoint S3, o Agente de Identidade de Pods do EKS, o Agente de monitoramento de nós do EKS, a AMI do Bottlerocket, o AWS Load Balancer Controller (LBC) e o Karpenter para gerenciar NodePools de CPUs e GPUs. Você usou o plug-in de dispositivo NVIDIA para habilitar a programação da GPU e configurou o S3 com PersistentVolume e PersistentVolumeClaim para acesso ao modelo. Você validou a configuração implantando um pod de GPU de exemplo, configurando o acesso ao modelo GPUnet-0 da NVIDIA no Amazon S3, habilitando a inicialização do pod e expondo o serviço de inferência por meio do Application Load Balancer. Para utilizar totalmente seu cluster, configure o Agente de monitoramento de nós do EKS com reparo automático. Certifique-se de realizar testes de benchmark, incluindo avaliações de performance, latência e throughput da GPU, para otimizar os tempos de resposta. Para saber mais, consulte Uso de ferramentas de monitoramento e observabilidade para suas workloads de IA/ML.

Limpeza

Para evitar cobranças futuras, você precisará excluir manualmente a pilha associada do CloudFormation para excluir todos os recursos criados durante este guia, incluindo a rede VPC.

Exclua a pilha do CloudFormation usando o sinalizador `--wait` com o eksctl:

```
eksctl delete cluster --region ${AWS_REGION} --name ${EKS_CLUSTER_NAME} --wait
```

Após a conclusão, você verá a seguinte saída de resposta:

```
2025-07-29 13:03:55 [#]  all cluster resources were deleted
```

Exclua o bucket do Amazon S3 criado durante este guia usando o console do Amazon S3.

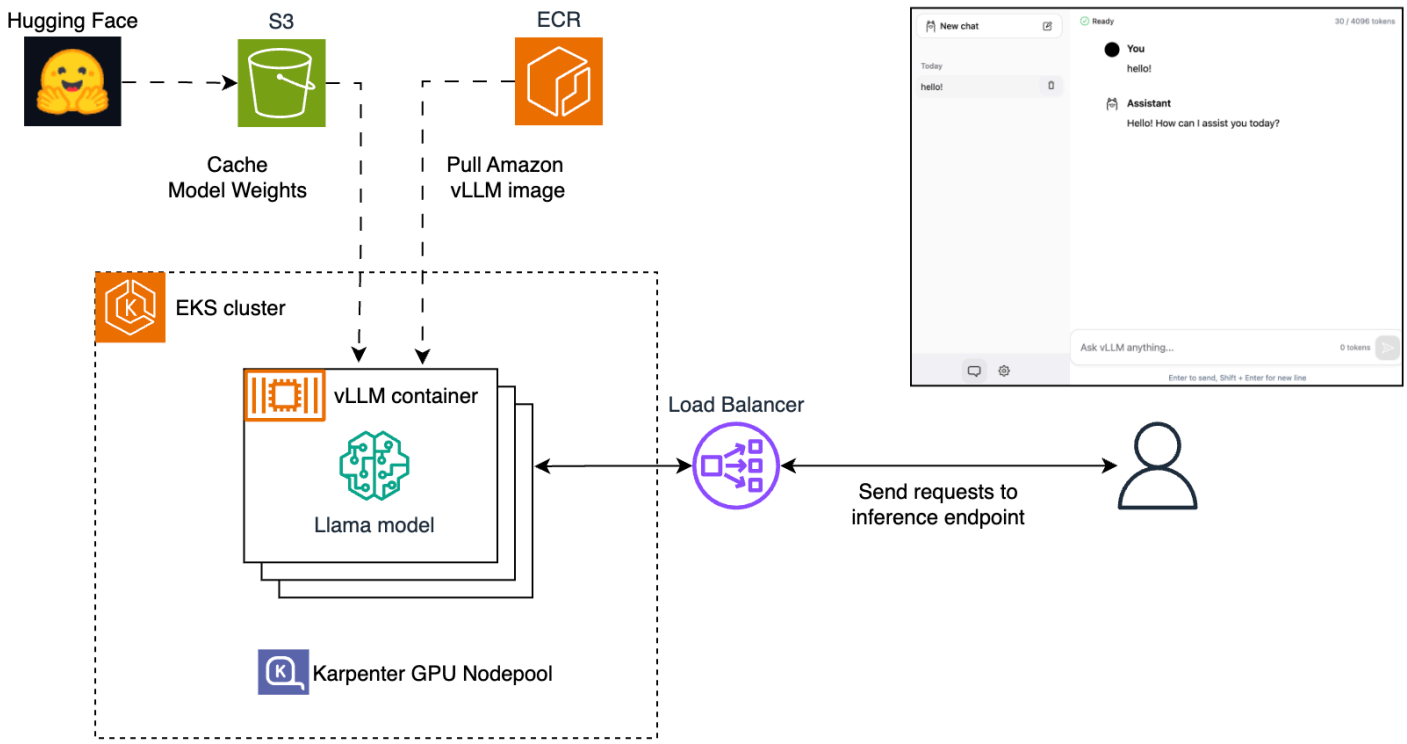
Início rápido: inferência de LLM de alto throughput usando vLLM no Amazon EKS

Introdução

Este guia de início rápido fornece um passo a passo para implantar grandes modelos de linguagem (LLMs) no Amazon EKS usando vLLM e GPUs para aplicações de inferência textual em tempo real.

A solução emprega o Amazon EKS para orquestração de contêiner e o vLLM para disponibilizar modelos de forma eficiente, permitindo o desenvolvimento de aplicações baseadas em IA escaláveis com aceleração por GPU e inferência de alto throughput. Para demonstração, utiliza-se o modelo Llama 3.1 8B Instruct, mas é possível implantar qualquer outro LLM compatível com o vLLM (consulte a documentação do vLLM para obter a lista de modelos compatíveis). Para testar a inferência de LLM, utilizamos uma aplicação de chatbot de exemplo baseada no projeto nextjs-vllm-ui. Por fim, usamos o GuideLLM para realizar avaliações comparativas e ajustar os parâmetros de configuração do vLLM, a fim de otimizar a performance da inferência.

Arquitetura do vLLM no EKS



Ao concluir este procedimento, você terá um endpoint de inferência vLLM otimizado para throughput e baixa latência e poderá interagir com um modelo do Llama por meio de uma aplicação frontend de

chat, demonstrando um caso de uso típico para assistentes de chatbot e outras aplicações baseadas em LLMs.

Para obter orientação adicional e recursos avançados de implantação, consulte nosso Guia de práticas recomendadas do EKS para workloads de IA/ML e a página AI on EKS inference charts pronta para implantação em ambientes de produção.

Antes de começar

Antes de começar, verifique se você conta com:

- Um cluster do Amazon EKS com os seguintes componentes principais: grupos de nós do Karpenter usando instâncias do EC2 das famílias G5 ou G6, o plug-in de dispositivo da NVIDIA instalado nos nós de processamento com a GPU habilitada e o driver CSI para o ponto de montagem do S3 instalado. Para criar essa configuração básica, siga as etapas apresentadas em the section called “Criação de cluster” até concluir a Etapa 4.
- Uma conta do Hugging Face. Para se cadastrar, acesse https://huggingface.co/login.

Configuração do armazenamento de modelos com o Amazon S3

Armazene arquivos grandes de LLM de forma eficiente no Amazon S3, separando o armazenamento dos recursos de computação. Essa abordagem agiliza as atualizações de modelos, reduz os custos e simplifica o gerenciamento em ambientes de produção. O S3 garante armazenamento confiável de arquivos massivos, e a integração com Kubernetes, usando o driver CSI para o ponto de montagem, permite que os pods acessem os modelos como se fossem armazenamento local, sem necessidade de downloads demorados a cada inicialização. Siga estas etapas para criar um bucket do S3, fazer upload de um LLM e montá-lo como um volume no seu contêiner de inferência.

Outras soluções de armazenamento também estão disponíveis no EKS para o armazenamento em cache de modelos, como o EFS e o FSx para Lustre. Para obter mais informações, consulte Práticas recomendadas do EKS.

Definição de variáveis de ambiente

Crie um nome exclusivo para um novo bucket do Amazon S3, que será criado posteriormente neste guia. Uma vez criado, use este mesmo nome de bucket em todas as etapas. Por exemplo:

```
MY_BUCKET_NAME=model-store-$(date +%s)
```

Defina variáveis de ambiente e armazene-as em um arquivo:

```
cat << EOF > .env-quickstart-vllm
export BUCKET_NAME=${MY_BUCKET_NAME}
export AWS_REGION=us-east-1
export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
EOF
```

Carregue as variáveis de ambiente no seu ambiente de shell. Se você fechar o ambiente de shell atual e abrir um novo, certifique-se de recarregar as variáveis de ambiente usando este mesmo comando:

```
source .env-quickstart-vllm
```

Criação de um bucket do S3 para armazenar os arquivos do modelo

Crie um bucket do S3 para armazenar os arquivos do modelo:

```
aws s3 mb s3://${BUCKET_NAME} --region ${AWS_REGION}
```

Download do modelo do Hugging Face

O Hugging Face é um dos principais hubs de modelos para acessar modelos de LLM. Para fazer o download do modelo Llama, é necessário aceitar a licença do modelo e configurar a autenticação via token:

1. Para continuar, aceite a licença do modelo Llama 3.1 8B Instruct em https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct.
2. Gere um token de acesso (acesse seu Perfil > Configurações > Tokens de acesso, e, em seguida, crie um novo token usando o tipo Leitura).

Configure uma variável de ambiente usando o token do Hugging Face:

```
export HF_TOKEN=your_token_here
```

Instale o pacote pip3, caso ele ainda não esteja presente em seu ambiente. Comando de exemplo no Amazon Linux 2023:

```
sudo dnf install -y python3-pip
```

Faça a instalação da CLI do Hugging Face:

```
pip install huggingface-hub
```

Faça o download do modelo Llama-3.1-8B-Instruct do Hugging Face (que tem aproximadamente 15 GB) usando o sinalizador `--exclude` para ignorar o formato legado PyTorch e fazer download apenas dos arquivos em formato safetensors otimizado, reduzindo o tamanho do download sem perder compatibilidade com os mecanismos de inferência mais usados:

```
huggingface-cli download meta-llama/Meta-Llama-3.1-8B-Instruct \
  --exclude "original/*" \
  --local-dir ./llama-3.1-8b-instruct \
  --token $HF_TOKEN
```

Verifique os arquivos que foram baixados:

```
$ ls llama-3.1-8b-instruct
```

A saída esperada deve ser semelhante a essa:

```
LICENSE        config.json                      model-00002-of-00004.safetensors
 model.safetensors.index.json  tokenizer_config.json
README.md      generation_config.json           model-00003-of-00004.safetensors
 special_tokens_map.json
USE_POLICY.md  model-00001-of-00004.safetensors  model-00004-of-00004.safetensors
 tokenizer.json
```

Upload dos arquivos do modelo

Habilite o AWS Common Runtime (CRT) para melhorar a performance de transferências no S3. O cliente de transferência baseado em CRT garante maior throughput e confiabilidade ao lidar com arquivos de grande porte:

```
aws configure set s3.preferred_transfer_client crt
```

Faça o upload do modelo:

```
aws s3 cp ./llama-3.1-8b-instruct s3://$BUCKET_NAME/llama-3.1-8b-instruct \
```

```
  --recursive
```

A saída esperada deve ser semelhante a essa:

```
...
upload: llama-3.1-8b-instruct/tokenizer.json to s3://model-store-1753EXAMPLE/
llama-3.1-8b-instruct/tokenizer.json
upload: llama-3.1-8b-instruct/model-00004-of-00004.safetensors to s3://model-
store-1753890326/llama-3.1-8b-instruct/model-00004-of-00004.safetensors
upload: llama-3.1-8b-instruct/model-00002-of-00004.safetensors to s3://model-
store-1753890326/llama-3.1-8b-instruct/model-00002-of-00004.safetensors
upload: llama-3.1-8b-instruct/model-00003-of-00004.safetensors to s3://model-
store-1753890326/llama-3.1-8b-instruct/model-00003-of-00004.safetensors
upload: llama-3.1-8b-instruct/model-00001-of-00004.safetensors to s3://model-
store-1753890326/llama-3.1-8b-instruct/model-00001-of-00004.safetensors
```

Configuração de permissões da CSI para o ponto de montagem do S3

O driver CSI para o ponto de montagem do S3 possibilita uma integração nativa entre o Kubernetes e o S3, permitindo que os pods acessem diretamente os arquivos do modelo como se fossem armazenamento local e eliminando a necessidade de cópias locais durante a inicialização do contêiner.

Crie uma política do IAM para permitir que o ponto de montagem do S3 realize a leitura do bucket do S3:

```
aws iam create-policy \
  --policy-name S3BucketAccess-${BUCKET_NAME} \
  --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\":
 \"Allow\", \"Action\": [\"s3:GetObject\", \"s3:GetObjectVersion\", \"s3:ListBucket
\", \"s3:GetBucketLocation\"], \"Resource\": [\"arn:aws:s3:::${BUCKET_NAME}\",
 \"arn:aws:s3:::${BUCKET_NAME}/*\"]}]}"
```

Localize o nome do perfil do IAM utilizado pela CSI para o ponto de montagem do S3 conferindo as anotações da conta de serviço do driver CSI do S3:

```
ROLE_NAME=$(kubectl get serviceaccount s3-csi-driver-sa -n kube-system -o
 jsonpath='{.metadata.annotations.eks\.amazonaws\.com/role-arn}' | cut -d'/' -f2)
```

Anexe a política do IAM ao perfil da CSI para o ponto de montagem do S3:

```
aws iam attach-role-policy \
 --role-name ${ROLE_NAME} \
 --policy-arn arn:aws:iam::${AWS_ACCOUNT_ID}:policy/S3BucketAccess-${BUCKET_NAME}
```

Caso a CSI para o ponto de montagem do S3 não esteja instalada no cluster, siga as etapas de implantação apresentadas em the section called “Criação de cluster”.

Montagem do bucket do S3 como um volume do Kubernetes

Crie um volume persistente (PV, na sigla em inglês) e uma solicitação de volume persistente (PVC, na sigla em inglês) para fornecer acesso somente à leitura ao bucket do S3 em diversos pods de inferência. O modo de acesso ReadOnlyMany permite que vários pods acessem simultaneamente os arquivos do modelo, com o driver CSI cuidando da montagem do bucket do S3:

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: PersistentVolume
metadata:
  name: model-store
spec:
  storageClassName: ""
  capacity:
    storage: 100Gi
  accessModes:
    - ReadOnlyMany
  persistentVolumeReclaimPolicy: Retain
  mountOptions:
    - region ${AWS_REGION}
  csi:
    driver: s3.csi.aws.com
    volumeHandle: model-store
    volumeAttributes:
      bucketName: ${BUCKET_NAME}
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: model-store
spec:
  storageClassName: ""
  volumeName: model-store
  accessModes:
```

```
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
EOF
```

Configuração da infraestrutura de GPU

Nós do cluster

Estamos usando o cluster do EKS criado no the section called “Criação de cluster”. Este cluster inclui grupos de nós do Karpenter capazes de provisionar nós com GPU e armazenamento suficiente para fazer download da imagem do contêiner vLLM. Se estiver usando um cluster do EKS personalizado, certifique-se de que ele consegue iniciar nós com GPU.

Seleção de instância

A seleção adequada da instância para inferência de LLM requer garantir que a memória disponível da GPU seja suficiente para carregar os pesos do modelo. Os pesos do modelo Llama 3.1 8B Instruct ocupam cerca de 16 GB (tamanho dos arquivos de modelo .safetensor). Portanto, é necessário disponibilizar, pelo menos, essa quantidade de memória para que o processo vLLM realize o carregamento do modelo.

As instâncias G5 do Amazon EC2 com GPUs A10G e as instâncias G6 do EC2 com GPUs L4 oferecem 24 GB de VRAM por GPU, garantindo memória suficiente para carregar os pesos do Llama 3.1 8B Instruct. Se você estiver implantando um modelo com pesos maiores, considere usar um tipo de instância com diversas GPUs ou uma configuração com vários nós.

Drivers de dispositivos da NVIDIA

Os drivers da NVIDIA fornecem o ambiente de runtime necessário para que os contêineres acessem os recursos da GPU de forma eficiente. Esses drivers permitem a alocação e o gerenciamento de recursos de GPU no Kubernetes, tornando as GPUs disponíveis como recursos que podem ser agendados.

O cluster usa AMIs para o EKS no Bottlerocket, que já incluem todos os drivers e plug-ins da NVIDIA necessários em todos os nós com GPU, permitindo que as workloads em contêineres acessem as GPUs imediatamente, sem necessidade de configurações adicionais. Caso esteja usando outros tipos de nós no EKS, certifique-se de que todos os drivers e os plug-ins necessários estejam devidamente instalados.

Teste da infraestrutura de GPU

Teste as funcionalidades de GPU do cluster executando as etapas abaixo para garantir que os pods possam acessar os recursos da GPU da NVIDIA e sejam agendados corretamente em nós com GPU.

Implante um pod de teste NVIDIA SMI:

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: gpu-nvidia-smi-test
spec:
  restartPolicy: OnFailure
  tolerations:
  - key: "nvidia.com/gpu"
    operator: "Exists"
    effect: "NoSchedule"
  nodeSelector:
    role: gpu-worker  # Matches GPU NodePool's label
  containers:
  - name: cuda-container
    image: nvidia/cuda:12.9.1-base-ubuntu20.04
    command: ["nvidia-smi"]
    resources:
      requests:
        memory: "24Gi"
      limits:
        nvidia.com/gpu: 1
EOF
```

Analise os logs do pod para confirmar que os detalhes da GPU estão listados, de forma semelhante à saída abaixo (não necessariamente o mesmo modelo de GPU):

```
$ kubectl wait --for=jsonpath='{.status.phase}'=Succeeded pod/gpu-nvidia-smi-test
$ kubectl logs gpu-nvidia-smi-test
```

```
Wed Jul 30 15:39:58 2025
+-----------------------------------------------------------------------------------------
+
```

```
| NVIDIA-SMI 570.172.08             Driver Version: 570.172.08     CUDA Version: 12.9
   |
|-----------------------------------------+------------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.
 ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute
 M. |
|                                         |                        |               MIG
 M. |
|=========================================+========================
+======================|
|   0  NVIDIA A10G                    On  |   00000000:00:1E.0 Off |
 0 |
|  0%   30C    P8              9W /  300W |       0MiB /  23028MiB |      0%
 Default |
|                                         |                        |
 N/A |
+-----------------------------------------+------------------------
+----------------------+

+-----------------------------------------------------------------------------------------
+
| Processes:
   |
|  GPU   GI   CI              PID   Type   Process name                        GPU
 Memory |
|        ID   ID                                                               Usage
   |
|
=============================================================================================|
|  No running processes found
   |
+-----------------------------------------------------------------------------------------
+
```

Esta saída mostra que os pods conseguem acessar os recursos de GPU com êxito.

IMPORTANTE: este pod utiliza uma configuração nodeSelector que corresponde aos grupos de nós do Karpenter descritos no the section called “Criação de cluster”. Se você estiver usando grupos de nós diferentes, certifique-se de que o pod corresponda corretamente ao nodeSelector e às Tolerations.

Implantação do contêiner de inferência

A pilha de serviço determina tanto a performance quanto as funcionalidades de escalabilidade da infraestrutura de inferência. O vLLM surgiu como uma das principais soluções para implantações em ambientes de produção. A arquitetura do vLLM oferece agrupamento em lotes contínuo de solicitações para processamento dinâmico, otimizações de kernel para inferência mais ágil e gerenciamento eficiente da memória da GPU por meio do PagedAttention. Esses recursos, combinados com uma API REST pronta para uso em produção e suporte a formatos de modelos amplamente utilizados, tornam essa solução uma escolha ideal para implantações de inferência com alta performance.

Seleção da imagem do contêiner de deep learning da AWS

Os contêineres de deep learning da AWS (DLCs) fornecem ambientes otimizados previamente com atualizações de segurança, compatibilidade com a infraestrutura da AWS e configurações de drivers otimizadas. Com isso, a implantação se torna mais simples e a sobrecarga de manutenção é reduzida, garantindo que o ambiente esteja pronto para produção.

Nesta implantação, usaremos o AWS DLC para vLLM 0.9, que inclui bibliotecas da NVIDIA e configurações de performance da GPU otimizadas, ajustadas especificamente para a inferência de modelos tipo transformador em instâncias de GPU da AWS.

```
image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/vllm:0.9-gpu-py312-ec2
```

Aplicação dos manifestos do vLLM com o Kubernetes

Existem várias formas de implantar o vLLM no EKS. Este guia demonstra a implantação do vLLM usando uma implantação do Kubernetes, que é uma abordagem nativa do Kubernetes e uma forma simples de começar. Para obter opções avançadas de implantação, consulte a documentação do vLLM e os esquemas baseados em IA no EKS.

Defina os parâmetros de implantação por meio de manifestos do Kubernetes para controlar a alocação de recursos, o posicionamento dos nós, as verificações de integridade, a exposição do serviço, entre outros aspectos. Configure a implantação para executar um pod com GPU usando a imagem do contêiner de deep learning da AWS para vLLM. Defina parâmetros otimizados para a inferência de LLM e exponha o endpoint do vLLM compatível com OpenAPI por meio do serviço AWS Load Balancer:

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: vllm-inference-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: vllm-inference-app
  template:
    metadata:
      labels:
        app: vllm-inference-app
    spec:
      tolerations:
      - key: "nvidia.com/gpu"
        operator: "Exists"
        effect: "NoSchedule"
      nodeSelector:
        role: gpu-worker
      containers:
      - name: vllm-inference
        image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/vllm:0.9-gpu-py312-ec2
        ports:
        - containerPort: 8000
        env:
        - name: MODEL_PATH
          value: "/mnt/models/llama-3.1-8b-instruct"
        args:
        - "--model=/mnt/models/llama-3.1-8b-instruct"
        - "--host=0.0.0.0"
        - "--port=8000"
        - "--tensor-parallel-size=1"
        - "--gpu-memory-utilization=0.9"
        - "--max-model-len=8192"
        - "--max-num-seqs=1"
        readinessProbe:
          httpGet:
            path: /health
            port: 8000
          initialDelaySeconds: 30
          periodSeconds: 5
          timeoutSeconds: 10
        resources:
          limits:
```

```
            nvidia.com/gpu: 1
          requests:
            memory: "24Gi"
            cpu: "4"
            ephemeral-storage: "25Gi"  # Ensure enough node storage for vLLM container
 image
        volumeMounts:
        - name: models
          mountPath: /mnt/models
          readOnly: true
      volumes:
      - name: models
        persistentVolumeClaim:
          claimName: model-store
---
apiVersion: v1
kind: Service
metadata:
  name: vllm-inference-svc
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: nlb
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 8000
      protocol: TCP
  selector:
    app: vllm-inference-app
EOF
```

Verifique se o pod do vLLM está no estado `Ready 1/1`:

```
kubectl get pod -l app=vllm-inference-app -w
```

Saída esperada:

```
NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
vllm-inference-app-65df5fddc8-5kmjm   1/1     1            1           5m
```

Pode demorar alguns minutos para que a imagem do contêiner seja baixada e o vLLM carregue os arquivos do modelo na memória da GPU. Avance apenas quando o pod estiver nos estados Pronto e Disponível.

Exposição do serviço

Deixe o endpoint de inferência acessível localmente por meio do encaminhamento de portas do Kubernetes, que facilita o desenvolvimento e os testes. Mantenha este comando em execução em uma janela de terminal separada:

```
export POD_NAME=$(kubectl get pod -l app=vllm-inference-app -o
 jsonpath='{.items[0].metadata.name}')
kubectl port-forward pod/$POD_NAME 8000:8000
```

O AWS Load Balancer Controller cria automaticamente um Network Load Balancer (NLB) que expõe o endpoint do serviço do vLLM externamente. Para recuperar o endpoint do NLB, execute:

```
NLB=$(kubectl get service vllm-inference-svc -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

Ainda não instalou o AWS Load Balancer Controller? Siga as etapas de implantação apresentadas em the section called “ AWS Load Balancer Controller”.

Execute a inferência

Validação do pod de inferência

Valide a funcionalidade do contêiner de inferência localmente por meio da porta encaminhada. Envie uma solicitação de conexão e confirme que a resposta inclua o código HTTP 200:

```
$ curl -IX GET "http://localhost:8000/v1/models"
```

```
HTTP/1.1 200 OK
date: Mon, 13 Oct 2025 23:24:57 GMT
server: uvicorn
content-length: 516
content-type: application/json
```

Teste as funcionalidades de inferência e valide a conectividade externa enviando uma solicitação de conclusão para o LLM pelo endpoint do NLB:

```
curl -X POST "http://$NLB:80/v1/completions" \
  -H "Content-Type: application/json" \
  -d '{
    "model": "/mnt/models/llama-3.1-8b-instruct",
    "prompt": "Explain artificial intelligence:",
    "max_tokens": 512,
    "temperature": 0.7
  }'
```

Este endpoint segue o formato da API OpenAI, tornando-o compatível com aplicações existentes, ao mesmo tempo em que oferece parâmetros configuráveis de geração, como tamanho da resposta e temperatura, para controlar a diversidade da saída.

Execução do aplicativo de chatbot

Para fins de demonstração, este guia executa uma aplicação de chatbot de amostra usando o projeto nextjs-vllm-ui para mostrar as interações do usuário com o modelo.

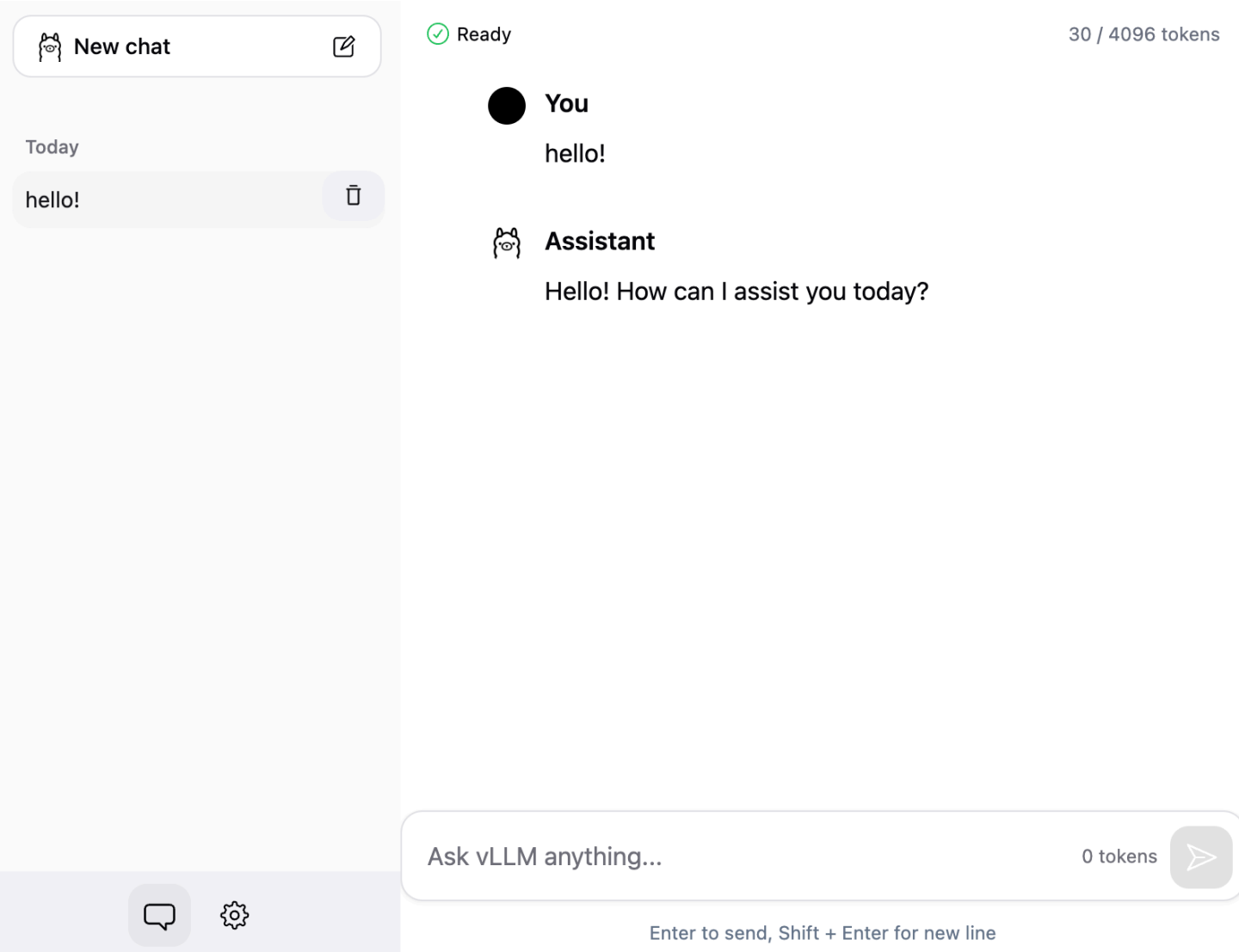
Execute a interface do chatbot como um contêiner do Docker, mapeando a porta 3000 para o localhost e conectando-se ao endpoint de NLB do vLLM:

```
docker run --rm \
  -p 3000:3000 \
  -e VLLM_URL="http://${NLB}:80" \
  --name nextjs-vllm-ui-demo \
  ghcr.io/yoziru/nextjs-vllm-ui:latest
```

Abra seu navegador e acesse: http://localhost:3000/

A interface de chat será exibida, permitindo que você interaja com o modelo Llama.

Interface da interface do usuário do chat



Otimização da performance de inferência

Mecanismos de inferência especializados, como o vLLM, oferecem recursos avançados que aumentam significativamente a performance da inferência, incluindo o agrupamento em lotes contínuo, o armazenamento KV eficiente e os mecanismos otimizados de atenção à memória. É possível ajustar os parâmetros de configuração do vLLM para melhorar a performance da inferência, atendendo aos requisitos específicos do seu caso de uso e aos padrões de workloads. A configuração adequada é essencial para atingir a saturação da GPU, garantindo que você extraia o máximo valor dos recursos de GPU, que são caros, ao mesmo tempo em que oferece alto throughput, baixa latência e operações econômicas. As otimizações apresentadas a seguir ajudarão você a maximizar a performance da implantação do vLLM no EKS.

Avaliação comparativa das configurações do vLLM

Para ajustar os parâmetros de configuração do vLLM ao seu caso de uso, compare diferentes configurações empregando uma ferramenta abrangente de avaliação comparativa de inferência, como o GuideLLM. Isso coletará métricas importantes, como o throughput de solicitações por segundo (RPS, na sigla em inglês), latência de ponta a ponta (E2E, na sigla em inglês), tempo até o primeiro token (TTFT, na sigla em inglês) e latência final (TPOT, na sigla em inglês) para comparar diferentes configurações.

Configuração de linha de base do vLLM

Esta é a configuração básica usada para executar o vLLM:

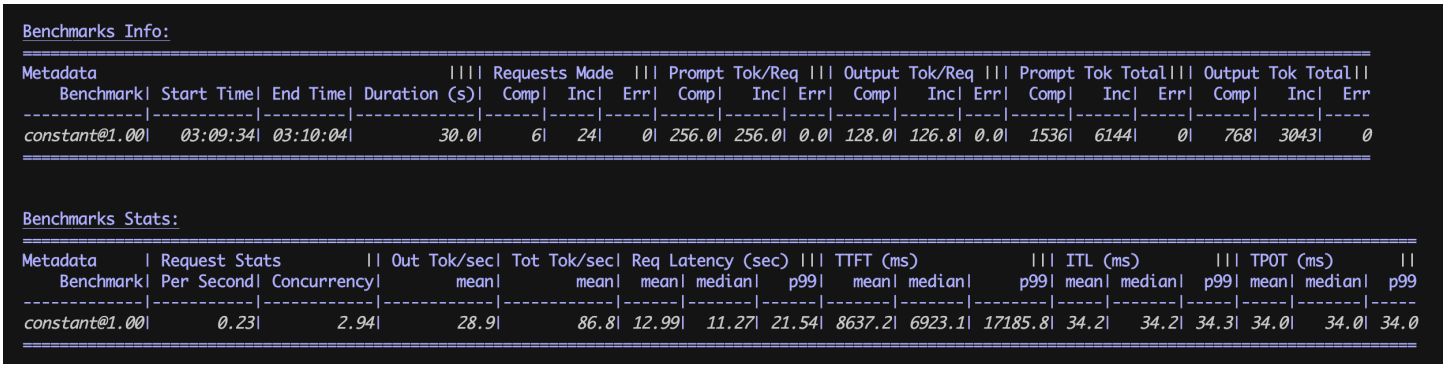
Parâmetro do vLLM	Descrição
tensor_parallel_size: 1	Distribuição do modelo em 1 GPU
gpu_memory_utilization: 0,90	Reserva 10% da memória da GPU para sobrecarga do sistema
max_sequence_length: 8.192	Comprimento máximo total da sequência (entrada + saída)
max_num_seqs: 1	Número máximo de solicitações simultâneas por GPU (agrupamento em lotes)

Execute o GuideLLM com essa configuração de linha de base para estabelecer uma linha de base para a performance. Neste teste, o GuideLLM está configurado para gerar 1 solicitação por segundo, com solicitações de 256 tokens e respostas de 128 tokens.

```
guidellm benchmark \
--target "http://${NLB}:80" \
--processor meta-llama/Llama-3.1-8B-Instruct \
--rate-type constant \
--rate 1 \
--max-seconds 30 \
--data "prompt_tokens=256,output_tokens=128"
```

Saída esperada:

Resultados da avaliação comparativa de referência



```
Benchmarks Info:
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
Metadata                                  |||| Requests Made  ||| Prompt Tok/Req ||| Output Tok/Req ||| Prompt Tok Total||| Output Tok Total||
    Benchmark| Start Time| End Time| Duration (s)|  Comp|  Inc|  Err|  Comp|   Inc| Err|  Comp|   Inc| Err|  Comp|   Inc|  Err|  Comp|   Inc|  Err
-------------|-----------|---------|-------------|------|-----|-----|------|------|----|------|------|----|------|------|-----|------|------|-----
constant@1.00|   03:09:34| 03:10:04|         30.0|     6|   24|    0| 256.0| 256.0| 0.0| 128.0| 126.8| 0.0|  1536|  6144|    0|   768|  3043|    0
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════


Benchmarks Stats:
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
Metadata     | Request Stats         || Out Tok/sec| Tot Tok/sec| Req Latency (sec) ||| TTFT (ms)           ||| ITL (ms)        ||| TPOT (ms)       ||
    Benchmark| Per Second| Concurrency|       mean|        mean|  mean|  median|   p99|   mean|  median|     p99| mean|  median|  p99| mean|  median|  p99
-------------|-----------|------------|-----------|------------|------|--------|------|-------|--------|--------|-----|--------|-----|-----|--------|-----
constant@1.00|       0.23|        2.94|       28.9|        86.8| 12.99|   11.27| 21.54| 8637.2| 6923.1| 17185.8| 34.2|    34.2| 34.3| 34.0|    34.0| 34.0
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
```

Configuração otimizada do vLLM

Ajuste os parâmetros do vLLM para melhor utilizar os recursos da GPU e a paralelização:

Parâmetro do vLLM	Descrição
tensor_parallel_size: 1	Mantenha em 1 GPU. A paralelização de tensores deve corresponder ao número de GPUs a serem usadas pelo vLLM.
gpu_memory_utilization: 0,92	Reduza a sobrecarga de memória da GPU, se possível, ao mesmo tempo em que garante que o vLLM continua a ser executado sem erros.
max_sequence_length: 4.096	Ajuste o comprimento máximo da sequência de acordo com os requisitos do seu caso de uso. Um comprimento máximo menor libera recursos que podem ser usados para aumentar a paralelização.
max_num_seqs: 8	Aumentar o número máximo de sequências aumenta o throughput, mas também a latência. Aumente este valor para maximizar o throughput, garantindo que a latência permaneça dentro dos requisitos do seu caso de uso.

Aplique essas alterações à implantação em execução usando o comando kubectl patch:

```
kubectl patch deployment vllm-inference-app --type='json' -p='[
  {"op": "replace", "path": "/spec/template/spec/containers/0/args/4", "value": "--gpu-
memory-utilization=0.92"},
  {"op": "replace", "path": "/spec/template/spec/containers/0/args/5", "value": "--max-
model-len=4096"},
```

```
  {"op": "replace", "path": "/spec/template/spec/containers/0/args/6", "value": "--max-
num-seqs=8"}
]'
```

Verifique se o pod do vLLM está no estado `Ready 1/1`:

```
kubectl get pod -l app=vllm-inference-app -w
```

Saída esperada:

```
NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
vllm-inference-app-65df5fddc8-5kmjm   1/1     1            1           5m
```

Em seguida, execute o GuideLLM novamente usando os mesmos valores da avaliação comparativa realizada anteriormente:

```
guidellm benchmark \
--target "http://${NLB}:80" \
--processor meta-llama/Llama-3.1-8B-Instruct \
--rate-type constant \
--rate 1 \
--max-seconds 30 \
--data "prompt_tokens=256,output_tokens=128"
```

Saída esperada:

Resultados da avaliação comparativa otimizados

```
Benchmarks Info:
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
Metadata                                  |||| Requests Made  ||| Prompt Tok/Req ||| Output Tok/Req||| Prompt Tok Total||| Output Tok Total  ||
    Benchmark| Start Time| End Time| Duration (s)|  Comp|  Inc|  Err|  Comp|   Inc| Err|  Comp|  Inc| Err|  Comp|   Inc|  Err|   Comp|   Inc|   Err
-------------|-----------|---------|-------------|------|-----|-----|------|------|----|------|-----|----|------|------|-----|-------|------|------
constant@1.00|   03:20:01| 03:20:31|         30.0|    25|    5|    0| 256.1| 256.0| 0.0| 128.0| 73.6| 0.0|  6403|  1280|    0|   3200|   368|     0
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════


Benchmarks Stats:
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
Metadata     | Request Stats         || Out Tok/sec| Tot Tok/sec| Req Latency (sec)  ||| TTFT (ms)         ||| ITL (ms)         ||| TPOT (ms)        ||
    Benchmark| Per Second| Concurrency|       mean|        mean|  mean|  median|   p99|  mean| median|   p99| mean| median|  p99| mean| median|  p99
-------------|-----------|------------|-----------|------------|------|--------|------|------|-------|------|-----|-------|-----|-----|-------|-----
constant@1.00|       0.86|        4.45|      109.6|       328.9|  5.19|    5.20|  5.23| 147.9|  148.4| 169.4| 39.7|   39.8| 39.8| 39.4|   39.5| 39.5
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
```

Resultados da avaliação comparativa

Os resultados da avaliação comparativa em uma tabela para as configurações de linha de base e otimizadas do vLLM são:

Valores médios	Configuração de linha de base	Configuração otimizada
RPS	0,23 solicitações por segundo	0,86 solicitações por segundo
E2E	12,99 s	5,19 s
TTFT	8.637,2 ms	147,9 ms
TPOT	34,0 ms	39,5 ms

As configurações otimizadas do vLLM melhoraram significativamente o throughput de inferência (RPS) e reduziram a latência (E2E e TTFT) com apenas um pequeno aumento de milissegundos na latência final (TPOT). Esses resultados demonstram como o vLLM melhora significativamente a performance da inferência, permitindo que cada contêiner processe mais solicitações em menos tempo, resultando em uma operação com melhor custo-benefício.

Configuração de cluster do Amazon EKS para workloads de IA/ML

Esta seção foi projetada para ajudar você a configurar clusters do Amazon EKS otimizados para workloads de IA/ML. Você encontrará orientações sobre como executar contêineres acelerados por GPU usando AMIs otimizadas para Linux e Windows, configurar clusters de treinamento com o Elastic Fabric Adapter (EFA) para redes de alta performance e criar clusters de inferência com instâncias do AWS Inferentia, incluindo pré-requisitos, procedimentos passo a passo e considerações de implantação.

Tópicos

- Uso de AMIs aceleradas e otimizadas para o EKS em instâncias de GPU
- Instalação do plug-in de dispositivo do Kubernetes para GPUs
- Executar contêineres acelerados por GPU (Windows no EC2 G-Series)
- Execute o treinamento de machine learning no Amazon EKS com o Elastic Fabric Adapter
- Usar instâncias do AWS Inferentia com o Amazon EKS para machine learning

Uso de AMIs aceleradas e otimizadas para o EKS em instâncias de GPU

O Amazon EKS oferece suporte a AMIs do Amazon Linux e do Bottlerocket otimizadas para o EKS em instâncias de GPU. As AMIs aceleradas e otimizadas para o EKS simplificam a execução

de workloads de IA e de ML em clusters do EKS ao fornecer imagens de sistemas operacionais previamente integradas e validadas para a pilha acelerada do Kubernetes. Além dos componentes principais do Kubernetes incluídos nas AMIs otimizadas para o EKS padrão, as AMIs aceleradas e otimizadas para o EKS incluem os módulos de kernel e os drivers necessários para executar as instâncias de GPU do EC2 para a NVIDIA nas famílias `G` e `P` e as instâncias de GPU do EC2 para a AWS no Inferentia e no Trainium em clusters do EKS.

A tabela abaixo apresenta os tipos de instâncias de GPU compatíveis com cada uma das variantes de AMIs aceleradas e otimizadas para o EKS. Consulte os lançamentos do AL2023 e os lançamentos do Bottlerocket otimizados para o EKS no GitHub para obter as atualizações mais recentes das variantes de AMI.

Variante de AMI do EKS	Tipos de instâncias do EC2
AL2023 x86_64 NVIDIA	p6-b300, p6-b200, p5, p5e, p5en, p4d, p4de, p3, p3dn, gr6, g6, g6e, g6f, gr6f, g5 e g4dn
AL2023 ARM NVIDIA	p6e-gb200 e g5g
AL2023 x86_64 Neuron	inf1, inf2, trn1 e trn2
Bottlerocket x86_64 aws-k8s-nvidia	p6-b300, p6-b200, p5, p5e, p5en, p4d, p4de, p3, p3dn, gr6, g6, g6e, g6f, gr6f, g5 e g4dn
Bottlerocket aarch64/arm64 aws-k8s-nvidia	g5g
Bottlerocket x86_64 aws-k8s	inf1, inf2, trn1 e trn2

AMIs da NVIDIA otimizadas para o EKS

Ao usar as AMIs da NVIDIA otimizadas para o EKS, você concorda com o Contrato de Licença de Usuário Final (EULA, na sigla em inglês) da NVIDIA para uso da nuvem.

Para encontrar as AMIs da NVIDIA otimizadas para o EKS mais recentes, consulte the section called “Obter os IDs mais recentes” e the section called “Obter os IDs mais recentes”.

Ao usar o Amazon Elastic Fabric Adaptor (EFA) com as AMIs da NVIDIA otimizadas para o EKS com o AL2023 ou o Bottlerocket, você deve instalar o plug-in de dispositivo de EFA separadamente. Para

obter mais informações, consulte the section called “Configuração de treinamento de clusters com o EFA”.

AMIs da NVIDIA para o EKS no AL2023

Ao usar o NVIDIA GPU Operator com as AMIs do AL2023 otimizadas para EKS com a NVIDIA, você deve desabilitar a instalação do driver e do toolkit pelo operador, pois estes já estão incluídos na AMI para o EKS. As AMIs do AL2023 otimizadas para o EKS com a NVIDIA não incluem o plug-in de dispositivo da NVIDIA para Kubernetes ou o driver de DRA da NVIDIA. Portanto, estes devem ser instalados separadamente. Para obter mais informações, consulte the section called “Instalação do plug-in de dispositivos da NVIDIA para Kubernetes”.

Além dos componentes padrão na AMI do EKS, as AMIs da NVIDIA otimizadas para o EKS no AL2023 incluem os componentes apresentados a seguir.

- Driver NVIDIA
- Driver de modo de usuário do NVIDIA CUDA
- NVIDIA Container Toolkit
- NVIDIA Fabric Manager
- Daemon “nvidia-persistenced”
- Driver de IMEX da NVIDIA
- Gerenciador de sub-rede NVIDIA NVLink
- EFA mínimo (contendo módulo de kernel e rdma-core)

Para obter detalhes sobre o driver de modo de usuário do NVIDIA CUDA e o runtime ou as bibliotecas do CUDA usados em contêineres de aplicações, consulte a documentação da NVIDIA. A versão do CUDA que o comando `nvidia-smi` mostra refere-se ao driver de modo de usuário do NVIDIA CUDA instalado no host. Essa versão precisa ser compatível com o runtime e com as bibliotecas do CUDA que as aplicações usam nos contêineres.

As AMIs da NVIDIA otimizadas para o EKS no AL2023 oferecem suporte ao kernel 6.12 para a versão 1.33 e para versões posteriores do Kubernetes, bem como à versão 580 do driver da NVIDIA para todas as versões do Kubernetes. O uso do CUDA 13 ou de suas versões posteriores requer a instalação do driver 580 da NVIDIA.

Consulte os lançamentos do AL2023 otimizados para o EKS no GitHub para obter detalhes das versões dos componentes incluídas nas AMIs. Consulte o script de instalação e o script de

carregamento do kernel das AMIs da NVIDIA para o EKS no AL2023 para obter detalhes sobre como as AMIs do EKS configuram as dependências da NVIDIA. É possível encontrar a lista de pacotes instalados e suas versões em uma instância do EC2 em execução com o comando `dnf list installed`.

Ao criar AMIs personalizadas usando AMIs otimizadas para o EKS como base, não é recomendado nem suportado executar uma atualização do sistema operacional (por exemplo, `dnf upgrade`) ou atualizar qualquer um dos pacotes do Kubernetes ou de GPU incluídos nas AMIs otimizadas para o EKS, pois isso pode comprometer a compatibilidade entre os componentes. Se você optar por atualizar o sistema operacional ou os pacotes incluídos nas AMIs otimizadas para o EKS, é recomendável realizar testes completos em um ambiente de desenvolvimento ou de preparação antes da implantação em um ambiente de produção.

Ao criar AMIs personalizadas para instâncias de GPU, recomenda-se desenvolver AMIs personalizadas distintas para cada geração e família de instâncias que você pretende executar. As AMIs aceleradas e otimizadas para o EKS realizam a instalação seletiva de drivers e pacotes durante o runtime, de acordo com a geração e a família do tipo de instância subjacente. Para obter mais informações, consulte os scripts de instalação e de runtime das AMIs do EKS.

AMIs da NVIDIA para o EKS no Bottlerocket

Ao usar o NVIDIA GPU Operator com as AMIs do Bottlerocket otimizadas para EKS com a NVIDIA, você deve desabilitar a instalação do driver, do toolkit e do plug-in de dispositivo pelo operador, pois estes já estão incluídos na AMI para o EKS.

Além dos componentes padrão na AMI do EKS, as AMIs da NVIDIA otimizadas para o EKS no Bottlerocket incluem os componentes apresentados a seguir. As dependências mínimas para EFA (contendo módulo do kernel e rdma-core) estão instaladas em todas as variantes do Bottlerocket.

- Plug-in de dispositivos da NVIDIA para Kubernetes
- Driver NVIDIA
- Driver de modo de usuário do NVIDIA CUDA
- NVIDIA Container Toolkit
- NVIDIA Fabric Manager
- Daemon “nvidia-persistenced”
- Driver de IMEX da NVIDIA
- Gerenciador de sub-rede NVIDIA NVLink

- Gerenciador de MIG da NVIDIA

Para obter detalhes sobre o driver de modo de usuário do NVIDIA CUDA e o runtime ou as bibliotecas do CUDA usados em contêineres de aplicações, consulte a documentação da NVIDIA. A versão do CUDA que o comando `nvidia-smi` mostra refere-se ao driver de modo de usuário do NVIDIA CUDA instalado no host. Essa versão precisa ser compatível com o runtime e com as bibliotecas do CUDA que as aplicações usam nos contêineres.

Consulte as informações da versão do Bottlerocket na documentação do Bottlerocket para obter detalhes sobre os pacotes instalados e suas versões. As AMIs da NVIDIA otimizadas para o EKS no Bottlerocket oferecem suporte ao kernel 6.12 para a versão 1.33 e para versões posteriores do Kubernetes, bem como à versão 580 do driver da NVIDIA na versão 1.34 e em versões posteriores do Kubernetes. O uso do CUDA 13 ou de suas versões posteriores requer a instalação do driver 580 da NVIDIA.

AMIs do Neuron otimizadas para o EKS

Para obter detalhes sobre como executar workloads de treinamento e de inferência usando o Neuron com o Amazon EKS, consulte as seguintes referências:

- Containers - Kubernetes - Getting Started na documentação do AWS Neuron
- Exemplo Training na página AWS Neuron EKS Samples no GitHub
- Implantação de workloads de inferência de ML com o Inferentia no Amazon EKS

Para encontrar as AMIs do Neuron otimizadas para o EKS mais recentes, consulte the section called “Obter os IDs mais recentes” e the section called “Obter os IDs mais recentes”.

Ao usar o Amazon Elastic Fabric Adaptor (EFA) com as AMIs do Neuron otimizadas para o EKS com o AL2023 ou o Bottlerocket, você deve instalar o plug-in de dispositivo de EFA separadamente. Para obter mais informações, consulte the section called “Configuração de treinamento de clusters com o EFA”.

AMIs do Neuron para o EKS no AL2023

As AMIs do Neuron otimizadas para o EKS no AL2023 não incluem o plug-in de dispositivo do Neuron para o Kubernetes nem a extensão de agendador do Neuron para Kubernetes, sendo necessário instalá-los separadamente. Para obter mais informações, consulte the section called “Instalação do plug-in de dispositivos do Neuron para Kubernetes”.

Além dos componentes padrão na AMI do EKS, as AMIs do Neuron otimizadas para o EKS no AL2023 incluem os componentes apresentados a seguir.

- Driver do Neuron (aws-neuronx-dkms)
- Ferramentas do Neuron (aws-neuronx-tools)
- EFA mínimo (contendo módulo de kernel e rdma-core)

Consulte o script de instalação da AMI do Neuron para o EKS no AL2023 para obter detalhes sobre como as AMIs do EKS configuram as dependências do Neuron. Consulte os lançamentos do AL2023 otimizados para o EKS no GitHub para verificar as versões dos componentes incluídas nas AMIs. É possível encontrar a lista de pacotes instalados e suas versões em uma instância do EC2 em execução com o comando `dnf list installed`.

AMIs do Neuron para o EKS no Bottlerocket

As variantes padrão do Bottlerocket (aws-k8s) incluem as dependências do Neuron, que são automaticamente detectadas e carregadas ao serem executadas em instâncias do EC2 no AWS Inferentia ou Trainium.

As AMIs otimizadas para o EKS no Bottlerocket não incluem o plug-in de dispositivo do Neuron para o Kubernetes nem a extensão de agendador do Neuron para Kubernetes, sendo necessário instalá-los separadamente. Para obter mais informações, consulte the section called “Instalação do plug-in de dispositivos do Neuron para Kubernetes”.

Além dos componentes padrão na AMI do EKS, as AMIs do Neuron otimizadas para o EKS no Bottlerocket incluem os componentes apresentados a seguir.

- Driver do Neuron (aws-neuronx-dkms)
- EFA mínimo (contendo módulo de kernel e rdma-core)

Ao usar as AMIs otimizadas para o EKS no Bottlerocket com instâncias do Neuron, as configurações a seguir devem ser realizadas nos dados do usuário do Bottlerocket. Essa configuração permite que o contêiner assuma a propriedade do dispositivo do Neuron montado com base nos valores `runAsUser` e `runAsGroup` fornecidos na especificação da workload. Para obter mais informações sobre o suporte ao Neuron no Bottlerocket, consulte o arquivo README de Quickstart on EKS no GitHub.

```
[settings]
```

```
[settings.kubernetes]
device-ownership-from-security-context = true
```

Consulte o Bottlerocket kernel kit changelog para obter informações sobre a versão do driver do Neuron incluída nas AMIs otimizadas para o EKS no Bottlerocket.

Instalação do plug-in de dispositivo do Kubernetes para GPUs

Os plug-ins de dispositivos do Kubernetes têm sido o principal mecanismo para divulgar infraestruturas especializadas, como GPUs, interfaces de rede e adaptadores de rede, como recursos consumíveis para workloads do Kubernetes. Embora a alocação dinâmica de recursos (DRA, na sigla em inglês) seja apresentada como o futuro do gerenciamento de dispositivos no Kubernetes, a maioria dos fornecedores de infraestrutura especializada ainda está em estágios iniciais de suporte aos drivers da DRA. Atualmente, os plug-ins de dispositivos do Kubernetes continuam sendo uma abordagem amplamente disponível para o uso de GPUs em clusters do Kubernetes.

Considerações

- Ao usar as AMIs do AL2023 otimizadas para o EKS com GPUs NVIDIA, é necessário instalar o plug-in de dispositivos da NVIDIA para Kubernetes. É possível instalar e gerenciar o plug-in de dispositivos da NVIDIA para Kubernetes com o Helm, qualquer ferramenta de Kubernetes de sua escolha ou o operador de GPU da NVIDIA.
- As AMIs do Bottlerocket otimizadas para o EKS já incluem nativamente o plug-in de dispositivos da NVIDIA para Kubernetes; portanto, nenhuma instalação adicional é necessária ao usar essas AMIs do Bottlerocket otimizadas para o EKS com GPUs NVIDIA. Isso se aplica também ao uso de instâncias de GPU no modo automático do EKS.
- Ao usar as AMIs do AL2023 ou do Bottlerocket otimizadas para o EKS com GPUs do AWS Inferentia ou Trainium, é necessário instalar o plug-in de dispositivos do Neuron para Kubernetes e, opcionalmente, instalar a extensão de agendador do Neuron para Kubernetes. Para obter mais informações, consulte a documentação do Neuron sobre execução no EKS.

Instalação do plug-in de dispositivos da NVIDIA para Kubernetes

O procedimento, apresentado a seguir, descreve como instalar o plug-in de dispositivos da NVIDIA para Kubernetes e executar um teste de exemplo em instâncias de GPU da NVIDIA.

Pré-requisitos

- Cluster do EKS criado
- Nós com GPUs da NVIDIA em execução no cluster usando a AMI do AL2023 otimizada para o EKS com a NVIDIA
- O Helm foi instalado em seu ambiente de linha de comando. Consulte as Instruções de configuração do Helm.

Procedimento

1. Adicione o repositório `nvdp` do chart do Helm.

```
helm repo add nvdp https://nvidia.github.io/k8s-device-plugin
```

2. Atualize o repositório local do Helm para confirmar que você tem os gráficos mais recentes.

```
helm repo update
```

3. Obtenha a versão mais recente do plug-in de dispositivos da NVIDIA para Kubernetes.

```
helm search repo nvdp --devel
```

```
NAME                        CHART VERSION APP VERSION DESCRIPTION
nvdp/gpu-feature-discovery  0.17.4        0.17.4      ...
nvdp/nvidia-device-plugin   0.17.4        0.17.4      ...
```

4. Instale o plug-in de dispositivos da NVIDIA para Kubernetes no cluster, substituindo a versão `0.17.4` pela versão mais recente obtida no comando anterior.

```
helm install nvdp nvdp/nvidia-device-plugin \
  --namespace nvidia \
  --create-namespace \
  --version 0.17.4 \
  --set gfd.enabled=true
```

5. Verifique se o plug-in de dispositivos da NVIDIA para Kubernetes está em execução no cluster. A saída, apresentada a seguir, exibe o resultado para um cluster com dois nós.

```
kubectl get ds -n nvidia nvdp-nvidia-device-plugin
```

```
NAME                        DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
nvdp-nvidia-device-plugin   2         2         2       2            2
 <none>          11m
```

6. Verifique se seus nós têm GPUs alocáveis com o comando apresentado a seguir.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

```
NAME                                           GPU
ip-192-168-11-225.us-west-2.compute.internal   1
ip-192-168-24-96.us-west-2.compute.internal    1
```

7. Crie um arquivo denominado `nvidia-smi.yaml` com os conteúdos a seguir. Este manifesto inicia uma imagem de contêiner do AL2023 mínima que executa o `nvidia-smi` em um nó.

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
    - name: gpu-demo
      image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
      command: ['/bin/sh', '-c']
      args: ['nvidia-smi && tail -f /dev/null']
      resources:
        limits:
          nvidia.com/gpu: 1
  tolerations:
    - key: 'nvidia.com/gpu'
      operator: 'Equal'
      value: 'true'
      effect: 'NoSchedule'
```

8. Aplique o manifesto com o comando a seguir.

```
kubectl apply -f nvidia-smi.yaml
```

9. Após o pod concluir a execução, visualize os logs com o comando a seguir.

```
kubectl logs nvidia-smi
```

Veja abaixo um exemplo de saída.

```
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI XXX.XXX.XX            Driver Version: XXX.XXX.XX     CUDA Version: XX.X
     |
|-----------------------------------------+------------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.
 ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util
 Compute M. |
|                                         |                        |
 MIG M. |
|=========================================+========================
+======================|
|   0  NVIDIA L4                      On  |   00000000:31:00.0 Off |
   0 |
| N/A   27C    P8             11W /   72W |       0MiB /  23034MiB |      0%
 Default |
|                                         |                        |
 N/A |
+-----------------------------------------+------------------------
+----------------------+

+-----------------------------------------------------------------------------------------
+
| Processes:
     |
|  GPU   GI   CI              PID   Type   Process name                        GPU
 Memory |
|        ID   ID                                                               Usage
     |
|
=========================================================================================|
|  No running processes found
     |
```

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

Instalação do plug-in de dispositivos do Neuron para Kubernetes

O procedimento, apresentado a seguir, descreve como instalar o plug-in de dispositivos do Neuron para Kubernetes e executar um teste de exemplo em uma instância do Inferentia.

Pré-requisitos

- Cluster do EKS criado
- Nós com GPUs do Neuron em execução no cluster usando a AMI do AL2023 otimizada para o EKS com o Neuron ou a AMI do Bottlerocket
- O Helm foi instalado em seu ambiente de linha de comando. Consulte as Instruções de configuração do Helm.

Procedimento

1. Instale o plug-in de dispositivos do Neuron para Kubernetes no cluster.

```
helm upgrade --install neuron-helm-chart oci://public.ecr.aws/neuron/neuron-helm-
chart \
    --set "npd.enabled=false"
```

2. Verifique se o plug-in de dispositivos do Neuron para Kubernetes está em execução no cluster. A saída, apresentada a seguir, exibe o resultado para um cluster com um único nó do Neuron.

```
kubectl get ds -n kube-system neuron-device-plugin
```

```
NAME                   DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
neuron-device-plugin   1         1         1       1            1           <none>
       72s
```

3. Verifique se seus nós têm NueronCores alocáveis com o comando apresentado a seguir.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,NeuronCore:.status.allocatable.aws\.amazon\.com/
neuroncore"
```

```
NAME                                           NeuronCore
ip-192-168-47-173.us-west-2.compute.internal   2
```

4. Verifique se seus nós têm NueronDevices alocáveis com o comando apresentado a seguir.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,NeuronDevice:.status.allocatable.aws\.amazon\.com/neuron"
```

```
NAME                                           NeuronDevice
ip-192-168-47-173.us-west-2.compute.internal   1
```

5. Crie um arquivo denominado `neuron-ls.yaml` com os conteúdos a seguir. Este manifesto executa um contêiner do Neuron Monitor com a ferramenta `neuron-ls` instalada.

```
apiVersion: v1
kind: Pod
metadata:
  name: neuron-ls
spec:
  restartPolicy: Never
  containers:
  - name: neuron-container
    image: public.ecr.aws/g4h4h0b5/neuron-monitor:1.0.0
    command: ["/bin/sh"]
    args: ["-c", "neuron-ls"]
    resources:
      limits:
        aws.amazon.com/neuron: 1
  tolerations:
  - key: "aws.amazon.com/neuron"
    operator: "Exists"
    effect: "NoSchedule"
```

6. Aplique o manifesto com o comando a seguir.

```
kubectl apply -f neuron-ls.yaml
```

7. Após o pod concluir a execução, visualize os logs com o comando a seguir.

```
kubectl logs neuron-ls
```

Um exemplo de saída está abaixo.

```
instance-type: inf2.xlarge
instance-id: ...
+--------+--------+--------+---------+
| NEURON | NEURON | NEURON |   PCI   |
| DEVICE | CORES  | MEMORY |   BDF   |
+--------+--------+--------+---------+
| 0      | 2      | 32 GB  | 00:1f.0 |
+--------+--------+--------+---------+
```

Executar contêineres acelerados por GPU (Windows no EC2 G-Series)

> **Important**
>
> O Kubernetes Device Plugin for DirectX da TensorWorks é uma ferramenta de terceiros que não é endossada, suportada ou mantida pela AWS. A AWS não assume qualquer responsabilidade pela segurança, confiabilidade ou performance desse plug-in.

Saiba como executar workloads de contêineres do Windows aceleradas por GPU no Amazon EKS (Elastic Kubernetes Service) usando GPUs NVIDIA com o Kubernetes Device Plugin for DirectX da TensorWorks. Para obter mais informações, consulte Kubernetes Device Plugin for DirectX.

Há duas abordagens principais para configurar a aceleração por GPU para os contêineres do Windows:

- Opção 1: crie uma AMI personalizada do EKS otimizada para o Windows com os drivers de GPU necessários pré-instalados.
  - Use essa abordagem quando precisar de um ambiente consistente e pré-configurado, pronto para executar contêineres do Windows acelerados por GPU, e puder investir esforços adicionais para criar e manter a AMI personalizada.
- Opção 2: instale os drivers de GPU necessários nos nós de processamento do EKS depois de executar a instância.

- Use essa abordagem quando quiser um processo de configuração mais simples e não se importar em instalar os drivers da GPU em cada novo nó de processamento. Mais adequado para um ambiente de desenvolvimento quando você está avaliando ou criando protótipos de workloads aceleradas por GPU.

Ambas as abordagens podem ser aproveitadas usando as etapas detalhadas neste guia.

Considerações

Este guia fornece as etapas para instalar e configurar a aceleração de GPU para os contêineres do Windows usando GPUs NVIDIA, drivers NVIDIA GRID e o Kubernetes Device Plugin for DirectX da TensorWorks. As etapas foram testadas e verificadas para fornecer aceleração por GPU para as workloads de contêineres do Windows no Amazon EKS. Consulte the section called “Limitações conhecidas” para obter mais informações sobre drivers e plug-ins de dispositivos compatíveis. Antes de prosseguir, observe o seguinte:

- Somente os tipos de instância G-family com drivers NVIDIA GRID foram testados e verificados para funcionar com este guia. Embora outras combinações de tipos de instância e drivers também possam ser capazes de executar contêineres do Windows acelerados por GPU, elas podem exigir etapas adicionais de configuração não abordadas neste guia.
- Somente workloads baseadas no DirectX foram testadas e verificadas para funcionar com este guia. Embora outras APIs de GPU, como OpenGL, Vulkan e OpenCL, provavelmente sejam compatíveis para executar contêineres do Windows acelerados por GPU, elas podem exigir etapas adicionais de configuração não abordadas neste guia.
- Há algumas limitações conhecidas das quais você deve estar ciente antes de executar contêineres do Windows acelerados por GPU. Consulte a seção the section called “Limitações conhecidas” para obter mais informações.

Pré-requisitos

Para habilitar a aceleração de GPU para os contêineres do Windows no Amazon EKS, você precisará preparar os seguintes requisitos antes de continuar:

- Execute um cluster de Amazon EKS com o Kubernetes v1.27 ou mais recente.
- Provisione nós do Windows com o Windows Server 2022 ou mais recente.
- Provisione nós do Windows nos tipos de instância G-family, como G4 ou G5.

- Provisione nós do Windows com um runtime de contêiner com o containerd `1.7.x` ou `2.x.x`. (Consulte the section called “Obter informações de versão” para verificar a versão do containerd na AMI otimizada para Amazon EKS.)

Instalar o driver da GPU em cada nó do Windows

Para instalar os drivers NVIDIA GRID nos nós de processamento do EKS, siga as etapas descritas em Drivers NVIDIA para a instância do Amazon EC2. Navegue até Opções de instalação - Opção 3: drivers GRID e siga as etapas de instalação.

Instalar o Server Core do Windows

Para o Server Core do Windows, que não tem uma experiência de desktop, instale os drivers NVIDIA GRID de forma silenciosa usando os seguintes comandos:

```
$nvidiaInstallerFilePath = nvidia-driver-installer.exe # Replace with path to installer
$installerArguments = "-s -clean -noreboot -noeula"
Start-Process -FilePath $nvidiaInstallerFilePath -ArgumentList $installerArguments -
Wait -NoNewWindow -PassThru
```

Verificar a instalação

Execute o seguinte comando do PowerShell para mostrar as informações de diagnóstico sobre as GPUs na instância:

```
nvidia-smi
```

Esse comando exibe a versão do driver NVIDIA, bem como informações sobre o hardware da GPU. Certifique-se de que a saída desse comando corresponda à versão do driver NVIDIA GRID que você esperava que fosse instalada.

Implantar o plug-in do dispositivo de GPU em cada nó

Para permitir a descoberta e a exposição dos recursos da GPU aos contêineres nos nós do Windows, você precisará de um plug-in de dispositivo. Implemente o DirectX Device Plugin da Tensorworks em cada nó de processamento, executando-o como um DaemonSet no cluster de EKS. Siga o guia de instalação especificado no README.md, que incluirá as etapas a seguir. É recomendado:

- Implantar o plug-in do dispositivo no namespace `kube-system`.

- Definir limites de recursos apropriados para o DaemonSet para garantir que ele não consuma recursos excessivos nos nós.

> **Note**
>
> O DaemonSet do plug-in do dispositivo será executado em cada nó como um contêiner de processo host com privilégios elevados. É recomendável implementar controles de RBAC para restringir o acesso a esse DaemonSet para que somente usuários autorizados possam executar comandos privilegiados.

Ao executar contêineres acelerados por GPU, o plug-in do dispositivo é compatível com dois modos:

- Modo de locação única: este modo dedica todos os recursos da GPU a um único contêiner na instância. Instale os plug-ins do dispositivo compatíveis com locação única usando o comando a seguir. Consulte README.md para obter mais informações.

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/default-daemonsets.yml"
```

- Modo de multilocação: este modo permite compartilhar recursos de GPU entre vários contêineres na instância. Instale os plug-ins do dispositivo compatíveis com multilocação usando o comando a seguir. Consulte README.md para obter mais informações.

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-inline.yml"
```

  Como alternativa, use um ConfigMap para especificar a multilocação.

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-configmap.yml"
```

Verificar a implantação do plug-in do dispositivo

Depois de ter implantado o plug-in do dispositivo, substitua `<namespace>` e execute o comando a seguir para verificar se o plug-in DirectX Device está funcionando corretamente em todos os nós do Windows.

```
kubectl get ds device-plugin-wddm -n <namespace>
```

Verificar se os contêineres estão prontos para implantação

Depois que o DaemonSet do plug-in do dispositivo estiver em execução nos nós de processamento do Windows com base em GPU, use o comando a seguir para verificar se cada nó tem GPUs alocáveis. O número correspondente deve corresponder ao número de dispositivos DirectX em cada nó.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,DirectX:.status.allocatable.directx\.microsoft\.com/
display"
```

Executar contêineres do Windows com aceleração por GPU

Antes de executar os pods, especifique o nome do recurso `directx.microsoft.com/display` em `.spec.containers[].resources`. Isso indicará que os contêineres exigem recursos habilitados para GPU, e o `kube-scheduler` tentará colocar os pods no nó do Windows pré-configurado com os recursos de GPU disponíveis.

Como exemplo, veja o comando de amostra abaixo que inicia um `Job` para executar uma simulação de Monte Carlo para estimar o valor de pi. Este exemplo é do repositório do GitHub do Kubernetes Device Plugins for DirectX, que tem vários exemplos para escolher que você pode executar para testar as capacidades de GPU do nó do Windows.

```
cat <<EOF | kubectl apply -f -
apiVersion: batch/v1
kind: Job
metadata:
  name: example-cuda-montecarlo-wddm
spec:
  template:
    spec:
      containers:
      - name: example-cuda-montecarlo-wddm
        image: "index.docker.io/tensorworks/example-cuda-montecarlo:0.0.1"
        resources:
          limits:
            directx.microsoft.com/display: 1
      nodeSelector:
```

```
        "kubernetes.io/os": windows
      restartPolicy: Never
  backoffLimit: 0
EOF
```

Limitações conhecidas

Todas as GPUs são funcionais

Todas as GPUs na instância poderão ser usadas por cada contêiner em execução no host, mesmo quando você solicitar um número específico de GPUs para um determinado contêiner. Além disso, o comportamento padrão é que todos os contêineres em execução no host usarão a GPU com índice 0, mesmo que haja várias GPUs disponíveis no nó. Portanto, para que as tarefas com várias GPUs funcionem corretamente, você deve designar explicitamente o dispositivo de GPU específico a ser utilizado no código da aplicação.

A implementação exata para alocar um dispositivo para uso na aplicação dependerá da linguagem de programação ou do framework que você está usando. Por exemplo, se você estiver usando a programação CUDA, para selecionar uma GPU específica, você poderá especificar explicitamente o dispositivo a ser usado no código da aplicação usando a função cudaSetDevice().

A necessidade de especificar explicitamente o dispositivo deve-se a um problema conhecido que afeta os contêineres do Windows. Você pode acompanhar o progresso na resolução desse problema em microsoft/Windows-Containers issue #333. A tabela a seguir mostra uma representação visual e um exemplo prático desse comportamento de alocação de GPUs.

Considere um cenário em que há um único nó do Windows do tipo de instância `g4dn.12xlarge` do EC2, que vem com quatro GPUs. Considere um cenário em que três pods sejam executados nessa instância. A tabela mostra que, independentemente do número de GPUs solicitadas por cada contêiner, todos os três pods têm acesso às quatro GPUs na instância e, por padrão, utilizarão a GPU com índice de dispositivo 0.

Pod	GPUs solicitadas	Acesso real à GPU	Uso padrão da GPU	Índices de GPU disponíveis	Total de instâncias de GPUs
Pod 1	1 GPU	Todas as 4 GPUs	GPU com índice 0	0, 1, 2, 3	4

Pod	GPUs solicitadas	Acesso real à GPU	Uso padrão da GPU	Índices de GPU disponíveis	Total de instâncias de GPUs
Pod 2	2 GPUs	Todas as 4 GPUs	GPU com índice 0	0, 1, 2, 3	4
Pod 3	1 GPU	Todas as 4 GPUs	GPU com índice 0	0, 1, 2, 3	4

Compatibilidade com o plug-in de dispositivo do Kubernetes

A implementação oficial do plugin de dispositivo do Kubernetes da NVIDIA não é compatível com o Windows. Você pode acompanhar o progresso na adição do suporte oficial ao Windows em NVIDIA/k8s-device-plugin issue #419.

Limitações da instância de computação da GPU

Dependendo da configuração da conta da AWS, você pode ter limites de serviço no número e nos tipos de instâncias de computação de GPU do Amazon EC2 que você pode executar. Se precisar de capacidade adicional, você pode solicitar um aumento de cota.

É necessário criar uma AMI do Windows otimizada para GPU

Não há nenhuma AMI do EKS para Windows otimizada para GPU ou componente gerenciado do EC2 Image Builder fornecido pelo Amazon EKS. Você precisará seguir as etapas deste guia para criar uma AMI personalizada do EKS otimizada para Windows com os drivers de GPU necessários pré-instalados, ou instalar os drivers de GPU necessários nos nós de processamento do EKS depois de executar as instâncias.

Inferentia e Trainium não compatíveis

As workloads baseadas no AWS Inferentia e no AWS Trainium não são compatíveis com o Windows.

Execute o treinamento de machine learning no Amazon EKS com o Elastic Fabric Adapter

Este tópico descreve como integrar o Elastic Fabric Adapter (EFA) com pods implantados no cluster do Amazon EKS. O Elastic Fabric Adapter (EFA) é uma interface de rede para instâncias do Amazon EC2 que permite que você execute aplicações que exigem altos níveis de comunicações entre nós

em escala na AWS. Sua interface de hardware de bypass do sistema operacional personalizada melhora a performance das comunicações entre instâncias, o que é essencial para escalar essas aplicações. Com a EFA, as aplicações de Computação de Alta Performance (HPC) que usam a Interface de Passagem de Mensagens (MPI) e as aplicações de machine learning (ML) que usam a NVIDIA Collective Communications Library (NCCL) podem ser escaladas para milhares de CPUs ou GPUs. Como resultado, você obtém a performance da aplicação de clusters de HPC on-premises, com a elasticidade e a flexibilidade sob demanda da nuvem AWS. A integração da EFA com aplicações executadas em clusters do Amazon EKS pode reduzir o tempo para concluir workloads de treinamento distribuídas em grande escala sem precisar adicionar instâncias adicionais ao cluster. Para obter mais informações sobre o EFA, consulte Elastic Fabric Adapter.

Tipos de instância com EFA

O plug-in de dispositivo EFA Kubernetes da AWS é compatível com todos os tipos de instância do Amazon EC2 que têm EFA. Para ver uma lista de tipos de instância que oferecem suporte a EFAs, consulte Tipos de instância compatíveis no Guia do usuário do Amazon EC2. No entanto, para executar aplicações de ML rapidamente, recomendamos que uma instância tenha chips de aceleração de hardware, como GPUs nVidia, chips do AWS Inferentia ou chips do AWS Trainium, além do EFA. Para ver uma lista de tipos de instância equipadas com chips de aceleração de hardware e EFA, consulte Computação acelerada no Guia do usuário do Amazon EC2.

Ao comparar os tipos de instância para escolher entre eles, considere o número de placas de rede EFA disponíveis para esse tipo de instância, bem como o número de placas aceleradoras, a quantidade de CPU e a quantidade de memória. É possível atribuir até uma EFA por placa de rede. Um EFA conta como uma interface de rede. Para ver quantos EFA estão disponíveis para cada tipo de instância equipada com EFA, consulte a lista Placas de rede no Guia do usuário do Amazon EC2.

Interfaces EFA e somente EFA

Um Elastic Fabric Adapter (EFA) é uma interface de rede que combina os recursos de um Adaptador de Rede Elástica (ENA) e uma interface de desvio de sistema operacional, alimentada pelo protocolo AWS Scalable Reliable Datagram (SRD). As funcionalidades do EFA permitem que as aplicações se comuniquem diretamente com o hardware para transporte de baixa latência. É possível acessar exclusivamente os recursos do EFA usando interfaces somente EFA, limitando a comunicação às interfaces dentro da mesma zona de disponibilidade.

Para criar nós que possam ter interfaces somente EFA, você deve usar um modelo de inicialização do EC2 personalizado e definir `InterfaceType` como `efa-only`. No modelo de inicialização personalizado, não é possível definir a placa de rede `0` como uma interface somente EFA, pois essa

é a placa de rede principal e a interface de rede da instância do EC2. Você deve ter o CNI da VPC versão `1.18.5` ou posterior para interfaces somente do tipo EFA. Se você estiver usando o Amazon Linux 2, a versão da ami deve ser `v20240928` ou posterior para interfaces somente do tipo EFA.

O procedimento a seguir orienta você a criar um cluster do EKS com o `eksctl` com nós que têm GPUs nVidia e interfaces do EFA. Não é possível usar o `eksctl` para criar nós e grupos de nós que usam interfaces somente EFA.

Pré-requisitos

- Um cluster do existente do Amazon EKS. Se você não tiver um cluster existente, crie um usando o Conceitos básicos. O cluster dove ser implantado em uma VPC que tenha pelo menos uma sub-rede privada com endereços IP disponíveis suficientes para implantar nós. A sub-rede privada deve ter acesso de saída à Internet fornecido por um dispositivo externo, como um gateway NAT.

  Se você planeja usar o `eksctl` para criar o grupo de nós, o `eksctl` também pode criar um cluster para você.
- Versão `2.12.3` ou posterior ou versão `1.27.160` ou posterior da AWS Command Line Interface (AWS CLI) instalada e configurada no seu dispositivo ou no AWS CloudShell. Para verificar sua versão atual, use `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Os gerenciadores de pacotes, como `yum`, `apt-get` ou Homebrew para macOS, geralmente estão várias versões atrás da versão mais recente da AWS CLI. Para instalar a versão mais recente, consulte Installing e Quick configuration with aws configure, no Guia do usuário da AWS Command Line Interface. A versão da AWS CLI instalada no AWS CloudShell também pode estar várias versões atrás da versão mais recente. Para atualizá-lo, consulte Instalar a AWS CLI no seu diretório pessoal, no Guia do usuário do AWS CloudShell.
- A ferramenta da linha de comando `kubectl` está instalada no seu dispositivo ou no AWS CloudShell. A versão pode ser a mesma ou até uma versão secundária anterior ou posterior à versão do Kubernetes do seu cluster. Por exemplo, se a versão do cluster for a `1.29`, você poderá usar o `kubectl` versão `1.28`, `1.29` ou `1.30` com ele. Para instalar ou atualizar o `kubectl`, consulte the section called “Configuração do kubectl e eksctl”.
- Você deve ter a versão `1.7.10` ou mais recente do plug-in CNI da Amazon VPC para Kubernetes instalada para que possa executar os nós de processamento que sejam compatíveis com vários Elastic Fabric Adapters, como o `p4d` ou `p5`. Para obter mais informações sobre como atualizar a versão do plug-in CNI da Amazon VPC para Kubernetes, consulte the section called “CNI da Amazon VPC”.
- Para instâncias p6-b200, você deve usar o plug-in do dispositivo EFA versão v0.5.6 ou posterior.

> **Important**
>
> Uma consideração importante, necessária para adotar o EFA com o Kubernetes, é configurar e gerenciar páginas enormes como um recurso no cluster. Para obter mais informações, consulte Manage Huge Pages (Gerenciar páginas grandes) na documentação do Kubernetes. As instâncias do Amazon EC2 com o driver EFA instalado pré-alocam 5128 páginas enormes de 2 MiB, as quais você pode solicitar como recursos para consumir em suas especificações de trabalho.

Crie grupos de nós.

O procedimento a seguir ajuda você a criar um grupo de nós que conta com o suporte do `p4d.24xlarge` e interfaces EFA e GPUDirect RDMA, além de executar um exemplo de teste NVIDIA Collective Communications Library (NCCL) para a performance do NCCL de vários nós usando EFAs. O exemplo pode ser usado em um modelo para treinamento de aprendizado profundo distribuído no Amazon EKS usando EFAs.

1. Determine quais tipos de instância do Amazon EC2 com suporte para EFA estão disponíveis na região da AWS em que você deseja implantar os nós. Substitua *`region-code`* pela região da AWS em que deseja implantar os grupo de nós.

   ```
   aws ec2 describe-instance-types --region region-code \
       --filters Name=network-info.efa-supported,Values=true \
       --query "InstanceTypes[*].[InstanceType]" --output text
   ```

   Quando você implanta nós, o tipo de instância que deseja implantar deve estar disponível na região AWS em que o cluster se encontra.

2. Determine em quais zonas de disponibilidade o tipo de instância que você deseja implantar está disponível. Neste tutorial, o tipo de instância `p5.48xlarge` é usado e deve ser retornado na saída para a região AWS que você especificou na etapa anterior. Ao implantar nós em um cluster do produção, substitua *`p5.48xlarge`* por qualquer tipo de instância retornado na etapa anterior.

   ```
   aws ec2 describe-instance-type-offerings --region region-code \
       --location-type availability-zone --filters Name=instance-
   type,Values=p4d.24xlarge,p5.48xlarge \
       --query 'InstanceTypeOfferings[*].Location' --output text
   ```

Veja abaixo um exemplo de saída.

```
us-west-2a    us-west-2c    us-west-2b
```

Observe as zonas de disponibilidade retornadas para uso em etapas posteriores. Quando você implanta nós em um cluster, a VPC deve ter sub-redes com endereços IP disponíveis em uma das zonas de disponibilidade retornadas na saída.

3. Crie um grupo de nós usando `eksctl`. Você precisa da versão `0.215.0` ou posterior da ferramenta de linha de comando `eksctl` instalada em seu dispositivo ou do AWS CloudShell. Para instalar ou atualizar o `eksctl`, consulte Instalação na documentação do `eksctl`.
   a. Copie o conteúdo a seguir em um arquivo chamado *`efa-cluster.yaml`*. Substitua os valores de exemplo pelos seus próprios. É possível substituir `p5.48xlarge` por uma instância diferente mas, se fizer isso, certifique-se de que os valores de `availabilityZones` sejam as zonas de disponibilidade que foram retornadas para o tipo de instância na etapa 1.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
```

   b. Crie um grupo de nós gerenciados em um cluster existente.

```
eksctl create nodegroup -f efa-cluster.yaml
```

Se você não tiver um cluster existente, recomendamos que siga um dos comandos a seguir para criar um cluster e o grupo de nós.

```
eksctl create cluster -f efa-cluster.yaml
```

> **Note**
>
> Como o tipo de instância usado neste exemplo tem GPUs, o `eksctl` instala automaticamente o plug-in de dispositivo NVIDIA para Kubernetes em cada instância para você ao usar o Amazon Linux 2. Isso não é necessário para o Bottlerocket, pois o plug-in do dispositivo NVIDIA está integrado na variante NVIDIA do Bottlerocket. Quando `efaEnabled` é definido como `true` na configuração do grupo de nós, o `eksctl` também implantará automaticamente o plug-in do dispositivo EFA nos nós.

Como usar o Bottlerocket com o EFA

A versão 1.28.0 e posteriores da AMI do Bottlerocket incluem suporte oficial para o EFA. Para usar o Bottlerocket para nós habilitados para o EFA, especifique `amiFamily: Bottlerocket` em sua configuração. Caso precise usar um ID de AMI personalizado, você deverá usar o padrão `nodeGroups` em vez de `managedNodeGroups`.

Confira um exemplo de configuração:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-bottlerocket-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]
```

```
managedNodeGroups:
  - name: my-efa-bottlerocket-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
    amiFamily: Bottlerocket
    bottlerocket:
      enableAdminContainer: true
      settings:
        kernel:
          sysctl:
            "vm.nr_hugepages": "3000"  # Configures 3000 * 2Mi = 6000Mi hugepages
```

A configuração `vm.nr_hugepages` de sysctl acima configura o número de hugepages de 2Mi. Neste exemplo, 3000 significa 3000 * 2Mi = 6000Mi de hugepages.

Verificar a instalação do plug-in do dispositivo EFA

Quando você cria um grupo de nós com `efaEnabled: true`, o `eksctl` implanta automaticamente o plug-in do dispositivo EFA para Kubernetes para você. Você pode verificar se o plug-in do dispositivo está instalado e funcionando corretamente:

1. Verifique o status do DaemonSet:

   ```
   kubectl get daemonsets -n kube-system
   ```

   Exemplo de resultado:

   ```
   NAME                                 DESIRED   CURRENT   READY   UP-TO-DATE
    AVAILABLE   NODE SELECTOR   AGE
   aws-efa-k8s-device-plugin-daemonset   2         2         2       2            2
         <none>          6m16s
   ...
   ```

   Aqui, o DaemonSet do plug-in do dispositivo EFA está sendo executado em dois nós. Ambos estão READY e AVAILABLE.

2. Em seguida, verifique os pods criados pelo DaemonSet:

```
kubectl get pods -n kube-system -l name=aws-efa-k8s-device-plugin
```

Exemplo de resultado:

```
NAME                                          READY   STATUS    RESTARTS   AGE
aws-efa-k8s-device-plugin-daemonset-d68bs   1/1     Running   0          6m16s
aws-efa-k8s-device-plugin-daemonset-w4l8t   1/1     Running   0          6m16s
```

Os pods do plug-in do dispositivo EFA estão no estado Running, confirmando que o plug-in foi implantado com êxito e está operacional.

3. Verifique o registro do recurso:

Você pode confirmar se o recurso `vpc.amazonaws.com/efa` está registrado no kubelet descrevendo os nós:

```
kubectl describe nodes
```

Se o recurso EFA estiver registrado corretamente, você o verá listado sob os recursos Capacidade e Alocável do nó. Por exemplo:

```
Capacity:
  ...
  vpc.amazonaws.com/efa:  4
Allocatable:
  ...
  vpc.amazonaws.com/efa:  4
```

Essa saída confirma que o nó reconhece o recurso EFA, tornando-o disponível para pods que o solicitarem.

(Opcional) Testar a performance do EFA

Recomendamos testar a configuração de EFA. Você pode usar os Testes NCCL no repositório `aws-samples/awsome-distributed-training` no GitHub. Os Testes NCCL avaliam a performance da rede usando a Nvidia Collective Communication Library. As etapas a seguir enviam testes de NCCL no Amazon EKS.

1. Implantar o operador Kubeflow MPI:

   Para os testes NCCL, você pode aplicar o Kubeflow MPI Operator. O MPI Operator facilita a execução do treinamento distribuído no estilo AllReduce no Kubernetes. Para obter mais informações, consulte MPI Operator no GitHub.

2. Execute o teste de performance NCCL em vários nós para verificar o GPUDirectRDMA/EFA:

   Para verificar a performance do NCCL com o GPUDirectRDMA no EFA, execute o teste de performance padrão do NCCL. Para obter mais informações, consulte o respositório NCCL-Tests (Testes de NCCL) no GitHub.

   Conclua as etapas a seguir para executar um teste de performance do NCCL de dois nós. No trabalho de exemplo de teste do NCCL, cada operador solicita oito GPUs, 5.210 Mi de `hugepages-2Mi`, quatro EFAs e 8.000 Mi de memória, o que significa efetivamente que cada operador consome todos os recursos de uma instância `p5.48xlarge`.

   a. Criar o manifesto MPIJob:

      Copie o seguinte para um arquivo chamado `nccl-tests.yaml`:

```
apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: nccl-tests
spec:
  runPolicy:
    cleanPodPolicy: Running
    backoffLimit: 20
  slotsPerWorker: 8
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
         spec:
          restartPolicy: OnFailure
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: test-nccl-launcher
            env:
             - name: PATH
               value: $PATH:/opt/amazon/efa/bin:/usr/bin
```

```
              - name: LD_LIBRARY_PATH
                value: /opt/amazon/openmpi/lib:/opt/nccl/build/lib:/opt/amazon/efa/
lib:/opt/aws-ofi-nccl/install/lib:/usr/local/nvidia/lib:$LD_LIBRARY_PATH
              - name: NCCL_DEBUG
                value: INFO
              - name: NCCL_BUFFSIZE
                value: '8388608'
              - name: NCCL_P2P_NET_CHUNKSIZE
                value: '524288'
              - name: NCCL_TUNER_PLUGIN
                value: /opt/aws-ofi-nccl/install/lib/libnccl-ofi-tuner.so
            command:
            - /opt/amazon/openmpi/bin/mpirun
            - --allow-run-as-root
            - --tag-output
            - -np
            - "16"
            - -N
            - "8"
            - --bind-to
            - none
            - -x
            - PATH
            - -x
            - LD_LIBRARY_PATH
            - -x
            - NCCL_DEBUG=INFO
            - -x
            - NCCL_BUFFSIZE
            - -x
            - NCCL_P2P_NET_CHUNKSIZE
            - -x
            - NCCL_TUNER_PLUGIN
            - --mca
            - pml
            - ^cm,ucx
            - --mca
            - btl
            - tcp,self
            - --mca
            - btl_tcp_if_exclude
            - lo,docker0,veth_def_agent
            - /opt/nccl-tests/build/all_reduce_perf
            - -b
```

```
            - "8"
            - -e
            - "16G"
            - -f
            - "2"
            - -g
            - "1"
            - -c
            - "1"
            - -n
            - "100"
    Worker:
      replicas: 2
      template:
        spec:
          nodeSelector:
            node.kubernetes.io/instance-type: "p5.48xlarge"
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: nccl-tests-worker
            volumeMounts:
            - name: shmem
              mountPath: /dev/shm
            resources:
              limits:
                nvidia.com/gpu: 8
                hugepages-2Mi: 5120Mi
                vpc.amazonaws.com/efa: 32
                memory: 32000Mi
              requests:
                nvidia.com/gpu: 8
                hugepages-2Mi: 5120Mi
                vpc.amazonaws.com/efa: 32
                memory: 32000Mi
          volumes:
          - name: shmem
            hostPath:
              path: /dev/shm
```

b. Aplicar MPIJob NCCL-tests:

Envie o `MPIJob` aplicando o manifesto. Isso criará duas instâncias `p5.48xlarge` do Amazon EC2.

```
kubectl apply -f nccl-tests.yaml
```

Veja abaixo um exemplo de saída.

```
mpijob.kubeflow.org/nccl-tests created
```

c. Verifique se o trabalho iniciou os pods:

Visualize os pods em execução.

```
kubectl get pods
```

Veja abaixo um exemplo de saída.

```
NAME                             READY   STATUS     RESTARTS   AGE
nccl-tests-launcher-nbql9    0/1     Init:0/1   0          2m49s
nccl-tests-worker-0          1/1     Running    0          2m49s
nccl-tests-worker-1          1/1     Running    0          2m49s
```

O MPI Operator cria um pod inicializador e dois pods de operador (um em cada nó).

d. Verifique se o trabalho está sendo executado com êxito com os logs:

Visualize o log do pod `nccl-tests-launcher`. Substitua *nbql9* pelo valor de sua saída.

```
kubectl logs -f nccl-tests-launcher-nbql9
```

Se o teste for concluído com êxito, você poderá implantar as aplicações que usam a Nvidia Collective Communication Library.

Usar instâncias do AWS Inferentia com o Amazon EKS para machine learning

Este tópico descreve como criar um cluster de Amazon EKS com nós que executam as instâncias Amazon EC2 Inf1 e (opcionalmente) implantar uma aplicação de exemplo. As instâncias Inf1 do Amazon EC2 são alimentadas por chips do AWS Inferentia que são personalizados pela AWS para fornecer alta performance e inferência de menor custo na nuvem. Os modelos de machine learning são implantados em contêineres usando o AWS Neuron, um kit de desenvolvimento de

software (SDK) especializado que consiste em um compilador, runtime e ferramentas de perfil que otimizam a performance de inferências de machine learning dos chips do AWS Inferentia. O Neuron é compatível com frameworks de machine learning muito usados, como TensorFlow, PyTorch e MXNet.

> **Note**
>
> Os IDs lógicos do dispositivo Neuron devem ser contíguos. Se um pod que solicita vários dispositivos Neuron estiver agendado em um tipo de instância `inf1.6xlarge` ou `inf1.24xlarge` (que tem mais de um dispositivo Neuron), esse pod não será iniciado se o agendador do Kubernetes selecionar IDs de dispositivo não contíguos. Para obter mais informações, consulte Device logical IDs must be contiguous no GitHub.

Pré-requisitos

- Ter o `eksctl` instalado no computador. Se ele não estiver instalado, consulte Instalação na documentação do `eksctl`.
- Ter o `kubectl` instalado no computador. Para obter mais informações, consulte the section called "Configuração do kubectl e eksctl".
- (Opcional) Instale o `python3` no computador. Se você não o tiver instalado, consulte as instruções de instalação de downloads do Python.

Criar um cluster

1. Crie um cluster com os nós da instância Inf1 do Amazon EC2. É possível substituir *`inf1.2xlarge`* por qualquer tipo de instância Inf1. O utilitário `eksctl` detecta que você está iniciando um grupo de nós com um tipo de instância `Inf1` e iniciará seus nós utilizando uma das AMIs aceleradas do Amazon Linux otimizadas para Amazon EKS.

   > **Note**
   >
   > Você não pode usar perfis do IAM para contas de serviço com o TensorFlow Serving.

   ```
   eksctl create cluster \
       --name inferentia \
   ```

```
    --region region-code \
    --nodegroup-name ng-inf1 \
    --node-type inf1.2xlarge \
    --nodes 2 \
    --nodes-min 1 \
    --nodes-max 4 \
    --ssh-access \
    --ssh-public-key your-key \
    --with-oidc
```

> **Note**
>
> Observe o valor da seguinte linha da saída. Ele é usado em uma etapa posterior (opcional).

```
[9]  adding identity "arn:aws:iam::111122223333:role/eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09" to auth ConfigMap
```

Ao iniciar um grupo de nós com instâncias `Inf1`, o `eksctl` instala automaticamente o plug-in de dispositivo Kubernetes do AWS Neuron. Este plug-in anuncia dispositivos Neuron como um recurso de sistema para o programador Kubernetes, que pode ser solicitado por um contêiner. Além das políticas padrão do IAM do nó do Amazon EKS, a política de acesso somente leitura do Amazon S3 é adicionada para que a aplicação de exemplo, abordada em uma etapa posterior, possa carregar um modelo treinado do Amazon S3.

2. Verifique se todos os pods foram iniciados corretamente.

```
kubectl get pods -n kube-system
```

Saída abreviada:

```
NAME                                   READY   STATUS    RESTARTS   AGE
[...]
neuron-device-plugin-daemonset-6djhp   1/1     Running   0          5m
neuron-device-plugin-daemonset-hwjsj   1/1     Running   0          5m
```

(Opcional) Implante uma imagem da aplicação do TensorFlow Serving

Um modelo treinado deve ser compilado para um destino do Inferentia antes que possa ser implantado em instâncias do Inferentia. Para continuar, você precisará de um modelo do TensorFlow otimizado para Neuron salvo no Amazon S3. Se você ainda não tem um SavedModel, siga o tutorial para Criar um modelo ResNet50 compatível com Neuron e carregue o SavedModel resultante no S3. O Resnet-50 é um modelo de machine learning popular usado para tarefas de reconhecimento de imagem. Para obter mais informações sobre a compilação de modelos Neuron, consulte O chip AWS Inferentia com DLAMI no Guia do desenvolvedor de AMIs de aprendizagem profunda AWS.

O exemplo de manifesto de implantação gerencia um contêiner de serviço de inferência pré-criado para o TensorFlow fornecido pelo AWS Deep Learning Containers. Dentro do recipiente está o AWS Neuron Runtime e a aplicação TensorFlow Serving. Uma lista completa de Deep Learning Containers pré-criados otimizados para Neuron é mantida no GitHub em Available Images (Imagens Disponíveis). Na inicialização, o DLC buscará o modelo do Amazon S3, iniciará o Neuron TensorFlow Serving com o modelo salvo e aguardará as solicitações de previsão.

O número de dispositivos Neuron alocados para a aplicação de serviço pode ser ajustado alterando o recurso `aws.amazon.com/neuron` no yaml de implantação. Observe que a comunicação entre o TensorFlow Serving e o runtime do Neuron acontece no GRPC, o que requer a transferência do recurso `IPC_LOCK` para o contêiner.

1. Adicione a política do IAM `AmazonS3ReadOnlyAccess` à perfil de instância de nó que foi criada na etapa 1 de Criar um cluster. Isso é necessário para que a aplicação de exemplo possa carregar um modelo treinado do Amazon S3.

   ```
   aws iam attach-role-policy \
       --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
       --role-name eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09
   ```

2. Crie um arquivo denominado `rn50_deployment.yaml` com o seguinte conteúdo: Atualize o código da região e o caminho do modelo para corresponder às configurações desejadas. O nome do modelo é para fins de identificação quando um cliente faz uma solicitação ao servidor do TensorFlow. Este exemplo usa um nome de modelo para corresponder a um script de cliente ResNet50 de exemplo que será usado em uma etapa posterior para enviar solicitações de previsão.

   ```
   aws ecr list-images --repository-name neuron-rtd --registry-id 790709498068 --region
    us-west-2
   ```

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
    role: master
spec:
  replicas: 2
  selector:
    matchLabels:
      app: eks-neuron-test
      role: master
  template:
    metadata:
      labels:
        app: eks-neuron-test
        role: master
    spec:
      containers:
        - name: eks-neuron-test
          image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
neuron:1.15.4-neuron-py37-ubuntu18.04
          command:
            - /usr/local/bin/entrypoint.sh
          args:
            - --port=8500
            - --rest_api_port=9000
            - --model_name=resnet50_neuron
            - --model_base_path=s3://${your-bucket-of-models}/resnet50_neuron/
          ports:
            - containerPort: 8500
            - containerPort: 9000
          imagePullPolicy: IfNotPresent
          env:
            - name: AWS_REGION
              value: "us-east-1"
            - name: S3_USE_HTTPS
              value: "1"
            - name: S3_VERIFY_SSL
              value: "0"
            - name: S3_ENDPOINT
              value: s3.us-east-1.amazonaws.com
```

```
            - name: AWS_LOG_LEVEL
              value: "3"
          resources:
            limits:
              cpu: 4
              memory: 4Gi
              aws.amazon.com/neuron: 1
            requests:
              cpu: "1"
              memory: 1Gi
          securityContext:
            capabilities:
              add:
                - IPC_LOCK
```

3. Implante o modelo.

```
kubectl apply -f rn50_deployment.yaml
```

4. Crie um arquivo denominado `rn50_service.yaml` com o conteúdo a seguir. As portas HTTP e gRPC são abertas para aceitar solicitações de previsão.

```
kind: Service
apiVersion: v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
spec:
  type: ClusterIP
  ports:
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: eks-neuron-test
    role: master
```

5. Crie um serviço Kubernetes para a sua aplicação de modelo do TensorFlow Serving.

```
kubectl apply -f rn50_service.yaml
```

(Opcional) Faça previsões em relação ao serviço do TensorFlow Serving

1. Para testar localmente, encaminhe a porta gRPC para o serviço `eks-neuron-test`.

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. Crie um script Python chamado `tensorflow-model-server-infer.py` com o conteúdo a seguir. Esse script executa a inferência via gRPC, que é um framework de serviço.

```
import numpy as np
   import grpc
   import tensorflow as tf
   from tensorflow.keras.preprocessing import image
   from tensorflow.keras.applications.resnet50 import preprocess_input
   from tensorflow_serving.apis import predict_pb2
   from tensorflow_serving.apis import prediction_service_pb2_grpc
   from tensorflow.keras.applications.resnet50 import decode_predictions

   if __name__ == '__main__':
       channel = grpc.insecure_channel('localhost:8500')
       stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
       img_file = tf.keras.utils.get_file(
           "./kitten_small.jpg",
           "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg")
       img = image.load_img(img_file, target_size=(224, 224))
       img_array = preprocess_input(image.img_to_array(img)[None, ...])
       request = predict_pb2.PredictRequest()
       request.model_spec.name = 'resnet50_inf1'
       request.inputs['input'].CopyFrom(
           tf.make_tensor_proto(img_array, shape=img_array.shape))
       result = stub.Predict(request)
       prediction = tf.make_ndarray(result.outputs['output'])
       print(decode_predictions(prediction))
```

3. Execute o script para enviar previsões ao seu serviço.

```
python3 tensorflow-model-server-infer.py
```

Veja um exemplo de saída abaixo.

```
[[(u'n02123045', u'tabby', 0.68817204), (u'n02127052', u'lynx', 0.12701613),
 (u'n02123159', u'tiger_cat', 0.08736559), (u'n02124075', u'Egyptian_cat',
 0.063844085), (u'n02128757', u'snow_leopard', 0.009240591)]]
```

Gerencie recursos computacionais para workloads de IA/ML no Amazon EKS

Esta seção foi criada para ajudar você a gerenciar recursos computacionais para workloads de machine learning no Amazon Elastic Kubernetes Service (EKS). Você encontrará detalhes sobre como reservar GPUs usando blocos de capacidade para grupos de nós gerenciados e nós autogerenciados, incluindo pré-requisitos, configuração do modelo de lançamento, configurações de escalabilidade, preparação da workload e considerações importantes para lidar com os ciclos de vida das reservas e o encerramento normal dos nós.

Tópicos

- Criar um grupo de nós gerenciados com blocos de capacidade para ML
- Criar nós autogerenciados com blocos de capacidade para ML
- Uso do P6e-GB200 UltraServers com o Amazon EKS

Criar um grupo de nós gerenciados com blocos de capacidade para ML

Os blocos de capacidade para machine learning (ML) permitem que você reserve instâncias com GPU para uma data futura, a fim de dar suporte a workloads de ML de curta duração. Para obter mais informações, consulte Blocos de capacidade para ML no Guia do Usuário do Amazon EC2 para Instâncias Linux.

Considerações

Important

- Os blocos de capacidade estão disponíveis apenas para determinados tipos de instância do Amazon EC2 e para regiões da AWS. Para obter informações sobre compatibilidade,

consulte Trabalhar com pré-requisitos de blocos de capacidade no Guia do usuário do Amazon EC2 para instâncias Linux.

- Para obter mais informações, consulte Usar blocos de capacidade para workloads de machine learning no Guia do usuário do Amazon EC2 Auto Scaling.
- Grupos de nós gerenciados com blocos de capacidade só podem ser criados com modelos de lançamento personalizados.
- Ao atualizar grupos de nós gerenciados com blocos de capacidade, certifique-se de que o tamanho desejado do grupo de nós esteja definido como `0`.

Criar um grupo de nós gerenciados com blocos de capacidade do Amazon EC2

Você pode usar blocos de capacidade com grupos de nós gerenciados do Amazon EKS para provisionar e escalar os nós de processamento acelerados por GPU. Os exemplos de modelos do AWS CloudFormation a seguir não abrangem todos os aspectos necessários em um cluster de produção. Normalmente, você também vai querer um script de bootstrapping para unir o nó ao cluster e especificar a AMI acelerada do Amazon EKS. Para obter mais informações, consulte the section called “Criar”.

1. Crie um modelo de execução apropriado para as workloads e que funcione com grupos de nós gerenciados do Amazon EKS. Para obter mais informações, consulte the section called “Modelos de execução”.

   Além dos requisitos dos procedimentos acima, certifique-se de que `LaunchTemplateData` inclua o seguinte:

   - `InstanceMarketOptions` com `MarketType` definido como `"capacity-block"`
   - `CapacityReservationSpecification: CapacityReservationTarget` com `CapacityReservationId` definido como bloco de capacidade (por exemplo:`cr-`*`02168da1478b509e0`*)
   - `InstanceType` definido como um tipo de instância que oferece suporte a blocos de capacidade (por exemplo: *`p5.48xlarge`*)

   Veja a seguir um trecho de um modelo do CloudFormation para criar um modelo de execução direcionado a um bloco de capacidade. Para criar um grupo personalizado de nós gerenciados da AMI, você também pode adicionar os parâmetros `ImageId` e `UserData`.

   ```
   NodeLaunchTemplate:
   ```

```
  Type: "AWS::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      InstanceType: p5.48xlarge
```

2. Use o modelo de execução para criar um grupo de nós gerenciados.

   Veja a seguir um exemplo de comando para criar grupos de nós para blocos de capacidade. Substitua *os valores de exemplo* pelos valores aplicáveis ao seu cluster.

   Ao criar o grupo de nós gerenciados do bloco de capacidade, faça o seguinte:

   - Defina `capacity-type` como `"CAPACITY_BLOCK"` . Se o tipo de capacidade não estiver definido como `"CAPACITY_BLOCK"`, ou se algum dos outros valores obrigatórios do modelo de execução acima estiver ausente, a solicitação de criação será rejeitada.
   - Ao especificar `subnets` na solicitação de criação, certifique-se de especificar somente a sub-rede na mesma zona de disponibilidade como a reserva de capacidade.
   - Se você especificar um `desiredSize` diferente de zero na solicitação de criação, o Amazon EKS atenderá a essa especificação ao criar o grupo do Auto Scaling (ASG). No entanto, se a solicitação de criação for feita antes que a reserva de capacidade esteja ativa, o ASG não poderá executar instâncias do Amazon EC2 até que elas se tornem ativas. Como resultado, as atividades de escalabilidade do ASG terão erros de execução. Sempre que a reserva se tornar ativa, a execução das instâncias terá êxito e o ASG terá a escala aumentada verticalmente para o `desiredSize` mencionado no momento da criação.

   ```
   aws eks create-nodegroup \
       --cluster-name my-cluster \
       --nodegroup-name my-mng \
       --node-role node-role-arn \
       --region region-code \
       --subnets subnet-id \
       --scaling-config minSize=node-group-min-size,maxSize=node-group-max-
   size,desiredSize=node-group-desired-size \
       --ami-type "AL2023_x86_64_NVIDIA" \
       --capacity-type "CAPACITY_BLOCK" \
       --launch-template id="lt-id",version=1
   ```

3. Verifique se os nós se unem após o aumento da escala verticalmente. Os clusters do Amazon EKS que usam grupos de nós gerenciados com blocos de capacidade não realizam nenhuma validação de que as instâncias executadas realmente se juntam e se registram no cluster.
4. Caso defina `desiredSize` como `0` no momento da criação, você terá opções diferentes para aumentar a escala verticalmente do grupo de nós quando a reserva de capacidade ficar ativa:
   - Crie uma política de escalabilidade programada para o ASG que se alinhe ao horário de início da reserva do bloco de capacidade. Para obter mais informações, consulte Escalabilidade agendada para o Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.
   - Use o console do Amazon EKS ou `eks update-nodegroup-config` para atualizar a configuração de escalabilidade e defina o tamanho desejado do grupo de nós.
   - Use o Kubernetes Cluster Autoscaler. Para obter mais informações, consulte Autoscaler do cluster na AWS.
5. O grupo de nós agora está pronto para que as workloads e os pods sejam agendados.
6. Para que os pods sejam facilmente drenados antes do término da reserva, o Amazon EKS usa uma política de escalabilidade agendada para reduzir a escala verticalmente do tamanho do grupo de nós para `0`. Essa escalabilidade programada será denominada `Amazon EKS Node Group Capacity Scaledown Before Reservation End`. Recomendamos não editar ou excluir essa ação.

   O Amazon EC2 começa a encerrar as instâncias 30 minutos antes do horário de término da reserva. Como resultado, o Amazon EKS vai configurar uma redução vertical da escala agendada no grupo de nós quarenta minutos antes do final da reserva, a fim de remover pods com segurança e facilidade.

Criar nós autogerenciados com blocos de capacidade para ML

Os blocos de capacidade para machine learning (ML) permitem que você reserve instâncias com GPU para uma data futura, a fim de dar suporte a workloads de ML de curta duração. Para obter mais informações, consulte Blocos de capacidade para ML no Guia do Usuário do Amazon EC2 para Instâncias Linux.

Considerações

> **Important**
>
> - Os blocos de capacidade estão disponíveis apenas para determinados tipos de instância do Amazon EC2 e para regiões da AWS. Para obter informações sobre compatibilidade, consulte Trabalhar com pré-requisitos de blocos de capacidade no Guia do usuário do Amazon EC2 para instâncias Linux.
> - Se você criar o grupo de nós autogerenciados antes de a reserva de capacidade se tornar ativa, defina a capacidade desejada como `0`.
> - Para permitir tempo suficiente para a drenagem adequada do(s) nó(s), sugerimos que você agende o dimensionamento para reduzir para zero mais de 30 minutos antes do horário de término da reserva do bloco de capacidade.
> - Para que os pods sejam drenados facilmente, recomendamos definir o AWS Node Termination Handler, conforme explicado nas próximas etapas.

Usar blocos de capacidade com nós autogerenciados

Você pode usar blocos de capacidade com o Amazon EKS para provisionar e escalar seus nós autogerenciados. As etapas a seguir fornecem um exemplo geral de visão geral. Os exemplos de modelos do AWS CloudFormation não abrangem todos os aspectos necessários em uma workload de produção. Normalmente, você também vai querer um script de bootstrapping para unir o nó ao cluster, especificar a AMI acelerada do Amazon EKS e um perfil de instância apropriado para unir ao cluster. Para obter mais informações, consulte the section called “Amazon Linux”.

1. Crie um modelo de execução que seja aplicável à workload. Para obter mais informações, consulte Usar blocos de capacidade para workloads de machine learning no Guia do usuário do Amazon EC2 Auto Scaling.

   Certifique-se de que `LaunchTemplateData` inclua o seguinte:

   - `InstanceMarketOptions` com `MarketType` definido como `"capacity-block"`
   - `CapacityReservationSpecification: CapacityReservationTarget` com `CapacityReservationId` definido como bloco de capacidade (por exemplo:`cr-`*`02168da1478b509e0`*)
   - `IamInstanceProfile` com o `Arn` definido como o *`iam-instance-profile-arn`* aplicável

- `ImageId` definido como o *`image-id`* aplicável
- `InstanceType` definido como um tipo de instância que oferece suporte a blocos de capacidade (por exemplo: *`p5.48xlarge`*)
- `SecurityGroupIds` definido para as IDs aplicáveis (por exemplo: *`sg-05b1d815d1EXAMPLE`*)
- `UserData` definido como os *`dados de usuário`* aplicáveis ao seu grupo de nós autogerenciado

  Veja a seguir um trecho de um modelo do CloudFormation para criar um modelo de execução direcionado a um bloco de capacidade.

```
NodeLaunchTemplate:
  Type: "aws::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      IamInstanceProfile:
        Arn: iam-instance-profile-arn
      ImageId: image-id
      InstanceType: p5.48xlarge
      KeyName: key-name
      SecurityGroupIds:
      - sg-05b1d815d1EXAMPLE
      UserData: user-data
```

  Você deve passar a sub-rede na zona de disponibilidade na qual a reserva é feita porque os blocos de capacidade são zonais.

2. Use o modelo de execução para criar um grupo de nós autogerenciados. Se você estiver criando o grupo de nós autogerenciados antes de a reserva de capacidade se tornar ativa, defina a capacidade desejada como `0`. Ao criar o grupo de nós, certifique-se de especificar apenas a sub-rede correspondente à Zona de Disponibilidade na qual a capacidade está reservada.

   Veja abaixo um modelo do CloudFormation de exemplo que pode ser utilizado como referência ao criar um modelo que seja aplicável à sua workload. Esse exemplo obtém o `LaunchTemplateId` e a `Version` do recurso `AWS::Amazon EC2::LaunchTemplate` mostrado no exemplo

anterior. Ele também obtém os valores para`DesiredCapacity`, `MaxSize`, `MinSize` e `VPCZoneIdentifier` que são declarados em outro lugar no mesmo modelo.

```
NodeGroup:
  Type: "AWS::AutoScaling::AutoScalingGroup"
  Properties:
    DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
    LaunchTemplate:
      LaunchTemplateId: !Ref NodeLaunchTemplate
      Version: !GetAtt NodeLaunchTemplate.LatestVersionNumber
    MaxSize: !Ref NodeAutoScalingGroupMaxSize
    MinSize: !Ref NodeAutoScalingGroupMinSize
    VPCZoneIdentifier: !Ref Subnets
    Tags:
      - Key: Name
        PropagateAtLaunch: true
        Value: !Sub ${ClusterName}-${NodeGroupName}-Node
      - Key: !Sub kubernetes.io/cluster/${ClusterName}
        PropagateAtLaunch: true
        Value: owned
```

3. Depois que o grupo de nós for criado com sucesso, certifique-se de registrar o `NodeInstanceRole` para o grupo de nós que foi criado. Você precisa disso para garantir que, quando o grupo de nós for escalado, os novos nós se associem ao cluster e o Kubernetes seja capaz de reconhecer os nós. Para obter mais informações, consulte as instruções do Console de gerenciamento da AWS em Criar nós autogerenciados do Amazon Linux.
4. Recomendamos que você crie uma política de escalabilidade programada para o grupo do Auto Scaling que se alinhe aos horários de reserva do bloco de capacidade. Para obter mais informações, consulte Escalabilidade agendada para o Amazon EC2 Auto Scaling no Manual do usuário do Amazon EC2 Auto Scaling.

   Você pode usar todas as instâncias reservadas até 30 minutos antes do horário final do bloco de capacidade. As instâncias que ainda estiverem em execução nesse momento começarão a ser encerradas. Para permitir tempo suficiente para a drenagem adequada do(s) nó(s), sugerimos que você agende o dimensionamento para reduzir para zero mais de 30 minutos antes do horário de término da reserva do bloco de capacidade.

   Se, em vez disso, quiser aumentar a escala manualmente sempre que a reserva de capacidade chegar a `Active`, você precisará atualizar a capacidade desejada do grupo do Auto Scaling no horário de início da reserva do bloco de capacidade. Em seguida, você também precisaria reduzir

a escala manualmente mais de 30 minutos antes do horário de término da reserva do bloco de capacidade.

5. O grupo de nós agora está pronto para que as workloads e os pods sejam agendados.
6. Para que os pods sejam drenados normalmente, recomendamos que você configure o AWS Node Termination Handler. Esse manipulador poderá observar os eventos do ciclo de vida "ASG Scale-in" do Amazon EC2 Auto Scaling usando o EventBridge e permitir que o ambiente de gerenciamento do Kubernetes execute as ações necessárias antes que a instância fique indisponível. Do contrário, seus objetos do Kubernetes e dos pods ficarão presos em um estado pendente. Para obter mais informações, consulte Manipulador do término do nó da AWS no GitHub.

   Se você não configurar um Node Termination Handler, recomendamos que comece a drenar seus pods manualmente antes de chegar à janela de trinta minutos, para que haja tempo suficiente para a drenagem adequada.

Uso do P6e-GB200 UltraServers com o Amazon EKS

Este tópico descreve como realizar a configuração e o uso do Amazon EKS com P6e-GB200 UltraServers. O tipo de instância `p6e-gb200.36xlarge` com 4 GPUs NVIDIA Blackwell está disponível apenas como P6e-GB200 UltraServers. Existem dois tipos de P6e-GB200 UltraServers. O UltraServer `u-p6e-gb200x36` conta com 9 instâncias `p6e-gb200.36xlarge` e o UltraServer `u-p6e-gb200x72` conta com 18 instâncias `p6e-gb200.36xlarge`.

Para saber mais informações, consulte a página da web do P6e-GB200 UltraServers do Amazon EC2.

Considerações

- O Amazon EKS oferece suporte ao P6e-GB200 UltraServers para a versão 1.33 e versões posteriores do Kubernetes. O lançamento desta versão do Kubernetes é compatível com a alocação dinâmica de recursos (DRA, na sigla em inglês), habilitada por padrão no EKS e nas AMIs aceleradas e otimizadas para o EKS do AL2023. A DRA é um requisito para usar o P6e-GB200 UltraServers com o EKS. A DRA não é compatível no Karpenter ou no modo automático do EKS, e recomenda-se o uso de grupos de nós autogerenciados do EKS ou de grupos de nós gerenciados pelo EKS ao usar o P6e-GB200 UltraServers com o EKS.

- O P6e-GB200 UltraServers é disponibilizado por meio de Blocos de Capacidade do EC2 para ML. Consulte the section called “Gerenciamento de capacidade” para obter informações sobre como iniciar nós do EKS com Blocos de Capacidade.
- Ao usar grupos de nós gerenciados do EKS com Blocos de Capacidade, é obrigatório o uso de modelos de inicialização personalizados. Ao atualizar os grupos de nós gerenciados do EKS com P6e-GB200 UltraServers, você deve definir o tamanho desejado do grupo de nós como `0` antes de atualizar.
- Recomenda-se usar a variante do AL2023 para a arquitetura ARM da NVIDIA nas AMIs aceleradas e otimizadas para EKS. Esta AMI inclui os componentes de nó e a configuração necessários para trabalhar com P6e-GB200 UltraServers. Se você decidir criar sua própria AMI, será responsável por instalar e validar a compatibilidade do nó e do software do sistema, incluindo os drivers. Para obter mais informações, consulte the section called “Uso de AMIs do Linux com suporte a GPU para o EKS”.
- Recomenda-se usar a versão `v20251103` ou versões posteriores da AMI otimizada para o EKS, que inclui o driver da NVIDIA na versão 580. Esta versão do driver da NVIDIA habilita o Coherent Driver-Based Memory Management (CDMM) para tratar possíveis gerações excessivas de relatórios de memória. Quando o CDMM está habilitado, as seguintes funcionalidades não são suportadas: GPU multi-instância (MIG, na sigla em inglês) da NVIDIA e vGPU. Para obter mais informações sobre o CDMM, consulte NVIDIA Coherent Driver-based Memory Management (CDMM).
- Ao usar o NVIDIA GPU Operator com a AMI do AL2023 otimizada para o EKS com a NVIDIA, você deve desabilitar a instalação do driver e do toolkit pelo operador, pois estes já estão incluídos na AMI. As AMIs do AL2023 otimizadas para o EKS com a NVIDIA não incluem o plug-in de dispositivo da NVIDIA para Kubernetes ou o driver de DRA da NVIDIA. Portanto, estes devem ser instalados separadamente.
- Cada instância `p6e-gb200.36xlarge` pode ser configurada com até 17 placas de rede e pode aproveitar o EFA para comunicação entre UltraServers. O tráfego de rede das workloads pode transitar entre diferentes UltraServers, mas, para obter a máxima performance, recomenda-se agendar as workloads no mesmo UltraServer, aproveitando o IMEX para comunicação entre GPUs no nível intra-UltraServer. Para obter mais informações, consulte Configuração do EFA para instâncias P6e-GB200.
- Cada instância `p6e-gb200.36xlarge` conta com 3x 7,5 TB de armazenamento de instância. Por padrão, a AMI otimizada para o EKS não realiza a formatação nem a montagem desses armazenamentos de instâncias. O armazenamento efêmero do nó pode ser compartilhado entre pods que solicitam armazenamento efêmero e imagens de contêiner baixadas no nó. Se

estiver usando a AMI do AL2023 otimizada para o EKS, isso pode ser configurado como parte da inicialização dos nós nos dados do usuário, definindo a política de armazenamento local da instância em NodeConfig como RAID0. Ao configurar o RAID0, a instância armazena os dados em faixas e configura o runtime do contêiner e o kubelet para usar esse armazenamento efêmero.

Componentes

Os componentes, apresentados a seguir, são recomendados para a execução de workloads no EKS com o P6e-GB200 UltraServers. Você pode, opcionalmente, usar o NVIDIA GPU Operator para instalar os componentes de nó da NVIDIA. Ao usar o NVIDIA GPU Operator com a AMI do AL2023 otimizada para o EKS com a NVIDIA, você deve desabilitar a instalação do driver e do toolkit pelo operador, pois estes já estão incluídos na AMI.

Stack	Componente
AMI acelerada e otimizada para o EKS	Kernel 6.12
	Driver de GPU da NVIDIA
	Driver de modo de usuário do NVIDIA CUDA
	NVIDIA Container Toolkit
	NVIDIA Fabric Manager
	Driver de IMEX da NVIDIA
	Gerenciador de sub-rede NVIDIA NVLink
	Driver do EFA
Componentes em execução no nó	CNI da VPC
	Plug-in de dispositivo do EFA
	Plug-in de dispositivo da NVIDIA para K8s
	Driver de DRA da NVIDIA
	NVIDIA Node Feature Discovery (NFD)

Stack	Componente
	NVIDIA GPU Feature Discovery (GFD)

Os componentes de nó listados na tabela anterior desempenham as seguintes funções:

- CNI da VPC: realiza a alocação de IPs da VPC como interface de rede principal para os pods que são executados no EKS.
- Plug-in de dispositivo do EFA: realiza a alocação de dispositivos do EFA como redes secundárias para pods que são executados no EKS. É responsável pelo tráfego de rede entre P6e-GB200 UltraServers. Para workloads de vários nós, o tráfego de GPU para GPU dentro de um UltraServer pode ocorrer por meio do NVLink de vários nós.
- Plug-in de dispositivo da NVIDIA para Kubernetes: realiza a alocação de GPUs como dispositivos para pods que são executados no EKS. Recomenda-se usar o plug-in de dispositivo da NVIDIA para Kubernetes até que a funcionalidade de alocação de GPU do driver de DRA da NVIDIA saia do estágio experimental. Consulte os lançamentos do driver de DRA da NVIDIA para obter informações atualizadas.
- Driver de DRA da NVIDIA: habilita recursos personalizados de ComputeDomain que facilitam a criação de domínios de IMEX que acompanham as workloads em execução no P6e-GB200 UltraServers.
  - O recurso ComputeDomain descreve um domínio de Internode Memory Exchange (IMEX). Quando workloads com um ResourceClaim para um ComputeDomain são implantadas no cluster, o driver de DRA da NVIDIA cria automaticamente um DaemonSet de IMEX que é executado nos nós correspondentes e estabelece os canais de IMEX entre os nós antes do início da workload. Para saber mais informações sobre o IMEX, consulte Overview of NVIDIA IMEX for multi-node NVLink systems.
  - O driver de DRA da NVIDIA usa um rótulo de ID de clique (`nvidia.com/gpu.clique`) aplicado pelo NVIDIA GFD, que transmite o conhecimento da topologia de rede e do domínio do NVLink.
  - É uma prática recomendada criar um ComputeDomain para cada trabalho de workload.
- NVIDIA Node Feature Discovery (NFD): dependência obrigatória para que o GFD aplique rótulos aos nós com base nos atributos descobertos no nível do nó.
- NVIDIA GPU Feature Discovery (GFD): aplica aos nós um rótulo de topologia padrão da NVIDIA chamado `nvidia.com/gpu.clique`. Os nós que compartilham o mesmo `nvidia.com/`

`gpu.clique` têm conectividade NVLink para vários nós, e você pode usar afinidades de pod em sua aplicação para agendar pods ao mesmo domínio do NVLink.

Procedimento

A seção a seguir pressupõe que você tenha um cluster do EKS executando o Kubernetes na versão 1.33 ou em versões posteriores com um ou mais grupos de nós com UltraServers P6e-GB200 que executam a AMI do AL2023 acelerada e otimizada para o EKS com o ARM da NVIDIA. Consulte os links apresentados em the section called “Gerenciamento de capacidade” para as etapas de pré-requisito para nós autogerenciados e nos grupos de nós gerenciados do EKS.

O procedimento a seguir usa os componentes apresentados abaixo.

Nome	Versão	Descrição
NVIDIA GPU Operator	25.3.4 ou posteriores	Para o gerenciamento do ciclo de vida dos plug-ins obrigatórios, como o plug-in de dispositivo da NVIDIA para Kubernetes e NFD/GFD.
Drivers de DRA da NVIDIA	25.8.0 ou posteriores	Para as CRDs de ComputeDomain e o gerenciamento de domínios de IMEX.
Plug-in de dispositivo do EFA	0.5.14 ou posteriores	Para a comunicação entre UltraServers.

Instalação do NVIDIA GPU Operator

O NVIDIA GPU Operator simplifica o gerenciamento dos componentes necessários para o uso de GPUs em clusters do Kubernetes. Como o driver de GPU da NVIDIA e o Container Toolkit já estão instalados como parte da AMI acelerada e otimizada para o EKS, estes devem ser definidos como `false` na configuração de valores do Helm.

1. Crie um arquivo de valores do Helm chamado `gpu-operator-values.yaml` com a configuração apresentada a seguir.

```
devicePlugin:
  enabled: true
nfd:
```

```
  enabled: true
gfd:
  enabled: true
driver:
  enabled: false
toolkit:
  enabled: false
migManager:
  enabled: false
```

2. Instale o NVIDIA GPU Operator no cluster utilizando o arquivo `gpu-operator-values.yaml` que você criou na etapa anterior.

```
helm repo add nvidia https://helm.ngc.nvidia.com/nvidia
helm repo update
```

```
helm install gpu-operator nvidia/gpu-operator \
 --namespace gpu-operator \
 --create-namespace \
 --version v25.3.4 \
 --values gpu-operator-values.yaml
```

Instalação do driver de DRA da NVIDIA

A partir da versão `v25.3.4` do NVIDIA GPU Operator, o driver de DRA da NVIDIA deve ser instalado separadamente. Recomenda-se acompanhar as notas de lançamento do NVIDIA GPU Operator, pois isso pode mudar em versões futuras.

1. Crie um arquivo de valores do Helm chamado `dra-values.yaml` com a configuração apresentada a seguir. Observe que os valores `nodeAffinity` e `tolerations` configuram o driver de DRA para ser implantado apenas em nós que contam com uma GPU da NVIDIA.

```
resources:
  gpus:
    enabled: false # set to false to disable experimental gpu support
  computeDomains:
    enabled: true

controller:
  nodeSelector: null
```

```
  affinity: null
  tolerations: []

kubeletPlugin:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: "nvidia.com/gpu.present"
            operator: In
            values:
            - "true"
  tolerations:
    - key: "nvidia.com/gpu"
      operator: Exists
      effect: NoSchedule
```

2. Instale o driver de DRA da NVIDIA no cluster utilizando o arquivo `dra-values.yaml` que você criou na etapa anterior.

```
helm repo add nvidia https://helm.ngc.nvidia.com/nvidia
helm repo update
```

```
helm install nvidia-dra-driver-gpu nvidia/nvidia-dra-driver-gpu \
  --version="25.8.0" \
  --namespace nvidia-dra-driver-gpu \
  --create-namespace \
  -f dra-values.yaml
```

3. Após a instalação, o driver de DRA cria recursos de `DeviceClass` que permitem ao Kubernetes compreender e alocar recursos de `ComputeDomain`, viabilizando o gerenciamento do IMEX para workloads de GPU distribuídas em UltraServers P6e-GB200.

   Confirme se os recursos de DRA estão disponíveis usando os comandos apresentados a seguir.

```
kubectl api-resources | grep resource.k8s.io
```

```
deviceclasses           resource.k8s.io/v1  false        DeviceClass
resourceclaims          resource.k8s.io/v1  true         ResourceClaim
resourceclaimtemplates  resource.k8s.io/v1  true         ResourceClaimTemplate
```

```
resourceslices          resource.k8s.io/v1  false        ResourceSlice
```

```
kubectl get deviceclasses
```

```
NAME
compute-domain-daemon.nvidia.com
compute-domain-default-channel.nvidia.com
```

Instalação do plug-in de dispositivo do EFA

Para usar a comunicação EFA entre UltraServers, você deve instalar o plug-in de dispositivo do Kubernetes para EFA. As instâncias P6e-GB200 podem ser configuradas com até 17 placas de rede, sendo que a NCI primária (índice 0) deve ser do tipo `interface` e fornecer suporte para até 100 Gbps de largura de banda ENA. Configure as interfaces EFA e ENA conforme seus requisitos durante o provisionamento do nó. Analise a documentação da AWS sobre a configuração de EFA para instâncias P6e-GB200 para obter mais detalhes relacionados a essa configuração.

1. Crie um arquivo de valores do Helm chamado `efa-values.yaml` com a configuração apresentada a seguir.

```
tolerations:
  - key: nvidia.com/gpu
    operator: Exists
    effect: NoSchedule
```

2. Instale o NVIDIA DRA Operator no cluster utilizando o arquivo `dra-values.yaml` que você criou na etapa anterior.

```
helm repo add eks https://aws.github.io/eks-charts
helm repo update
```

```
helm install efa eks/aws-efa-k8s-device-plugin -n kube-system \
  --version="0.5.14" \
  -f efa-values.yaml
```

Como exemplo, se você configurou as instâncias com uma interface apenas de EFA em cada grupo de NCI, ao descrever um nó, a expectativa é encontrar quatro dispositivos de EFA alocáveis.

```
kubectl describe node/<gb200-node-name>
```

```
Capacity:
  ...
  vpc.amazonaws.com/efa:  4
Allocatable:
  ...
  vpc.amazonaws.com/efa:  4
```

Validação de IMEX no NVLink de vários nós

Confira o repositório awesome-distributed-training do GitHub para realizar testes de NCCL para o NCLink de vários nós e outras avaliações comparativas de performance. As etapas a seguir mostram como executar um teste de NVLink de vários nós com o nvbandwidth.

1. Para executar um teste de largura de banda de vários nós entre dois nós no domínio NVL72, primeiro instale o MPI Operator:

   ```
   kubectl create -f https://github.com/kubeflow/mpi-operator/releases/download/v0.7.0/
   mpi-operator.yaml
   ```

2. Crie um arquivo de valores do Helm chamado `nvbandwidth-test-job.yaml` que define o manifesto do teste. Observe o uso da afinidade do pod `nvidia.com/gpu.clique` para garantir que os processamentos sejam agendados no mesmo domínio do NVLink que tem conectividade NVLink de vários nós.

   A partir da versão `v25.8.0` do driver de DRA da NVIDIA, os ComputeDomains são elásticos e o campo `.spec.numNodes` pode ser definido como `0` na definição do ComputeDomain. Consulte as últimas notas de lançamento do driver de DRA da NVIDIA para obter atualizações.

   ```
   ---
   apiVersion: resource.nvidia.com/v1beta1
   kind: ComputeDomain
   metadata:
   ```

```
  name: nvbandwidth-test-compute-domain
spec:
  numNodes: 0 # This can be set to 0 from NVIDIA DRA Driver version v25.8.0+
  channel:
    resourceClaimTemplate:
      name: nvbandwidth-test-compute-domain-channel

---
apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: nvbandwidth-test
spec:
  slotsPerWorker: 4 # 4 GPUs per worker node
  launcherCreationPolicy: WaitForWorkersReady
  runPolicy:
    cleanPodPolicy: Running
  sshAuthMountPath: /home/mpiuser/.ssh
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
        metadata:
          labels:
            nvbandwidth-test-replica: mpi-launcher
        spec:
          containers:
          - image: ghcr.io/nvidia/k8s-samples:nvbandwidth-v0.7-8d103163
            name: mpi-launcher
            securityContext:
              runAsUser: 1000
            command:
            - mpirun
            args:
            - --bind-to
            - core
            - --map-by
            - ppr:4:node
            - -np
            - "8"
            - --report-bindings
            - -q
            - nvbandwidth
            - -t
```

```
            - multinode_device_to_device_memcpy_read_ce
    Worker:
      replicas: 2 # 2 worker nodes
      template:
        metadata:
          labels:
            nvbandwidth-test-replica: mpi-worker
        spec:
          affinity:
            podAffinity:
              requiredDuringSchedulingIgnoredDuringExecution:
              - labelSelector:
                  matchExpressions:
                  - key: nvbandwidth-test-replica
                    operator: In
                    values:
                    - mpi-worker
                topologyKey: nvidia.com/gpu.clique
          containers:
          - image: ghcr.io/nvidia/k8s-samples:nvbandwidth-v0.7-8d103163
            name: mpi-worker
            securityContext:
              runAsUser: 1000
            env:
            command:
            - /usr/sbin/sshd
            args:
            - -De
            - -f
            - /home/mpiuser/.sshd_config
            resources:
              limits:
                nvidia.com/gpu: 4  # Request 4 GPUs per worker
              claims:
              - name: compute-domain-channel # Link to IMEX channel
          resourceClaims:
          - name: compute-domain-channel
            resourceClaimTemplateName: nvbandwidth-test-compute-domain-channel
```

3. Crie o recurso ComputeDomain e inicie o trabalho com o comando apresentado a seguir.

```
kubectl apply -f nvbandwidth-test-job.yaml
```

4. Após a criação do ComputeDomain, você poderá verificar que o ComputeDomain da workload tem dois nós:

```
kubectl get computedomains.resource.nvidia.com -o yaml
```

```
status:
  nodes:
  - cliqueID: <ClusterUUID>.<Clique ID>
    ipAddress: <node-ip>
    name: <node-hostname>
  - cliqueID: <ClusterUUID>.<Clique ID>
    ipAddress: <node-ip>
    name: <node-hostname>
  status: Ready
```

5. Analise os resultados do trabalho com o comando apresentado a seguir.

```
kubectl logs --tail=-1 -l job-name=nvbandwidth-test-launcher
```

6. Quando o teste estiver concluído, exclua-o com o comando apresentado a seguir.

```
kubectl delete -f nvbandwidth-test-job.yaml
```

Receitas para otimizar seu cluster Amazon de EKS para workloads de IA/ML

Esta seção foi projetada para fornecer receitas curtas para otimizar seu cluster de Amazon EKS, especialmente para workloads de IA/ML envolvendo hardware especializado. Você encontrará orientações sobre como evitar que os pods sejam programados em nós específicos adicionando taints aos grupos de nós gerenciados, incluindo pré-requisitos, procedimentos passo a passo e considerações de implantação.

Tópicos

- Receita: evitar que pods sejam agendados em nós específicos

Receita: evitar que pods sejam agendados em nós específicos

Visão geral

A execução de nós com processadores especializados, como GPUs, pode ser mais cara do que a de nós executados em máquinas padrão. Para proteger esses nós de workloads que não exijam hardware especial, é possível usar os taints do Kubernetes. Os taints marcam os nós para repelir os pods que não tenham tolerâncias correspondentes, garantindo que somente workloads compatíveis sejam programadas. Para obter mais informações, consulte Taints and Tolerations na documentação do Kubernetes.

As taints de nós do Kubernetes podem ser aplicadas a grupos de nós gerenciados novos e existentes usando o Console de gerenciamento da AWS ou por meio da API do Amazon EKS. Esta receita mostra como aplicar taints aos grupos de nós gerenciados do Amazon EKS usando a AWS CLI. Para obter informações sobre como criar um grupo de nós com um taint usando o Console de gerenciamento da AWS, consulte the section called “Criar”.

Pré-requisitos

- Um cluster existente do Amazon EKS.
- AWS CLI instalada e configurada com as permissões apropriadas.

Etapas

Etapa 1: criação de um grupo de nós com taints

Use o comando `aws eks create-nodegroup` para criar um novo grupo de nós gerenciados com taints. Este exemplo aplica uma taint com chave `dedicated`, valor `gpuGroup` e efeito `NO_SCHEDULE`.

```
aws eks create-nodegroup \
 --cli-input-json '
{
  "clusterName": "my-cluster",
  "nodegroupName": "node-taints-example",
  "subnets": [
     "subnet-1234567890abcdef0",
     "subnet-abcdef01234567890",
     "subnet-021345abcdef67890"
```

```
    ],
  "nodeRole": "arn:aws:iam::111122223333:role/AmazonEKSNodeRole",
  "taints": [
     {
         "key": "dedicated",
         "value": "gpuGroup",
         "effect": "NO_SCHEDULE"
     }
    ]
}'
```

Para obter mais informações e exemplos, consulte taint na documentação de referência do Kubernetes.

Etapa 2: atualizar os taints em um grupo de nós existente

Use o comando aws eks update-nodegroup-config da AWS CLI para adicionar, remover ou substituir taints de grupos de nós gerenciados.

```
aws eks update-nodegroup-config
  --cluster-name my-cluster
  --nodegroup-name node-taints-example
  --taints 'removeTaints=[{key=dedicated,value=gpuGroup,effect=NO_SCHEDULE}]'
```

Observações

- Os taints podem ser atualizados após a criação do grupo de nós usando a API `UpdateNodegroupConfig`.
- A chave da taint deve começar com uma letra ou um número. Pode conter letras, números, hifens (`-`), pontos (`.`) e sublinhados (_). Pode ter até 63 caracteres.
- Opcionalmente, a chave da taint pode começar com um prefixo de subdomínio DNS e um `/` único. Se começar com um prefixo de subdomínio DNS, poderá ter 253 caracteres.
- O valor é opcional e deve começar com uma letra ou um número. Pode conter letras, números, hifens (`-`), pontos (`.`) e sublinhados (_). Pode ter até 63 caracteres.
- Ao usar o Kubernetes diretamente ou o Console de gerenciamento da AWS, o efeito do taint deve ser `NoSchedule`, `PreferNoSchedule` ou `NoExecute`. No entanto, ao usar a AWS CLI ou API, o efeito de taint deve ser `NO_SCHEDULE`, `PREFER_NO_SCHEDULE` ou `NO_EXECUTE`.
- Um máximo de 50 taints são permitidos por grupo de nós.

- Se os taints criados usando um grupo de nós gerenciados forem removidos manualmente de um nó, o Amazon EKS não os adicionará de volta ao nó. Isso será válido mesmo se as contaminações forem especificadas na configuração do grupo de nós gerenciados.

Recursos para começar a usar IA/ML no Amazon EKS

Para entrar no machine learning no EKS, comece escolhendo entre esses padrões prescritivos para preparar rapidamente um cluster de EKS e software e hardware de ML para começar a executar workloads de ML.

Workshops

Workshop de IA generativa no Amazon EKS

Saiba como começar a usar aplicações e inferências de grande modelo de linguagem (LLM) no Amazon EKS. Descubra como implantar e gerenciar workloads de LLM em nível de produção. Por meio de laboratórios práticos, você explorará como aproveitar o Amazon EKS junto com serviços da AWS e ferramentas de código aberto para criar soluções robustas de LLM. O ambiente do workshop fornece toda a infraestrutura e as ferramentas necessárias, permitindo que você se concentre no aprendizado e na implementação.

IA generativa no Amazon EKS usando Neuron

Saiba como começar a usar aplicações e inferências de grande modelo de linguagem (LLM) no Amazon EKS. Descubra como implantar e gerenciar workloads de LLM em nível de produção, implementar padrões RAG avançados com bancos de dados de vetores e criar aplicações de LLM baseadas em dados usando estruturas de código aberto. Por meio de laboratórios práticos, você explorará como aproveitar o Amazon EKS junto com serviços da AWS e ferramentas de código aberto para criar soluções robustas de LLM. O ambiente do workshop fornece toda a infraestrutura e as ferramentas necessárias, permitindo que você se concentre no aprendizado e na implementação.

Melhores práticas

Os tópicos focados em IA/ML no guia de melhores práticas do Amazon EKS fornecem recomendações detalhadas nas áreas a seguir para otimizar suas workloads de IA/ML no Amazon EKS.

Computação e escalonamento automático de IA/ML

Esta seção descreve as melhores práticas para otimizar a computação de IA/ML e o escalonamento automático no Amazon EKS, com foco no gerenciamento de recursos de GPU, resiliência de nós e escalabilidade de aplicações. Ela fornece estratégias como programar workloads com rótulos e afinidade de nós conhecidos, usar blocos de capacidade de ML ou reservas de capacidade sob demanda e implementar verificações de integridade de nós com ferramentas como o Agente de monitoramento de nós do EKS.

Rede de IA/ML

Esta seção descreve as melhores práticas para otimizar a rede de IA/ML no Amazon EKS para melhorar a performance e a escalabilidade, incluindo estratégias como selecionar instâncias com maior largura de banda da rede ou Elastic Fabric Adapter (EFA) para treinamento distribuído, instalar ferramentas como MPI e NCCL e habilitar a delegação de prefixos para aumentar os endereços IP e melhorar os tempos de lançamento do pod.

Segurança de IA/ML

Esta seção se concentra em proteger o armazenamento de dados e garantir a conformidade das workloads de IA/ML no Amazon EKS, incluindo práticas como o uso do Amazon S3 com o AWS Key Management Service (KMS) para criptografia do lado do servidor (SSE-KMS), a configuração de buckets com chaves regionais do KMS e chaves de bucket do S3 para reduzir custos, a concessão de permissões do IAM para ações do KMS, como decodificação em pods do EKS, e auditoria com os logs do AWS CloudTrail.

Armazenamento de IA/ML

Esta seção fornece as melhores práticas para otimizar o armazenamento em workloads de IA/ML no Amazon EKS, incluindo práticas como implantar modelos usando drivers CSI para montar serviços como S3, FSx para Lustre ou EFS como volumes persistentes, selecionar armazenamento com base nas necessidades da workload (por exemplo, FSx para Lustre para treinamento distribuído com opções como Scratch-SSD ou SSD persistente) e habilitar recursos como compressão e distribuição de dados.

Observabilidade de IA/ML

Esta seção se concentra no monitoramento e na otimização da utilização da GPU para workloads de IA/ML no Amazon EKS para melhorar a eficiência e reduzir custos, incluindo estratégias como o

objetivo do alto uso da GPU com ferramentas como o CloudWatch Container Insights e o DCGM-Exporter da NVIDIA integrado com Prometheus e Grafana, além de métricas que recomendamos que você analise para suas workload sde IA/ML.

Performance de IA/ML

Esta seção se concentra em aprimorar a escalabilidade e a performance de aplicaçãoes para workloads de IA/ML no Amazon EKS por meio do gerenciamento de imagens de contêineres e da otimização de startups, incluindo práticas como o uso de imagens básicas pequenas e leves ou contêineres do AWS Deep Learning com compilações de vários estágios, pré-carregamento de imagens por meio de snapshots do EBS ou pré-inserção em cache de runtime usando DaemonSets ou implantações.

Arquiteturas de referência

Explore esses repositórios do GitHub para obter arquiteturas de referência, código de amostra e utilitários para implementar treinamento e inferência distribuídos para workloads de IA/ML no Amazon EKS e em outros serviços da AWS.

AWSome Distributed Training

Este repositório oferece uma coleção de melhores práticas, arquiteturas de referência, exemplos de treinamento de modelos e utilitários para treinar modelos grandes na AWS. Ele oferece suporte a treinamento distribuído com o Amazon EKS, incluindo modelos do CloudFormation para clusters EKS, compilações personalizadas de AMI e contêiner, casos de teste para estruturas como PyTorch (DDP/FSDP, MegatronLM, NeMo) e JAX e ferramentas para validação, observabilidade e monitoramento de performance, como o exportador EFA Prometheus e a Nvidia Nsight Systems.

AWSome Inference

Este repositório fornece arquiteturas de referência e casos de teste para otimizar soluções de inferência na AWS, com foco no Amazon EKS e nas instâncias aceleradas do EC2. Ele inclui configurações de infraestrutura para clusters VPC e EKS, projetos para estruturas como NVIDIA NIMs, TensorRT-LLM, Triton Inference Server e RayService, com exemplos de modelos como Llama3-8B e Llama 3.1 405B. Apresenta implantações de vários nós usando K8s LeaderWorkerSet, ajuste de escala automático do EKS, GPUs de várias instâncias (MIG) e casos de uso reais, como um bot de áudio para ASR, inferência e TTS.

Tutoriais

Se você tem interesse em configurar plataformas e estruturas de machine learning no EKS, explore os tutoriais descritos nesta seção. Esses tutoriais abrangem tudo, desde padrões para fazer o melhor uso dos processadores de GPU até a escolha de ferramentas de modelagem e a criação de estruturas para setores especializados.

Criar plataformas de IA generativa no EKS

- Implemente modelos de IA generativa no Amazon EKS
- Crie plataformas do JupyterHub de vários locatários no Amazon EKS

Executar estruturas de IA generativa especializadas no EKS

- Acelere suas workloads de treinamento de IA generativa distribuídas com o NVIDIA NeMo Framework no Amazon EKS
- Execução do TorchServe no Amazon Elastic Kubernetes Service

Maximize a performance da GPU NVIDIA para ML no EKS

- Implemente o compartilhamento de GPU para usar com eficiência as GPUs NVIDIA em seus clusters do EKS:

  Compartilhamento de GPU no Amazon EKS com divisão de tempo da NVIDIA e instâncias do EC2 aceleradas
- Use GPUs de múltiplas instâncias (MIGs) e microsserviços NIM para executar mais pods por GPU em seus clusters do EKS:

  Maximização da utilização de GPU com a GPU de múltiplas instâncias (MIG) da NVIDIA no Amazon EKS: execução de mais pods por GPU para melhorar a performance
- Crie e implante um sistema de machine learning escalável no Kubernetes com o Kubeflow na AWS

Executar workloads de codificação de vídeo no EKS

- Entrega de conteúdo de vídeo com GPUs fracionárias em contêineres no Amazon EKS

Acelerar o carregamento de imagens para workloads de inferência

- Como a H2O.ai otimizou e protegeu sua infraestrutura de IA/ML com o Karpenter e o Bottlerocket

Monitoramento de workloads de ML

- Monitoramento de workloads de GPU no Amazon EKS usando serviços de código aberto gerenciados pela AWS
- Habilite métricas de GPU baseadas em pods no Amazon CloudWatch

Ferramentas do Amazon EKS

O Amazon EKS oferece um conjunto de ferramentas da AWS que ajudam você a gerenciar, solucionar problemas e interagir com seus clusters do EKS de forma mais eficiente. Essas ferramentas aproveitam recursos baseados em IA e interfaces padronizadas para simplificar as operações do cluster, acelerar a solução de problemas e permitir interações em linguagem natural com os recursos do Kubernetes. Entre as ferramentas disponíveis, estão:

- Servidor do protocolo de contexto para modelos (MCP) do Amazon EKS: um serviço totalmente gerenciado que possibilita experiências baseadas em IA para o gerenciamento de clusters do EKS. O servidor MCP do EKS integra-se a assistentes de programação baseados em IA compatíveis com MCP para fornecer conhecimento contextual em tempo real sobre os clusters e os recursos do Kubernetes. Use essa ferramenta para gerenciar clusters, implantar aplicações, solucionar problemas e consultar a documentação do EKS por meio de interações em linguagem natural.
- Amazon Q no console do Amazon EKS: uma integração de soluções de problemas baseada em IA, disponível diretamente no Console de Gerenciamento da AWS. O Amazon Q analisa automaticamente problemas no cluster e fornece resultados de investigação, análise de causa raiz e etapas sugeridas de correção ao detectar erros ou alertas no console de gerenciamento do EKS.

Essas ferramentas trabalham em conjunto para oferecer suporte abrangente durante todo o ciclo de vida do cluster do EKS, desde a configuração inicial até as operações contínuas e a solução de problemas. Esteja desenvolvendo aplicações, gerenciando a infraestrutura ou resolvendo problemas em ambientes de produção, essas ferramentas da AWS ajudam você a trabalhar com mais eficiência nos clusters do EKS.

Tópicos

- Servidor do protocolo de contexto para modelos (MCP) do Amazon EKS
- Uso do Amazon Q Developer no console do Amazon EKS

Servidor do protocolo de contexto para modelos (MCP) do Amazon EKS

O servidor MCP do Amazon EKS é um serviço totalmente gerenciado que possibilita experiências impulsionadas por IA para desenvolvimento e operações. O protocolo de contexto para modelos (MCP) fornece uma interface padronizada que enriquece aplicações e agentes de IA com

conhecimento contextual em tempo real dos clusters do EKS e recursos do Kubernetes. Isso viabiliza respostas mais precisas, maior consciência de contexto e fluxos de trabalho baseados em IA durante todo o ciclo de vida da aplicação, desde a configuração inicial até a otimização em produção e a solução de problemas.

> **Note**
>
> O servidor MCP do Amazon EKS está em versão de pré-visualização para o Amazon EKS e está sujeito a alterações.

Visão geral

O servidor MCP do EKS pode ser facilmente integrado a qualquer assistente de codificação por IA compatível com MCP para aprimorar o fluxo de trabalho de desenvolvimento, como o Kiro, a CLI do Amazon Q Developer ou ferramentas de terceiros como Cursor ou Cline. Durante os conceitos básicos, o servidor MCP do Amazon EKS guia você na criação do cluster, realizando automaticamente o provisionamento de pré-requisitos e a aplicação das práticas recomendadas da AWS. Durante o desenvolvimento, o servidor simplifica as operações do EKS e do Kubernetes ao fornecer fluxos de trabalho de alto nível para a implantação de aplicações e para o gerenciamento de clusters. Para a depuração e solução de problemas, o servidor acelera a resolução de falhas por meio dos auxílios integrados de solução de problemas e do acesso à base de conhecimento disponível no console do EKS e em seus assistentes de IA favoritos. Essas funcionalidades estão acessíveis por meio de interações em linguagem natural, possibilitando a execução de operações complexas do Kubernetes de maneira mais intuitiva e eficaz.

Hospedado na Nuvem AWS, o servidor MCP do EKS, que é totalmente gerenciado, dispensa a necessidade de instalação e de manutenção local. O serviço fornece funcionalidades de nível empresarial, incluindo atualizações e aplicação de patches automáticas, segurança centralizada por meio da integração ao AWS IAM, registros em log de auditoria detalhados usando o AWS CloudTrail, além da escalabilidade, confiabilidade e suporte comprovados da AWS. O servidor MCP do EKS totalmente gerenciado e hospedado na Nuvem AWS oferece os seguintes benefícios principais:

- Elimina a instalação e a manutenção. Com a hospedagem do servidor MCP do EKS na Nuvem AWS, não é mais necessário gerenciar atualizações de versão nem solucionar problemas relacionados ao servidor local. Basta configurar o assistente de IA para se conectar ao endpoint do servidor MCP do EKS hospedado e você estará com tudo pronto para começar a trabalhar com os clusters do EKS.

- Realiza o gerenciamento de acesso centralizado. O servidor MCP do EKS integra-se ao IAM, fornecendo uma maneira centralizada e segura de controlar o acesso ao servidor. Todas as solicitações são assinadas usando a SigV4 da AWS por meio de um proxy leve, o que possibilita a integração direta com as políticas do IAM e credenciais da AWS já existentes.
- Fornece monitoramento e visibilidade aprimorados. A integração com o AWS CloudTrail captura inicializações e chamadas de ferramentas de acesso total realizadas por meio do serviço hospedado, permitindo trilhas de auditoria e relatórios de conformidade detalhados.
- Está sempre atualizado. Obtenha novas ferramentas, funcionalidades e correções de erros de forma automática, dispensando a atualização de instalações locais. O serviço hospedado é aprimorado continuamente com base no seu feedback e nas práticas recomendadas da AWS.

Exemplos de integração

O servidor MCP do EKS fornece diversas ferramentas que você pode usar para:

- Gerenciar o cluster: criar, configurar e gerenciar clusters do EKS com as práticas recomendadas automatizadas.
- Gerenciar recursos do Kubernetes: implantar aplicações, gerenciar recursos do Kubernetes e inspecionar o estado do cluster.
- Solucionar problemas do cluster: diagnosticar problemas usando ferramentas de solução de problemas integradas e uma base de conhecimento de runbooks.
- Consultar documentação: pesquisar e recuperar documentação relevante do EKS de forma contextual.

Análise dos clusters

```
Show me all EKS clusters and their status
What insights does EKS have about my production-cluster?
Show me the VPC configuration for my staging cluster
```

Verificação de recursos do Kubernetes

```
List all deployments in the production namespace
Show me pods that are not in Running state
Get the logs from the api-server pod in the last 30 minutes
```

Solução de problemas do

```
Why is my nginx-ingress-controller pod failing to start?
Search the EKS troubleshooting guide for pod networking issues
Show me events related to the failed deployment in the staging namespace
```

Criação de recursos (se o modo “escrita” estiver habilitado)

```
Create a new EKS cluster named demo-cluster with VPC and Auto Mode
Deploy my containerized app from ECR to the production namespace with 3 replicas
Generate a Kubernetes deployment YAML for my Node.js app running on port 3000
```

Conceitos básicos

Para começar, consulte o the section called “Conceitos básicos”.

Conceitos básicos do servidor MCP do Amazon EKS

Este guia apresenta as etapas de configuração e de uso do servidor MCP do EKS com assistentes de código de IA. Você aprenderá a configurar o ambiente, estabelecer conexão com o servidor MCP e começar a gerenciar os clusters do EKS por meio de interações em linguagem natural.

> **Note**
>
> O servidor MCP do Amazon EKS está em versão de pré-visualização para o Amazon EKS e está sujeito a alterações.

Pré-requisitos

Antes de começar, certifique-se de ter realizado as seguintes tarefas:

- Criar uma conta da AWS com acesso ao Amazon EKS
- Instalar e configurar a AWS CLI com credenciais
- Instalar o Python na versão 3.10 ou em versões posteriores
- Instalar o uv

Configuração

1. Verifique os pré-requisitos do

```
# Check that your Python version is 3.10 or higher
python3 --version

# Check uv installation
uv --version

# Verify CLI configuration
aws configure list
```

2. Configuração de permissões do IAM

Para se conectar ao servidor MCP do EKS, o perfil do IAM deve ter as seguintes políticas anexadas: **`eks-mcp:InvokeMcp`** (permissões necessárias para inicialização e recuperação de informações sobre as ferramentas disponíveis), **`eks-mcp:CallReadOnlyTool`** (permissões necessárias para o uso de ferramentas destinadas somente à leitura) e **`eks-mcp:CallPrivilegedTool`** (permissões necessárias para o uso de ferramentas de acesso total [gravação]). As permissões `eks-mcp` mencionadas fazem parte das políticas gerenciadas pela AWS de acesso destinadas somente à leitura e de acesso total disponibilizadas abaixo.

- Abra o console do IAM.
- No painel de navegação à esquerda, escolha Usuários, Grupos de usuários ou Perfis, dependendo da identidade à qual você deseja anexar a política e, em seguida, o nome do usuário, grupo ou perfil específico.
- Escolha a aba Permissões.
- Escolha Anexar políticas (ou Adicionar permissões, se for a primeira vez).
- Na lista de políticas, pesquise e selecione a política gerenciada que você deseja anexar:
- Operações destinadas somente à leitura: AmazonEKSMCPReadOnlyAccess
- Escolha Anexar políticas (ou Próximo e, em seguida, Adicionar permissões para confirmar).

Ao fazer isso, a política é anexada e as permissões entram em vigor imediatamente. Você pode anexar várias políticas à mesma identidade, e cada uma pode conter diversas permissões. Para saber mais sobre essas políticas, consulte Políticas gerenciadas pela AWS para o Amazon Elastic Kubernetes Service.

3. Escolha de um assistente de IA

Escolha um dos seguintes assistentes de IA compatíveis com o MCP ou qualquer ferramenta compatível com o MCP:

- Instalar a CLI do Amazon Q Developer
- Instalar o Kiro
- Instalar o Cursor
- Instalar a extensão Cline do VS Code

Etapa 1: configuração do assistente de IA

Escolha uma das opções apresentadas a seguir para configurar o assistente de código de IA. A conclusão desta etapa configura o assistente de código de IA para usar o Proxy do MCP para AWS, que é necessário para o acesso seguro e autenticado ao servidor MCP do Amazon EKS. Isso envolve adicionar ou editar o arquivo de configuração do MCP (por exemplo, `~/.aws/amazonq/mcp.json` para a CLI do Amazon Q Developer). O proxy atua como uma ponte no lado do cliente, gerenciando a autenticação SigV4 da AWS com as credenciais locais da AWS e viabilizando a descoberta dinâmica de ferramentas para interagir com servidores MCP de backend da AWS, como o servidor MCP do EKS. Para saber mais, consulte o repositório MCP Proxy for AWS.

Opção A: CLI do Amazon Q Developer

A CLI do Q Developer oferece a experiência mais integrada com o servidor MCP do EKS.

1. Localize o arquivo de configuração do MCP

- macOS/Linux: `~/.aws/q/mcp.json`
- Windows: `%USERPROFILE%\.aws\q\mcp.json`

2. Adicione a configuração do servidor MCP

Crie o arquivo de configuração, se ele ainda não existir. Certifique-se de substituir o marcador de região (`{region}`) pela região desejada.

Para Mac/Linux:

```
{
  "mcpServers": {
```

```
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Para Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Aviso de segurança: `--read-only` para restringir o uso somente a ferramentas de leitura.

3. Verifique a configuração

Reinicie a CLI do Amazon Q Developer e verifique as ferramentas disponíveis:

```
q /tools
```

Opção B: IDE do Kiro

O Kiro é um espaço de trabalho de codificação focado em IA com suporte nativo ao MCP.

1. Abra as configurações do Kiro

- Abra o Kiro
- Acesse Kiro → Settings e pesquise por “MCP Config”
- Como alternativa, use o atalho `Cmd+Shift+P,` (Mac) ou `Ctrl+Shift+P,` (Windows/Linux) e pesquise por “MCP Config”

2. Adicione a configuração do servidor MCP

- Clique em “Open Workspace MCP Config” ou “Open User MCP Config” para realizar a edição direta do arquivo de configuração.

Certifique-se de substituir o marcador de região (`{region}`) pela região desejada.

Para Mac/Linux:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
```

```
        "{region}"
      ]
    }
  }
}
```

Para Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Aviso de segurança: `--read-only` para restringir o uso somente a ferramentas de leitura.

Opção C: IDE do Cursor

O Cursor oferece suporte nativo ao MCP com uma interface gráfica de configuração.

1. Abra as configurações do Cursor

- Abra o Cursor
- Acesse Settings → Cursor Settings → Tools & MCP
- Como alternativa, use o atalho `Cmd+Shift+P` (Mac) ou `Ctrl+Shift+P` (Windows) e pesquise por “MCP”

2. Adicione a configuração do servidor MCP

- Clique em “New MCP Server”

Crie o arquivo de configuração, se ele ainda não existir. Certifique-se de substituir o marcador de região (`{region}`) pela região desejada.

Para Mac/Linux:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Para Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
```

```
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Aviso de segurança: `--read-only` para restringir o uso somente a ferramentas de leitura.

3. Reinicie o Cursor

Encerre e abra novamente o Cursor para que as alterações entrem em vigor.

4. Verifique o chat do Cursor

Acesse o painel de chat e teste o seguinte comando:

```
What EKS MCP tools are available?
```

O assistente deve exibir a lista de ferramentas de gerenciamento do EKS disponíveis.

Opção D: Cline (extensão do VS Code)

O Cline é uma extensão muito utilizada no VS Code para integrar assistência de IA ao seu editor.

1. Abra as configurações do Cline

- Abra o Cline
- Use o atalho `Cmd+Shift+P` (Mac) ou `Ctrl+Shift+P` (Windows) e pesquise por “MCP”

2. Adicione a configuração do servidor MCP

- Selecione “Add Server”
- Escolha “Open User Configuration”

Crie o arquivo de configuração, se ele ainda não existir. Certifique-se de substituir o marcador de região (`{region}`) pela região desejada.

Para Mac/Linux:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Para Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
```

```
  }
}
```

Aviso de segurança: `--read-only` para restringir o uso somente a ferramentas de leitura.

2. Recarregue o VS Code

Use o atalho `Cmd+Shift+P` ou `Ctrl+Shift+P` e selecione “Developer: Reload Window”

3. Verificar a configuração

Abra o Cline e pergunte:

```
List the available MCP tools for EKS
```

Etapa 2: (Opcional) criação de uma política de “gravação”

Se desejar, você pode criar uma política do IAM gerenciada pelo cliente para liberar acesso total ao servidor MCP do Amazon EKS. Esta política concede permissões para usar todas as ferramentas no servidor MCP do EKS, incluindo tanto ferramentas privilegiadas, que podem envolver operações de gravação, quanto ferramentas destinadas somente à leitura. Lembre-se de que permissões de alto risco, como ações Delete* ou o acesso irrestrito a recursos do IAM, fazem parte desta política, visto que são indispensáveis para a criação ou exclusão de recursos de cluster usando a ferramenta manage_eks_stacks.

```
aws iam create-policy \
 --policy-name EKSMcpWriteManagementPolicy \
 --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect
\": \"Allow\", \"Action\": [\"eks:DescribeCluster\", \"eks:ListClusters\",
 \"eks:DescribeNodegroup\", \"eks:ListNodegroups\", \"eks:DescribeAddon\",
 \"eks:ListAddons\", \"eks:DescribeAccessEntry\", \"eks:ListAccessEntries\",
 \"eks:DescribeInsight\", \"eks:ListInsights\", \"eks:AccessKubernetesApi\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"eks:CreateCluster
\", \"eks:DeleteCluster\", \"eks:CreateAccessEntry\", \"eks:TagResource\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"iam:GetRole\",
 \"iam:ListRolePolicies\", \"iam:ListAttachedRolePolicies\", \"iam:GetRolePolicy\",
 \"iam:GetPolicy\", \"iam:GetPolicyVersion\"], \"Resource\": \"*\"}, {\"Effect\":
 \"Allow\", \"Action\": [\"iam:TagRole\", \"iam:CreateRole\", \"iam:AttachRolePolicy
\", \"iam:PutRolePolicy\", \"iam:DetachRolePolicy\", \"iam:DeleteRole\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"iam:PassRole\"],
 \"Resource\": \"*\", \"Condition\": {\"StringEquals\": {\"iam:PassedToService
\": [\"eks.amazonaws.com\", \"ec2.amazonaws.com\"]}}}, {\"Effect\": \"Allow\",
```

```
 \"Action\": [\"ec2:CreateVpc\", \"ec2:CreateSubnet\", \"ec2:CreateRouteTable\",
 \"ec2:CreateRoute\", \"ec2:CreateInternetGateway\", \"ec2:CreateNatGateway\",
 \"ec2:CreateSecurityGroup\", \"ec2:AttachInternetGateway\", \"ec2:AssociateRouteTable
\", \"ec2:ModifyVpcAttribute\", \"ec2:ModifySubnetAttribute\", \"ec2:AllocateAddress
\", \"ec2:CreateTags\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action
\": [\"ec2:DeleteVpc\", \"ec2:DeleteSubnet\", \"ec2:DisassociateRouteTable\",
 \"ec2:DeleteRouteTable\", \"ec2:DeleteRoute\", \"ec2:DetachInternetGateway\",
 \"ec2:DeleteInternetGateway\", \"ec2:DeleteNatGateway\", \"ec2:ReleaseAddress\",
 \"ec2:DeleteSecurityGroup\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action
\": [\"ec2:DescribeVpcs\", \"ec2:DescribeSubnets\", \"ec2:DescribeRouteTables\",
 \"ec2:DescribeInternetGateways\", \"ec2:DescribeNatGateways\", \"ec2:DescribeAddresses
\", \"ec2:DescribeSecurityGroups\", \"ec2:DescribeAvailabilityZones\"], \"Resource
\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"cloudformation:CreateStack
\", \"cloudformation:UpdateStack\", \"cloudformation:DeleteStack\",
 \"cloudformation:DescribeStacks\", \"cloudformation:TagResource\"], \"Resource
\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"sts:GetCallerIdentity\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"logs:StartQuery\",
 \"logs:GetQueryResults\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\":
 [\"cloudwatch:GetMetricData\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action
\": [\"eks-mcp:*\"], \"Resource\": \"*\"}]}"
```

Etapa 3: verificação da configuração

Testar conexão

Faça uma pergunta simples ao seu assistente de IA para verificar a conexão:

```
List all EKS clusters in my {aws} account
```

O assistente deve exibir a lista dos clusters do EKS.

Etapa 4: execução das primeiras tarefas

Exemplo 1: conhecer seus clusters

```
Show me all EKS clusters and their status
What insights does EKS have about my production-cluster?
Show me the VPC configuration for my staging cluster
```

Exemplo 2: verificar os recursos do Kubernetes

```
Get the details of all the kubernetes resources deployed in my EKS cluster
```

```
Show me pods that are not in Running state or pods with any restarts
Get the logs from the aws-node daemonset in the last 30 minutes
```

Exemplo 3: solucionar problemas

```
Why is my nginx-ingress-controller pod failing to start?
Search the EKS troubleshooting guide for pod networking issues
Show me events related to the failed deployment in the staging namespace
```

Exemplo 4: criar recursos (se o modo “gravação” estiver habilitado)

```
Create a new EKS cluster named demo-cluster with VPC and Auto Mode
Deploy my containerized app from ECR to the production namespace with 3 replicas
Generate a Kubernetes deployment YAML for my Node.js app running on port 3000
```

Configurações comuns

Cenário 1: vários perfis da AWS

Se você trabalha com várias contas da AWS, crie configurações do servidor MCP separadas.

Para Mac/Linux:

```
{
  "mcpServers": {
    "eks-mcp-prod": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2"
      ]
    },
    "eks-mcp-dev": {
      "disabled": false,
```

```
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

Para Windows:

```
{
  "mcpServers": {
    "eks-mcp-prod": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2"
      ]
    },
    "eks-mcp-dev": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
```

```
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

Cenário 2: somente leitura para ambientes de produção

Crie uma configuração destinada somente à leitura para ambientes de produção.

Para Mac/Linux:

```
{
  "mcpServers": {
    "eks-mcp-prod-readonly": {
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2",
        "--read-only"
      ],
      "autoApprove": [
        "list_k8s_resources",
        "get_pod_logs",
        "get_k8s_events"
      ]
    }
  }
}
```

Para Windows:

```
{
  "mcpServers": {
    "eks-mcp-prod-readonly": {
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2",
        "--read-only"
      ],
      "autoApprove": [
        "list_k8s_resources",
        "get_pod_logs",
        "get_k8s_events"
      ]
    }
  }
}
```

Cenário 3: desenvolvimento com acesso total

Para ambientes de desenvolvimento com acesso total à gravação.

Para Mac/Linux:

```
{
  "mcpServers": {
    "eks-mcp-dev-full": {
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
```

```
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

Para Windows:

```
{
  "mcpServers": {
    "eks-mcp-dev-full": {
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

Considerações

Segurança

Não transmita segredos ou informações confidenciais por meio de mecanismos de entrada permitidos:

- Não inclua segredos ou credenciais em arquivos YAML que são aplicados com apply_yaml.
- Não forneça informações confidenciais diretamente nos prompts enviados ao modelo.
- Não inclua segredos em modelos do CloudFormation ou em manifestos de aplicações.

- Evite o uso de ferramentas do servidor MCP para a criação de Kubernetes Secrets, uma vez que isso demandaria o envio de dados sigilosos ao modelo.
- Evite o registro de dados sensíveis nos logs das aplicações que são executadas em pods do Kubernetes.

Segurança do conteúdo em YAML:

- Use somente arquivos YAML originados de fontes seguras.
- O servidor MCP utiliza a validação nativa da API do Kubernetes para processar o conteúdo YAML, sem contar com um mecanismo de validação próprio.
- Audite os arquivos YAML antes de aplicá-los ao cluster.

Alternativas ao envio de segredos pelo MCP:

- Use o AWS Secrets Manager ou o Parameter Store para armazenar informações sensíveis.
- Configure o RBAC do Kubernetes adequadamente para as contas de serviço.
- Adote perfis do IAM para contas de serviço (IRSA, na sigla em inglês) para permitir que os pods acessem serviços da AWS com segurança.

A seguir

Para obter uma lista completa de ferramentas e configurações, consulte the section called “Ferramentas”.

Referência de configuração da ferramenta de servidor MCP do Amazon EKS

Este guia apresenta todas as configurações disponíveis para a ferramenta mcp-proxy-for-aws do lado do cliente, que permite o estabelecimento de conexão com o servidor MCP do Amazon EKS totalmente gerenciado por parte do IDE.

> **Note**
>
> O servidor MCP do Amazon EKS está em versão de pré-visualização para o Amazon EKS e está sujeito a alterações.

Exemplo

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.us-west-2.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "us-east-1",
        "--read-only"
      ]
    }
  }
}
```

permissões do IAM

O perfil usado para se conectar ao servidor MCP requer a permissão **`eks-mcp:InvokeMcp`** para inicialização e para a obtenção de informações sobre as ferramentas disponíveis. A permissão **`eks-mcp:CallReadOnlyTool`** é necessária para o uso de ferramentas destinadas somente à leitura, e a permissão **`eks-mcp:CallPrivilegedTool`** é necessária para o uso de ferramentas com acesso completo (gravação).

Variáveis de ambiente

`AWS_PROFILE`: (opcional) nome do perfil de credenciais da AWS a ser usado. Pode ser substituído pelo argumento de linha de comandos `--profile`.

- Exemplo: `export AWS_PROFILE=production`

`AWS_REGION`: (opcional) região da AWS para assinatura SigV4. O padrão é `us-west-2`, se a região não estiver definida.

- Exemplo: `export AWS_REGION=us-east-1`

Argumentos

URL do endpoint do MCP para a assinatura SigV4: (obrigatório) o URL do endpoint do MCP para conexão.

- Exemplo: `https://eks-mcp.us-west-2.api.aws/mcp`

`--service`: (opcional) nome do serviço da AWS para a assinatura SigV4. É detectado automaticamente com base no nome do host do endpoint, se não for fornecido.

- Exemplo: `--service eks-mcp`

`--profile`: (opcional) perfil de credenciais da AWS a ser usado. O padrão será a variável de ambiente `AWS_PROFILE`, se não for especificado.

- Exemplo: `--profile production`

`--region`: região da AWS a ser usada. Usa a variável de ambiente `AWS_REGION`; se não for definida, o padrão será `us-east-1`.

- Exemplo: `--region us-west-2`

`--read-only`: (opcional) desabilita as ferramentas que podem requerer permissões de gravação (ferramentas que NÃO requerem permissão de gravação estão anotadas com “readOnlyHint=true”). Por padrão, todas as ferramentas são habilitadas.

- Exemplo: `--read-only`

Para obter mais opções de configuração, consulte Parâmetros de configuração.

Ferramentas

O servidor disponibiliza as ferramentas do MCP apresentadas a seguir.

Ferramentas somente leitura

Esta seção descreve as ferramentas destinadas somente à leitura disponíveis para o servidor MCP do EKS. Lembre-se de que todas as operações de leitura da API do Kubernetes podem acessar tanto:

- Clusters privados (consulte Endpoint privado do cluster)
- Clusters públicos

search_eks_documentation Realize pesquisas na documentação do EKS para obter informações e orientações atualizadas. Esta ferramenta fornece acesso à documentação mais recente do EKS, incluindo os novos recursos e as últimas melhorias que podem ser desconhecidos pelos agentes.

Parâmetros:

- query (obrigatório): sua dúvida específica ou consulta de pesquisa relacionada à documentação, aos recursos ou às práticas recomendadas do EKS.
- limit (opcional): quantidade máxima de resultados da documentação que devem ser retornados (1 a 10). Padrão: 5.

search_eks_troubleshooting_guide: pesquisa no guia de solução de problemas do EKS para encontrar informações de depuração com base em uma consulta. Auxilia na identificação de problemas recorrentes e oferece instruções detalhadas para resolução.

Parâmetros:

- query (obrigatório): sua dúvida específica ou descrição do problema relacionada à solução de problemas do EKS.

describe_eks_resource: recupera informações detalhadas sobre um recurso específico do cluster do EKS, incluindo a configuração, o status e os metadados.

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS (obrigatório para recursos com escopo de cluster).
- resource_type (obrigatório): o tipo de recurso do EKS que será descrito. Valores válidos:
- accessentry (requer cluster_name e resource_name como principalArn).

- addon (requer cluster_name e resource_name como nome do addon).
- cluster (requer cluster_name), nodegroup (requer cluster_name e resource_name como nome do nodegroup).
- resource_name (opcional): nome do recurso específico que será descrito (obrigatório para a maioria dos tipos de recurso).

list_eks_resources: lista os recursos do EKS de um determinado tipo, fornecendo um resumo de todos os recursos do tipo especificado que estão acessíveis.

Parâmetros:

- resource_type (obrigatório): o tipo de recurso do EKS que será listado. Valores válidos:
- accessentry (requer cluster_name).
- addon (requer cluster_name).
- cluster (nenhum parâmetro adicional é requerido).
- nodegroup (requer cluster_name).
- cluster_name (opcional): nome do cluster do EKS (obrigatório para recursos com escopo de cluster).

get_eks_insights: obtém insights e recomendações do cluster do EKS voltados para a otimização. Fornece insights acionáveis para segurança, performance e otimização de custos com base nas práticas recomendadas da AWS e na análise do cluster.

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS.
- category (opcional): categoria opcional para filtrar os insights (por exemplo, “MISCONFIGURATION” ou “UPGRADE_READINESS”).
- insight_id (opcional): ID opcional de um insight específico sobre o qual se deseja obter informações.
- next_token (opcional): token opcional de paginação para acessar o próximo conjunto de resultados.

get_eks_vpc_config: obtém as configurações da VPC de um cluster do EKS, abrangendo sub-redes, tabelas de rotas e conectividade de rede.

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS para o qual se deseja obter a configuração da VPC.
- vpc_id (opcional): ID da VPC específica para consulta (como é opcional, caso não seja informado, será utilizada a VPC do cluster).

get_k8s_events: obtém eventos do Kubernetes relacionados a recursos específicos para solução de problemas e monitoramento.

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS em que o recurso está localizado.
- kind (obrigatório): tipo do objeto envolvido (por exemplo, “Pod”, “Deployment” e “Service”). Deve corresponder exatamente ao tipo do recurso.
- name (obrigatório): nome do objeto envolvido do qual se deseja obter os eventos.
- namespace (opcional): namespace do objeto envolvido. Obrigatório para recursos vinculados a um namespace (como Pods e Deployments). Não é necessário para recursos de escopo de cluster (como Nodes e PersistentVolumes).

get_pod_logs Recupera logs de pods em um cluster do EKS com opções de filtragem.

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS em que o pod está sendo executado.
- namespace (obrigatório): namespace do Kubernetes em que o pod está localizado.
- pod_name (obrigatório): nome do pod do qual se deseja recuperar os logs.
- container_name (opcional): nome do contêiner específico do qual se deseja obter os logs. Obrigatório apenas se o pod contiver vários contêineres.
- limit_bytes (opcional): número máximo de bytes a serem retornados. Padrão: 10 KB (10.240 bytes).
- previous (opcional): retorna os logs do contêiner anterior que foi encerrado (o padrão é “false”). Útil para obter logs de pods que estão sendo reiniciados.
- since_seconds (opcional): retorna somente os logs mais recentes do que a quantidade de segundos definida. Útil para obter logs recentes sem recuperar todo o histórico.

- tail_lines (opcional): número de linhas a serem retornadas do final dos logs. Padrão: 100.

list_api_versions Lista todas as versões de API disponíveis no cluster do Kubernetes especificado.

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS.

list_k8s_resources Lista recursos do Kubernetes de um tipo específico em um cluster do EKS.

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS em que os recursos estão localizados.
- kind (obrigatório): o tipo dos recursos do Kubernetes a serem listados (por exemplo, “Pod”, “Service” e “Deployment”). Use a ferramenta `list_api_versions` para localizar os tipos de recursos disponíveis.
- api_version (obrigatório): versão da API dos recursos do Kubernetes (por exemplo, “v1”, “apps/v1” e “networking.k8s.io/v1”). Use a ferramenta `list_api_versions` para localizar as versões da API disponíveis.
- field_selector (opcional): seletor de campos para filtrar recursos (por exemplo, “metadata.name=mypod,status.phase=Running”). Usa a mesma sintaxe da sinalização “--field-selector” do kubectl.
- label_selector (opcional): seletor de rótulos para a filtragem de recursos (por exemplo, “app=nginx,tier=frontend”). Usa a mesma sintaxe da sinalização `--selector` do kubectl.
- namespace (opcional): namespace dos recursos do Kubernetes a serem listados. Se não for informado, os recursos serão listados em todos os namespaces (para recursos vinculados a um namespace).

read_k8s_resource Obtém informações detalhadas sobre um recurso específico do Kubernetes em um cluster do EKS.

Parâmetros:

- api_version (obrigatório): versão da API do recurso do Kubernetes (por exemplo, “v1”, “apps/v1” e “networking.k8s.io/v1”).
- cluster_name (obrigatório): nome do cluster do EKS em que o recurso está localizado.

- kind (obrigatório): tipo do recurso do Kubernetes (por exemplo, “Pod”, “Service” e “Deployment”).
- name (obrigatório): nome do recurso do Kubernetes destinado à leitura.
- namespace (opcional): namespace do recurso do Kubernetes. Obrigatório para recursos vinculados a um namespace. Não é necessário para recursos de escopo de cluster (como Nodes e PersistentVolumes).

generate_app_manifest Gera manifestos padronizados de “deployment” e “service” do Kubernetes para aplicações conteinerizadas.

Parâmetros:

- app_name (obrigatório): nome da aplicação. Usado para nomes de implantação e de serviços, bem como para rótulos.
- image_uri (obrigatório): URI completo da imagem no ECR com uma etiqueta (por exemplo, “123456789012.dkr.ecr.region.amazonaws.com/repo:tag”). É necessário informar o caminho completo do repositório seguido da etiqueta.
- load_balancer_scheme (opcional): tipo de esquema do AWS Load Balancer. Valores válidos:
- “interno” (somente para VPC privada)
- “voltado para a internet” (acesso público)
- Padrão: “interno”.
- cpu (opcional): solicitação de CPU para cada contêiner (por exemplo, “100 m” para 0,1 núcleos de CPU, “500 m” para meio núcleo). Padrão: “100 m”.
- memory (opcional): solicitação de memória para cada contêiner (por exemplo, “128 Mi” para 128 MiB, “1 Gi” para 1 GiB). Padrão: “128 Mi”.
- namespace (opcional): namespace do Kubernetes no qual a aplicação será implantada. Padrão: “padrão”.
- port (opcional): porta do contêiner configurada para receber o tráfego da aplicação. Padrão: 80
- replicas (opcional): número de réplicas a serem implantadas. Padrão: 2

get_cloudwatch_logs Realiza consultas nos logs do CloudWatch com filtros personalizados baseados nos parâmetros informados e suporte aos grupos de logs padrão para a observabilidade do cluster do EKS.

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS em que o recurso está localizado. Usado para criar o nome do grupo de logs do CloudWatch.
- resource_type (obrigatório): tipo de recurso para o qual se deseja pesquisar por logs. Valores válidos: “pod”, “node”, “container” ou “cluster”. Determina o critério de filtragem dos logs.
- log_type (obrigatório): tipo de log destinado à consulta. Valores válidos:
- “application”: logs de aplicação e de contêiner.
- “host”: logs do sistema no nível do nó.
- performance”: logs de métricas de performance.
- “control-plane”: logs do ambiente de gerenciamento do EKS.
- “your-log-group-name”: informe diretamente um nome personalizado para o grupo de logs do CloudWatch.
- resource_name (opcional): nome do recurso que você deseja pesquisar nos registros de log (por exemplo, o nome do pod, do nó ou do contêiner). Usado para filtrar os logs referentes a esse recurso específico.
- minutes (opcional): quantidade de minutos retroativos para a busca de logs. Padrão: 15. Este parâmetro será desconsiderado caso o “start_time” seja informado. Use valores baixos para investigar problemas atuais e valores mais altos para realizar uma análise do histórico.
- start_time (opcional): horário de início no formato ISO (por exemplo, “2023-01-01T00:00:00Z”). Se for fornecido, substitui o parâmetro “minutes”.
- end_time (opcional): horário de término no formato ISO (por exemplo, “2023-01-01T01:00:00Z”). Se não for informado, o padrão é o horário atual.
- fields (opcional): campos personalizados para incluir nos resultados da consulta (o padrão é “@timestamp, @message”). Use a sintaxe de campos do CloudWatch Logs Insights.
- filter_pattern (opcional): padrão de filtragem adicional do CloudWatch Logs a ser aplicado. Usa a sintaxe do CloudWatch Logs Insights (por exemplo, “ERROR” e “field=value”).
- limit (opcional): número máximo de entradas de log a serem retornadas. Use valores baixos (10 a 50) para consultas mais rápidas e valores altos (100 a 1.000) para resultados mais abrangentes. Valores mais altos podem afetar o desempenho.

get_cloudwatch_metrics Obtém métricas e pontos de dados do CloudWatch para monitoramento e análise da performance de clusters do EKS. Gerencia métricas do Container Insights e métricas personalizadas, além de períodos de tempo e dimensões configuráveis.

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS para o qual as métricas serão obtidas.
- dimensions (obrigatório): dimensões a serem utilizadas na consulta de métricas do CloudWatch, fornecidas como uma string JSON. Deve incluir as dimensões apropriadas para o tipo de recurso e a métrica (por exemplo, “{“ClusterName”: “my-cluster”, “PodName”: “my-pod”, “Namespace”: “default”}”).
- metric_name (obrigatório): nome da métrica a ser recuperada. Exemplos comuns:
- `cpu_usage_total`: uso total de CPU
- `memory_rss`: uso de memória “Resident Set Size” (RSS)
- `network_rx_bytes`: bytes de rede recebidos
- `network_tx_bytes`: bytes de rede transmitidos
- namespace (obrigatório): namespace do CloudWatch em que a métrica está armazenada. Valores comuns:
- `"ContainerInsights"`: para métricas de contêiner
- `"AWS/EC2"`: para métricas de instâncias do EC2
- `"AWS/EKS"`: para métricas do ambiente de gerenciamento do EKS
- minutes (opcional): quantidade de minutos retroativos para a busca de métricas. Padrão: 15. Este parâmetro será desconsiderado caso o “start_time” seja informado.
- start_time (opcional): horário de início no formato ISO (por exemplo, “2023-01-01T00:00:00Z”). Se for fornecido, substitui o parâmetro “minutes”.
- end_time (opcional): horário de término no formato ISO (por exemplo, “2023-01-01T01:00:00Z”). Se não for informado, o padrão é o horário atual.
- limit (opcional): número máximo de pontos de dados a serem retornados. Valores mais altos (100 a 1.000) oferecem maior detalhamento dos dados, mas podem afetar a performance da consulta. Padrão: 50.
- period (opcional): período em segundos para os pontos de dados da métrica. Valor padrão: 60 (1 minuto). Valores baixos (1 a 60) oferecem alta resolução, embora possam ter menor disponibilidade.
- stat (opcional): estatística a ser usada para a agregação da métrica. Padrão: “Average”. Valores válidos:
- `Average`: valor médio durante o período
- `Sum:`: valor total durante o período
- `Maximum`: valor máximo atingido no período

- `Minimum`: valor mínimo atingido no período
- `SampleCount`: quantidade de amostras coletadas no período

get_eks_metrics_guidance Obtém orientações sobre métricas do CloudWatch para tipos de recursos específicos em clusters do EKS. Útil para o agente ao determinar as dimensões corretas a serem usadas com a ferramenta “get_cloudwatch_metrics”.

Parâmetros:

- resource_type (obrigatório): tipo de recurso para o qual se deseja obter orientações relacionadas às métricas (“cluster”, “node”, “pod”, “namespace” e “service”).

get_policies_for_role Recupera todas as políticas anexadas a um perfil do IAM especificado, incluindo a política de assumir perfil, as políticas gerenciadas e as políticas em linha.

Parâmetros:

- role_name (obrigatório): nome do perfil do IAM para o qual se deseja obter as políticas. O perfil deve existir em sua conta da AWS.

Ferramentas de acesso total (gravação)

Esta seção descreve as ferramentas destinadas somente à leitura disponíveis para o servidor MCP do EKS. Lembre-se de que, até o momento, todas as operações de gravação da API do Kubernetes podem acessar apenas:

- Clusters públicos (endpointPublicAccess=true)

manage_k8s_resource Gerencia um único recurso do Kubernetes por meio de operações de gravação (criar, atualizar, aplicar patches ou excluir).

Parâmetros:

- operation (obrigatório): operação a ser realizada no recurso. Valores válidos:
- `create:`: cria um novo recurso
- `replace`: substitui um recurso existente
- `patch`: atualiza campos específicos de um recurso existente

- `delete`: exclui um recurso existente
- Observação: use “read_k8s_resource” para a consulta de recursos e “list_k8s_resources” para obter uma lista com vários recursos de uma vez.
- cluster_name (obrigatório): nome do cluster do EKS em que o recurso está localizado ou será criado.
- kind (obrigatório): tipo do recurso do Kubernetes (por exemplo, “Pod”, “Service” e “Deployment”).
- api_version (obrigatório): versão da API do recurso do Kubernetes (por exemplo, “v1”, “apps/v1” e “networking.k8s.io/v1”).
- body (opcional): definição do recurso como uma string JSON. Obrigatório para operações de criação, substituição e aplicação de patches. Para a criação e a substituição, deve ser usada uma definição completa do recurso. Para a aplicação de patches, deve conter apenas os campos a serem atualizados.
- name (opcional): nome do recurso do Kubernetes. Obrigatório em todas as operações, menos a operação de criação (visto que pode ser informado no próprio corpo).
- namespace (opcional): namespace do recurso do Kubernetes. Obrigatório para recursos vinculados a um namespace. Não é necessário para recursos de escopo de cluster (como Nodes e PersistentVolumes).

apply_yaml Aplica manifestos YAML do Kubernetes a um cluster do EKS.

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS em que os recursos serão criados ou atualizados.
- namespace (obrigatório): namespace do Kubernetes ao qual os recursos serão aplicados. Será usado para recursos vinculados a namespaces que não especifiquem um namespace próprio.
- yaml_content (obrigatório): conteúdo YAML a ser aplicado ao cluster. Pode conter vários documentos separados por “---”.
- force (opcional): indica se os recursos devem ser atualizados caso já existam (semelhante ao “kubectl apply”). Defina como “false” para apenas criar novos recursos.

manage_eks_stacks Gerencia pilhas do CloudFormation para o EKS com operações para gerar modelos, implantar, descrever e excluir clusters do EKS e a infraestrutura subjacente. A criação do cluster normalmente demora de 15 a 20 minutos para ser concluída. Para operações de implantação

e exclusão, a pilha deve ter sido criada por esta ferramenta (ou seja, marcada com a etiqueta “CreatedBy=EksMcpServer”).

Parâmetros:

- cluster_name (obrigatório): nome do cluster do EKS (para operações de geração, implantação, descrição e exclusão). Este nome será usado para derivar o nome da pilha do CloudFormation e será incorporado aos recursos do cluster.
- operation (obrigatório): operação a ser realizada. Valores válidos:
- `generate`: gerar um modelo do CloudFormation
- `deploy`: implantar uma pilha do CloudFormation (requer `template_content`)
- `describe`: descrever ou ler uma pilha do CloudFormation (somente leitura)
- `delete`: excluir uma pilha do CloudFormation
- template_content (opcional): conteúdo do modelo do CloudFormation (para operações de implantação). Deve ser o conteúdo completo do modelo em YAML ou JSON. Fornece suporte tanto a recursos únicos quanto a conteúdo YAML de vários documentos separados por “---”.

add_inline_policy Adiciona uma nova política em linha a um perfil do IAM.

Parâmetros:

- permissions (obrigatório): permissões a serem incluídas na política como strings JSON que representam declarações da política do IAM. Pode ser uma única string JSON ou uma matriz de strings JSON.
- policy_name (obrigatório): Nome da política em linha a ser criada. Deve ser exclusivo dentro do perfil.
- role_name (obrigatório): nome do perfil do IAM ao qual a política será adicionada. O perfil deve existir.

Uso do Amazon Q Developer no console do Amazon EKS

O Amazon Elastic Kubernetes Service (EKS) integra-se ao Amazon Q para fornecer soluções de problemas baseadas em IA diretamente no Console de Gerenciamento da AWS. Essa integração possibilita investigar e solucionar rapidamente problemas relacionados ao cluster, ao ambiente de gerenciamento, ao nó e à workload, com a assistência da IA do Amazon Q.

Como funciona

O console do Amazon EKS exibe os botões Inspecionar com o Amazon Q de maneira contextual, ao lado de erros ou de problemas em diversas partes do console. Ao clicar nesse botão, o Amazon Q analisa automaticamente o problema e abre um painel de chat no lado direito do console, apresentando os resultados da investigação, a análise da causa raiz e as etapas sugeridas para a mitigação.

A integração está disponível nos seguintes locais no console do Amazon EKS:

- Integridade do cluster: investigue problemas relacionados à integridade do cluster e mensagens de status na guia “Integridade do cluster” no painel de observabilidade
- Ambiente de gerenciamento: solucione erros e alertas relacionados aos componentes do ambiente de gerenciamento em “Monitoramento do ambiente de gerenciamento” no painel de observabilidade
- Insights do upgrade: analise possíveis bloqueadores de atualizações e problemas de compatibilidade em “Insights do upgrade” no painel de observabilidade
- Integridade de nó: investigue problemas em nível de nó que afetam a capacidade do cluster em “Problemas de integridade de nó” no painel de observabilidade
- Workloads: analise eventos do Kubernetes, que indiquem falhas ou problemas, em pods

Uso do Amazon Q para solução de problemas

Como investigar um problema com o Amazon Q

1. Abra o console do Amazon EKS.
2. Selecione o nome do cluster que deseja investigar.
3. Ao se deparar com uma mensagem de erro ou um indicador de problema, procure o botão Inspecionar com o Amazon Q. O botão é apresentado de forma contextual ao lado do problema ou na visualização dos detalhes do status de erro.
4. Escolha Inspecionar com o Amazon Q.
5. O Amazon Q realiza automaticamente a investigação do problema e apresenta a análise em um painel de chat à direita do console.
6. Confira os resultados da investigação, incluindo a análise da causa raiz e as etapas sugeridas para a mitigação.

7. É possível continuar a interação com o Amazon Q ao fazer perguntas adicionais sobre o problema.

Observação: a integração com o Amazon Q é exibida somente quando há um erro, aviso ou problema que precise ser investigado. Ela não é exibida quando os recursos se encontram em condições íntegras.

Considerações

Considere as seguintes informações ao usar o Amazon Q com o Amazon EKS:

- Operações somente leitura: a integração do Amazon Q executa somente operações de leitura nos recursos do cluster. Nenhuma ação de gravação ou de alteração na configuração do cluster ou nas workloads é realizada.
- Processamento entre regiões: o Amazon Q pode processar dados em diversas regiões da AWS para fornecer análises baseadas em IA. Para obter mais informações sobre o processamento entre regiões, consulte Processamento entre regiões no Guia do usuário do Amazon Q Developer.
- Exclusivo para uso no Console de Gerenciamento da AWS: esta integração está disponível apenas por meio do Console de Gerenciamento da AWS. Não é possível acessá-la pela AWS CLI, pelas APIs da AWS ou por ferramentas de infraestrutura como código.

Saiba mais

Para obter mais informações sobre como usar o Amazon Q, consulte Chat com o Amazon Q no Guia do usuário do Amazon Q Developer.

Versionamento no Amazon EKS

Esta seção foi criada para ajudar você a aprender como o versionamento do Kubernetes opera no Amazon Elastic Kubernetes Service (EKS). Você encontrará detalhes sobre as versões com suporte, nossas políticas de descontinuação e suporte, processos de atualização e as principais considerações para gerenciar as versões do seu cluster.

Tópicos

- Compreender o ciclo de vida da versão do Kubernetes no EKS
- Veja as versões da plataforma Amazon EKS para cada versão do Kubernetes

Compreender o ciclo de vida da versão do Kubernetes no EKS

O Kubernetes evolui rapidamente com novos recursos, atualizações de design e correções de bugs. A comunidade lança novas versões secundárias do Kubernetes (como `1.33`), em média, uma vez a cada quatro meses. O Amazon EKS segue o ciclo upstream de lançamento e descontinuação para versões secundárias. Conforme novas versões do Kubernetes são disponibilizadas no Amazon EKS, recomendamos que você atualize proativamente seus clusters para usarem a versão mais recente disponível.

Uma versão secundária terá suporte padrão no Amazon EKS durante os primeiros 14 meses após seu lançamento. Depois que uma versão excede a data de término do suporte padrão, ela entra no suporte estendido pelos 12 meses seguintes. O suporte estendido permite que você permaneça em uma versão específica do Kubernetes por mais tempo por um custo adicional por hora de cluster. Se você não atualizou seu cluster antes do término do período de suporte estendido, seu cluster é atualizado automaticamente para a versão estendida mais antiga atualmente suportada.

O suporte estendido está habilitado por padrão. Para desabilitar, consulte Desabilitação do suporte estendido do EKS.

Recomendamos que você crie o cluster com a versão mais recente disponível do Kubernetes e compatível com o Amazon EKS. Se a aplicação exigir uma versão específica do Kubernetes, você poderá selecionar versões mais antigas. Você pode criar novos clusters do Amazon EKS em qualquer versão oferecida no suporte padrão ou estendido.

Versões disponíveis no suporte padrão

Atualmente, as seguintes versões do Kubernetes estão disponíveis no suporte padrão do Amazon EKS:

- `1.34`
- `1.33`
- `1.32`

Para conhecer alterações importantes para cada versão no suporte padrão, consulte Suporte padrão a versões do Kubernetes.

Versões disponíveis com suporte estendido

Atualmente, as seguintes versões do Kubernetes estão disponíveis no suporte estendido do Amazon EKS:

- `1.31`
- `1.30`
- `1.29`

Para conhecer alterações importantes para cada versão no suporte estendido, consulte Suporte estendido a versões do Kubernetes.

Calendário de lançamento do Amazon EKS Kubernetes

A tabela a seguir mostra datas importantes de lançamento e suporte a serem consideradas para cada versão do Kubernetes. A cobrança do suporte estendido começa no início do dia em que a versão atinge o final do suporte padrão, no fuso horário UTC+0. As datas na tabela a seguir usam o fuso horário UTC+0.

> **Note**
>
> As datas com apenas mês e ano são aproximadas e são atualizadas com uma data exata quando ela é conhecida.

Para receber notificações de todas as alterações do arquivo de origem nesta página específica da documentação, você pode se inscrever no seguinte URL com um leitor de RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/clusters/kubernetes-versions.adoc.atom
```

Versão do Kubernetes	Versão upstream	Lançamento do Amazon EKS	Fim do suporte padrão	Fim do suporte estendido
1.34	27 de agosto de 2025	2.º de outubro de 2025	2 de dezembro de 2026	2 de dezembro de 2027
1.33	23 de abril de 2025	29 de maio de 2025	29 de julho de 2026	29 de julho de 2027
1.32	11 de dezembro de 2024	23 de janeiro de 2025	23 de março de 2026	23 de março de 2027
1.31	13 de agosto de 2024	26 de setembro de 2024	26 de novembro de 2025	26 de novembro de 2026
1.30	17 de abril de 2024	23 de maio de 2024	23 de julho de 2025	23 de julho de 2026
1.29	13 de dezembro de 2023	23 de janeiro de 2024	23 de março de 2025	23 de março de 2026

Obter informações sobre a versão com a AWS CLI

É possível usar a AWS CLI para obter informações sobre as versões do Kubernetes disponíveis no EKS, incluindo a data de término do suporte padrão.

Para recuperar informações sobre as versões disponíveis do Kubernetes no EKS usando a AWS CLI

1. Abra o terminal.
2. Verifique se a AWS CLI está instalada e configurada. Para obter mais informações, consulte Instalar ou atualizar a versão mais recente da CLI.

3. Execute o seguinte comando:

```
aws eks describe-cluster-versions
```

4. O comando retornará uma saída JSON com detalhes sobre as versões de cluster disponíveis. Veja abaixo um exemplo de saída:

```
{
    "clusterVersions": [
        {
            "clusterVersion": "1.31",
            "clusterType": "eks",
            "defaultPlatformVersion": "eks.21",
            "defaultVersion": true,
            "releaseDate": "2024-09-25T17:00:00-07:00",
            "endOfStandardSupportDate": "2025-11-25T16:00:00-08:00",
            "endOfExtendedSupportDate": "2026-11-25T16:00:00-08:00",
            "status": "STANDARD_SUPPORT",
            "kubernetesPatchVersion": "1.31.3"
        }
    ]
}
```

A saída fornece as seguintes informações para cada versão de cluster:

- `clusterVersion`: a versão do Kubernetes do cluster do EKS
- `clusterType`: o tipo de cluster (por exemplo, "eks")
- `defaultPlatformVersion`: a versão padrão da plataforma do EKS
- `defaultVersion`: se esta é a versão padrão.
- `releaseDate`: a data em que esta versão foi lançada
- `endOfStandardSupportDate`: a data em que o suporte padrão termina
- `endOfExtendedSupportDate`: a data em que o suporte estendido termina
- `status`: o status do suporte atual da versão, como `STANDARD_SUPPORT` ou `EXTENDED_SUPPORT`
- `kubernetesPatchVersion`: a versão específica do patch do Kubernetes

Perguntas frequentes da versão do Amazon EKS

Quantas versões do Kubernetes estão disponíveis no suporte padrão?

De acordo com o suporte da comunidade do Kubernetes para as versões do Kubernetes, o Amazon EKS tem o compromisso de oferecer suporte para ao menos três versões do Kubernetes em qualquer momento. Anunciaremos a data de encerramento do suporte padrão de uma determinada versão secundária do Kubernetes com pelo menos sessenta dias de antecedência. Devido ao processo de lançamento e qualificação do Amazon EKS para novas versões do Kubernetes, a data de encerramento de uma versão do Kubernetes no Amazon EKS será a data em que o projeto do Kubernetes encerrar o suporte ao upstream da versão.

Por quanto tempo um Kubernetes recebe o suporte padrão do Amazon EKS?

Uma versão do Kubernetes recebeu o suporte padrão por 14 meses após estar disponível pela primeira vez no Amazon EKS. Isso também será válido mesmo que o Upstream Kubernetes não seja mais compatível com uma versão disponível no Amazon EKS. Fornecemos patches de segurança de backport que são aplicáveis às versões do Kubernetes compatíveis com o Amazon EKS.

Serei notificado quando o suporte estiver terminando para uma versão do Kubernetes no Amazon EKS?

Sim. Se algum cluster da sua conta estiver executando a versão que está chegando ao fim do suporte, o Amazon EKS enviará um aviso por meio do AWS Health Dashboard aproximadamente 12 meses após o lançamento da versão do Kubernetes no Amazon EKS. O aviso inclui a data de fim do suporte. Isso é, pelo menos, 60 dias após a data do aviso.

Quais recursos do Kubernetes são compatíveis com o Amazon EKS?

O Amazon EKS é compatível com todos os recursos disponíveis para o público geral (GA) da API do Kubernetes. As novas APIs beta não estão habilitadas nos clusters por padrão. As APIs beta anteriores existentes e as novas versões das APIs beta existentes continuam a ser habilitadas por padrão. Não há suporte a recursos alfa.

Os grupos de nós gerenciados pelo Amazon EKS são atualizados automaticamente junto com a versão do ambiente de gerenciamento de cluster?

Não. Um grupo de nós gerenciados cria instâncias do Amazon EC2 em sua conta. Essas instâncias não são atualizadas automaticamente quando você ou o Amazon EKS atualizam seu ambiente de gerenciamento. Para obter mais informações, consulte the section called “Atualizar”. Recomendamos manter a mesma versão do Kubernetes em seu plano de controle e nos nós.

Os grupos de nós autogerenciados são atualizados automaticamente junto com a versão do ambiente de gerenciamento de cluster?

Não. Um grupo de nós autogerenciado inclui instâncias do Amazon EC2 em sua conta. Essas instâncias não são atualizadas automaticamente quando você ou o Amazon EKS atualizam a versão do seu ambiente de gerenciamento em seu nome. Um grupo de nós autogerenciado não tem nenhuma indicação no console de que ele precisa ser atualizado. Você pode visualizar a versão do `kubelet` instalada em um nó selecionando o nó na lista Nodes (Nós) da guia Overview (Visão geral) do cluster para determinar quais nós precisam ser atualizados. Você deve atualizar manualmente os nós. Para obter mais informações, consulte the section called “Métodos de atualização”.

O projeto do Kubernetes testa a compatibilidade entre o ambiente de gerenciamento e os nós para até duas versões secundárias. Por exemplo, os nós `1.30` continuam a operar quando orquestrados por um ambiente de gerenciamento `1.33`. No entanto, a execução de um cluster com nós que são persistentemente três versões secundárias atrás do ambiente de gerenciamento não é recomendada. Para obter mais informações, consulte a Política de suporte da distorção da versão e da versão do Kubernetes na documentação do Kubernetes. Recomendamos manter a mesma versão do Kubernetes em seu plano de controle e nós.

Os pods em execução no Fargate são atualizados automaticamente com uma atualização automática da versão do ambiente de gerenciamento do cluster?

Não. É extremamente recomendável executar os pods do Fargate como parte de um controlador de replicação, como uma implantação do Kubernetes. Em seguida, faça uma reinicialização sucessiva de todos os pods do Fargate. A nova versão do pod do Fargate é implantada com uma versão do `kubelet`, que é a mesma versão da versão atualizada do ambiente de gerenciamento do cluster. Para obter mais informações, consulte Deployments (Implantações) na documentação do Kubernetes.

> **Important**
>
> Se você atualizar o ambiente de gerenciamento, ainda será necessário atualizar os nós Fargate por conta própria. Para atualizar os nós do Fargate, exclua o pod do Fargate representado pelo nó e reimplante o pod. O novo pod é implantado com uma versão do `kubelet` que é a mesma versão do cluster.

Quais versões do Kubernetes são compatíveis com nós híbridos?

O Amazon EKS Hybrid Nodes é compatível com as mesmas versões do Kubernetes dos clusters do Amazon EKS com outros tipos de computação de nós, incluindo suporte à versão padrão e estendida do Kubernetes. Os nós híbridos não são atualizados automaticamente quando você atualiza a versão do ambiente de gerenciamento e você é responsável por atualizá-los. Para obter mais informações, consulte the section called “Atualizar os nós híbridos”.

Perguntas frequentes sobre o suporte estendido do Amazon EKS

A terminologia padrão de suporte e suporte estendido é nova para mim. O que significam esses termos?

O suporte padrão para uma versão do Kubernetes no Amazon EKS começa quando uma versão é lançada no Amazon EKS, e terminará 14 meses após a data de lançamento. O suporte estendido para uma versão do Kubernetes começará imediatamente após o término do suporte padrão, e terminará após mais 12 meses. Por exemplo, o suporte padrão para a versão `1.23` no Amazon EKS terminou em 11 de outubro de 2023. O suporte estendido para a versão `1.23` começou em 12 de outubro de 2023 e terminou em 11 de outubro de 2024.

O que eu preciso fazer para obter suporte estendido para clusters do Amazon EKS?

Você precisará habilitar o suporte estendido (consulte Suporte estendido do EKS) para o seu cluster alterando a política de upgrade do cluster para EXTENDED. Por padrão, para todos os clusters novos e existentes, a política de upgrade é definida como EXTENDED, a menos que seja especificado de outra forma. Consulte a Política de atualização de clusters para ver a política de atualização do seu cluster. O suporte padrão começará quando uma versão do Kubernetes for lançada no Amazon EKS, e terminará 14 meses após a data de lançamento. O suporte estendido para uma versão do Kubernetes começará imediatamente após o término do suporte padrão, e terminará após mais 12 meses.

Para quais versões do Kubernetes eu posso obter suporte estendido?

Você pode executar clusters em qualquer versão por até 12 meses após o fim do suporte padrão para essa versão. Isso significa que cada versão terá suporte por 26 meses no Amazon EKS (14 meses de suporte padrão mais 12 meses de suporte estendido).

E se eu não quiser usar o suporte estendido?

Se você não quiser se inscrever automaticamente no suporte estendido, você pode atualizar o cluster para uma versão do Kubernetes são que esteja no suporte padrão do Amazon EKS.

Para desabilitar o suporte estendido, consulte Desabilitação do suporte estendido do EKS. Observação: se você desabilitar o suporte estendido, o cluster será atualizado automaticamente no final do suporte padrão.

O que acontecerá ao final de 12 meses de suporte estendido?

Os clusters executados em uma versão do Kubernetes que completou seu ciclo de vida de 26 meses (14 meses de suporte padrão, mais 12 meses de suporte estendido) serão atualizados automaticamente para a próxima versão. A atualização automática inclui apenas o ambiente de gerenciamento do Kubernetes. Se você tiver nós do Modo Automático do EKS, eles poderão ser atualizados automaticamente. Os nós autogerenciados e os grupos de nós gerenciados do EKS permanecerão na versão anterior.

Na data de fim do suporte estendido, não será mais possível criar novos clusters do Amazon EKS com a versão sem suporte. Os ambientes de gerenciamento existentes são atualizados automaticamente pelo Amazon EKS para a versão mais antiga com suporte por meio de um processo de implantação gradual após a data de término do suporte. Após a atualização automática do ambiente de gerenciamento, certifique-se de atualizar manualmente os complementos do cluster e os nós do Amazon EC2. Para obter mais informações, consulte the section called “Atualizar a versão do Kubernetes”.

Quando exatamente meu ambiente de gerenciamento será atualizado automaticamente após a data do fim do suporte estendido?

O Amazon EKS não pode fornecer prazos específicos. Atualizações automáticas podem ocorrer a qualquer momento após a data de fim do suporte estendido. Você não receberá nenhuma notificação antes da atualização. Recomendamos que você atualize proativamente seu ambiente de gerenciamento sem depender do processo de atualização automática do Amazon EKS. Para obter mais informações, consulte the section called “Atualizar a versão do Kubernetes”.

Posso deixar meu ambiente de gerenciamento em uma versão do Kubernetes indefinidamente?

Não, segurança da nuvem na AWS é a nossa maior prioridade. Após um determinado ponto (geralmente um ano), a comunidade do Kubernetes para de lançar patches para vulnerabilidades e exposições comuns (CVE) e não aconselha o envio de CVEs para versões não compatíveis. Isso quer dizer que vulnerabilidades específicas de uma versão anterior do Kubernetes podem nem mesmo ter sido relatadas. Isso deixa clusters expostos sem aviso prévio e sem opções de correção em caso de vulnerabilidade. Considerando isso, o Amazon EKS não permite que ambientes de gerenciamento permaneçam em uma versão que tenha atingido o fim do suporte estendido.

Há algum custo adicional para obter suporte estendido?

Sim, haverá um custo adicional para clusters do Amazon EKS executados em suporte estendido. Para obter detalhes sobre preços, consulte os preços do suporte estendido do Amazon EKS para a versão Kubernetes, no blog AWS ou em nossa página de preços.

O que está incluído no suporte estendido?

Os clusters do Amazon EKS no suporte estendido recebem patches de segurança contínuos para o ambiente de gerenciamento do Kubernetes. Além disso, o Amazon EKS lançará patches para os complementos CNI da Amazon VPC, `kube-proxy` e CoreDNS para as versões Extended Support. O Amazon EKS também lançará patches para AMIs otimizadas do Amazon EKS publicadas pela AWS para Amazon Linux, Bottlerocket e Windows, bem como nós do Fargate para Amazon EKS para essas versões. Todos os clusters do suporte estendido continuarão a ter acesso ao suporte técnico do AWS.

Há alguma limitação nos patches para componentes que não sejam do Kubernetes no suporte estendido?

Embora o suporte estendido abranja todos os componentes específicos do Kubernetes da AWS, ele fornecerá suporte apenas para as AMIs otimizadas do Amazon EKS publicadas pela AWS para Amazon Linux, Bottlerocket e Windows em todos os momentos. Isso significa que você potencialmente terá componentes mais novos (como sistema operacional ou kernel) em sua AMI otimizada para Amazon EKS enquanto estiver usando o suporte estendido. Por exemplo, quando o Amazon Linux 2 chegar ao fim de seu ciclo de vida em 2025, as AMIs do Amazon Linux otimizadas para Amazon EKS serão criadas usando um sistema operacional Amazon Linux mais novo. O Amazon EKS anunciará e documentará as discrepâncias importantes no ciclo de vida do suporte, como esta, para cada versão do Kubernetes.

Posso criar novos clusters usando uma versão com suporte estendido?

Sim.

Revisão das notas de release das versões do Kubernetes com suporte padrão

Este tópico fornece as mudanças importantes que você deve conhecer em cada versão do Kubernetes com suporte padrão. Ao fazer o upgrade, analise cuidadosamente as alterações que ocorreram entre a versão antiga e a nova do seu cluster.

Kubernetes 1.34

O Kubernetes `1.34` agora está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.34`, consulte o anúncio oficial de lançamento.

Important

- Containerd atualizado para a versão 2.1 na versão 1.34 de lançamento.
  - Se você enfrentar problemas após a atualização, verifique as notas de lançamento do containerd 2.1.
- A AWS não está lançando uma AMI do Amazon Linux 2 otimizada para EKS para o Kubernetes 1.34.
  - A AWS incentiva você a migrar para o Amazon Linux 2023. Saiba como the section called “Atualizar para AL2023”.
  - Para obter mais informações, consulte the section called “Descontinuação da AMI do Amazon Linux 2”.
- O AppArmor foi descontinuado no Kubernetes 1.34.
  - Recomendamos a migração para soluções alternativas de segurança de contêineres, como seccomp ou Pod Security Standards.
- A VolumeAttributesClass (VAC) chega ao estágio de GA no Kubernetes 1.34, com a migração da API beta (`storage.k8s.io/v1beta1`) para a API estável (`storage.k8s.io/v1`).
  - O driver CSI do EBS com contêineres com arquivos associados gerenciados pela AWS (dos componentes da CSI na Galeria do ECR) e a modificação de volumes continuarão funcionando perfeitamente em clusters do EKS nas versões 1.31 a 1.33. A AWS aplicará patches nos arquivos associados para oferecer suporte às APIs beta da VAC até o final do suporte padrão do EKS 1.33, que é em 29 de julho de 2026.
  - Se você gerencia seus próprios contêineres com arquivos associados da CSI, pode ser necessário fixar versões mais antigas do arquivo associado em clusters anteriores à versão 1.34 para manter a funcionalidade da VAC.
  - Para usar os recursos da GA destinados à VolumeAttributesClass (como a reversão de modificações), atualize para o EKS 1.34 ou para versões posteriores.

- APIs principais de alocação dinâmica de recursos (DRA) para a versão em GA: a alocação dinâmica de recursos atingiu o estágio estável, permitindo o gerenciamento eficiente de hardware especializado como GPUs por meio de interfaces de alocação padronizadas. Isso simplifica a gestão de aceleradores de hardware e aumenta a eficiência no uso de recursos especializados.
- Tokens de ServiceAccount projetados para o Kubelet para a versão Beta: esta melhoria aumenta a segurança ao utilizar credenciais de curta duração para a extração de imagens de contêineres em vez de segredos de longa duração. Isso reduz o risco de exposição de credenciais e fortalece a postura geral de segurança dos clusters.
- Solicitações e limites de recursos em nível de pod para a versão Beta: este recurso simplifica o gerenciamento de recursos, permitindo grupos de recursos compartilhados para pods com vários contêineres. Isso possibilita uma alocação e utilização de recursos mais eficientes para aplicações complexas com diversos contêineres.
- Contagem alocável de nós da CSI mutáveis para a versão Beta: o recurso `MutableCSINodeAllocatableCount` está habilitado por padrão no EKS 1.34, tornando o atributo de contagem máxima de volumes anexáveis do CSINode mutável e introduzindo um mecanismo para atualizá-lo dinamicamente com base na configuração do usuário no nível do driver CSI. Essas atualizações podem ser acionadas por intervalos periódicos ou pela detecção de falhas, aumentando a confiabilidade do agendamento de pods com estados ao resolver divergências entre a capacidade de anexação relatada e a real nos nós.
  - Para obter mais informações, consulte Kubernetes v1.34: Mutable CSI Node Allocatable Count no blog do Kubernetes.
- Aviso de descontinuação para a configuração do driver cgroup: a configuração manual do driver cgroup está sendo descontinuada e será substituída pela detecção automática.
  - Impacto para o cliente: se você define atualmente a sinalização `--cgroup-driver` manualmente em sua configuração do kubelet, recomenda-se planejar a remoção dessa definição.
  - Ação necessária: planeje a atualização dos scripts de inicialização dos nós e das configurações de AMIs personalizadas para remover as definições manuais do driver cgroup antes que o recurso seja removido em uma versão futura do Kubernetes.
  - Para obter mais informações, consulte a documentação do driver cgroup.

Para ver o changelog completo do Kubernetes `1.34`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.34.md.

Kubernetes 1.33

O Kubernetes `1.33` agora está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.33`, consulte o anúncio oficial de lançamento.

Important

- A API beta de alocação dinâmica de recursos do Kubernetes está habilitada.
  - Essa API beta melhora a experiência de agendamento e monitoramento de workloads que exigem recursos, como GPUs.
  - A API beta é definida pela comunidade Kubernetes e poderá mudar em versões futuras do Kubernetes.
  - Analise cuidadosamente os estágios dos recursos na documentação do Kubernetes para compreender as implicações do uso de APIs beta.
- A AWS não está lançando uma AMI do Amazon Linux 2 otimizada para EKS para o Kubernetes 1.33.
  - A AWS incentiva você a migrar para o Amazon Linux 2023. Saiba como the section called “Atualizar para AL2023”.
  - Para obter mais informações, consulte the section called “Descontinuação da AMI do Amazon Linux 2”.

- Redimensionamento de recursos de pod no local (beta): o redimensionamento de recursos no local foi promovido para beta, permitindo atualizações dinâmicas dos recursos de CPU e memória para pods existentes sem reinicializações, o que possibilita a escalabilidade vertical de workloads com estado sem tempo de inatividade e ajustes contínuos de recursos com base nos padrões de tráfego.
- Contêineres sidecar agora estáveis: os contêineres sidecar foram promovidos para estáveis, implementando sidecars como contêineres init especiais com `restartPolicy: Always` que iniciam antes dos contêineres de aplicações, são executados durante todo o ciclo de vida do pod e suportam sondas para sinalização do estado operacional.
  - Para obter mais informações, consulte Sidecar Containers na documentação do Kubernetes.
- Descontinuação da API Endpoints: a API Endpoints agora está oficialmente obsoleta e retornará avisos quando acessada. Migre workloads e scripts para usar a API EndpointSlices, que é

compatível com recursos modernos, como rede de pilha dupla, e lida com vários EndpointSlices por serviço.

  - Para obter mais informações, consulte Kubernetes v1.33: Continuing the transition from Endpoints to EndpointSlice no Blog do Kubernetes.
- Compatibilidade com o Elastic Fabric Adapter: o grupo de segurança padrão para clusters do Amazon EKS agora é compatível com o tráfego do Elastic Fabric Adapter (EFA). O grupo de segurança padrão tem uma nova regra de saída que permite o tráfego do EFA com o destino do mesmo grupo de segurança. Isso permite o tráfego do EFA dentro do cluster.
  - Para obter mais informações, consulte Elastic Fabric Adapter para workloads de IA/ML e HPC no Amazon EC2 no Guia do usuário do Amazon Elastic Compute Cloud.

Para ver o changelog completo do Kubernetes `1.33`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.33.md.

Kubernetes 1.32

O Kubernetes `1.32` agora está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.32`, consulte o anúncio oficial de lançamento.

> **Important**
>
> - A versão da API `flowcontrol.apiserver.k8s.io/v1beta3` de FlowSchema e PriorityLevelConfiguration foi removida na versão `1.32`. Se você estiver usando essas APIs, deverá atualizar suas configurações para usar a versão compatível mais recente antes de realizar a atualização.
> - A anotação `metadata.annotations[kubernetes.io/enforce-mountable-secrets]` do ServiceAccount foi descontinuada na versão `1.32` e será removida em uma futura versão secundária do Kubernetes. É recomendado usar namespaces separados para isolar o acesso aos segredos montados.
> - A versão `1.32` do Kubernetes é a última versão para a qual o Amazon EKS lançará AMIs do Amazon Linux 2 (AL2). A partir da versão `1.33`, o Amazon EKS continuará a lançar AMIs baseadas no Amazon Linux 2023 (AL2023) e no Bottlerocket.

- O recurso Memory Manager foi promovido para o status de disponível ao público em geral (GA) na versão `1.32` do Kubernetes. Essa melhoria proporciona uma alocação de memória mais eficiente

e previsível para aplicações em contêineres, o que é especialmente vantajoso para workloads com requisitos específicos de memória.

- Agora, os PersistentVolumeClaims (PVCs) criados por StatefulSets incluem a funcionalidade de limpeza automática. Quando os PVCs não forem mais necessários, eles serão excluídos automaticamente, mantendo a persistência de dados durante as atualizações do StatefulSet e durante as operações de manutenção de nós. Esse recurso simplifica o gerenciamento de armazenamento e ajuda a evitar PVCs isolados em seu cluster.
- Foi introduzida a funcionalidade de Seletor de campos para recursos personalizados, permitindo que os desenvolvedores adicionem seletores de campo aos recursos personalizados. Esse recurso fornece as mesmas funcionalidades de filtragem disponíveis para objetos do Kubernetes integrados aos recursos personalizados, possibilitando uma filtragem de recursos mais precisa e eficiente, além de promover melhores práticas de design de APIs.

Para ver o changelog completo do Kubernetes `1.32`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.32.md.

Alterações na autenticação anônima

A partir da versão `1.32` do Amazon EKS, a autenticação anônima passa a ser restrita aos seguintes endpoints de verificação de integridade do servidor da API:

- `/healthz`
- `/livez`
- `/readyz`

As solicitações enviadas para qualquer outro endpoint usando o usuário `system:unauthenticated` receberão uma resposta HTTP `401 Unauthorized`. Esse aprimoramento de segurança ajuda a evitar acessos não intencionais ao cluster que poderiam ocorrer devido a configurações incorretas das políticas de RBAC.

> **Note**
>
> O perfil RBAC `public-info-viewer` permanece válido para os endpoints de verificação de integridade listados acima.

Descontinuação da AMI do Amazon Linux 2

A versão `1.32` do Kubernetes é a última para a qual o Amazon EKS lançou AMIs baseadas no AL2. A partir da versão `1.33`, o Amazon EKS continuará a lançar AMIs baseadas no AL2023 e no Bottlerocket. Para obter mais informações, consulte the section called “Descontinuação da AMI do AL2”.

Revisão das notas de release das versões do Kubernetes com suporte estendido

O Amazon EKS é compatível com as versões do Kubernetes por mais tempo do que as versões upstream, com suporte padrão para as versões secundárias do Kubernetes por 14 meses a partir do lançamento no Amazon EKS, e suporte estendido para as versões secundárias do Kubernetes por mais 12 meses de suporte (26 meses no total por versão).

Este tópico fornece mudanças importantes que você deve conhecer em cada Kubernetes versão do suporte estendido. Ao fazer o upgrade, analise cuidadosamente as alterações que ocorreram entre a versão antiga e a nova do seu cluster.

Kubernetes 1.31

O Kubernetes `1.31` agora está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.31`, consulte o anúncio oficial de lançamento.

> **Important**
>
> - O sinalizador do kubelet `--keep-terminated-pod-volumes`, obsoleto desde 2017, foi removido como parte da versão `1.31`. Essa alteração afeta o modo como os volumes de pods encerrados são tratados pelo kubelet. Se você usa esse sinalizador nas configurações do seu nó, deverá atualizar seus scripts de bootstrap e modelos de inicialização para removê-lo antes da atualização.

- A porta do recurso `VolumeAttributesClass` beta e o recurso de API estão habilitados na versão `1.31` do Amazon EKS. Esse recurso permite que operadores de cluster modifiquem propriedades mutáveis de volumes persistentes (PVs) gerenciados por drivers da CSI compatíveis, incluindo o driver da CSI do Amazon EBS. Para aproveitar esse recurso, certifique-se de que o driver CSI seja compatível com o recurso `VolumeAttributesClass` (no caso do driver CSI do Amazon EBS, atualize para a versão `1.35.0` ou posterior para ativar automaticamente o

recurso). Você poderá criar objetos `VolumeAttributesClass` para definir os atributos de volume desejados, como tipo de volume e throughput, e associá-los às suas declarações de volume persistentes (PVCs). Consulte a documentação oficial do Kubernetes e a documentação do driver da CSI para obter mais informações.

  - Para obter mais informações sobre o driver CSI do Amazon EBS, consulte the section called "Amazon EBS".

- O suporte do Kubernetes a AppArmor avançou para estável e agora está disponível para o público em geral. Esse recurso permite que você proteja seus contêineres com o AppArmor definindo o campo `appArmorProfile.type` no `securityContext` do contêiner. Antes da versão `1.30` do Kubernetes, o AppArmor era controlado por anotações. Começando na versão `1.30`, ele é controlado usando campos. Para aproveitar esse recurso, recomendamos que você se afaste das anotações e use o campo `appArmorProfile.type` para garantir que suas cargas de trabalho sejam compatíveis.
- O recurso de tempo de transição da última fase de PersistentVolume avançou para estável e agora está disponível para uso pelo público em geral na versão `1.31` do Kubernetes. Esse recurso introduz um novo campo, `.status.lastTransitionTime`, em PersistentVolumeStatus que fornece um registro de data e hora de quando um PersistentVolume avançou pela última vez para uma fase diferente. Esse aprimoramento permite um melhor rastreamento e gerenciamento de PersistentVolumes, especialmente em cenários em que a compreensão do ciclo de vida dos volumes é importante.

Para ver o changelog completo do Kubernetes `1.31`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.31.md

Kubernetes 1.30

O Kubernetes `1.30` agora está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.30`, consulte o anúncio oficial de lançamento.

- A partir da versão `1.30` do Amazon EKS ou em versões mais recentes, qualquer grupo de nós gerenciados recém-criados será automaticamente padronizado para usar o Amazon Linux 2023 (AL2023) como o sistema operacional do nó. Para obter mais informações sobre como especificar o sistema operacional para um grupo de nós gerenciados, consulte the section called "Criar".
- Com o Amazon EKS `1.30`, o rótulo `topology.k8s.aws/zone-id` é adicionado a nós de processamento. É possível visualizar os IDs de Zonas de disponibilidade (IDs de AZs) para determinar o local de recursos em uma conta em relação aos recursos em outra conta. Para obter

mais informações, consulte IDs de zona de disponibilidade para os recursos do AWS no Guia do Usuário do AWS RAM.

- A partir da versão `1.30`, o Amazon EKS não incluirá mais a anotação `default` no recurso `gp2 StorageClass` aplicado a clusters recém-criados. Isso não terá impacto se você estiver referenciando essa classe de armazenamento pelo nome. Você deverá tomar medidas se estiver confiando em ter um `StorageClass` padrão no cluster. É necessário referenciar `StorageClass` pelo nome `gp2`. Como alternativa, você pode implantar a classe de armazenamento padrão recomendada pelo Amazon EBS definindo o parâmetro `defaultStorageClass.enabled` como verdadeiro ao instalar a versão `1.31.0` ou posterior do `aws-ebs-csi-driver add-on`.
- A política do IAM mínima exigida para o perfil do IAM do cluster do Amazon EKS foi alterada. A ação `ec2:DescribeAvailabilityZones` é obrigatória. Para obter mais informações, consulte the section called “Função do IAM do cluster”.

Para ver o changelog completo do Kubernetes `1.30`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.30.md.

Kubernetes 1.29

O Kubernetes `1.29` agora está disponível no Amazon EKS. Para obter mais informações sobre o Kubernetes `1.29`, consulte o anúncio oficial de lançamento.

> **Important**
>
> - A versão obsoleta da API `flowcontrol.apiserver.k8s.io/v1beta2` de `FlowSchema` e `PriorityLevelConfiguration` não é mais fornecida na versão `1.29` do Kubernetes. Se você tiver manifestos ou software-cliente que usa o grupo de API beta obsoleto, altere-os antes de fazer o upgrade para a versão `1.29`.

- O campo `.status.kubeProxyVersion` para objetos de nó agora está obsoleto, e o projeto do Kubernetes está sugerindo remover esse campo em uma versão futura. O campo obsoleto não é preciso e historicamente foi gerenciado por `kubelet` que, na verdade, não conhece a versão `kube-proxy` ou mesmo se `kube-proxy` está em execução. Se você estiver usando este campo no software cliente, pare. As informações não são confiáveis e o campo agora está obsoleto.
- No Kubernetes `1.29`, para reduzir a potencial superfície de ataque, o recurso `LegacyServiceAccountTokenCleanUp` rotulará os tokens legados baseados em segredos

gerados automaticamente como inválidos se não forem usados por um longo período (um ano, por padrão), e os removerá automaticamente se não houver uma tentativa de uso por um longo período após serem marcados como inválidos (um ano adicional, por padrão). Para identificar esses tokens, você pode executar:

```
kubectl get cm kube-apiserver-legacy-service-account-token-tracking -n kube-system
```

Para ver o changelog completo do Kubernetes `1.29`, consulte https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.29.md#changelog-since-v1280.

Visualizar o período atual de suporte do cluster

A seção período de suporte do cluster do console da AWS indica se seu cluster está atualmente em suporte padrão ou estendido. Se o período de suporte do cluster for Suporte estendido, você está sendo cobrado pelo suporte estendido do EKS.

Para obter mais informações sobre o suporte padrão e o suporte estendido, consulte Entenda o ciclo de vida da versão do Kubernetes no EKS.

1. Navegue até a página Clusters na seção EKS do Console da AWS. Confirme se o console está configurado na mesma região da AWS do cluster que deseja revisar.
2. Revise a coluna Período de suporte. Se o valor for Suporte padrão até..., você não está recebendo cobranças pelo suporte estendido no momento. Você está dentro do período de suporte padrão. Se o valor for Suporte estendido..., significa que o cluster está sendo cobrado atualmente pelo suporte estendido.

> **Note**
>
> O Período de suporte não pode ser recuperado com a AWS CLI ou API.

Visualizar a política atual de upgrade do cluster

A política de upgrade de um cluster determina o que acontece quando um cluster chega ao final do período de suporte padrão. Se a política de upgrade for `EXTENDED`, o cluster não será atualizado automaticamente e entrará no suporte estendido. Se política de upgrade for `STANDARD`, ele será atualizado automaticamente.

Os controles do Amazon EKS para a política de versão do Kubernetes permitem escolher o fim do comportamento de suporte padrão para seus clusters do EKS. Com esses controles, você pode decidir quais clusters devem entrar no suporte estendido e quais clusters devem ser atualizados automaticamente ao final do suporte padrão para uma versão do Kubernetes.

Uma versão secundária terá suporte padrão no Amazon EKS durante os primeiros 14 meses após seu lançamento. Depois que uma versão excede a data de término do suporte padrão, ela entra no suporte estendido pelos 12 meses seguintes. O suporte estendido permite que você permaneça em uma versão específica do Kubernetes por mais tempo por um custo adicional por hora de cluster. É possível habilitar ou desabilitar o suporte estendido para um cluster do EKS. Se você desabilitar o suporte estendido, a AWS atualizará automaticamente seu cluster para a próxima versão ao final do suporte padrão. Se você habilitar o suporte estendido, poderá permanecer na versão atual por um custo adicional por um período limitado de tempo. Planeje fazer a atualização do cluster do Kubernetes regularmente, mesmo que use o suporte estendido.

Você pode definir a política de versão para clusters novos e existentes usando a propriedade `supportType`. Há duas opções que podem ser usadas para definir a política de suporte a versões:

- **`STANDARD`** : seu cluster EKS está qualificado para upgrade automático no final do suporte padrão. Você não incorrerá em cobranças de suporte estendido com essa configuração, mas o cluster do EKS será atualizado automaticamente para a próxima versão do Kubernetes compatível com o suporte padrão.
- **`EXTENDED`** : o cluster do EKS entrará em suporte estendido quando a versão do Kubernetes atingir o fim do suporte padrão. Você incorrerá em taxas de suporte estendidas com essa configuração. É possível atualizar o cluster para uma versão do Kubernetes compatível com o suporte padrão a fim de parar de incorrer em cobranças de suporte estendido. Clusters executados com suporte estendido estarão qualificados para atualização automática ao final do suporte estendido.

O suporte estendido está habilitado por padrão para todos os novos clusters e para os clusters existentes. É possível verificar se o suporte estendido está habilitado para um cluster via Console de gerenciamento da AWS ou AWS CLI.

> **Important**
>
> Se você quiser que o cluster permaneça na versão atual do Kubernetes para se beneficiar do período de suporte estendido, será necessário habilitar a política de atualização de suporte estendido antes do final do período de suporte padrão.

Só é possível definir a política de suporte de versão para seus clusters enquanto ela é executada na versão do Kubernetes em suporte padrão. Depois que a versão entrar no suporte estendido, você não poderá alterar essa configuração até que execute uma versão no suporte padrão.

Por exemplo, se você definiu sua política de suporte de versão como `standard`, não poderá alterar essa configuração depois que a versão do Kubernetes em execução no seu cluster atingir o fim do suporte padrão. Se você definiu sua política de suporte de versão como `extended`, não poderá alterar essa configuração depois que a versão do Kubernetes em execução no seu cluster atingir o fim do suporte padrão. Para alterar a configuração da política de suporte de versão, seu cluster deve estar sendo executado em uma versão padrão compatível do Kubernetes.

Visualizar a política de upgrade do cluster (Console da AWS)

1. Navegue até a página Clusters na seção EKS do Console da AWS. Confirme se o console está configurado na mesma região da AWS do cluster que deseja revisar.
2. Revise a coluna Política de upgrade. Se o valor for Suporte padrão, o cluster não entrará no suporte estendido. Se o valor for Suporte estendido, o cluster entrará no suporte estendido.

Visualizar a política de upgrade do cluster (AWS CLI)

1. Verifique se a AWS CLI está instalada e se você está conectado. Saiba como atualizar e instalar a AWS CLI.
2. Determine o nome do seu cluster do EKS. Defina a CLI para a mesma região da AWS do seu cluster do EKS.
3. Execute o seguinte comando:

```
aws eks describe-cluster \
--name <cluster-name> \
--query "cluster.upgradePolicy.supportType"
```

4. Se o valor for `STANDARD`, o cluster não entrará no suporte estendido. Se o valor for `EXTENDED`, o cluster entrará no suporte estendido.

Adicione flexibilidade para planejar atualizações de versão do Kubernetes habilitando o suporte estendido do EKS

Este tópico descreve como definir a política de atualização de um cluster de EKS para habilitar o suporte estendido. A política de atualização de um cluster de EKS determina o que acontece quando um cluster chega ao final do período de suporte padrão. Se uma política de upgrade de cluster tiver o suporte estendido habilitado, ela entrará no período de suporte estendido no final do período de suporte padrão. O cluster não será atualizado automaticamente no final do período de suporte padrão.

Na verdade, os clusters, no período de suporte estendido, incorrem em custos mais altos. Se um cluster simplesmente tiver a política de atualização definida para permitir o suporte estendido e estiver no período de suporte padrão, ele incorrerá em custos padrão.

Se você criar um cluster no Console da AWS, ele terá a política de atualização definida para desabilitar o suporte estendido. Se você criar um cluster de outra forma, ele terá a política de atualização definida para desabilitar o suporte estendido. Por exemplo, clusters criados com a API da AWS têm o suporte estendido habilitado.

Para obter mais informações sobre políticas de atualização, consulte Política de atualização do cluster.

> **Important**
>
> Se você quiser que o cluster permaneça na versão atual do Kubernetes para se beneficiar do período de suporte estendido, será necessário habilitar a política de atualização de suporte estendido antes do final do período de suporte padrão.
> Se você não ativar o suporte estendido, seu cluster será atualizado automaticamente.

Habilitar suporte estendido do EKS (Console da AWS)

1. Navegue para seu cluster de EKS no Console da AWS. Selecione a guia Visão geral na página Informações do cluster.
2. Na seção Configurações de versão do Kubernetes, selecione Gerenciar.

3. Selecione Suporte estendido e, em seguida, Salvar alterações.

Habilitar suporte estendido do EKS (AWS CLI)

1. Verifique se a AWS CLI está instalada e se você está conectado. Saiba como atualizar e instalar a AWS CLI.
2. Determine o nome do seu cluster de EKS.
3. Execute o seguinte comando:

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=EXTENDED
```

Evitar o aumento dos custos do cluster desabilitando o suporte estendido do EKS

Este tópico descreve como definir a política de atualização de um cluster de EKS para desabilitar o suporte estendido. A política de atualização de um cluster de EKS determina o que acontece quando um cluster chega ao final do período de suporte padrão. Se uma política de atualização de cluster tiver o suporte estendido desabilitado, ela será atualizada automaticamente para a próxima versão do Kubernetes.

Para obter mais informações sobre políticas de atualização, consulte Política de atualização do cluster.

> **Important**
>
> Não é possível desabilitar o suporte estendido após o cluster entrar nele. O suporte estendido pode ser desabilitado somente para clusters no suporte padrão.
> A AWS recomenda atualizar o cluster para uma versão no período de suporte padrão.

Desabilitar suporte estendido do EKS (Console da AWS)

1. Navegue para seu cluster de EKS no Console da AWS. Selecione a guia Visão geral na página Informações do cluster.

2. Na seção Configurações de versão do Kubernetes, selecione Gerenciar.
3. Selecione Suporte padrão e, em seguida, Salvar alterações.

Desabilitar suporte estendido do EKS (AWS CLI)

1. Verifique se a AWS CLI está instalada e se você está conectado. Saiba como atualizar e instalar a AWS CLI.
2. Determine o nome do seu cluster de EKS.
3. Execute o seguinte comando:

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=STANDARD
```

Veja as versões da plataforma Amazon EKS para cada versão do Kubernetes

As versões da plataforma Amazon EKS representam os recursos do ambiente de gerenciamento do cluster do Amazon EKS, como quais sinalizadores do servidor de API do Kubernetes estão habilitados e também a versão de patch atual do Kubernetes. Cada versão secundária do Kubernetes tem uma ou mais versões de plataforma do Amazon EKS associadas. As versões da plataforma para diferentes versões secundárias do Kubernetes são independentes. Você pode recuperar a versão atual da plataforma do seu cluster usando a AWS CLI ou o Console de gerenciamento da AWS. Se você tiver um cluster local no AWS Outposts, consulte the section called "Versões da plataforma do EKS" em vez deste tópico.

Quando uma nova versão secundária do Kubernetes estiver disponível no Amazon EKS, como a 1.33, a versão inicial da plataforma do Amazon EKS para essa versão secundária do Kubernetes iniciará em `eks.1`. No entanto, o Amazon EKS lança novas versões de plataforma periodicamente para habilitar novas configurações do ambiente de gerenciamento do Kubernetes e para fornecer correções de segurança.

Quando novas versões de plataforma do Amazon EKS são disponibilizadas para uma versão secundária:

- O número da versão da plataforma do Amazon EKS é incrementado (`eks.<n+1>`).

- O Amazon EKS atualiza automaticamente todos os clusters existentes da versão de plataforma mais recente do Amazon EKS para a versão secundária correspondente do Kubernetes. As atualizações automáticas de versões de plataformas existentes do Amazon EKS são implementadas de forma incremental. O processo de implantação pode levar algum tempo. Se precisar dos recursos da versão da plataforma mais recente do Amazon EKS imediatamente, você deverá criar outro cluster do Amazon EKS.

  Se o cluster tiver mais de duas versões de plataforma atrás da versão atual da plataforma, é possível que o Amazon EKS não tenha conseguido atualizar automaticamente o cluster. Para obter detalhes sobre o que pode causar isso, consulte the section called “A versão da plataforma Amazon EKS está mais de duas versões atrás da versão atual da plataforma”.
- O Amazon EKS pode publicar uma nova AMI do nó com uma versão de patch correspondente. No entanto, todas as versões de patch são compatíveis entre o ambiente de gerenciamento do EKS e as AMIs do nó para determinada versão secundária do Kubernetes.

Novas versões de plataforma do Amazon EKS não apresentam alterações que podem causar interrupções nem causam interrupções do serviço.

Os clusters são sempre criados com a versão da plataforma mais recente disponível do Amazon EKS (`eks.<n>`) para a versão especificada do Kubernetes. Se você atualizar o cluster para uma nova versão secundária do Kubernetes, o cluster receberá a versão atual da plataforma do Amazon EKS da versão secundária do Kubernetes para a qual você atualizou.

As versões recentes e atuais da plataforma do Amazon EKS estão descritas nas tabelas a seguir.

> **Note**
>
> A AWS desabilitou recentemente algumas versões da plataforma publicadas em junho de 2024. As versões da plataforma apresentavam problemas de estabilidade. Nenhuma ação é necessária.

Para receber notificações de todas as alterações do arquivo de origem nesta página específica da documentação, você pode se inscrever no seguinte URL com um leitor de RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/clusters/platform-versions.adoc.atom
```

Versão do Kubernetes **1.34**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma `1.34`: `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
`1.34.1`	`eks.9`	Nova versão da plataforma com correções de segurança e melhorias.	18 de novembro de 2025
`1.34.1`	`eks.8`	Nova versão da plataforma com correções de segurança e melhorias.	17 de novembro de 2025
`1.34.1`	`eks.7`	Nova versão da plataforma com correções de segurança e melhorias.	6 de novembro de 2025
`1.34.1`	`eks.6`	Nova versão da plataforma com correções de segurança e melhorias.	30 de outubro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.34.1	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	15 de outubro de 2025
1.34.1	eks.4	Lançamento inicial do Kubernetes versão 1.34 para o EKS. Para obter mais informações, consulte the section called “Kubernetes 1.34”.	2.º de outubro de 2025

Versão do Kubernetes **1.33**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.33: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota.

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.33.5	eks.23	Nova versão da plataforma com correções de segurança e melhorias.	18 de novembro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.33.5	eks.22	Nova versão da plataforma com correções de segurança e melhorias.	17 de novembro de 2025
1.33.5	eks.21	Nova versão da plataforma com correções de segurança e melhorias.	6 de novembro de 2025
1.33.5	eks.20	Nova versão da plataforma com correções de segurança e melhorias.	30 de outubro de 2025
1.33.5	eks.19	Nova versão da plataforma com correções de segurança e melhorias.	28 de outubro de 2025
1.33.5	eks.18	Nova versão da plataforma com correções de segurança e melhorias.	15 de outubro de 2025
1.33.5	eks.17	Nova versão da plataforma com correções de segurança e melhorias.	15 de outubro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.33.5	eks.16	Nova versão da plataforma com correções de segurança e melhorias.	2.º de outubro de 2025
1.33.4	eks.15	Nova versão da plataforma com correções de segurança e melhorias.	29 de setembro de 2025
1.33.4	eks.14	Nova versão da plataforma com correções de segurança e melhorias.	29 de setembro de 2025
1.33.4	eks.13	Nova versão da plataforma com correções de segurança e melhorias.	16 de setembro de 2025
1.33.4	eks.12	Nova versão da plataforma com correções de segurança e melhorias.	5 de setembro de 2025
1.33.3	eks.11	Nova versão da plataforma com correções de segurança e melhorias.	25 de agosto de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.33.3	eks.10	Nova versão da plataforma com correções de segurança e melhorias.	17 de agosto de 2025
1.33.2	eks.9	Nova versão da plataforma com correções de segurança e melhorias. eks.7 1.33 e eks.8 1.33 foram descartadas internamente e nunca foram lançadas.	30 de julho de 2025
1.33.1	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	26 de junho de 2025
1.33.1	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	11 de junho de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.33.1	eks.4	Lançamento inicial do Kubernetes versão 1.33 para o EKS. Para obter mais informações, consulte the section called “Kubernetes 1.33”.	28 de maio de 2025

Versão **1.32** do Kubernetes

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.32: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota.

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.32.9	eks.30	Nova versão da plataforma com correções de segurança e melhorias.	18 de novembro de 2025
1.32.9	eks.29	Nova versão da plataforma com correções de segurança e melhorias.	17 de novembro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.32.9	eks.28	Nova versão da plataforma com correções de segurança e melhorias.	6 de novembro de 2025
1.32.9	eks.27	Nova versão da plataforma com correções de segurança e melhorias.	30 de outubro de 2025
1.32.9	eks.26	Nova versão da plataforma com correções de segurança e melhorias.	28 de outubro de 2025
1.32.9	eks.25	Nova versão da plataforma com correções de segurança e melhorias.	15 de outubro de 2025
1.32.9	eks.24	Nova versão da plataforma com correções de segurança e melhorias.	15 de outubro de 2025
1.32.9	eks.23	Nova versão da plataforma com correções de segurança e melhorias.	2.º de outubro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.32.8	eks.22	Nova versão da plataforma com correções de segurança e melhorias.	29 de setembro de 2025
1.32.8	eks.21	Nova versão da plataforma com correções de segurança e melhorias.	29 de setembro de 2025
1.32.8	eks.20	Nova versão da plataforma com correções de segurança e melhorias.	16 de setembro de 2025
1.32.8	eks.19	Nova versão da plataforma com correções de segurança e melhorias.	5 de setembro de 2025
1.32.7	eks.18	Nova versão da plataforma com correções de segurança e melhorias.	25 de agosto de 2025
1.32.7	eks.17	Nova versão da plataforma com correções de segurança e melhorias.	17 de agosto de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.32.6	eks.16	Nova versão da plataforma com correções de segurança e melhorias. eks.14 1.32 e eks.15 1.32 foram descartadas internamente e nunca foram lançadas.	30 de julho de 2025
1.32.5	eks.13	Nova versão da plataforma com correções de segurança e melhorias.	26 de junho de 2025
1.32.5	eks.12	Nova versão da plataforma com correções de segurança e melhorias.	11 de junho de 2025
1.32.5	eks.11	Nova versão da plataforma com correções de segurança e melhorias.	30 de maio de 2025
1.32.3	eks.10	Nova versão da plataforma com correções de segurança e melhorias.	16 de maio de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.32.3	eks.9	Nova versão da plataforma com correções de segurança e melhorias.	29 de abril de 2025
1.32.3	eks.8	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2025
1.32.3	eks.7	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2025
1.32.3	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	2 de abril de 2025
1.32.2	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	17 de março de 2025
1.32.2	eks.4	Nova versão da plataforma com correções de segurança e melhorias.	4 de março de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.32.1	eks.3	Nova versão da plataforma com correções de segurança e melhorias.	24 de fevereiro de 2025
1.32.0	eks.2	Lançamento inicial do Kubernetes versão 1.32 para o EKS. Para obter mais informações, consulte the section called “Kubernetes 1.32”.	Janeiro de 2025

Versão do Kubernetes **1.31**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.31: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota, ObjectCount.

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.31.13	eks.46	Nova versão da plataforma com correções de segurança e melhorias.	18 de novembro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.31.13	eks.45	Nova versão da plataforma com correções de segurança e melhorias.	17 de novembro de 2025
1.31.13	eks.44	Nova versão da plataforma com correções de segurança e melhorias.	6 de novembro de 2025
1.31.13	eks.43	Nova versão da plataforma com correções de segurança e melhorias.	30 de outubro de 2025
1.31.13	eks.42	Nova versão da plataforma com correções de segurança e melhorias.	28 de outubro de 2025
1.31.13	eks.41	Nova versão da plataforma com correções de segurança e melhorias.	15 de outubro de 2025
1.31.13	eks.40	Nova versão da plataforma com correções de segurança e melhorias.	15 de outubro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.31.13	eks.39	Nova versão da plataforma com correções de segurança e melhorias.	2.º de outubro de 2025
1.31.12	eks.38	Nova versão da plataforma com correções de segurança e melhorias.	29 de setembro de 2025
1.31.12	eks.37	Nova versão da plataforma com correções de segurança e melhorias.	29 de setembro de 2025
1.31.12	eks.36	Nova versão da plataforma com correções de segurança e melhorias.	16 de setembro de 2025
1.31.12	eks.35	Nova versão da plataforma com correções de segurança e melhorias.	5 de setembro de 2025
1.31.11	eks.34	Nova versão da plataforma com correções de segurança e melhorias.	25 de agosto de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.31.11	eks.33	Nova versão da plataforma com correções de segurança e melhorias.	17 de agosto de 2025
1.31.10	eks.32	Nova versão da plataforma com correções de segurança e melhorias. eks.30 1.31 e eks.31 1.31 foram descartadas internamente e nunca foram lançadas.	30 de julho de 2025
1.31.9	eks.29	Nova versão da plataforma com correções de segurança e melhorias.	26 de junho de 2025
1.31.9	eks.28	Nova versão da plataforma com correções de segurança e melhorias.	11 de junho de 2025
1.31.9	eks.27	Nova versão da plataforma com correções de segurança e melhorias.	30 de maio de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.31.7	eks.26	Nova versão da plataforma com correções de segurança e melhorias.	16 de maio de 2025
1.31.7	eks.25	Nova versão da plataforma com correções de segurança e melhorias.	29 de abril de 2025
1.31.7	eks.24	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2025
1.31.7	eks.23	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2025
1.31.7	eks.22	Nova versão da plataforma com correções de segurança e melhorias.	2 de abril de 2025
1.31.6	eks.21	Nova versão da plataforma com correções de segurança e melhorias.	17 de março de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.31.6	eks.20	Nova versão da plataforma com correções de segurança e melhorias.	4 de março de 2025
1.31.5	eks.19	Nova versão da plataforma com correções de segurança e melhorias.	24 de fevereiro de 2025
1.31.5	eks.18	Nova versão da plataforma com correções de segurança e melhorias.	24 de fevereiro de 2025
1.31.4	eks.17	Nova versão da plataforma com correções de segurança e melhorias.	17 de janeiro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.31.2	eks.12	Nova versão da plataforma compatível com o Amazon EKS Hybrid Nodes e aprimoramentos na observabilidade do ambiente de gerenciamento. Consulte the section called “Nós híbridos” e Amazon EKS enhances performance observability, respectivamente.	15 de novembro de 2024
1.31.1	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	21 de outubro de 2024
1.31.0	eks.4	Lançamento inicial do Kubernetes versão 1.31 para o EKS. Para obter mais informações, consulte the section called “Kubernetes 1.31”.	26 de setembro de 2024

Versão do Kubernetes **1.30**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.30: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle,

`LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
`1.30.14`	`eks.54`	Nova versão da plataforma com correções de segurança e melhorias.	18 de novembro de 2025
`1.30.14`	`eks.53`	Nova versão da plataforma com correções de segurança e melhorias.	17 de novembro de 2025
`1.30.14`	`eks.52`	Nova versão da plataforma com correções de segurança e melhorias.	6 de novembro de 2025
`1.30.14`	`eks.51`	Nova versão da plataforma com correções de segurança e melhorias.	30 de outubro de 2025
`1.30.14`	`eks.50`	Nova versão da plataforma com correções	28 de outubro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
		de segurança e melhorias.	
1.30.14	eks.49	Nova versão da plataforma com correções de segurança e melhorias.	15 de outubro de 2025
1.30.14	eks.48	Nova versão da plataforma com correções de segurança e melhorias.	15 de outubro de 2025
1.30.14	eks.47	Nova versão da plataforma com correções de segurança e melhorias.	2.º de outubro de 2025
1.30.14	eks.46	Nova versão da plataforma com correções de segurança e melhorias.	29 de setembro de 2025
1.30.14	eks.45	Nova versão da plataforma com correções de segurança e melhorias.	29 de setembro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.30.14	eks.44	Nova versão da plataforma com correções de segurança e melhorias.	16 de setembro de 2025
1.30.14	eks.43	Nova versão da plataforma com correções de segurança e melhorias.	5 de setembro de 2025
1.30.14	eks.42	Nova versão da plataforma com correções de segurança e melhorias.	25 de agosto de 2025
1.30.14	eks.41	Nova versão da plataforma com correções de segurança e melhorias.	17 de agosto de 2025
1.30.14	eks.40	Nova versão da plataforma com correções de segurança e melhorias. eks.38 1.30 e eks.39 1.30 foram descartadas internamente e nunca foram lançadas.	30 de julho de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.30.13	eks.37	Nova versão da plataforma com correções de segurança e melhorias.	26 de junho de 2025
1.30.13	eks.36	Nova versão da plataforma com correções de segurança e melhorias.	11 de junho de 2025
1.30.13	eks.35	Nova versão da plataforma com correções de segurança e melhorias.	30 de maio de 2025
1.30.11	eks.34	Nova versão da plataforma com correções de segurança e melhorias.	16 de maio de 2025
1.30.11	eks.33	Nova versão da plataforma com correções de segurança e melhorias.	29 de abril de 2025
1.30.11	eks.32	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.30.11	eks.31	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2025
1.30.11	eks.30	Nova versão da plataforma com correções de segurança e melhorias.	2 de abril de 2025
1.30.10	eks.29	Nova versão da plataforma com correções de segurança e melhorias.	17 de março de 2025
1.30.10	eks.28	Nova versão da plataforma com correções de segurança e melhorias.	4 de março de 2025
1.30.9	eks.27	Nova versão da plataforma com correções de segurança e melhorias.	24 de fevereiro de 2025
1.30.8	eks.25	Nova versão da plataforma com correções de segurança e melhorias.	17 de janeiro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.30.8	eks.24	Nova versão da plataforma com correções de segurança e melhorias.	3 de janeiro de 2025
1.30.7	eks.23	Nova versão da plataforma com correções de segurança e melhorias.	13 de dezembro de 2024
1.30.6	eks.22	Nova versão da plataforma com correções de segurança e melhorias.	13 de dezembro de 2024
1.30.6	eks.21	Nova versão da plataforma com correções de segurança e melhorias.	13 de dezembro de 2024

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.30.6	eks.20	Nova versão da plataforma compatível com o Amazon EKS Hybrid Nodes e aprimoramentos na observabilidade do ambiente de gerenciamento. Consulte the section called “Nós híbridos” e Amazon EKS enhances performance observability, respectivamente.	15 de novembro de 2024
1.30.5	eks.12	Nova versão da plataforma com correções de segurança e melhorias.	21 de outubro de 2024
1.30.4	eks.8	Nova versão da plataforma com correções de segurança e melhorias.	3 de setembro de 2024
1.30.3	eks.7	Nova versão da plataforma com correções de segurança e melhorias.	28 de agosto de 2024

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.30.3	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	9 de agosto de 2024
1.30.2	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	2 de julho de 2024
1.30.0	eks.2	Lançamento inicial do Kubernetes versão 1.30 para o EKS. Para obter mais informações, consulte the section called “Kubernetes 1.30”.	23 de maio de 2024

Versão do Kubernetes **1.29**

Os seguintes controladores de admissão estão habilitados para todas as versões da plataforma 1.29: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota.

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.29.15	eks.57	Nova versão da plataforma com correções de segurança e melhorias.	18 de novembro de 2025
1.29.15	eks.56	Nova versão da plataforma com correções de segurança e melhorias.	17 de novembro de 2025
1.29.15	eks.55	Nova versão da plataforma com correções de segurança e melhorias.	6 de novembro de 2025
1.29.15	eks.54	Nova versão da plataforma com correções de segurança e melhorias.	30 de outubro de 2025
1.29.15	eks.53	Nova versão da plataforma com correções de segurança e melhorias.	28 de outubro de 2025
1.29.15	eks.52	Nova versão da plataforma com correções	15 de outubro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
		de segurança e melhorias.	
1.29.15	eks.51	Nova versão da plataforma com correções de segurança e melhorias.	15 de outubro de 2025
1.29.15	eks.50	Nova versão da plataforma com correções de segurança e melhorias.	2.º de outubro de 2025
1.29.15	eks.49	Nova versão da plataforma com correções de segurança e melhorias.	29 de setembro de 2025
1.29.15	eks.48	Nova versão da plataforma com correções de segurança e melhorias.	29 de setembro de 2025
1.29.15	eks.47	Nova versão da plataforma com correções de segurança e melhorias.	16 de setembro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.29.15	eks.46	Nova versão da plataforma com correções de segurança e melhorias.	5 de setembro de 2025
1.29.15	eks.45	Nova versão da plataforma com correções de segurança e melhorias.	25 de agosto de 2025
1.29.15	eks.44	Nova versão da plataforma com correções de segurança e melhorias.	17 de agosto de 2025
1.29.15	eks.43	Nova versão da plataforma com correções de segurança e melhorias. eks.41 1.29 e eks.42 1.29 foram descartadas internamente e nunca foram lançadas.	30 de julho de 2025
1.29.15	eks.40	Nova versão da plataforma com correções de segurança e melhorias.	26 de junho de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.29.15	eks.39	Nova versão da plataforma com correções de segurança e melhorias.	11 de junho de 2025
1.29.15	eks.38	Nova versão da plataforma com correções de segurança e melhorias.	30 de maio de 2025
1.29.15	eks.37	Nova versão da plataforma com correções de segurança e melhorias.	16 de maio de 2025
1.29.15	eks.36	Nova versão da plataforma com correções de segurança e melhorias.	29 de abril de 2025
1.29.15	eks.35	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2025
1.29.15	eks.34	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.29.15	eks.33	Nova versão da plataforma com correções de segurança e melhorias.	2 de abril de 2025
1.29.14	eks.32	Nova versão da plataforma com correções de segurança e melhorias.	17 de março de 2025
1.29.14	eks.31	Nova versão da plataforma com correções de segurança e melhorias.	4 de março de 2025
1.29.13	eks.30	Nova versão da plataforma com correções de segurança e melhorias.	24 de fevereiro de 2025
1.29.13	eks.29	Nova versão da plataforma com correções de segurança e melhorias.	24 de fevereiro de 2025
1.29.12	eks.28	Nova versão da plataforma com correções de segurança e melhorias.	20 de janeiro de 2025

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.29.12	eks.27	Nova versão da plataforma com correções de segurança e melhorias.	3 de janeiro de 2025
1.29.11	eks.26	Nova versão da plataforma com correções de segurança e melhorias.	13 de dezembro de 2024
1.29.10	eks.25	Nova versão da plataforma com correções de segurança e melhorias.	13 de dezembro de 2024
1.29.10	eks.24	Nova versão da plataforma com correções de segurança e melhorias.	13 de dezembro de 2024

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.29.10	eks.23	Nova versão da plataforma compatível com o Amazon EKS Hybrid Nodes e aprimoramentos na observabilidade do ambiente de gerenciamento. Consulte the section called “Nós híbridos” e Amazon EKS enhances performance observability, respectivamente.	15 de novembro de 2024
1.29.9	eks.17	Nova versão da plataforma com correções de segurança e melhorias.	21 de outubro de 2024
1.29.8	eks.13	Nova versão da plataforma com correções de segurança e melhorias.	3 de setembro de 2024
1.29.7	eks.12	Nova versão da plataforma com correções de segurança e melhorias.	28 de agosto de 2024

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.29.7	eks.11	Nova versão da plataforma com correções de segurança e melhorias.	9 de agosto de 2024
1.29.6	eks.10	Nova versão da plataforma com correções de segurança e melhorias.	2 de julho de 2024
1.29.4	eks.7	Nova versão da plataforma com autoescalabilidade do CoreDNS, correções de segurança e melhorias. Para obter mais informaçõ es sobre autoescal abilidade do CordeDNS, consulte the section called “Escalar para tráfego intenso”.	16 de maio de 2024
1.29.3	eks.6	Nova versão da plataforma com correções de segurança e melhorias.	18 de abril de 2024

Versão do Kubernetes	Versão da plataforma do EKS	Notas da versão	Data de lançamento
1.29.1	eks.5	Nova versão da plataforma com correções de segurança e melhorias.	29 de março de 2024
1.29.1	eks.4	Nova versão da plataforma com correções de segurança e melhorias.	20 de março de 2024
1.29.1	eks.3	Nova versão da plataforma com correções de segurança e melhorias.	12 de março de 2024
1.29.0	eks.1	Lançamento inicial do Kubernetes versão 1.29 para o EKS. Para obter mais informações, consulte the section called “Kubernetes 1.29”.	23 de janeiro de 2024

Obter a versão atual da plataforma

1. Abra o console do Amazon EKS.
2. No painel de navegação, escolha Clusters.
3. Na lista de clusters, escolha o Nome do cluster para verificar a versão da plataforma.
4. Escolha a guia Visão geral.
5. A Versão da plataforma está disponível na seção Detalhes.

6. Determine o Nome do cluster do qual você deseja verificar a versão da plataforma.
7. Execute o seguinte comando:

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

Veja abaixo um exemplo de saída.

```
"eks.10"
```

Alterar versão da plataforma

Não é possível alterar a versão da plataforma de um cluster do EKS. Quando novas versões da plataforma do Amazon EKS são disponibilizadas para uma versão do Kubernetes, o EKS atualiza automaticamente todos os clusters existentes para a versão da plataforma mais recente do Amazon EKS para a versão do Kubernetes correspondente. As atualizações automáticas de versões de plataformas existentes do Amazon EKS são implementadas de forma incremental. Você não pode usar o Console da AWS ou a CLI para alterar a versão da plataforma.

Se você atualizar a versão do Kubernetes, o cluster migrará para a versão mais recente da plataforma para a versão do Kubernetes.

Solucionar problemas com clusters e nós do Amazon EKS

Este capítulo aborda alguns erros comuns que você poderá encontrar ao usar o Amazon EKS e como resolvê-los. Se você precisar solucionar problemas em áreas específicas do Amazon EKS, consulte os tópicos the section called “Solução de problemas”, the section called “Solucionar problemas do EKS Connector” e Solução de problemas do ADOT utilizando complementos do EKS.

Para obter outras informações sobre solução de problemas, consulte o conteúdo do Centro de Conhecimento sobre o Amazon Elastic Kubernetes Service em AWS re:Post.

Capacidade insuficiente

Se você receber o seguinte erro ao tentar criar um cluster do Amazon EKS, isso significa que uma das zonas de disponibilidade que você especificou não tem capacidade suficiente para oferecer suporte a um cluster.

```
Cannot create cluster 'example-cluster' because region-1d, the targeted Availability Zone, does not currently have sufficient capacity to support the cluster. Retry and choose from these Availability Zones: region-1a, region-1b, region-1c
```

Tente criar o cluster com sub-redes na VPC do cluster que são hospedadas nas zonas de disponibilidade retornadas por essa mensagem de erro.

Existem zonas de disponibilidade nas quais um cluster não pode residir. Compare as zonas de disponibilidade das suas sub-redes com a lista de zonas de disponibilidade especificada nos requisitos e considerações de sub-rede.

Falha nos nós ao ingressar no cluster

Há alguns motivos comuns que impedem que os nós ingressem no cluster:

- Se os nós forem nós gerenciados, o Amazon EKS adiciona entradas ao `ConfigMap aws-auth` quando você cria o grupo de nós. Se a entrada tiver sido removida ou modificada, você precisará adicioná-la novamente. Para obter mais informações, insira `eksctl create iamidentitymapping --help` no seu terminal. É possível verificar a versão atual das entradas do `ConfigMap aws-auth` ao substituir *`my-cluster`* no comando a seguir pelo nome do seu cluster e, depois, executar o comando modificado: `eksctl get iamidentitymapping --`

`cluster` *`my-cluster`* . O ARN do perfil que você especifica não pode incluir um caminho diferente de `/`. Por exemplo, se o nome do seu perfil for `development/apps/my-role`, você precisará alterá-lo para `my-role` ao especificar o ARN para o perfil. Verifique se o ARN do perfil do IAM do nó (não o ARN do perfil de instância) está especificado.

Caso os nós sejam autogerenciados e não haja entradas de acesso criadas para o ARN do perfil do IAM do nó, execute os comandos listados para os nós gerenciados. Se você criou uma entrada de acesso para o ARN do perfil do IAM do nó, talvez ela não esteja configurada corretamente na entrada de acesso. Verifique se o ARN do perfil do IAM do nó (não o ARN do perfil de instância) está especificado como o ARN principal na entrada `ConfigMap` `aws-auth` ou na entrada de acesso. Para obter mais informações sobre as entradas de acesso, consulte the section called "Entradas de acesso".

- O ClusterName em seu nó AWS O modelo CloudFormation não corresponde exatamente ao nome do cluster ao qual você deseja que seus nós se juntem. O fornecimento de um valor incorreto a esse campo resulta em uma configuração incorreta do arquivo `/var/lib/kubelet/kubeconfig` do nó, e os nós não ingressarão no cluster.
- O nó não é marcado como sendo de propriedade do cluster. Seus nós devem ter a seguinte tag aplicada a eles, onde *`my-cluster`* é substituído pelo nome do seu cluster.

Chave	Valor
`kubernetes.io/cluster/` *`my-cluster`*	`owned`

- Talvez não seja possível que os nós acessem o cluster usando um endereço IP público. Verifique se os nós implantados em sub-redes públicas recebem um endereço IP público. Caso contrário, é possível associar um endereço IP elástico a um nó depois que ele for executado. Para obter mais informações, consulte Associar um endereço IP elástico a uma instância em execução ou interface de rede. Se a sub-rede pública não estiver definida para atribuir automaticamente endereços IP públicos a instâncias implantadas nela, recomendamos habilitar essa configuração. Para obter mais informações, consulte Modificar o atributo de endereçamento IPv4 público para a sub-rede. Se o nó for implantado em uma sub-rede privada, a sub-rede deverá ter uma rota para um gateway NAT que tenha um endereço IP público atribuído a ele.
- O endpoint AWS STS para a região AWS na qual você está implantando os nós não está ativado para a sua conta. Para ativar a região, consulte Ativação e desativação do AWS STS em uma região AWS.

- O nó não tem uma entrada DNS privada, resultando no erro `node "" not found` no log do `kubelet`. Verifique se a VPC na qual o nó foi criado tem valores definidos para `domain-name` e `domain-name-servers` como `Options` em um `DHCP options set`. Os valores padrão são `domain-name:<region>.compute.internal` e `domain-name-servers:AmazonProvidedDNS`. Para obter mais informações, consulte Conjuntos de opções DHCP no Guia do usuário da Amazon VPC.
- Se os nós do grupo de nós gerenciados não se conectarem ao cluster em 15 minutos, um problema de integridade "NodeCreationFailure" será emitido e o status do console será definido como `Create failed`. Para AMIs do Windows com tempos de execução lentos, esse problema pode ser resolvido usando a inicialização rápida.

Você pode usar o runbook `AWSSupport-TroubleshootEKSWorkerNode` para identificar e solucionar problemas de causas comuns que impeçam o ingresso de nós de processamento em um cluster. Para obter mais informações, consulte `AWSSupport-TroubleshootEKSWorkerNode` na referência do runbook AWS Systems Manager Automation.

Acesso negado ou não autorizado (**`kubectl`**)

Se você receber um dos erros a seguir ao executar os comandos do `kubectl`, é porque o `kubectl` não está configurado corretamente para o Amazon EKS ou as credenciais para a entidade principal do IAM (perfil ou usuário) que estão sendo usadas não estão mapeadas para um nome de usuário do Kubernetes com permissões suficientes para objetos do Kubernetes no cluster do Amazon EKS.

- `could not get token: AccessDenied: Access denied`
- `error: You must be logged in to the server (Unauthorized)`
- `error: the server doesn't have a resource type "svc"`

Isso pode ocorrer em função de um dos seguintes motivos:

- O cluster foi criado com credenciais para uma entidade principal do IAM e o `kubectl` está configurado para usar credenciais para uma entidade principal do IAM diferente. Para resolver essa situação, atualize seu arquivo `kube config` para usar as credenciais que criaram o cluster. Para obter mais informações, consulte the section called "Acessar meu cluster com o kubectl".
- Caso seu cluster satisfaça os requisitos mínimos de plataforma descritos na seção de pré-requisitos da página Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso ao EKS, não haverá uma entrada de acesso associada à sua entidade principal do IAM. Se

a entrada existir, ela não terá os nomes de grupo do Kubernetes necessários definidos para ela, ou não terá a política de acesso adequada associada a ela. Para obter mais informações, consulte the section called “Entradas de acesso”.

- Se o cluster atender aos requisitos mínimos de plataforma na seção de pré-requisitos de Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso EKS, não existirá uma entrada de acesso com a sua entidade principal do IAM no utilitário `aws-auth` do `ConfigMap`. Se a entrada existir, ela não será mapeada para nomes de grupos do Kubernetes vinculados a `Role` ou `ClusterRole` do Kubernetes com as permissões necessárias. Para obter mais informações sobre os objetos de autorização baseada em perfil (RBAC) do Kubernetes, consulte Using RBAC authorization na documentação do Kubernetes. É possível verificar a versão atual das entradas do `ConfigMap` `aws-auth` ao substituir *`my-cluster`* no comando a seguir pelo nome do seu cluster e, depois, executar o comando modificado: `eksctl get iamidentitymapping --cluster` *`my-cluster`* . Caso não exista uma entrada com o ARN da sua entidade principal do IAM no `ConfigMap`, insira `eksctl create iamidentitymapping --help` em seu terminal para obter informações sobre como criar uma entrada.

Se você instalar e configurar a AWS CLI, poderá configurar as credenciais do IAM que usa. Para obter mais informações, consulte Configurar a AWS CLI, no Guia do usuário da AWS Command Line Interface. Também é possível configurar o `kubectl` para usar um perfil do IAM, caso você o assuma para acessar objetos do Kubernetes no cluster. Para obter mais informações, consulte the section called “Acessar meu cluster com o kubectl”.

`hostname doesn’t match`

Seu sistema precisa ter a versão `2.7.9` ou superior do Python. Caso contrário, você receberá erros `hostname doesn’t match` com chamadas AWS CLI para o Amazon EKS. Para obter mais informações, consulte What are "hostname doesn't match" errors? nas Perguntas frequentes sobre solicitações do Python.

`getsockopt: no route to host`

O Docker é executado no intervalo `172.17.0.0/16` do CIDR em clusters do Amazon EKS. Recomendamos que as sub-redes da VPC do cluster não sobreponham esse intervalo. Caso contrário, você receberá o seguinte erro:

```
Error: : error upgrading connection: error dialing backend: dial tcp
 172.17.<nn>.<nn>:10250: getsockopt: no route to host
```

Instances failed to join the Kubernetes cluster

Se você receber o erro `Instances failed to join the Kubernetes cluster` no Console de gerenciamento da AWS, certifique-se de que o acesso ao endpoint privado do cluster esteja habilitado ou que você configurou corretamente os blocos CIDR para acesso ao endpoint público. Para obter mais informações, consulte the section called “Acesso ao endpoint do cluster”.

Códigos de erro para o grupo de nós gerenciados

Se o grupo de nós gerenciados encontrar um problema de integridade de hardware, o Amazon EKS retornará um código de erro para ajudar você a diagnosticar o problema. Essas verificações de integridade não detectam problemas de software porque são baseadas em verificações de integridade do Amazon EC2. A lista a seguir descreve os códigos de erro.

AccessDenied

> O Amazon EKS ou um ou mais nós gerenciados não conseguem autenticar nem autorizar com o servidor de API do cluster do Kubernetes. Para obter mais informações sobre como resolver uma causa comum, consulte the section called “Como corrigir uma causa comum de erros `AccessDenied` para grupos de nós gerenciados”. As AMIs privadas do Windows também podem causar esse código de erro juntamente com a mensagem de erro `Not authorized for images`. Para obter mais informações, consulte the section called “`Not authorized for images`”.

AmiIdNotFound

> Não foi possível encontrar o ID da AMI associada ao seu modelo de inicialização. Certifique-se de que a AMI existe e é compartilhada com sua conta.

AutoScalingGroupNotFound

> Não foi possível encontrar o grupo do Auto Scaling associado ao grupo de nós gerenciados. Você pode recriar um grupo do Auto Scaling com as mesmas configurações para fazer a recuperação.

ClusterUnreachable

O Amazon EKS ou um ou mais nós gerenciados não conseguem se comunicar com o servidor de API do cluster do Kubernetes. Isso pode acontecer se houver interrupções na rede ou se os servidores de API estiverem expandindo solicitações de processamento.

Ec2SecurityGroupNotFound

Não foi possível localizar o grupo de segurança do cluster. Você deve recriar seu cluster.

Ec2SecurityGroupDeletionFailure

Não foi possível excluir o grupo de segurança de acesso remoto para o grupo de nós gerenciados. Remova quaisquer dependências do grupo de segurança.

Ec2LaunchTemplateNotFound

Não foi possível encontrar o modelo de inicialização do Amazon EC2 para o grupo de nós gerenciados. Você deve recriar seu grupo de nós para a recuperação.

Ec2LaunchTemplateVersionMismatch

A versão do modelo de inicialização do Amazon EC2 para o grupo de nós gerenciados não corresponde à versão que o Amazon EKS criou. Você pode reverter para a versão criada pelo Amazon EKS para fazer a recuperação.

IamInstanceProfileNotFound

Não foi possível encontrar o perfil de instância do IAM para o grupo de nós gerenciados. Você pode recriar um perfil de instância com as mesmas configurações para fazer a recuperação.

IamNodeRoleNotFound

Não foi possível encontrar o perfil do IAM para o grupo de nós gerenciados. Você pode recriar um perfil do IAM com as mesmas configurações para fazer a recuperação.

AsgInstanceLaunchFailures

O grupo do Auto Scaling está sofrendo falhas ao tentar iniciar instâncias.

NodeCreationFailure

As instâncias iniciadas não conseguem se registrar no cluster do Amazon EKS. As causas comuns dessa falha são permissões insuficientes do perfil do IAM do nó ou a falta de acesso à Internet na saída para os nós. Seus nós devem atender a qualquer um dos seguintes requisitos:

- Capacidade de acessar a Internet utilizando um endereço IP público. O grupo de segurança associado à sub-rede em que o nó se encontra deve permitir a comunicação. Para obter mais informações, consulte the section called “Requisitos e considerações para sub-redes” e the section called “Requisitos para grupos de segurança”.
- Os seus nós e a VPC devem satisfazer os requisitos apresentados na página Implementar clusters privados com acesso limitado à Internet.

InstanceLimitExceeded

Sua conta da AWS não pode iniciar mais nenhuma instância do tipo especificado. Você pode solicitar um aumento do limite de instâncias do Amazon EC2 para fazer a recuperação.

InsufficientFreeAddresses

Uma ou mais sub-redes associadas ao grupo de nós gerenciados não têm endereços IP disponíveis suficientes para novos nós.

InternalFailure

Esses erros geralmente são causados por um problema no servidor do Amazon EKS.

Como corrigir uma causa comum de erros **`AccessDenied`** para grupos de nós gerenciados

A causa mais comum de erros `AccessDenied` ao executar operações em grupos de nós gerenciados, é a ausência de `eks:node-manager`, `ClusterRole` ou `ClusterRoleBinding`. O Amazon EKS configura esses recursos no cluster como parte da integração com grupos de nós gerenciados, e eles são necessários para gerenciar os grupos de nós.

O `ClusterRole` pode mudar ao longo do tempo, mas deve ser semelhante ao seguinte exemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
```

```
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
```

O `ClusterRoleBinding` pode mudar ao longo do tempo, mas deve ser semelhante ao seguinte exemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Verifique se o `eks:node-manager` `ClusterRole` existe.

```
kubectl describe clusterrole eks:node-manager
```

Se estiver presente, compare a saída com ao exemplo de `ClusterRole` anterior.

Verifique se o `eks:node-manager` `ClusterRoleBinding` existe.

```
kubectl describe clusterrolebinding eks:node-manager
```

Se estiver presente, compare a saída com ao exemplo de `ClusterRoleBinding` anterior.

Se você identificou a ausência de `ClusterRole` ou `ClusterRoleBinding` como a causa de um erro `AcessDenied` ao solicitar operações de grupo de nó gerenciado, você pode restaurá-los. Salve o conteúdo a seguir em um arquivo chamado *`eks-node-manager-role.yaml`*.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
```

```
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Aplique o arquivo.

```
kubectl apply -f eks-node-manager-role.yaml
```

Tente novamente a operação do grupo de nós para ver se isso resolveu o problema.

`Not authorized for images`

Uma possível causa de uma mensagem de erro `Not authorized for images` é usar uma AMI privada do Windows para Amazon EKS para executar grupos de nós gerenciados do Windows. Depois de lançar novas AMIs do Windows, a AWS torna privadas as AMIs com mais de quatro meses, o que as torna inacessíveis. Se o grupo de nós gerenciados estiver usando uma AMI privada do Windows, considere a possibilidade de atualizar o grupo de nós gerenciados do Windows. Embora não possamos garantir que poderemos fornecer acesso a AMIs que se tornaram privadas, você pode solicitar acesso ao preencher um tíquete no AWS Support. Para obter mais informações, consulte Patches no Guia do usuário do Amazon EC2.

O nó está no estado **`NotReady`**

Se o nó entrar no status de `NotReady`, isso provavelmente indicará que ele não está íntegro e está indisponível para o agendamento de novos pods. Isso pode ocorrer por vários motivos, como a falta de recursos suficientes para a CPU, a memória ou o espaço em disco disponível no nó.

Para AMIs do Windows otimizadas para o Amazon EKS, não há reserva para recursos de computação especificados por padrão na configuração do `kubelet`. Para ajudar a evitar problemas de recursos, você pode reservar recursos de computação para processos do sistema, fornecendo ao `kubelet` valores de configuração para kube-reserved e/ou system-reserved. Você faz isso ao usar o parâmetro de linha de comando `-KubeletExtraArgs` no script de inicialização. Para obter mais informações, consulte Reserve Compute Resources for System Daemons na documentação do

Kubernetes e the section called “Parâmetros de configuração do script de bootstrap” neste guia de usuário.

Coletor de logs do EKS

Para solucionar problemas com os nós do Amazon EKS, há um script pré-criado disponível nos nós localizados em `/etc/eks/log-collector-script/eks-log-collector.sh`. Você pode usar o script para coletar logs de diagnóstico para casos de suporte e solução de problemas gerais.

Use o seguinte comando para executar o script em seu nó:

```
sudo bash /etc/eks/log-collector-script/eks-log-collector.sh
```

> **Note**
>
> Se o script não estiver presente no local. Você pode baixar e executar o script manualmente com o seguinte comando:
>
> ```
> curl -O https://amazon-eks.s3.amazonaws.com/support/log-collector-script/linux/
> eks-log-collector.sh
> sudo bash eks-log-collector.sh
> ```

O script coleta as seguintes informações de diagnóstico.

```
$ sudo bash /etc/eks/log-collector-script/eks-log-collector.sh

      This is version 0.7.8. New versions can be found at https://github.com/awslabs/
amazon-eks-ami/blob/main/log-collector-script/

Trying to collect common operating system logs...
Trying to collect kernel logs...
Trying to collect mount points and volume information...
...
...

 Done... your bundled logs are located in /var/log/eks_i-EXAMPLE_2025-03-25_0000-
UTC_0.7.8.tar.gz
```

As informações de diagnóstico são coletadas e armazenadas em:

```
/var/log/eks_i-EXAMPLE_2025-03-25_0000-UTC_0.7.8.tar.gz
```

Para recuperar o pacote de logs dos nós do Bottlerocket, consulte Logs do Bottlerocket para obter mais detalhes.

A rede de runtime do contêiner não está pronta

Você pode receber um erro `Container runtime network not ready` e erros de autorização semelhantes a:

```
4191 kubelet.go:2130] Container runtime network not ready: NetworkReady=false
 reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config
 uninitialized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
4191 kubelet_node_status.go:106] Unable to register node
 "ip-10-40-175-122.ec2.internal" with API server: Unauthorized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
```

Isso pode acontecer devido a um dos seguintes motivos:

1. Ou você não tem um `ConfigMap` `aws-auth` em seu cluster ou ele não inclui entradas para o perfil do IAM com o qual você configurou seus nós.

   Para resolver o problema, verifique as entradas existentes no `ConfigMap` ao substituir *`my-cluster`* no comando a seguir pelo nome do cluster e execute o comando modificado: `eksctl get iamidentitymapping --cluster` *`my-cluster`* . Se você receber uma mensagem de erro ao executar o comando, isso pode ocorrer porque o seu cluster não tem um utilitário `aws-auth` do `ConfigMap`. O comando a seguir adiciona uma entrada ao `ConfigMap`. Caso o `ConfigMap` não exista, o comando também o cria. Substitua *`111122223333`* pelo ID da conta da AWS do perfil do IAM e *`myAmazonEKSNodeRole`* pelo nome do perfil do seu nó.

   ```
   eksctl create iamidentitymapping --cluster my-cluster \
       --arn arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --group
    system:bootstrappers,system:nodes \
       --username system:node:{{EC2PrivateDNSName}}
   ```

O ARN do perfil que você especifica não pode incluir um caminho diferente de `/`. Por exemplo, se o nome do seu perfil for `development/apps/my-role`, você precisará alterá-lo para `my-role` ao especificar o ARN do perfil. Verifique se o ARN do perfil do IAM do nó (não o ARN do perfil de instância) está especificado.

2. Os seus nós autogerenciados estão em um cluster com uma versão da plataforma na versão mínima listada nos pré-requisitos do tópico Conceder aos usuários do IAM acesso ao Kubernetes com entradas de acesso EKS, mas não há uma entrada listada no utilitário `aws-auth` do `ConfigMap` (consulte o item anterior) para o perfil do IAM do nó, ou não existe uma entrada de acesso para o perfil. Para resolver o problema, visualize as entradas de acesso existentes ao substituir *`my-cluster`* no comando a seguir pelo nome do cluster e execute o comando modificado: `aws eks list-access-entries --cluster-name` *`my-cluster`* . O comando apresentado a seguir adiciona uma entrada de acesso para o perfil do IAM associado ao nó. Substitua *`111122223333`* pelo ID da conta da AWS do perfil do IAM e *`myAmazonEKSNodeRole`* pelo nome do perfil do seu nó. Se você tiver um nó do Windows, substitua *`EC2_LINUX`* por `EC2_Windows`. Verifique se o ARN do perfil do IAM do nó (não o ARN do perfil de instância) está especificado.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --type EC2_LINUX
```

Tempo limite de handshake TLS

Quando um nó não conseguir estabelecer uma conexão com o endpoint do servidor de API público, talvez você veja um erro semelhante ao erro a seguir.

```
server.go:233] failed to run Kubelet: could not init cloud provider "aws": error
 finding instance i-1111f2222f333e44c: "error listing AWS instances: \"RequestError:
 send request failed\\ncaused by: Post  net/http: TLS handshake timeout\""
```

O processo `kubelet` gerará e testará continuamente o endpoint do servidor de API. O erro também pode ocorrer temporariamente durante algum procedimento que realize uma atualização contínua do cluster no ambiente de gerenciamento, como uma alteração de configuração ou atualização de versão.

Para resolver o problema, verifique a tabela de rotas e os grupos de segurança para garantir que o tráfego dos nós possa atingir o endpoint público.

InvalidClientTokenId

Se você estiver usando os perfis do IAM para contas de serviço de um pod ou DaemonSet implantado em um cluster na região da China da AWS e ainda não tiver definido a variável de ambiente `AWS_DEFAULT_REGION` na especificação, o pod ou o DaemonSet poderá receber o seguinte erro:

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation:
 The security token included in the request is invalid
```

Para resolver o problema, é necessário adicionar a variável de ambiente `AWS_DEFAULT_REGION` à especificação do pod ou DaemonSet, conforme mostrado na seguinte especificação do pod de exemplo.

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: AWS_DEFAULT_REGION
      value: "region-code"
```

Os grupos de nós devem corresponder à versão do Kubernetes antes da atualização do ambiente de gerenciamento

Antes de atualizar um ambiente de gerenciamento para uma nova versão do Kubernetes, a versão secundária dos nós gerenciados e do Fargate no cluster dove ser a mesma da versão atual do ambiente de gerenciamento. A API `update-cluster-version` do Amazon EKS rejeita solicitações até que você atualize todos os nós gerenciados do Amazon EKS para a versão atual do cluster. O Amazon EKS fornece APIs para atualizar nós gerenciados. Para obter informações sobre como atualizar a versão do grupo de nós gerenciados do Kubernetes, consulte the section called “Atualizar”. Para atualizar a versão de um nó do Fargate, exclua o pod representado pelo nó

e reimplante o pod depois de atualizar o ambiente de gerenciamento. Para obter mais informações, consulte the section called “Atualizar a versão do Kubernetes”.

Ao iniciar muitos nós, ocorrem erros `Too Many Requests`

Se você iniciar muitos nós ao mesmo tempo, poderá ver uma mensagem de erro nos logs de execução de Dados do usuário do Amazon EC2 que indica `Too Many Requests`. Isso pode ocorrer porque o ambiente de gerenciamento está sendo sobrecarregado por chamadas `describeCluster`. A sobrecarga resulta em controle de utilização, nós com falha ao executar o script de bootstrap e nós que não são completamente incorporados ao cluster.

Verifique se os argumentos `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` estão sendo transmitidos ao script de bootstrap do nó. Ao incluir esses argumentos, não há necessidade de o script bootstrap criar uma chamada `describeCluster`, o que ajuda a evitar a sobrecarga do ambiente de gerenciamento. Para obter mais informações, consulte the section called “Fornecer dados de usuário para passar argumentos para o arquivo `bootstrap.sh` incluído com uma AMI do Linux/Bottlerocket otimizada para Amazon EKS”.

Resposta de erro HTTP 401 - Não autorizado em solicitações do servidor de API do Kubernetes

Esses erros serão mostrados se um token de conta de serviço de um pod tiver expirado em um cluster.

O servidor de API do Kubernetes do cluster do Amazon EKS rejeita solicitações com tokens com mais de 90 dias. Em versões anteriores do Kubernetes, os tokens não tinham expiração. Isso significa que os clientes que dependem desses tokens precisam atualizá-los em até uma hora. Para evitar que o servidor de API do Kubernetes rejeite sua solicitação devido a um token inválido, a versão do SDK do cliente do Kubernetes usada por sua workload deve ser igual ou posterior às seguintes versões:

- Versão do Go `0.15.7` e superiores
- Versão do Python `12.0.0` e posteriores
- Versão Java `9.0.0` e posterior
- Versão JavaScript `0.10.3` e posterior
- Ramificação Ruby `master`

- Versão do Haskell `0.3.0.0`
- Versão `7.0.5` e mais recente do C#

Você pode identificar todos os pods existentes no cluster que estão usando tokens obsoletos. Para obter mais informações, consulte the section called “Tokens de contas de serviço”.

A versão da plataforma Amazon EKS está mais de duas versões atrás da versão atual da plataforma

Isso pode acontecer quando o Amazon EKS não consegue atualizar automaticamente a versão da plataforma do cluster. Embora existem muitas causas para isso, veja a seguir algumas das causas comuns. Se algum desses problemas se aplicar ao cluster, ele ainda poderá funcionar; apenas a versão da plataforma não será atualizada pelo Amazon EKS.

Problema

O perfil do IAM do cluster foi excluído; essa função foi especificada quando o cluster foi criado. Você pode verificar qual função foi especificada com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut
 -d / -f 2
```

Veja abaixo um exemplo de saída.

```
eksClusterRole
```

Solução

Crie um novo perfil do IAM do cluster com o mesmo nome.

Problema

Uma sub-rede especificada durante a criação do cluster foi excluída; as sub-redes a serem usadas com o cluster foram especificadas durante a criação do cluster. Você pode verificar quais sub-redes foram especificadas com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

Veja abaixo um exemplo de saída.

```
[
"subnet-EXAMPLE1",
"subnet-EXAMPLE2"
]
```

Solução

Confirme se os IDs de sub-rede existem na conta.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --query
 "Subnets[*].SubnetId"
```

Veja abaixo um exemplo de saída.

```
[
"subnet-EXAMPLE3",
"subnet-EXAMPLE4"
]
```

Se os IDs de sub-rede retornados na saída não corresponderem aos IDs de sub-rede especificados quando o cluster foi criado, se você quiser que o Amazon EKS atualize o cluster, será necessário alterar as sub-redes usadas pelo cluster. Isso ocorre porque, se você especificou mais de duas sub-redes ao criar o cluster, o Amazon EKS selecionará aleatoriamente as sub-redes que você especificou para a criação de novas interfaces. Essas interfaces de rede permitem que o ambiente de gerenciamento se comunique com os nós. O Amazon EKS não atualizará o cluster se a sub-rede selecionada não existir. Você não tem controle sobre qual das sub-redes especificadas na criação do cluster o Amazon EKS escolhe para criar uma nova interface de rede.

Quando você inicia uma atualização de versão do Kubernetes para o cluster, a atualização pode falhar pelo mesmo motivo.

Problema

Um grupo de segurança especificado durante a criação do cluster foi excluído. Se você especificou grupo de segurança durante a criação do cluster, poderá ver os IDs com o comando a seguir. Substitua *my-cluster* pelo nome do cluster.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.securityGroupIds
```

Veja abaixo um exemplo de saída.

```
[
    "sg-EXAMPLE1"
]
```

Se `[]` for retornado, nenhum grupo de segurança foi especificado quando o cluster foi criado e a falta de um grupo de segurança não é o problema. Se grupos de segurança forem retornados, confirme se eles existem na conta.

Solução

Confirme se esses grupos de segurança existem na conta.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$vpc_id" --query
 "SecurityGroups[*].GroupId"
```

Veja abaixo um exemplo de saída.

```
[
"sg-EXAMPLE2"
]
```

Se os IDs dos grupos de segurança retornados na saída não corresponderem aos IDs dos grupos de segurança especificados quando o cluster foi criado, se você quiser que o Amazon EKS atualize o cluster, será necessário alterar os grupos de segurança usados pelo cluster. O Amazon EKS não atualizará um cluster se os IDs de grupo de segurança especificados na criação do cluster não existirem.

Quando você inicia uma atualização de versão do Kubernetes para o cluster, a atualização pode falhar pelo mesmo motivo.

- Você não tem pelo menos seis (embora recomendemos 16) endereços IP disponíveis em cada uma das sub-redes especificadas ao criar o cluster. Se não tiver endereços IP disponíveis

suficientes na sub-rede, você precisará liberar endereços IP na sub-rede ou alterar as sub-redes usadas pelo cluster para usar sub-redes com endereços IP disponíveis suficientes.

- Você habilitou a criptografia de segredos durante a criação do cluster, e a chave do AWS KMS especificada foi excluída. Se você quiser que o Amazon EKS atualize o cluster, precisará criar um novo cluster

Perguntas frequentes sobre a integridade do cluster e códigos de erro com caminhos de resolução

O Amazon EKS detecta problemas com os clusters do EKS e com a infraestrutura do cluster e os armazena no objeto integridade do recurso do cluster do EKS. Você pode detectar, solucionar problemas e resolver problemas de cluster mais rapidamente com a ajuda das informações de integridade do cluster. Isso permite que você crie ambientes de aplicações mais seguros e atualizados. Além disso, pode ser impossível atualizar para versões mais recentes do Kubernetes ou instalar atualizações de segurança do Amazon EKS em um cluster dogradado como resultado de problemas com a infraestrutura ou a configuração do cluster necessária. O Amazon EKS pode levar 3 horas para detectar problemas ou detectar que um problema foi resolvido.

A integridade de um cluster do Amazon EKS é uma responsabilidade compartilhada entre o Amazon EKS e seus usuários. Você é responsável pela infraestrutura de pré-requisitos dos perfis do IAM e das sub-redes do Amazon VPC, bem como por outras infraestruturas necessárias, que devem ser fornecidas com antecedência. O Amazon EKS detecta mudanças na configuração dessa infraestrutura e do cluster.

Para acessar a integridade do seu cluster no console do Amazon EKS, procure uma tabela chamada Problemas de integridade na guia Problemas de integridade do cluster do painel de observabilidade acessado na página de detalhes do cluster do Amazon EKS. Esses dados também estarão disponíveis chamando a ação `DescribeCluster` na API do EKS, por exemplo, na interface de linha de comando AWS.

Por que devo usar esse recurso?

> Você terá maior visibilidade da integridade do cluster do Amazon EKS, diagnosticará e corrigirá rapidamente quaisquer problemas, sem precisar perder tempo depurando ou abrindo casos de suporte da AWS. Por exemplo, se você excluir acidentalmente uma sub-rede do cluster do Amazon EKS, o Amazon EKS não poderá criar interfaces de rede entre contas e comandos da AWS CLI do Kubernetes, como exec do `kubectl` ou logs do `kubectl`. Eles falharão com o erro:

`Error from server: error dialing backend: remote error: tls: internal error.` Agora você verá um problema de integridade do Amazon EKS que diz: `subnet-da60e280 was deleted: could not create network interface`.

Como este recurso se relaciona ou funciona com outros serviços da AWS?

Os perfis do IAM e as sub-redes do Amazon VPC são dois exemplos de infraestrutura de pré-requisito com a qual a integridade do cluster dotecta problemas. O recurso retornará informações detalhadas se esses recursos não estiverem configurados corretamente.

Um cluster com problemas de integridade gera custos?

Sim, cada cluster do Amazon EKS é cobrado de acordo com o preço padrão do Amazon EKS. O recurso de integridade do cluster está disponível sem custo adicional.

Esse recurso funciona com clusters do Amazon EKS em AWS Outposts?

Sim, os problemas de cluster são detectados para clusters EKS na nuvem AWS, incluindo clusters estendidos em AWS Outposts e clusters locais em AWS Outposts. A integridade do cluster não detecta problemas com o Amazon EKS Anywhere ou o Amazon EKS Distro (EKS-D).

Posso ser notificado quando novos problemas forem detectados?

Sim. A AWS envia um e-mail e uma notificação no Personal Health Dashboard quando novos problemas de integridade de clusters são detectados.

O console me avisa sobre problemas de integridade?

Sim, qualquer cluster com problemas de integridade incluirá um banner na parte superior do console.

As duas primeiras colunas são o que é necessário para os valores de resposta da API. O terceiro campo do objeto Health ClusterIssue é resourceIds, cujo retorno depende do tipo de problema.

Código	Mensagem	ResourceIds	Cluster recuperável?
SUBNET_NOT_FOUND	Não foi possível encontrar uma ou mais sub-redes que estão atualmente associadas ao seu cluster. Chame a API	IDs de sub-rede	Sim

Código	Mensagem	ResourceIds	Cluster recuperável?
	update-cluster-config do Amazon EKS para atualizar sub-redes.		
SECURITY_ GROUP_NOT _FOUND	Não foi possível encontrar um ou mais grupos de segurança que estão atualment e associados ao seu cluster. Chame a API update-cluster-config do Amazon EKS para atualizar grupos de segurança	IDs de grupos de segurança	Sim
IP_NOT_AVAILABLE	Uma ou mais sub-redes associada s ao seu cluster não têm endereços IP disponíveis suficientes para o Amazon EKS realizar operações de gerenciamento de cluster. Libere endereços na(s) sub-rede(s) ou associe diferentes sub-redes ao seu cluster usando a API update-cluster-config do Amazon EKS.	IDs de sub-rede	Sim

Código	Mensagem	ResourceIds	Cluster recuperável?
VPC_NOT_FOUND	Não foi possível encontrar a VPC associada ao seu cluster. Você deve excluir e recriar seu cluster.	ID da VPC	Não
ASSUME_ROLE_ACCESS_DENIED	Seu cluster não está usando o perfil vinculado ao serviço do Amazon EKS. Não foi possível assumir a função associada ao seu cluster para realizar as operações de gerenciamento necessárias do Amazon EKS. Confira se o perfil já existe e apresenta a política de confiança necessária.	O perfil do IAM do cluster	Sim

Código	Mensagem	ResourceIds	Cluster recuperável?
PERMISSIO N_ACCESS_DENIED	Seu cluster não está usando o perfil vinculado ao serviço do Amazon EKS. A função associada ao seu cluster não concede permissõe s suficientes para que o Amazon EKS execute as operações de gerenciamento necessárias. Verifique as políticas anexadas à função de cluster e se alguma política de negação separada foi aplicada.	O perfil do IAM do cluster	Sim
ASSUME_RO LE_ACCESS _DENIED_U SING_SLR	Não foi possível assumir o perfil vinculado ao serviço de gerenciam ento de clusters do Amazon EKS. Confira se o perfil já existe e apresenta a política de confiança necessária.	A função vinculada ao serviço do Amazon EKS	Sim

Código	Mensagem	ResourceIds	Cluster recuperável?
PERMISSIO N_ACCESS_ DENIED_USING_SLR	O perfil vinculado ao serviço de gerenciam ento de clusters do Amazon EKS não concede permissõe s suficientes para que o Amazon EKS execute as operações de gerenciamento necessárias. Verifique as políticas anexadas à função de cluster e se alguma política de negação separada foi aplicada.	A função vinculada ao serviço do Amazon EKS	Sim
OPT_IN_REQUIRED	Sua conta não tem uma assinatura de serviço do Amazon EC2. Atualize as assinaturas da sua conta na página de configurações da conta.	N/D	Sim
STS_REGIO NAL_ENDPO INT_DISABLED	O endpoint regional do STS está desabilit ado. Habilite o endpoint do Amazon EKS para realizar as operações de gerenciamento de cluster necessárias.	N/D	Sim

Código	Mensagem	ResourceIds	Cluster recuperável?
KMS_KEY_DISABLED	A chave AWS KMS associada ao seu cluster está desativada. Habilite novamente a chave para recuperar o cluster.	O Arn da chave do KMS	Sim
KMS_KEY_NOT_FOUND	Não foi possível encontrar a chave AWS KMS associada ao seu cluster. Você precisa excluir e recriar o cluster.	O ARN da chave do KMS	Não
KMS_GRANT_REVOKED	As concessões para a chave AWS KMS associada ao seu cluster são revogadas. Você precisa excluir e recriar o cluster.	O Arn da chave do KMS	Não

Estender os recursos do Amazon EKS com projetos de código aberto

Esses projetos de código aberto ampliam a funcionalidade de clusters do Kubernetes em execução na AWS ou fora dela, incluindo clusters gerenciados pelo Amazon EKS.

Suporte para software implantado no EKS

Ao analisar a documentação do Amazon EKS, você encontrará referências a várias ferramentas e softwares de código aberto em todos os nossos procedimentos e exemplos. Essas ferramentas incluem o Servidor de métricas do Kubernetes e o Gerenciador de certificados.

Observe que qualquer software de terceiros ou de código aberto que você escolher implantar estará fora do escopo de seus Contratos do AWS Support. Um benefício de usar o Kubernetes é a comunidade ativa de código aberto. Recomendamos trabalhar diretamente com as comunidades de código aberto relevantes e os mantenedores do projeto para estabelecer canais de suporte apropriados para esses componentes. Para obter mais informações, consulte os projetos de mudanças e incubação associados à Cloud Native Computing Foundation (CNCF).

O ecossistema do Kubernetes inclui vários projetos e componentes que vêm com diferentes níveis de suporte da comunidade, tempos de resposta e casos de uso pretendidos. Ao implementar essas tecnologias junto com o EKS, certifique-se de entender a matriz de suporte para cada componente.

A AWS mantém os componentes de código aberto que integramos no ambiente de gerenciamento do EKS. Isso inclui nosso abrangente pipeline de segurança que cobre verificação de compilação, varredura de vulnerabilidades, testes de validação e gerenciamento de patches para todas as imagens de contêineres e binários que distribuímos. Por exemplo, a AWS é responsável pelo servidor de API do Kubernetes. O servidor de API do Kubernetes é coberto pelo Acordo de Serviço do Amazon EKS. Você pode usar seu Plano de Suporte da Amazon Web Services para resolver problemas com o servidor de API do Kubernetes ou obter orientação geral.

Você precisa analisar cuidadosamente o suporte oferecido para os vários complementos do Amazon EKS. Os complementos da AWS são o único tipo de complemento do Amazon EKS totalmente suportado pela AWS. Os complementos do AWS Marketplace são suportados principalmente pelos parceiros da AWS. Os complementos da comunidade recebem suporte básico do ciclo de vida da AWS. Para obter mais informações, consulte Suporte para complementos.

Todo complemento do EKS, independentemente do tipo, recebe suporte básico ao ciclo de vida do EKS, incluindo os complementos do Marketplace. O suporte básico do ciclo de vida inclui a instalação e a desinstalação do complemento. Para obter mais informações sobre os tipos de complementos do Amazon EKS disponíveis e os níveis de suporte associados, consulte Escopo do suporte para complementos do Amazon EKS. Para ver os complementos totalmente suportados pelo AWS, consulte os complementos da Amazon Web Services.

- Para obter mais informações sobre nossas práticas de segurança e limites de suporte, consulte Segurança no Amazon EKS.
- Para obter mais informações sobre os complementos da comunidade e do AWS Marketplace disponíveis por meio dos complementos do Amazon EKS, consulte Suporte a complementos do EKS.

Ferramentas de gerenciamento

Ferramentas de gerenciamento relacionadas para clusters do Amazon EKS e do Kubernetes.

eksctl

`eksctl`O é uma simples ferramenta de CLI para criar clusters no Amazon EKS.

- URL do projeto
- Documentação do projeto
- Blog de código aberto da AWS: eksctl: cluster do Amazon EKS com um comando

AWS Controllers for Kubernetes

Com o AWS Controllers para Kubernetes, você pode criar e gerenciar os recursos da AWS diretamente do seu cluster do Kubernetes.

Disponível nas funcionalidades do EKS.

- URL do projeto
- Blog de código aberto da AWS: o operador de serviço da AWS para Kubernetes já está disponível

kro (Kube Resource Orchestrator)

Com o kro, você pode criar APIs personalizadas do Kubernetes que agrupam vários recursos em abstrações de alto nível. As equipes responsáveis pela plataforma podem definir padrões reutilizáveis com barreiras de proteção, enquanto as equipes responsáveis pela aplicação usam APIs simples e de alto nível para provisionar e gerenciar recursos.

Disponível nas funcionalidades do EKS.

- URL do projeto
- Documentação do projeto

Argo CD

O Argo CD é uma ferramenta de entrega contínua declarativa e baseada em GitOps para Kubernetes. Essa ferramenta monitora continuamente os repositórios do Git e sincroniza automaticamente as alterações com seus clusters.

Disponível nas funcionalidades do EKS.

- URL do projeto
- Documentação do projeto

Flux CD

Flux é uma ferramenta que você pode usar para gerenciar a configuração do cluster usando o Git. Ele usa um operador no cluster para acionar implantações dentro do Kubernetes. Para obter mais informações sobre operadores, consulte OperatorHub.io no GitHub.

- URL do projeto
- Documentação do projeto

CDK para Kubernetes

Com o CDK para Kubernetes (cdk8s), você pode definir as aplicações e os componentes do Kubernetes usando linguagens de programação familiares. As aplicações cdk8s sintetizam em manifestos padrão do Kubernetes, que podem ser aplicadas a qualquer cluster do Kubernetes.

- URL do projeto
- Documentação do projeto
- AWS Blog de contêineres: Introdução às APIs cdk8s+: orientadas pela intenção para objetos do Kubernetes

Redes

Projetos de redes relacionados para clusters do Amazon EKS e do Kubernetes.

Plug-in CNI da Amazon VPC para Kubernetes

O Amazon EKS oferece suporte a redes de VPC nativas por meio do plug-in CNI da Amazon VPC para Kubernetes. O plug-in atribui um endereço IP da VPC a cada pod.

- URL do projeto
- Documentação do projeto

AWS Load Balancer Controller para Kubernetes

O AWS Load Balancer Controller ajuda a gerenciar AWS Elastic Load Balancers para um cluster do Kubernetes. Ele satisfaz os recursos do Kubernetes Ingress provisionando os AWS Application Load Balancers. Ele satisfaz os recursos do serviço do Kubernetes provisionando os AWS Network Load Balancers.

- URL do projeto
- Documentação do projeto

ExternalDNS

O ExternalDNS sincroniza os serviços expostos do Kubernetes e entra com provedores de DNS incluindo o Amazon Route 53 e o AWS Service Discovery.

- URL do projeto
- Documentação do projeto

Machine learning

Projetos de machine learning relacionados para clusters do Amazon EKS e do Kubernetes.

Kubeflow

Um toolkit de machine learning para o Kubernetes.

- URL do projeto
- Documentação do projeto
- Blog de código aberto da AWS: Kubeflow no Amazon EKS

ajuste de escala automático

Projetos de Auto Scaling relacionados para clusters do Amazon EKS e do Kubernetes.

Cluster autoscaler

O Autoscaler do cluster é uma ferramenta que ajusta automaticamente o tamanho do cluster do Kubernetes com base na CPU e na pressão de memória.

- URL do projeto
- Documentação do projeto
- Workshop do Amazon EKS: escalador automático de clusters

Karpenter

O Karpenter é um autoscaler de nós do Kubernetes criado para oferecer flexibilidade, performance e simplicidade.

- URL do projeto
- Documentação do projeto
- Workshop do Amazon EKS: Karpenter

Escalator

O Escalator é um autoscaler horizontal otimizado para trabalho ou lote para o Kubernetes.

- URL do projeto
- Documentação do projeto

Monitoramento

Projetos de monitoramento relacionados para clusters do Amazon EKS e do Kubernetes.

Prometheus

O Prometheus é um toolkit de código aberto para alertas e monitoramento de sistemas.

- URL do projeto
- Documentação do projeto
- Workshop o Amazon EKS: https://eksworkshop.com/intermediate/240_monitoring/

Integração contínua/implantação contínua

Projetos de CI/CD imperativos associados a clusters do Amazon EKS e do Kubernetes.

Jenkins X

Solução de CI/CD para aplicações de nuvem modernas nos clusters do Amazon EKS e do Kubernetes.

- URL do projeto
- Documentação do projeto

Saiba mais sobre os novos recursos e o roteiro do Amazon EKS

Você pode saber mais sobre os novos recursos do Amazon EKS rolando para o feed Novidades na página de Novidades da AWS. Você também pode revisar o roadmap (roteiro) no GitHub, que permite que você saiba sobre os próximos recursos e prioridades para que possa planejar como deseja usar o Amazon EKS no futuro. Você pode nos fornecer feedback direto sobre as prioridades do roteiro.

Histórico do documento

A tabela a seguir descreve as principais atualizações e os novos recursos do Guia do usuário do Amazon EKS. Para receber notificações quando esta tabela receber uma nova entrada, você pode assinar o seguinte URL com um leitor de RSS:

```
https://docs.aws.amazon.com/eks/latest/userguide/doc-history.rss
```

Alteração	Descrição	Data
Recursos do Amazon EKS	Agora, o Amazon EKS disponibiliza funcionalidades de cluster totalmente gerenciadas com as funcionalidades do EKS. O recurso inclui suporte para implantação contínua declarativa com o Argo CD, gerenciamento de recursos da AWS com o AWS Controllers for Kubernetes, e composição e orquestração de recursos do Kubernetes com o Kube Resource Orchestrator.	30 de novembro de 2025
Nova política gerenciada pela AWS	O Amazon EKS lançou uma nova política gerenciada, a `AmazonEKSMCPReadOnlyAccess`, para habilitar ferramentas destinadas somente à leitura no Servidor MCP do Amazon EKS para observabilidade e solução de problemas. Para obter mais informações, consulte Atualizações do Amazon EKS	21 de novembro de 2025

	para políticas gerenciadas pela AWS.	
Observabilidade da rede	Agora, o Amazon EKS disponibiliza uma observabilidade aprimorada da rede de contêineres, contando com monitoramento de performance e visibilidade do tráfego das workloads.	19 de novembro de 2025
AWS Atualização da política gerenciada	A permissão `ec2:CopyVolumes` foi adicionada à política `AmazonEBSCSIDriverPolicy`, possibilitando que o driver CSI do EBS realize a cópia direta de volumes do EBS.	17 de novembro de 2025
AWS Atualização da política gerenciada	Foram adicionadas as permissões `ec2:DescribeRouteTables` e `ec2:DescribeNetworkAcls` para a política `AmazonEKSServiceRolePolicy`. Isso permite que o Amazon EKS detecte problemas de configuração em tabelas de rotas da VPC e ACLs de rede para nós híbridos, como parte dos insights do cluster.	22 de outubro de 2025

Atualização do perfil vinculado ao serviço	Foi adicionada a permissão `ssmmessages:OpenDataChannel` para a política `AmazonEKSConnectorServiceRolePolicy` .	15 de outubro de 2025
Versão do Kubernetes `1.34`	Suporte adicionado para o Kubernetes versão `1.34` para novos clusters e atualizações de versão.	2.º de outubro de 2025
Reparo automático de nós configurável	Agora, o Amazon EKS oferece um controle mais granular sobre o comportamento da funcionalidade de reparo automático de nós.	22 de setembro de 2025
Atualizar insights do cluster	Você agora pode atualizar manualmente os insights do cluster.	27 de agosto de 2025
AWS Atualização da política gerenciada	Adicionada uma permissão a `AmazonEKSServiceRolePolicy` . Esse perfil pode anexar uma nova política de acesso `AmazonEKSEventPolicy` . Permissões restritas para `ec2:DeleteLaunchTemplate` e `ec2:TerminateInstances` .	26 de agosto de 2025

Prevenção do problema do confused deputy entre serviços	Adição de um tópico com um exemplo de uma política de confiança que você pode aplicar para prevenção do problema do confused deputy entre serviços. O Amazon EKS aceita as condições `aws:SourceArn` e `aws:SourceAccount` na política de confiança de um perfil de cluster do EKS.	19 de agosto de 2025
Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança . Isto inclui novas versões de patch do Kubernetes `1.33.2`, `1.32.6`, `1.31.10` e `1.30.14`.	30 de julho de 2025

Recurso de CNI da VPC de várias NICs para pods com vários hospedeiros	O Amazon EKS adiciona pods com vários hospedeiros à CNI da VPC. Agora é possível configurar uma workload e os endereços IP atribuídos pela CNI da VPC de cada NIC na instância do EC2 para cada pod. A aplicação pode fazer conexões simultâneas para usar a largura de banda de cada NIC. Cada interface de rede é configurada na mesma sub-rede e nos mesmos grupos de segurança do nó. Anteriormente, você precisava usar o Multus CNI para executar várias CNIs e criar pods com várias conexões. A documentação e as etapas para fazer isso foram movidas para eks/latest/userguide/pod-multus.html.	15 de julho de 2025
Atualização de conteúdo da solução de problemas da CNI da VPC	Página de solução de problemas expandida para a política de rede do Kubernetes na CNI da VPC. Adicionadas as permissões de CRDs e RBAC e 12 problemas conhecidos.	30 de junho de 2025

Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança . Não há novas versões de patch do Kubernetes nessa versão da plataforma.	26 de junho de 2025
AWS Atualização da política gerenciada	Adicionada a permissão `ssmmessages:OpenDataChannel` a `AmazonEKSLocalOutpostServiceRolePolicy` .	26 de junho de 2025
AWS Atualização da política gerenciada	Adicionadas permissões a `AmazonEKSServiceRolePolicy` e `AmazonEKSComputePolicy` para permitir que o Modo automático do Amazon EKS inicialize instâncias usando as reservas de capacidade sob demanda do EC2 em sua conta. Além disso, foram adicionadas as permissões a `AmazonEKSComputePolicy` para que o Amazon EKS crie o perfil vinculado ao serviço de instâncias spot do EC2 em seu nome.	20 de junho de 2025
Atualizações da política gerenciada	Adicionada a política `AmazonEKSDashboardConsoleReadOnly` .	19 de junho de 2025

Atualização do modo automático do Amazon EKS para `NodeClass`	O modelo do `NodeClass` para nós do Modo automático adicionou uma configuração para sub-redes de pods separadas. Isso adiciona as chaves opcionais `podSubnetSelectorTerms` e `podSecurityGroupSelectorTerms` para definir as sub-redes e os grupos de segurança para os pods.	13 de junho de 2025
Direcionar perfis secundários e entre contas com as Identidades de Pods do EKS	O Amazon EKS adiciona perfis do IAM de destino às Identidades de Pods do EKS para encadeamento automatizado de perfis. Você pode usar isso para assumir automaticamente um perfil em outra conta e a Identidade de Pods do EKS alternará as credenciais temporárias. Cada associação de Identidade de Pods deve ter um perfil do IAM na mesma conta para assumir primeiro, depois ela usa esse perfil para assumir o perfil de destino.	11 de junho de 2025
Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança. Não há novas versões de patch do Kubernetes nessa versão da plataforma.	11 de junho de 2025

Expansão de regiões da AWS do Amazon EKS	O Amazon EKS agora está disponível na região Ásia-Pacífico (Taipei) (`ap-east-2`) da AWS.	6 de junho de 2025
`IPv6` Controle de acesso `IPv6` para endpoints públicos de pilha dupla para novos clusters	O Amazon EKS adiciona blocos CIDR `IPv6` para controlar o acesso ao endpoint público do cluster para novos clusters `IPv6`. Anteriormente, você só podia adicionar blocos CIDR `IPv4` para permitir o tráfego para o endpoint público do cluster, mesmo para endpoints de cluster de pilha dupla.	5 de junho de 2025
Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança. Isso inclui novas versões de patch do Kubernetes `1.32.5`, `1.31.9` e `1.30.13`.	30 de maio de 2025
Novos insights de clusters para o EKS Hybrid Nodes	O Amazon EKS adiciona novos insights de clusters que verificam a configuração dos nós híbridos. Essas verificações de insights alertarão você sobre problemas com nós e pods on-premises e a configuração de rede remota do cluster.	29 de maio de 2025

Versão do Kubernetes `1.33`	Suporte adicionado para o Kubernetes versão `1.33` para novos clusters e atualizações de versão.	29 de maio de 2025
Suporte ao complemento para o driver CSI do Amazon FSx	Você agora pode usar o Console de Gerenciamento da AWS, a AWS CLI e a API para gerenciar o driver CSI do Amazon FSx.	23 de maio de 2025
Editar os extratores do Prometheus no console	Você agora pode editar os extratores do Amazon Managed Service for Prometheus no console do Amazon EKS.	22 de maio de 2025
Atualizações da política gerenciada	Adicionada a política `AmazonEKSDashboard ServiceRolePolicy` .	21 de maio de 2025
Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança . Não há novas versões de patch do Kubernetes nessas versões da plataforma.	16 de maio de 2025
Novas páginas sobre performance do Amazon FSx para Lustre	Adição de novos tópicos com detalhes sobre como otimizar a performance do Amazon FSx para Lustre.	2 de maio de 2025

Atualização do modo automático do Amazon EKS para `NodeClass`	O modelo `NodeClass` para nós do modo automático agora inclui a configuração para proxies de rede de encaminhamento. Isso adiciona a chave opcional `advancedNetworking` para configurar seu proxy HTTPS.	30 de abril de 2025
Bottlerocket para nós híbridos	O Bottlerocket agora está disponível para o EKS Hybrid Nodes.	29 de abril de 2025
Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança. Não há novas versões de patch do Kubernetes nessas versões da plataforma.	29 de abril de 2025
Novas páginas de conceitos para redes híbridas	Adicionadas páginas de conceitos do EKS Hybrid Nodes. Eles abrangem a rede on-premises e na nuvem em detalhes com diagramas.	18 de abril de 2025

AWS Atualização da política gerenciada	Adicionadas permissões à política `AmazonEKSClusterPolicy` para permitir que o Amazon EKS use interfaces de rede elástica criadas pela CNI da VPC. Isso é necessário para que o EKS possa limpar as interfaces de rede elástica que serão esquecidas se a CNI da VPC for encerrada inesperadamente.	16 de abril de 2025
AWS Atualização da política gerenciada	Adicionadas permissões à política `AmazonEKSServiceRolePolicy` para permitir que os clientes de IA e ML do EKS adicionem regras de saída ao grupo de segurança padrão do cluster do EKS.	14 de abril de 2025
Integridade do nó para EKS Hybrid Nodes	Você pode usar `eks-node-monitoring-agent` em nós híbridos, a partir da versão `1.2.0-eksbuild.1` . Execute `eks-node-monitoring-agent` como um complemento do Amazon EKS para detectar e mostrar problemas de integridade.	31 de março de 2025

EKS Hybrid Nodes para clusters existentes	Agora você pode adicionar, alterar ou remover a configuração de nós híbridos dos clusters existentes. Anteriormente, você só podia adicionar a configuração de nós híbridos aos novos clusters quando os criava. Com o Amazon EKS Hybrid Nodes, você pode usar a infraestrutura on-premises e de borda como nós nos clusters do Amazon EKS. A AWS gerencia o ambiente de gerenciamento do Kubernetes hospedado na AWS do cluster do Amazon EKS, e você gerencia os nós híbridos que são executados nos ambientes on-premises ou de borda.	31 de março de 2025
Reversão: evite atualizações acidentais com insights de clusters	O Amazon EKS reverteu temporariamente um recurso que exigia que você usasse um sinalizador `--force` para atualizar o cluster quando havia certos problemas de insights do cluster. Para obter mais informações, consulte Temporary rollback of enforcing upgrade insights on update cluster version no GitHub.	28 de março de 2025

AMIs do FIPS do Bottlerocket	As AMIs do FIPS do Bottlerocket agora estão disponíveis em grupos padrão de nós gerenciados.	27 de março de 2025
AWS Atualização da política gerenciada	Foram adicionadas permissões à `AmazonEKSServiceRolePolicy` para que o Amazon EKS possa encerrar instâncias do EC2 criadas pelo Modo automático. Foram adicionadas permissões à `AmazonEKSServiceRolePolicy` para que o Amazon EKS possa encerrar instâncias do EC2 criadas pelo Modo automático.	28 de fevereiro de 2025
Estratégias de atualização para grupos de nós gerenciados	Você agora pode usar estratégias de atualização para configurar o processo de atualização de versão para grupos de nós gerenciados. Isso introduz a estratégia de atualização mínima para encerrar os nós antes de criar novos, o que é útil em ambientes com restrição de capacidade. A estratégia de atualização padrão mantém o comportamento existente.	27 de janeiro de 2025
Versão do Kubernetes `1.32`	Suporte adicionado para o Kubernetes versão `1.32` para novos clusters e atualizações de versão.	23 de janeiro de 2025

Expansão de regiões da AWS do Amazon EKS	Agora, o Amazon EKS está disponível nas regiões da AWS Ásia-Pacífico (Tailândia) (`ap-southeast-7`) e México (Centro) (`mx-central-1`). O modo automático do EKS e os endpoints da VPC para a API do EKS não estão disponíveis em nenhuma das regiões.	14 de janeiro de 2025
AWS Atualização da política gerenciada	Diversas permissões foram adicionadas à política `AmazonEBSCSIDriverPolicy` para permitir que o driver da CSI do Amazon EBS restaure todos os snapshots, habilite a restauração rápida de snapshots (FSR) nos volumes do EBS e modifique as etiquetas dos volumes.	13 de janeiro de 2025
AWS Atualização da política gerenciada	Foram adicionadas permissões a `AmazonEKSLoadBalancingPolicy`.	26 de dezembro de 2024

Insights de clusters atualizados	Agora, os insights de atualização do Amazon EKS alertarão sobre mais problemas de integridade de clusters e compatibilidade de versões. Eles podem detectar problemas entre diferentes componentes do Kubernetes e do Amazon EKS, como o `kubelet`, o `kube-proxy` e complementos do Amazon EKS.	20 de dezembro de 2024
Agente de monitoramento de nós e reparo automático	Você pode usar o novo `eks-node-monitoring-agent` como um complemento do Amazon EKS para detectar e mostrar problemas de integridade. Você também pode habilitar o reparo automático de nós para substituir automaticamente os nós quando forem detectados problemas.	16 de dezembro de 2024

Amazon EKS Hybrid Nodes	Agora você pode executar nós on-premises conectados aos clusters do Amazon EKS. Com o Amazon EKS Hybrid Nodes, você pode usar a infraestrutura on-premises e de borda como nós nos clusters do Amazon EKS. A AWS gerencia o ambiente de gerenciamento do Kubernetes hospedado na AWS do cluster do Amazon EKS, e você gerencia os nós híbridos que são executados nos ambientes on-premises ou de borda.	1.º de dezembro de 2024
Modo automático do Amazon EKS	O Modo automático do Amazon EKS automatiza totalmente o gerenciamento da infraestrutura de clusters do Kubernetes para computação, armazenamento e rede na AWS. Ele simplifica o gerenciamento do Kubernetes provisionando automaticamente a infraestrutura, selecionando as instâncias de computação ideais, escalando dinamicamente os recursos, otimizando continuamente os custos, aplicando patches aos sistemas operacionais e integrando-se aos serviços de segurança da AWS.	1.º de dezembro de 2024

Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança . Isto inclui novas versões de patch do Kubernetes `1.31.2`, `1.30.6`, `1.29.10` e `1.28.15`.	22 de novembro de 2024
AWS Atualização da política gerenciada	Foi atualizado o `AWSServiceRoleForAmazonEKSNodegroup` para compatibilidade com as regiões da China.	22 de novembro de 2024
A versão 1.30 do Kubernetes já está disponível para clusters locais no AWS Outposts	Você agora pode criar um cluster local do Amazon EKS em um AWS Outposts usando a versão 1.30 do Kubernetes.	21 de novembro de 2024
AWS Atualização da política gerenciada	O EKS atualizou a política gerenciada `AmazonEKSLocalOutpostClusterPolicy` da AWS. Foi adicionada a permissão `ec2:DescribeAvailabilityZones` para que o AWS Cloud Controller Manager no ambiente de gerenciamento do cluster possa identificar a zona de disponibilidade em que cada nó está.	21 de novembro de 2024

AMIs do Bottlerocket que usam FIPS 140-3	As AMIs do Bottlerocket pré-configuradas estão disponíveis para o uso de módulos de criptografia validados por FIPS 140-3. Inclui os módulos Amazon Linux 2023 Kernel Crypto API Cryptographic e AWS-LC Cryptographic.	20 de novembro de 2024
AWS Atualização da política gerenciada	Foi atualizada a política `AWSServiceRoleForAmazonEKSNodegroup` para permitir `ec2:RebootInstances` para instâncias criadas por grupos de nós gerenciados pelo Amazon EKS. Foram restringidas as permissões `ec2:CreateTags` para recursos do Amazon EC2.	20 de novembro de 2024
Painel de observabilidade	O painel de observabilidade ajuda você a detectar, solucionar e remediar problemas rapidamente. Também há novas métricas fornecidas ao CloudWatch disponíveis no namespace `AWS/EKS`.	18 de novembro de 2024

AWS Atualização da política gerenciada	O EKS atualizou a política gerenciada `AmazonEKS ServiceRolePolicy` da AWS. Foram adicionadas permissões para políticas de acesso ao EKS, gerenciam ento de balanceador de carga e limpeza automatizada de recursos de cluster.	16 de novembro de 2024
Criação de novo perfil no console para complementos com suporte a Identidade de Pods do EKS	Há novas etapas ao usar o console para criar ou atualizar complementos com suporte a Identidade de Pods do EKS, nas quais é possível gerar automaticamente perfis do IAM com o nome, a política do perfil e a política de confiança apropriados para o complemento.	15 de novembro de 2024
Grupos de nós gerenciados em zonas locais da AWS	Os grupos de nós gerenciados agora podem ser criados nas zonas locais da AWS.	15 de novembro de 2024
Há novas métricas disponíveis	Há novas métricas disponíveis no grupo de APIs `metrics.e ks.amazonaws.com` .	11 de novembro de 2024
AWS Atualização da política gerenciada	O EKS atualizou a política gerenciada `AmazonEKS ComputePolicy` da AWS. Foram atualizadas as permissões de recurso para a ação `iam:AddRo leToInstanceProfile` .	7 de novembro de 2024

AWS Atualização da política gerenciada	O EKS adicionou uma nova política gerenciada pela AWS: `AmazonEKSComputePolicy`	1.º de novembro de 2024
AWS Atualização da política gerenciada	Foram adicionadas permissões a `AmazonEKSClusterPolicy` . Foi adicionada a permissão `ec2:DescribeInstanceTopology` para permitir que o Amazon EKS anexe informações de topologia ao nó como rótulos.	1.º de novembro de 2024
AWS Atualização da política gerenciada	O EKS adicionou uma nova política gerenciada pela AWS: `AmazonEKSBlockStoragePolicy`	30 de outubro de 2024
AWS Atualização da política gerenciada	O EKS adicionou uma nova política gerenciada pela AWS: `AmazonEKSLoadBalancingPolicy`	30 de outubro de 2024
AWS Atualização da política gerenciada	Foram adicionadas permissões `cloudwatch:PutMetricData` à `AmazonEKSServiceRolePolicy` para permitir que o Amazon EKS publique métricas no Amazon CloudWatch.	29 de outubro de 2024
AWS Atualização da política gerenciada	O EKS adicionou uma nova política gerenciada pela AWS: `AmazonEKSNetworkingPolicy`	28 de outubro de 2024

Endpoints de pilha dupla para novos clusters IPv6	Conecte-se a novos clusters IPv6 com um endpoint eks-cluster. *region*.api.aws de pilha dupla. Esse endpoint é retornado quando você descreve esses clusters. kubectl e outros clientes de API do Kubernetes em IPv4, IPv6 ou ambientes de pilha dupla podem resolver e se conectar a esses endpoints para clusters públicos ou privados.	21 de outubro de 2024
AWS Atualização da política gerenciada	Adição de autoscaling:ResumeProcesses , autoscaling:SuspendProcesses e permissões associadas a nas regiões da AWSServiceRoleForAmazonEKSNodegroup China para integração com o Amazon Application Recovery Controller para EKS. Não há alterações em outras regiões.	21 de outubro de 2024
AMIs aceleradas AL2023	Agora, você pode usar instâncias NVIDIA e AWS Neuron aceleradas para AMIs baseadas no AL2023.	11 de outubro de 2024

Novo formato de origem	Mudamos para um novo formato de origem com algumas alterações de layout. Existem pequenos problemas temporários de formatação que estamos resolvendo.	10 de outubro de 2024
AWS Atualização da política gerenciada	Foram adicionadas permissões a `AmazonEKSServicePolicy` e `AmazonEKSServiceRolePolicy` . Adição de `ec2:GetSecurityGroupsForVpc` e permissões de etiqueta associadas para permitir que o EKS leia as informações do grupo de segurança e atualize as etiquetas relacionadas.	10 de outubro de 2024
AWS Atualizações de políticas gerenciadas pela : nova política	O EKS adicionou uma nova política gerenciada pela AWS.	3 de outubro de 2024
Versão do Kubernetes `1.31`	Suporte adicionado para o Kubernetes versão `1.31` para novos clusters e atualizações de versão.	24 de setembro de 2024
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	21 de agosto de 2024
A versão 1.29 do Kubernetes já está disponível para clusters locais no AWS Outposts	Você agora pode criar um cluster local do Amazon EKS em um AWS Outposts usando a versão 1.29 do Kubernetes.	20 de agosto de 2024

Identidade de Pod do EKS em AWS GovCloud (EUA)	As Identidades de Pods do Amazon EKS associam um perfil do IAM a uma conta de serviço do Kubernetes. Com esse recurso, você não precisa mais fornecer permissões estendidas ao perfil do IAM do nó. Dessa forma, os pods nesse nó podem chamar as APIs da AWS. Ao contrário dos perfis do IAM para contas de serviço, as Identidades de Pods do EKS estão completamente dentro do EKS. Assim, não é preciso um provedor de identidade OIDC.	14 de agosto de 2024
Atualizações de conteúdo baseadas em cenários	Renomeamos e atualizamos os tópicos para torná-los mais orientados por cenários ao longo de todo o guia.	9 de agosto de 2024
Endpoints de interface da VPC de pilha dupla para o Amazon EKS	Agora você pode criar endpoints de interface da VPC de pilha dupla para o Amazon EKS com endereços IP e nomes DNS `IPv4` e `IPv6`.	7 de agosto de 2024

Novos endpoints de pilha dupla para as APIs do Amazon EKS com endereços IPv6	A API do EKS para criar e gerenciar clusters e os URLs do emissor OIDC para clusters têm novos endpoints de pilha dupla. O novo nome DNS da API do Amazon EKS é `eks.region.api.aws`, que é resolvido em endereços `IPv4` e `IPv6`. Os novos clusters têm um novo URL de emissor OIDC de pilha dupla (`oidc-eks.region.api.aws`).	1º de agosto de 2024
Blocos de capacidade para grupos de nós gerenciados	Você agora pode usar blocos de capacidade para grupos de nós gerenciados.	1.º de julho de 2024
Coleta de métricas de grupo do Auto Scaling habilitada por padrão	Os grupos de nós gerenciados pelo Amazon EKS agora têm as métricas de grupo do Amazon EC2 Auto Scaling habilitadas por padrão, sem custo adicional. Anteriormente, era necessário executar várias etapas para habilitar esse recurso.	28 de junho de 2024
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	27 de junho de 2024
Versão do Kubernetes `1.26`	A versão `1.26` do Kubernetes agora está com suporte estendido.	12 de junho de 2024

Melhorias nas referências de informações da AMI	Fizemos melhorias nas referências de informações de AMIs, especialmente para o Bottlerocket.	12 de junho de 2024
Versão do Kubernetes `1.30`	Suporte adicionado para o Kubernetes versão `1.30` para novos clusters e atualizações de versão.	23 de maio de 2024
Ajuste de escala automático do CoreDNS	O autoscaler do CoreDNS adaptará dinamicamente o número de réplicas da implantação do CoreDNS em um cluster do EKS com base no número de nós e núcleos de CPU. Esse recurso funciona para o CoreDNS `v1.9` e a versão da plataforma mais recente da versão `1.25` ou mais recente do EKS.	14 de maio de 2024
Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança . Isso inclui novas versões de patch do Kubernetes `1.29.4`, `1.28.9` e `1.27.13`.	14 de maio de 2024
CloudWatch Container Insights compatível com o Windows	O complemento do Amazon CloudWatch Observability Operator agora também permite o Container Insights em nós de processamento do Windows no cluster.	10 de abril de 2024

Conceitos do Kubernetes	Adição de um novo tópico sobre conceitos do Kubernetes.	5 de abril de 2024
Reestruturação do conteúdo sobre acesso e IAM	Mova as páginas existentes relacionadas aos tópicos de acesso e IAM, como mapa de configuração de autenticação, entradas de acesso, ID do pod e IRSA, para a nova seção. Revise o conteúdo da visão geral.	2 de abril de 2024
Sistema operacional do Bottlerocket compatível com o driver CSI do Amazon S3	Agora, o driver CSI do Mountpoint para Amazon S3 é compatível com o Bottlerocket.	13 de março de 2024
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	4 de março de 2024
Amazon Linux 2023	O Amazon Linux 2023 (AL2023) é um novo sistema operacional baseado no Linux que foi projetado para fornecer um ambiente seguro, estável e de alta performance para as aplicações em nuvem.	29 de fevereiro de 2024

Identidade de Pods do EKS e IRSA compatíveis com sidecars no Kubernetes 1.29	No Kubernetes 1.29, os contêineres associados estão disponíveis nos clusters do Amazon EKS. Há suporte para os contêineres associados com o uso de perfis do IAM para contas de serviço ou da Identidade de Pods do EKS. Para obter mais informaçõ es sobre sidecars, consulte Sidecar Containers na documentação do Kubernetes.	26 de fevereiro de 2024
Versão do Kubernetes 1.29	Suporte adicionado para o Kubernetes versão 1.29 para novos clusters e atualizações de versão.	23 de janeiro de 2024
Versão completa: Amazon EKS Extended Support compatível com versões do Kubernetes	A versão Extended do Kubernetes permite que você permaneça em uma versão específica do Kubernetes por mais de 14 meses.	16 de janeiro de 2024

Detecção da integridade do cluster do Amazon EKS na Nuvem AWS	O Amazon EKS detecta problemas com os clusters do Amazon EKS e com a infraestrutura dos pré-requisitos do cluster na integridade do cluster. É possível visualizar os problemas com os clusters do EKS no Console de gerenciamento da AWS e na `health` do cluster na API do EKS. Esses problemas são adicionais aos problemas detectados e exibidos pelo console. Anteriormente, a integridade do cluster estava disponível apenas para clusters locais no AWS Outposts.	28 de dezembro de 2023
Insights de cluster	Agora você pode obter recomendações sobre seu cluster com base em verificações recorrentes.	20 de dezembro de 2023
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS agora está disponível na região da AWS Oeste do Canadá (Calgary) (`ca-west-1`).	20 de dezembro de 2023

Agora você pode conceder aos perfis e usuários do IAM acesso ao cluster usando entradas de acesso	Antes da introdução das entradas de acesso, o acesso ao cluster era concedido aos perfis e usuários do IAM ao adicionar entradas ao `ConfigMap aws-auth`. Agora, cada cluster tem um modo de acesso e você pode passar a usar entradas de acesso em sua agenda. Depois de alternar os modos, você pode adicionar usuários adicionando entradas de acesso na AWS CLI, no AWS CloudFormation e nos SDKs do AWS.	18 de dezembro de 2023
Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança . Isto inclui novas versões de patch do Kubernetes `1.28.4`, `1.27.8`, `1.26.11` e `1.25.16`.	12 de dezembro de 2023
Driver CSI do Mountpoint para Amazon S3	Você agora pode instalar o driver CSI do Mountpoint para Amazon S3 em clusters do Amazon EKS.	27 de novembro de 2023

Ativar as métricas do Prometheus ao criar um cluster	No Console de gerenciam ento da AWS, você agora pode ativar as métricas do Prometheus ao criar um cluster. Você também pode ver os detalhes do extrator do Prometheus na guia Observabilidade.	26 de novembro de 2023
Amazon EKS Pod Identities	As Identidades de Pods do Amazon EKS associam um perfil do IAM a uma conta de serviço do Kubernete s. Com esse recurso, você não precisa mais fornecer permissões estendidas ao perfil do IAM do nó. Dessa forma, os pods nesse nó podem chamar as APIs da AWS. Ao contrário dos perfis do IAM para contas de serviço, as Identidad es de Pods do EKS estão completamente dentro do EKS. Assim, não é preciso um provedor de identidade OIDC.	26 de novembro de 2023
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	26 de novembro de 2023
Controlador de snapshots da CSI	Agora é possível instalar o controlador de snapshots da CSI para uso com drivers da CSI compatíveis, como o driver da CSI do Amazon EBS.	17 de novembro de 2023

Reescrever o tópico do ADOT Operator	A seção sobre o complemento do Amazon EKS compatível com o operador do ADOT era redundante com a documentação do AWS Distro para OpenTelemetry. Migramos as informações essenciais restantes para esse recurso para reduzir informações desatualizadas e inconsistentes.	14 de novembro de 2023
Complemento CoreDNS do EKS compatível com métricas do Prometheus	As versões `v1.10.1-eksbuild.5` , `v1.9.3-eksbuild.9` e `v1.8.7-eksbuild.8` do complemento do EKS para CoreDNS expõem a porta na qual o CoreDNS publicou as métricas, no serviço `kube-dns`. Isso facilita a inclusão das métricas do CoreDNS nos sistemas de monitoramento.	10 de novembro de 2023
Complemento Amazon EKS CloudWatch Observability Operator	Adição da página Amazon EKS CloudWatch Observability Operator.	6 de novembro de 2023
Blocos de capacidade para instâncias P5 autogerenciadas no Leste dos EUA (Ohio)	No Leste dos EUA (Ohio), agora é possível usar blocos de capacidade para instâncias P5 autogerenciadas.	31 de outubro de 2023

Clusters oferecem suporte à modificação de sub-redes e grupos de segurança	Você pode atualizar o cluster para alterar quais sub-redes e grupos de segurança o cluster usa. Você pode atualizar no Console de gerenciamento da AWS, na versão mais recente da AWS CLI, no AWS CloudFormation e no `eksctl` versão `v0.164.0-rc.0` ou posterior. Talvez seja necessário fazer isso para fornecer às sub-redes mais endereços IP disponíveis para atualizar com sucesso uma versão de cluster.	24 de outubro de 2023
O perfil de cluster e o perfil de grupo de nós gerenciados são compatíveis com as políticas do AWS Identity and Access Management gerenciadas pelo cliente	Você pode usar uma política do IAM personalizada no perfil de cluster, em vez da política gerenciada AmazonEKS ClusterPolicy do AWS. Além disso, você pode usar uma política do IAM personali zada no perfil de nó em um grupo de nós gerenciados, em vez da política gerenciad a AmazonEKSWorkerNod ePolicy do AWS. Faça isso para criar uma política com privilégio mínimo para atender aos rígidos requisitos de conformidade.	23 de outubro de 2023
Corrija o link para a instalação do eksctl	Corrija o link de instalação do eksctl depois que a página foi movida.	6 de outubro de 2023

Versão prévia: Amazon EKS Extended Support para versões do Kubernetes	A versão Extended do Kubernetes permite que você permaneça em uma versão específica do Kubernetes por mais de 14 meses.	4 de outubro de 2023
Remover referências à integração do AWS App Mesh	As integrações do Amazon EKS com o AWS App Mesh permanecem apenas para os clientes existentes do App Mesh.	29 de setembro de 2023
Versão do Kubernetes `1.28`	Suporte adicionado para o Kubernetes versão `1.28` para novos clusters e atualizações de versão.	26 de setembro de 2023
Os clusters existentes são compatíveis com a aplicação de políticas de rede do Kubernetes no plug-in CNI da Amazon VPC para Kubernetes	Você pode usar a política de rede do Kubernetes em clusters existentes com o plug-in CNI da Amazon VPC para Kubernetes, em vez de precisar de uma solução de terceiros. Você pode usar a política de rede do Kubernetes em clusters existentes com o plug-in CNI da Amazon VPC para Kubernetes, em vez de precisar de uma solução de terceiros.	15 de setembro de 2023

Complemento CoreDNS do Amazon EKS compatível com a modificação do PDB	Você pode modificar o `PodDisruptionBudget` do complemento do EKS para CoreDNS nas versões `v1.9.3-eksbuild.7` e mais recente e `v1.10.1-eksbuild.4` e mais recente.	15 de setembro de 2023
Compatibilidade do Amazon EKS com sub-redes compartilhadas	Novos requisitos e considerações sobre sub-redes compartilhadas para criar clusters do Amazon EKS em sub-redes compartilhadas.	7 de setembro de 2023
Atualizações de O que é o Amazon EKS?	Novos tópicos foram adicionados, Casos de uso comuns e Arquitetura. Outros tópicos foram atualizados.	6 de setembro de 2023
Aplicação de política de rede do Kubernetes no plug-in CNI da Amazon VPC para Kubernetes	Você pode usar a política de rede do Kubernetes com o plug-in CNI da Amazon VPC para Kubernetes, em vez de precisar de uma solução de terceiros. Você pode usar a política de rede do Kubernetes com o plug-in CNI da Amazon VPC para Kubernetes, em vez de precisar de uma solução de terceiros.	29 de agosto de 2023
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS agora está disponível na região da AWS Israel (Tel Aviv) (`il-central-1`).	1.º de agosto de 2023

Armazenamento temporário configurável para o Fargate	Você pode aumentar a quantidade total de armazenamento temporário para cada pod em execução no Fargate para Amazon EKS.	31 de julho de 2023
Suporte ao complemento para o driver da CSI do Amazon EFS	Agora, você pode usar o Console de gerenciamento da AWS, a AWS CLI e a API para gerenciar o driver CSI do Amazon EFS.	26 de julho de 2023
AWS Atualizações de políticas gerenciadas da - Nova política	O Amazon EKS adicionou uma nova política gerenciada da AWS.	26 de julho de 2023
As atualizações de versão do Kubernetes para 1.27, 1.26, 1.25 e 1.24 já estão disponíveis para clusters locais no AWS Outposts	As atualizações de versão do Kubernetes para 1.27.3, 1.26.6, 1.25.11 e 1.24.15 já estão disponíveis para clusters locais no AWS Outposts	20 de julho de 2023
Prefixos de IP compatíveis com nós do Windows	A atribuição de prefixos de IP aos seus nós possibilita a hospedagem de um número significativamente maior de pods em seus nós do que a atribuição de endereços IP secundários individuais aos seus nós.	6 de julho de 2023
Driver da CSI do Amazon FSx para OpenZFS	Agora é possível instalar o driver da CSI do Amazon FSx para OpenZFS em clusters do Amazon EKS.	30 de junho de 2023

Os pods em nós do Linux em clusters do `IPv4` agora podem se comunicar com endpoints do `IPv6`.	Após a atribuição de um endereço IPv6 ao seu nó, o endereço `IPv4` dos pods é o endereço de rede convertido no endereço `IPv6` do nó em que ele está sendo executado.	19 de junho de 2023
Grupos de nós gerenciados do Windows em regiões AWS GovCloud (EUA)	Nas regiões AWS GovCloud (EUA), os grupos de nós gerenciados do Amazon EKS agora podem executar contêineres do Windows.	30 de maio de 2023
Versão do Kubernetes `1.27`	Suporte adicionado para o Kubernetes versão `1.27` para novos clusters e atualizações de versão.	24 de maio de 2023
Versão do Kubernetes `1.26`	Suporte adicionado para o Kubernetes versão `1.26` para novos clusters e atualizações de versão.	11 de abril de 2023
gMSA sem domínio	Você agora pode usar o gMSA sem domínio com pods do Windows.	27 de março de 2023
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS agora está disponível na região da AWS Ásia-Pacífico (Melbourne) (`ap-southeast-4`).	10 de março de 2023
Driver da CSI do Amazon File Cache	Atualmente, é possível instalar o driver da CSI do Amazon File Cache em clusters do Amazon EKS.	3 de março de 2023

A versão 1.25 do Kubernetes já está disponível para clusters locais no AWS Outposts	Você já pode criar um cluster local do Amazon EKS em um Outpost usando as versões `1.22` a `1.25` do Kubernetes.	1 de março de 2023
Versão do Kubernetes `1.25`	Suporte adicionado para o Kubernetes versão `1.25` para novos clusters e atualizações de versão.	22 de fevereiro de 2023
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	7 de fevereiro de 2023
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível nas regiões da AWS Ásia-Pacífico (Hyderabad) (`ap-south-2`), Europa (Zurique) (`eu-central-2` e Europa (Espanha) (`eu-south-2`).	6 de fevereiro de 2023
As versões `1.21` a `1.24` do Kubernetes agora estão disponíveis para clusters locais no AWS Outposts.	Você já pode criar um cluster local do Amazon EKS em um Outpost usando as versões `1.21` a `1.24` do Kubernetes. Anteriormente, apenas a versão `1.21` estava disponível.	17 de janeiro de 2023
O Amazon EKS agora é compatível com o AWS PrivateLink	Você pode usar um AWS PrivateLink para criar uma conexão privada entre sua VPC e o Amazon EKS.	16 de dezembro de 2022

Compatibilidade do Windows com grupos de nós gerenciados	Você agora pode usar o Windows para os grupos de nós gerenciados do Amazon EKS.	15 de dezembro de 2022
Os complementos do Amazon EKS de fornecedores de software independentes agora estão disponíveis no AWS Marketplace	Agora, você pode procurar e assinar complementos do Amazon EKS de fornecedores de software independentes por meio do AWS Marketplace.	28 de novembro de 2022
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	17 de novembro de 2022
Versão do Kubernetes `1.24`	Suporte adicionado para o Kubernetes versão `1.24` para novos clusters e atualizações de versão.	15 de novembro de 2022
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS agora está disponível na região da AWS Oriente Médio (EAU) (`me-central-1`).	3 de novembro de 2022
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	24 de outubro de 2022
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	20 de outubro de 2022
Os clusters locais em AWS Outposts já estão disponíveis	Agora, você pode criar um cluster local do Amazon EKS em um Outpost.	19 de setembro de 2022

Cotas baseadas em vCPU do Fargate	O Fargate está fazendo a transição de cotas baseadas em pods para cotas baseadas em vCPU.	8 de setembro de 2022
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	31 de agosto de 2022
Monitoramento de custos	O Amazon EKS agora é compatível com o Kubecost, o que permite que você monitore os custos detalhados por recursos do Kubernetes, incluindo pods, nós, namespaces e rótulos.	24 de agosto de 2022
AWS Atualizações de políticas gerenciadas da - Nova política	O Amazon EKS adicionou uma nova política gerenciada da AWS.	24 de agosto de 2022
AWS Atualizações de políticas gerenciadas da - Nova política	O Amazon EKS adicionou uma nova política gerenciada da AWS.	23 de agosto de 2022
Recursos de tag para faturamento	Adicionada tag de alocação de custos gerada por `aws:eks:cluster-name` para todos os clusters.	16 de agosto de 2022
Curingas do perfil do Fargate	Foi adicionado suporte para curingas do perfil do Fargate nos critérios do seletor para namespaces, chaves de rótulo e valores de rótulos.	16 de agosto de 2022

Versão do Kubernetes 1.23	Suporte adicionado para o Kubernetes versão 1.23 para novos clusters e atualizações de versão.	11 de agosto de 2022
Visualizar os recursos do Kubernetes no Console de gerenciamento da AWS	Você agora pode ver informações sobre os recursos do Kubernetes implantados no cluster usando o Console de gerenciamento da AWS.	3 de maio de 2022
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS agora está disponível na região da AWS Ásia-Pacífico (Jacarta) (ap-southeast-3).	2 de maio de 2022
Página de observabilidade e suporte ao complemento ADOT	Adição da página Observabilidade e do AWS Distro for OpenTelemetry (ADOT).	21 de abril de 2022
Versão do Kubernetes 1.22	Suporte adicionado para o Kubernetes versão 1.22 para novos clusters e atualizações de versão.	4 de abril de 2022
AWS Atualizações de políticas gerenciadas da - Nova política	O Amazon EKS adicionou uma nova política gerenciada da AWS.	4 de abril de 2022
Adicionados os detalhes da aplicação de patches em pods do Fargate	Ao fazer a atualização de pods do Fargate, o Amazon EKS primeiro tenta remover os pods com base nos seus orçamentos de interrupções de pods. Você pode criar regras de eventos para reagir a remoções com falha antes que os pods sejam excluídos.	1.º de abril de 2022

Lançamento completo: suporte ao complemento para o driver da CSI do Amazon EBS	Agora, você pode usar o Console de gerenciamento da AWS, a AWS CLI e a API para gerenciar o driver CSI do Amazon EBS.	31 de março de 2022
AWS Atualização do conteúdo do Outposts	Instruções para implementar um cluster do Amazon EKS no AWS Outposts.	22 de março de 2022
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	21 de março de 2022
Suporte ao Windows `containerd`	Agora, é possível selecionar o runtime do `containerd` para nós do Windows.	14 de março de 2022
Adição de considerações sobre o Conector do Amazon EKS à documentação de segurança	Descreve o modelo de responsabilidade compartil hada no que diz respeito a clusters conectados.	25 de fevereiro de 2022
Atribuição de endereços `IPv6` aos pods e serviços	Você agora pode criar um cluster na versão `1.21` ou mais recente que atribua endereços `IPv6` aos pods e serviços.	6 de janeiro de 2022
AWS Atualizações de política gerenciada pela : atualização de uma política existente	O Amazon EKS atualizou uma política gerenciada existente da AWS.	13 de dezembro de 2021
Lançamento da versão de demonstração: suporte ao complemento para o driver da CSI do Amazon EBSSI	Agora, você pode visualizar o uso do Console de gerenciam ento da AWS, da AWS CLI e da API para gerenciar o driver CSI do Amazon EBS.	9 de dezembro de 2021

Suporte ao escalador automático Karpenter	Agora você pode usar o projeto de código aberto Karpenter para escalar seus nós automaticamente.	29 de novembro de 2021
Suporte ao filtro Fluent Bit Kubernetes no registro do Fargate	Agora você pode usar o filtro Fluent Bit Kubernetes com o registro do Fargate.	10 de novembro de 2021
Suporte ao Windows disponível no ambiente de gerenciamento	O suporte ao Windows já está disponível em seu ambiente de gerenciamento. Não é mais necessário habilitá-lo no plano de dados.	9 de novembro de 2021
Bottlerocket adicionado como um tipo de AMI para grupos de nós gerenciados	Antes, o Bottlerocket só estava disponível como uma opção de nó autogerenciado. Agora, ele pode ser configura do como um grupo de nós gerenciados, reduzindo os esforços necessários para atender aos requisitos de conformidade dos nós.	28 de outubro de 2021
Suporte ao driver DL1	As AMIs personalizadas do Amazon Linux agora oferecem suporte a workloads de aprendizado profundo para Amazon Linux 2. Essa ativação permite uma configuração de linha de base genérica on-premises ou na nuvem.	25 de outubro de 2021

Suporte a vídeo VT1	As AMIs personalizadas do Amazon Linux agora oferecem suporte para VT1 para algumas distribuições. Essa habilitação anuncia dispositivos Xilinx U30 em seu cluster do Amazon EKS.	13 de setembro de 2021
O Amazon EKS Connector já está disponível	Você pode usar o Amazon EKS Connector para registrar e conectar à AWS qualquer cluster do Kubernetes compatível e visualizá-lo no console do Amazon EKS.	8 de setembro de 2021
O Amazon EKS Anywhere já está disponível	O Amazon EKS Anywhere é uma nova opção de implantação para o Amazon EKS que você pode usar para criar e operar clusters on-premises do Kubernetes.	8 de setembro de 2021
Driver de CSI do Amazon FSx for NetApp ONTAP	Adicionado tópico que resume o driver de CSI do Amazon FSx for NetApp ONTAP e fornece links para outras referências.	2 de setembro de 2021
Grupos de nós gerenciados agora calculam automaticamente o máximo recomendado de pods do Amazon EKS para os nós	Grupos de nós gerenciados agora calculam automaticamente o máximo recomendado de pods do Amazon EKS para os nós que você implanta sem um modelo de execução, ou com um modelo de execução em que você não especificou um ID de AMI.	30 de agosto de 2021

Remover o gerenciamento de configurações de complemento do Amazon EKS sem remover o software do complemento do Amazon EKS	Agora você pode remover um complemento do Amazon EKS sem remover o software do complemento do cluster.	20 de agosto de 2021
Criar pods com vários hospedeiros usando o Multus	Agora, você pode adicionar várias interfaces de rede a um pod usando o Multus.	2 de agosto de 2021
Adicionar mais endereços IP aos nós do Amazon EC2 do Linux	Agora você pode adicionar muito mais endereços IP aos nós do Linux Amazon EC2. Isso significa que você pode executar uma densidade maior de pods em cada nó. Agora você pode adicionar muito mais endereços IP aos nós do Linux Amazon EC2. Isso significa que você pode executar uma densidade maior de pods em cada nó.	27 de julho de 2021
`containerd` Bootstrap de runtime do	A imagem de máquina da Amazon (AMI) do Amazon Linux acelerada e otimizada para o Amazon EKS agora contém um sinalizador de bootstrap que você pode usar para habilitar o runtime do `containerd` em AMIs do Bottlerocket e otimizadas para o Amazon EKS. Este sinalizador está disponível em todas as versões do Kubernetes compatíveis da AMI.	19 de julho de 2021

Versão do Kubernetes `1.21`	Suporte adicionado para o Kubernetes versão `1.21`.	19 de julho de 2021
Tópico de políticas gerenciadas adicionado	Uma lista de todas as políticas gerenciadas do IAM no Amazon EKS e alterações feitas nessas políticas desde 17 de junho de 2021.	17 de junho de 2021
Usar grupos de segurança para pods com o Fargate	Você agora pode usar grupos de segurança para pods com o Fargate, além de usá-los com nós do Amazon EC2.	1º de junho de 2021
Complementos do Amazon EKS para CoreDNS e `kube-proxy` adicionados	O Amazon EKS agora pode ajudar você a gerenciar os complementos do Amazon EKS para CoreDNS e `kube-proxy` para o cluster.	19 de maio de 2021
Versão do Kubernetes `1.20`	Suporte adicionado para o Kubernetes versão `1.20` para novos clusters e atualizações de versão.	18 de maio de 2021
AWS Load Balancer Controller `2.2.0` lançado	Agora é possível usar o AWS Load Balancer Controller para criar balanceadores de carga elásticos usando destinos IP ou de instância.	14 de maio de 2021
Taints de nós para grupos de nós gerenciados	O Amazon EKS agora oferece suporte para a adição de taints de notas a grupos de nós gerenciados.	11 de maio de 2021

Criptografia de segredos para os clusters existentes	O Amazon EKS agora oferece suporte à adição de criptografia de segredos para os clusters existentes.	26 de fevereiro de 2021
Versão do Kubernetes `1.19`	Suporte adicionado para o Kubernetes versão `1.19` para novos clusters e atualizações de versão.	16 de fevereiro de 2021
O Amazon EKS agora oferece suporte a provedores de identidade OpenID Connect (OIDC) como método para autenticar usuários no cluster de versão `1.16` ou superior.	Provedores de identidade OIDC podem ser usados com ou como alternativa ao AWS Identity and Access Management (IAM).	12 de fevereiro de 2021
Visualizar recursos de nó e workload no Console de gerenciamento da AWS	Você agora pode visualizar detalhes sobre os nós gerenciados, autogerenciados e do Fargate, assim como as workloads implantadas do Kubernetes no Console de gerenciamento da AWS.	1º de dezembro de 2020
Implantar tipos de instância spot em um grupo de nós gerenciados	Agora você pode implantar vários tipos de instância spot ou sob demanda em um grupo de nós gerenciados.	1º de dezembro de 2020
O Amazon EKS agora pode gerenciar complementos específicos para o cluster	O próprio usuário pode gerenciar os complementos ou pode permitir que o Amazon EKS controle o lançamento e a versão de um complemento por meio da API do Amazon EKS.	1º de dezembro de 2020

Compartilhar um ALB em várias entradas	Agora é possível compartilhar um AWS Application Load Balancer entre vários ingressos do Kubernetes. No passado, você tinha que implantar um balanceador de carga da aplicação separado para cada ingresso.	23 de outubro de 2020
Suporte para o destino de IP do NLB	Agora, você pode implantar um Network Load Balancer com destinos de IP. Isso significa que você pode usar um NLB para balancear a carga de tráfego da rede para os pods do Fargate e diretamente para os pods executados nos nós do Amazon EC2.	23 de outubro de 2020
Versão do Kubernetes `1.18`	Suporte adicionado para o Kubernetes versão `1.18` para novos clusters e atualizações de versão.	13 de outubro de 2020
Especifique um bloco CIDR personalizado para atribuição de endereços IP de serviço do Kubernetes.	Agora você pode especificar um bloco CIDR personalizado a partir do qual o Kubernetes atribui os endereços IP do serviço.	29 de setembro de 2020
Atribuir grupos de segurança a pods individuais	Você agora pode associar grupos de segurança diferentes a alguns dos pods individuais que são executados em muitos tipos de instância do Amazon EC2.	9 de setembro de 2020

Implantar o Bottlerocket nos nós	Agora você pode implantar nós que executam o Bottlerocket.	31 de agosto de 2020
A capacidade de executar nós Arm está disponível ao público em geral	Agora você pode executar nós Arm em grupos de nós gerenciados e autogerenciados.	17 de agosto de 2020
Modelos de execução de grupos de nós gerenciados e AMI personalizada	Agora você pode implantar um grupo de nós gerenciados usando um modelo de lançamento do Amazon EC2. Caso você queira, o modelo de execução poderá especificar uma AMI personalizada.	17 de agosto de 2020
Suporte do EFS para o AWS Fargate	Agora você pode usar o Amazon EFS com o AWS Fargate.	17 de agosto de 2020
Atualização de versão da plataforma do Amazon EKS	Existe uma versão da plataforma com correções e aprimoramentos de segurança. Isso inclui suporte UDP para serviços do tipo `LoadBalancer` ao usar Network Load Balancers com a versão `1.15` ou mais recente do Kubernetes. Para obter mais informações, consulte o problema Allow UDP for AWS Network Load Balancer no GitHub.	12 de agosto de 2020

Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível nas regiões África (Cidade do Cabo) (`af-south-1`) e Europa (Milão) (`eu-south-1`) AWS.	6 de agosto de 2020
Métricas de uso do Fargate	AWS A Fargate fornece métricas de uso do CloudWatch que proporcionam visibilidade do uso dos recursos do Fargate On-Demand pela sua conta.	3 de agosto de 2020
Versão do Kubernetes `1.17`	Suporte adicionado para o Kubernetes versão `1.17` para novos clusters e atualizações de versão.	10 de julho de 2020
Criação e gerenciamento de recursos do App Mesh de dentro do Kubernetes com o controlador do App Mesh para Kubernetes	Você pode criar e gerenciar recursos do App Mesh no Kubernetes. O controlador também injeta automaticamente o proxy Envoy e os contêineres de init nos pods que você implanta.	18 de junho de 2020
O Amazon EKS agora oferece suporte para nós Inf1 do Amazon EC2	Você pode adicionar nós Inf1 do Amazon EC2 ao seu cluster.	4 de junho de 2020
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível nas regiões AWS GovCloud (EUA-Leste) (`us-gov-east-1`) e AWS GovCloud (EUA-Oeste) (`us-gov-west-1`) AWS.	13 de maio de 2020

O Kubernetes 1.12 não é mais compatível com o Amazon EKS	A versão 1.12 do Kubernetes não é mais compatível com o Amazon EKS. Atualize qualquer cluster 1.12 para a versão 1.13 ou superior para evitar interrupção do serviço.	12 de maio de 2020
Versão do Kubernetes 1.16	Suporte adicionado para o Kubernetes versão 1.16 para novos clusters e atualizações de versão.	30 de abril de 2020
Adicionada a função vinculada ao serviço AWSServiceRoleForAmazonEKS	Adicionado o perfil vinculado ao serviço AWSServiceRoleForAmazonEKS.	16 de abril de 2020
Versão do Kubernetes 1.15	Suporte adicionado para o Kubernetes versão 1.15 para novos clusters e atualizações de versão.	10 de março de 2020
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível nas regiões (`cn-northwest-1`) AWS Pequim (`cn-north-1`) e Ningxia.	26 de fevereiro de 2020
Driver de CSI do FSx for Lustre	Tópico adicionado para a instalação do driver CSI do FSx para Lustre em clusters do Kubernetes 1.14 para Amazon EKS.	23 de dezembro de 2019

Restringir o acesso à rede ao endpoint de acesso público de um cluster	Com esta atualização, você pode usar o Amazon EKS para restringir os intervalos de CIDR que podem se comunicar com o endpoint de acesso público do servidor de API do Kubernetes.	20 de dezembro de 2019
Resolver o endereço de endpoint de acesso privado para um cluster do fora de uma VPC	Com esta atualização, você pode usar o Amazon EKS para resolver o endpoint de acesso privado do servidor de API do Kubernetes de fora de uma VPC.	13 de dezembro de 2019
(Beta) Nós de instâncias A1 do Amazon EC2	Execute os nós de instâncias A1 do Amazon EC2 registrados no cluster do Amazon EKS.	4 de dezembro de 2019
Criar um cluster no AWS Outposts	O Amazon EKS agora suporta a criação de clusters no AWS Outposts.	3 de dezembro de 2019
AWS Fargate no Amazon EKS	Agora os clusters do Kubernetes para Amazon EKS são compatíveis com a execução de pods no Fargate.	3 de dezembro de 2019
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível na região (`ca-central-1`) AWS Canadá (Central) ().	21 de novembro de 2019

Grupos de nós gerenciados	Os grupos de nós gerenciados do Amazon EKS automatizam o provisionamento e o gerenciamento do ciclo de vida dos nós (instâncias do Amazon EC2) para clusters de Kubernetes do Amazon EKS.	18 de novembro de 2019
Atualização de versão da plataforma do Amazon EKS	Novas versões de plataforma para resolver CVE-2019-11253.	6 de novembro de 2019
O Kubernetes `1.11` não é mais compatível com o Amazon EKS	A versão `1.11` do Kubernetes não é mais compatível com o Amazon EKS. Atualize qualquer cluster `1.11` para a versão `1.12` ou superior para evitar interrupção do serviço.	4 de novembro de 2019
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível na região (`sa-east-1`) AWS América do Sul (São Paulo).	16 de outubro de 2019
Suporte ao Windows	Os clusters do Amazon EKS que executam a versão `1.14` do Kubernetes agora são compatíveis com workloads do Windows.	7 de outubro de 2019
Autoescalabilidade	Adicionado um capítulo para abordar alguns dos diferentes tipos de escalabilidade automática do Kubernetes com suporte nos clusters do Amazon EKS.	30 de setembro de 2019

Atualização do painel do Kubernetes	Tópico atualizado para instalar o painel do Kubernetes em clusters do Amazon EKS para usar a versão beta `2.0`.	28 de setembro de 2019
Driver da CSI do Amazon EFS	Tópico adicionado para instalar o driver CSI do Amazon EFS em clusters do Kubernetes `1.14` para Amazon EKS.	19 de setembro de 2019
Parâmetro do Amazon EC2 Systems Manager para o ID da AMI otimizada para o Amazon EKS	Tópico adicionado para recuperar o ID da AMI otimizada para o Amazon EKS usando um parâmetro do Amazon EC2 Systems Manager. O parâmetro elimina a necessidade de pesquisar IDs de AMIs.	18 de setembro de 2019
Marcação de recursos do Amazon EKS	Você pode gerenciar a marcação dos clusters do Amazon EKS.	16 de setembro de 2019
Driver da CSI do Amazon EBS	Tópico adicionado para instalar o driver CSI do Amazon EBS em clusters do Kubernetes `1.14` para Amazon EKS.	9 de setembro de 2019
Nova AMI com patch otimizada para Amazon EKS para `CVE-2019-9512` e `CVE-2019-9514`	O Amazon EKS atualizou a AMI otimizada para Amazon EKS, a fim de abordar os CVE-2019-9512 e CVE-2019-9514.	6 de setembro de 2019

Anúncio de descontinuação do Kubernetes `1.11` no Amazon EKS	O Amazon EKS encerrou o suporte para o Kubernetes versão `1.11` em 4 de novembro de 2019.	4 de setembro de 2019
Versão do Kubernetes `1.14`	Suporte adicionado para o Kubernetes versão `1.14` para novos clusters e atualizações de versão.	3 de setembro de 2019
Perfis do IAM para contas de serviço	Com os perfis do IAM para contas de serviço em clusters do Amazon EKS, é possível associar um perfil do IAM a uma conta de serviço do Kubernetes. Com esse recurso, você não precisa mais fornecer permissões estendidas ao perfil do IAM do nó. Dessa forma, os pods nesse nó podem chamar as APIs da AWS.	3 de setembro de 2019
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível na região (`me-south-1`) AWS Oriente Médio (Bahrein).	29 de agosto de 2019
Atualização de versão da plataforma do Amazon EKS	Novas versões de plataforma para abordar CVE-2019-9512 e CVE-2019-9514.	28 de agosto de 2019
Atualização de versão da plataforma do Amazon EKS	Novas versões de plataforma para abordar CVE-2019-11247 e CVE-2019-11249.	5 de agosto de 2019

Expansão de regiões do Amazon EKS	O Amazon EKS já está disponível na região (`ap-east-1`) AWS Ásia-Pacífico (Hong Kong).	31 de julho de 2019
O Kubernetes `1.10` não é mais compatível com o Amazon EKS	A versão `1.10` do Kubernetes não é mais compatível com o Amazon EKS. Atualize qualquer cluster `1.10` para a versão `1.11` ou superior para evitar interrupção do serviço.	30 de julho de 2019
Adicionado tópico adicionado sobre o ALB Ingress Controller	O ALB Ingress Controller para Kubernetes da AWS é um controlador que faz com que um ALB seja criado quando recursos de entrada são criados.	11 de julho de 2019
Nova AMI otimizada para o Amazon EKS	A remoção de binários `kubectl` desnecessários das AMIs.	3 de julho de 2019
Versão do Kubernetes `1.13`	Suporte adicionado para o Kubernetes versão `1.13` para novos clusters e atualizações de versão.	18 de junho de 2019
Nova AMI com patch otimizada para Amazon EKS para `AWS-2019-005`	O Amazon EKS atualizou a AMI otimizada do Amazon EKS para solucionar as vulnerabilidades descritas em link:security/security-bulletins/AWS-2019-005/[AWS-2019-005,type="marketing"].	17 de junho de 2019

Anúncio de encerramento do suporte do Kubernetes `1.10` no Amazon EKS	O Amazon EKS encerrou o suporte para o Kubernetes versão `1.10` em 22 de julho de 2019.	21 de maio de 2019
Atualização de versão da plataforma do Amazon EKS	Nova versão da plataforma para clusters do Kubernetes `1.11` e `1.10` para ser compatível com nomes de DNS personalizados no certificado do `kubelet` e melhorar a performance do `etcd`.	21 de maio de 2019
Getting started with `eksctl`	Este guia de conceitos básicos ajuda a instalar todos os recursos necessários para começar a usar o Amazon EKS usando `eksctl`. Trata-se de um simples utilitário de linha de comando para criar e gerenciar clusters do Kubernetes no Amazon EKS.	10 de maio de 2019

AWS Comando da `get-token` CLI	O comando `aws eks get-token` foi adicionado à AWS CLI. Você não precisa mais instalar o AWS IAM Authenticator para Kubernetes a fim de criar tokens de segurança de clientes para comunicação com o servidor de API do cluster. Atualize sua instalação da AWS CLI para a versão mais recente para usar essa nova funcionalidade. Para obter mais informações, consulte Instalar a AWS Command Line Interface no Guia do usuário da AWS Command Line Interface.	10 de maio de 2019
Atualização de versão da plataforma do Amazon EKS	Nova versão da plataforma para clusters do Kubernetes `1.12` para ser compatível com nomes de DNS personalizados no certificado do `kubelet` e melhorar a performance do `etcd`. Isso corrige um erro que fazia com que os daemons do `kubelet` do nó solicitassem um novo certificado a cada poucos segundos.	8 de maio de 2019
Tutorial do Prometheus	Tópico adicionado para implantar o Prometheus ao cluster do Amazon EKS.	5 de abril de 2019

Registro em log do ambiente de gerenciamento do Amazon EKS	Com essa atualização, você pode obter logs de auditoria e diagnóstico diretamente do painel de controle do Amazon EKS. Você pode usar esses CloudWatch Logs em sua conta como referência para proteger e executar os clusters.	4 de abril de 2019
Versão do Kubernetes `1.12`	Suporte adicionado para o Kubernetes versão `1.12` para novos clusters e atualizações de versão.	28 de março de 2019
Adicionado o Guia de conceitos básicos do App Mesh	Documentação adicionada para os conceitos básicos do App Mesh e Kubernetes.	27 de março de 2019
Acesso privado ao endpoint do servidor de API do Amazon EKS	Documentação adicionad a para desabilitar o acesso público para o endpoint do servidor de API do Kubernetes do cluster do Amazon EKS.	19 de março de 2019
Adicionado tópico sobre como instalar o servidor de métricas do Kubernetes	O servidor de métricas do Kubernetes é um agregador de dados de uso de recursos no cluster.	18 de março de 2019
Adição da lista de projetos de código aberto relacionados	Esses projetos de código aberto ampliam a funcional idade de clusters do Kubernetes em execução na AWS, incluindo clusters gerenciados pelo Amazon EKS.	15 de março de 2019

Adição de tópico sobre como instalar Helm localmente	O gerenciador de pacotes `helm` para Kubernetes ajuda a instalar e gerenciar aplicações no cluster do Kubernetes. Esse tópico mostra como instalar e executar os binários do `helm` e `tiller` localmente. Dessa maneira, é possível instalar e gerenciar gráficos utilizando a CLI do Helm no seu sistema local.	11 de março de 2019
Atualização de versão da plataforma do Amazon EKS	Nova versão da plataforma que atualiza os clusters do Kubernetes `1.11` para Amazon EKS para o nível de patch `1.11.8` para resolver o CVE-2019-1002100.	8 de março de 2019
Aumento do limite de clusters	O Amazon EKS aumentou o número de clusters que você pode criar em uma região da AWS de 3 para 50.	13 de fevereiro de 2019
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível nas regiões (`ap-south-1`) AWS Europa (Londres) (`eu-west-2`), Europa (Paris) (`eu-west-3`) e Ásia-Pacífico (Mumbai).	13 de fevereiro de 2019
Nova AMI com patch otimizada para Amazon EKS para `ALAS-2019-1156`	O Amazon EKS atualizou a AMI otimizada do Amazon EKS para resolver a vulnerabilidade descrita em ALAS-2019-1156.	11 de fevereiro de 2019

Nova AMI com patch otimizada para Amazon EKS para `ALAS2-2019-1141`	O Amazon EKS atualizou a AMI otimizada do Amazon EKS para resolver os CVEs referenciados em ALAS2-2019-1141.	9 de janeiro de 2019
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível na região (`ap-northeast-2`) AWS Ásia-Pacífico (Seul).	9 de janeiro de 2019
Expansão da região do Amazon EKS	O Amazon EKS agora está disponível nas seguintes regiões adicionais da AWS: Europa (Frankfurt) (`eu-central-1`), Ásia-Pacífico (Tóquio) (`ap-northeast-1`), Ásia-Pacífico (Cingapura) (`ap-southeast-1`) e Ásia-Pacífico (Sydney) (`ap-southeast-2`).	19 de dezembro de 2018
Atualizações do cluster do Amazon EKS	A documentação das atualizações de versão do Kubernetes para clusters e as substituições de nós do Amazon EKS foi adicionada.	12 de dezembro de 2018
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível na região (`eu-north-1`) AWS Europa (Estocolmo).	11 de dezembro de 2018

Atualização de versão da plataforma do Amazon EKS	Nova versão da plataforma atualizando o Kubernetes para o nível de patch `1.10.11` para resolver link:security/security-bulletins/AWS-2018-020/ [CVE-2018-1002105,type="marketing"].	4 de dezembro de 2018
Adição de suporte para a versão `1.0.0` do controlador de ingresso do ALB	O controlador de ingresso do ALB lança a versão `1.0.0` com suporte formal da AWS.	20 de novembro de 2018
Adicionado suporte para a configuração de rede do CNI	O plug-in CNI da Amazon VPC para o Kubernetes versão `1.2.1`, agora é compatível com a configuração de rede personalizada para interfaces de rede de pods secundária.	16 de outubro de 2018
Adicionado suporte para `MutatingAdmissionWebhook` e `ValidatingAdmissionWebhook`	A versão `1.10-eks.2` da plataforma do Amazon EKS agora oferece suporte ao `MutatingAdmissionWebhook` e aos controladores de admissão `ValidatingAdmissionWebhook`.	10 de outubro de 2018
Adicionadas informações de AMIs de parceiros	A Canonical fez uma parceria com o Amazon EKS para criar AMIs de nós que você pode usar nos clusters.	3 de outubro de 2018

Foram adicionadas instruções para o comando `update-kubeconfig` da AWS CLI	O Amazon EKS adicionou o `update-kubeconfig` à AWS CLI para simplificar o processo de criação de um arquivo `kubeconfig` para acessar seu cluster.	21 de setembro de 2018
Novas AMIs otimizadas para o Amazon EKS	O Amazon EKS atualizou as AMIs otimizadas para o Amazon EKS (com e sem suporte a GPU) para fornecer várias correções de segurança e otimizações da AMI.	13 de setembro de 2018
Expansão de regiões da AWS do Amazon EKS	O Amazon EKS já está disponível na região Europa (Irlanda) (`eu-west-1`).	5 de setembro de 2018
Atualização de versão da plataforma do Amazon EKS	Nova versão da plataforma com suporte a camada de agregação e Horizontal Pod Autoscaler(HPA) do Kubernetes.	31 de agosto de 2018
Novas AMIs otimizadas para o Amazon EKS e suporte a GPU	O Amazon EKS atualizou a AMI otimizada do Amazon EKS para usar um novo modelo de nó e script de inicialização do AWS CloudFormation. Além disso, uma nova AMI otimizada para o Amazon EKS com suporte a GPU está disponível.	22 de agosto de 2018

Nova AMI com patch otimizada para Amazon EKS para `ALAS2-2018-1058`	O Amazon EKS atualizou a AMI otimizada do Amazon EKS para resolver os CVEs referenciados em ALAS2-2018-1058.	14 de agosto de 2018
Scripts de compilação de AMI otimizados para o Amazon EKS	O Amazon EKS usa scripts de compilação de código aberto para criar a AMI otimizada para o Amazon EKS. Esses scripts de compilação agora estão disponíveis no GitHub.	10 de julho de 2018
Lançamento do Amazon EKS	Documentação inicial para a inicialização do serviço	5 de junho de 2018

Contribuir para o Guia do usuário do EKS

A AWS introduziu uma experiência aprimorada para contribuições no Guia do usuário do EKS.

Agora, é possível editar o conteúdo do Guia do usuário do EKS diretamente no GitHub.

Agora, a documentação usa AsciiDoc, uma linguagem avançada para escrita, parecida com o Markdown. O AsciiDoc combina sintaxe simples com recursos de documentação corporativa, como formatação avançada, referência cruzada e controles de segurança.

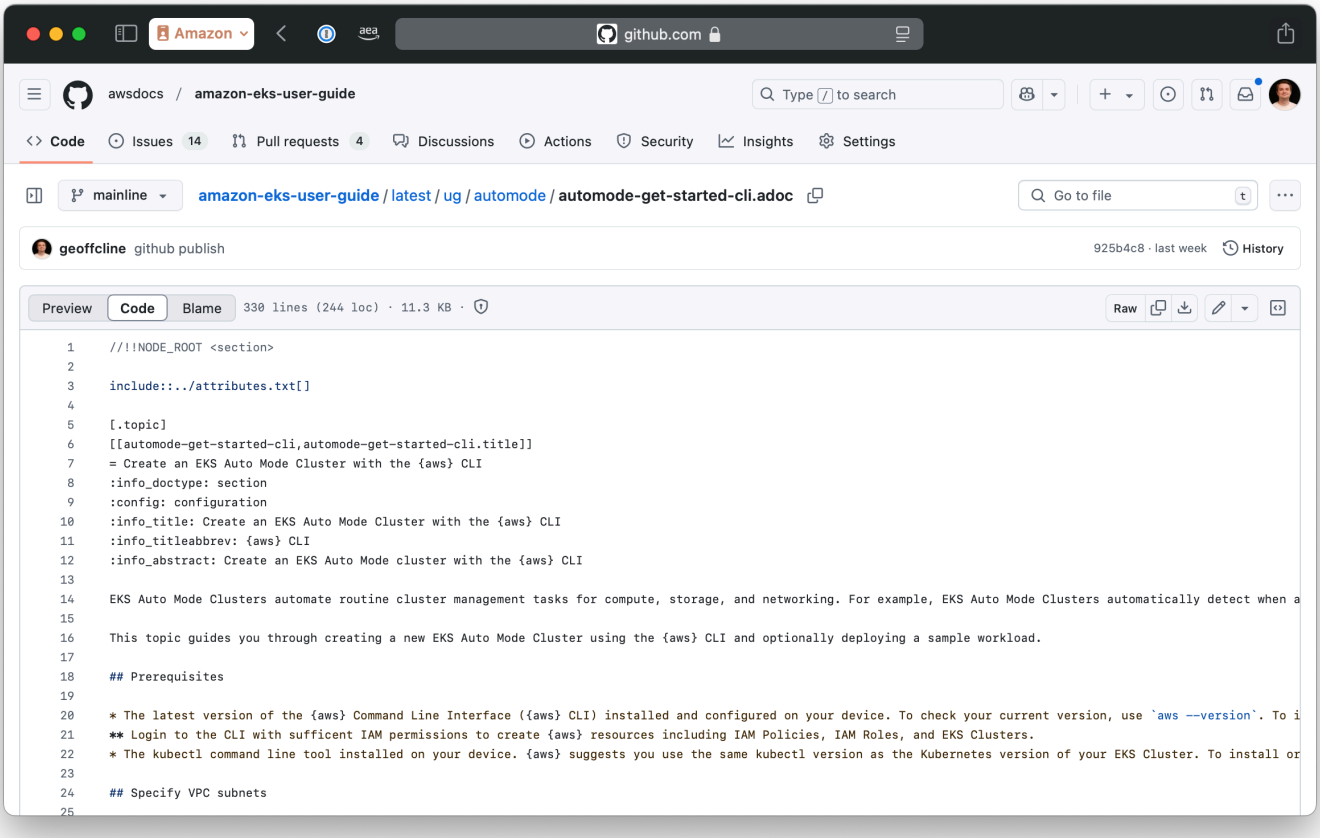
Você pode editar a documentação do EKS diretamente no GitHub a partir de agora. Nosso processo racionalizado inclui:

- Processamento mais rápido de solicitações pull
- Redução nas etapas manuais
- Verificações automatizadas da qualidade do conteúdo

Aguardamos ansiosamente suas contribuições.

Edição de uma única página usando um navegador da Web

É possível editar com facilidade uma única página no Guia do usuário do EKS diretamente por meio do seu navegador da Web.



Se você deseja editar diversas páginas usando o navegador da Web, consulte the section called "Edição de arquivos com o GitHub".

Pré-requisitos

- Página de documentação para alterações aberta no navegador da web.
- Sessão iniciada no GitHub.

Procedimento

1. Acesse a página que deseja editar no Guia do usuário do Amazon EKS.
2. No painel direito, escolha o link Editar esta página no GitHub.
3. No GitHub, abra o editor de uma das seguintes maneiras:
   - Pressionando a tecla e no seu teclado.
   - Clicando no ícone de lápis e selecionando Editar no local no menu suspenso.

- Caso não tenha a opção de editar, será necessário fazer login no GitHub. Sua conta do GitHub não precisa de nenhuma permissão especial para sugerir alterações. No entanto, os colaboradores internos da Amazon devem vincular seus perfis do GitHub.

4. Realize as alterações necessárias no conteúdo no editor do GitHub.
   - O editor oferece funcionalidades de destaque de sintaxe e de pré-visualização.
   - É possível usar a marcação do AsciiDoc para formatar suas alterações.
   - Você pode usar `ctrl-f` para abrir uma interface de localização/substituição.
5. (Opcional) Visualize previamente suas alterações.
   - Use a guia `preview` para visualizar previamente suas alterações com formatação avançada.
   - Use a opção `show diff` para destacar as seções alteradas. As seções removidas têm um indicador em vermelho na margem esquerda. As novas seções têm um indicador em verde na margem esquerda.
6. Ao concluir a edição, clique no botão Confirmar alterações... na parte superior do editor
7. Na caixa de diálogo de confirmação:
   - Verifique se seu endereço de e-mail está correto.
   - Adicione uma breve mensagem de confirmação, mas descritiva, sobre as alterações.
   - Opcionalmente, adicione uma descrição mais longa, se necessário.
   - Selecione a opção para criar uma nova ramificação e solicitação pull.

Você criou uma solicitação pull incluindo as alterações propostas.

Visão geral da solicitação pull

Quando você cria uma solicitação pull:

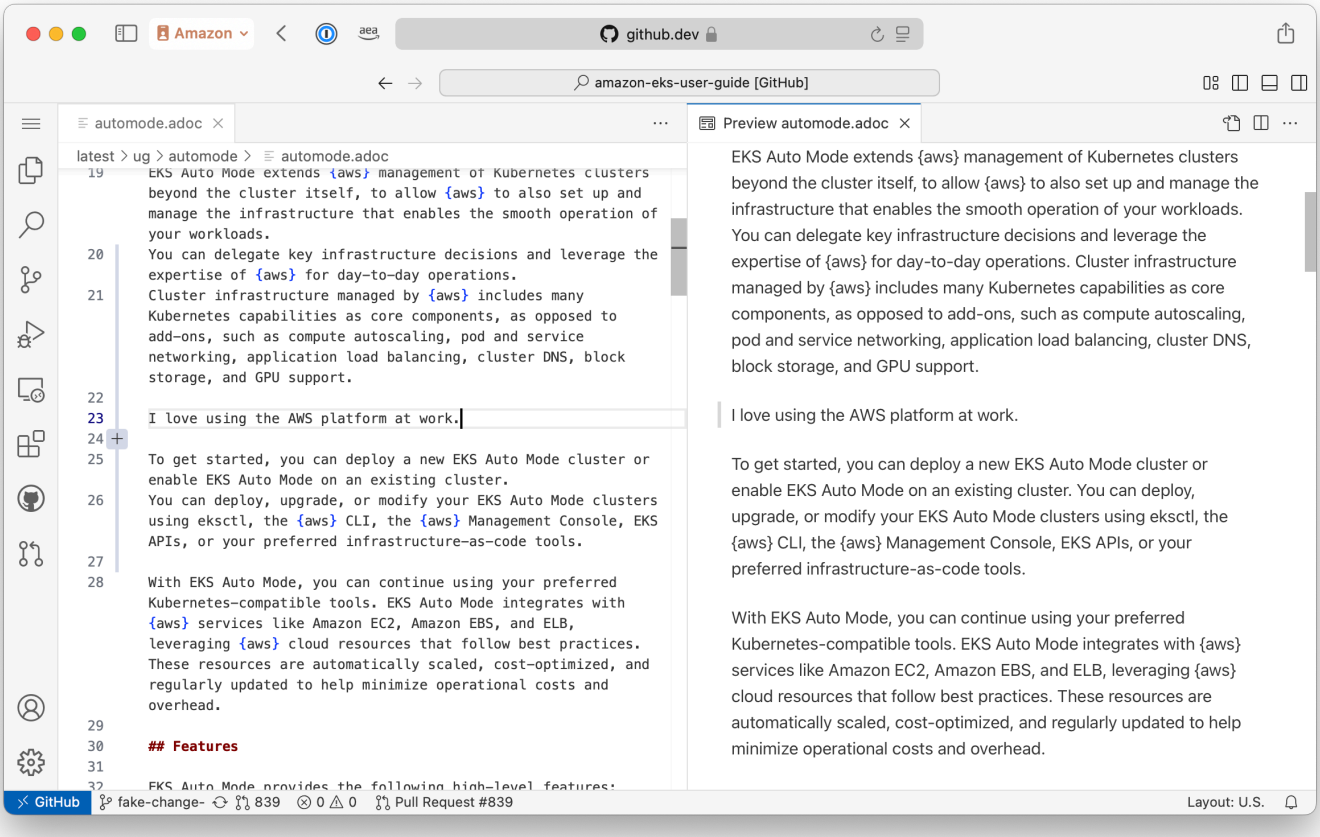
- As alterações são enviadas para análise dos responsáveis pelo repositório.
- Os revisores podem fazer comentários sobre suas alterações e solicitar modificações.
- Testes automatizados podem ser executados para validar suas alterações.
- Após a aprovação, suas alterações podem ser integradas ao repositório principal.

As solicitações pull ajudam a garantir a qualidade e fornecem uma maneira de debater as alterações antes de elas serem integradas.

Aprenda como as solicitações pull são analisadas e aprovadas na documentação do GitHub.

Edição de diversos arquivos usando um navegador da Web com o editor baseado na Web do GitHub

Se você deseja propor alterações em diversas páginas ou criar uma nova página de documentação, use o editor baseado na Web do GitHub.dev. Este editor na Web é baseado no popular editor de texto Visual Studio Code.



Pré-requisitos

- Estar conectado ao GitHub
- Ter familiaridade com o editor Visual Studio Code
- Ter familiaridade com o Git

Procedimento

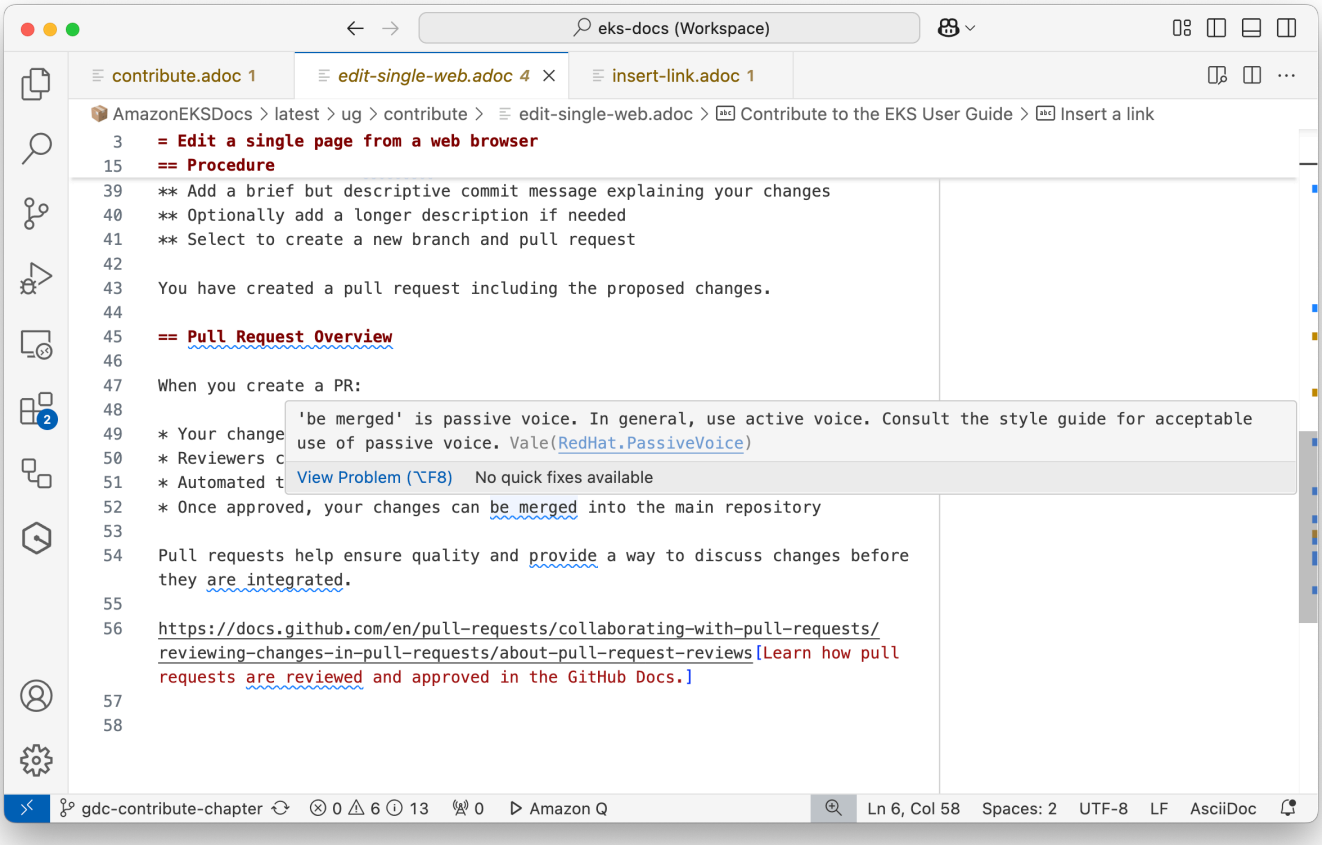
> **Note**
>
> A equipe de documentação do EKS criou um arquivo de espaço de trabalho que inclui configurações sugeridas para o editor, como quebra de texto e destaque de sintaxe do AsciiDoc. Recomendamos que você carregue este arquivo de espaço de trabalho.

1. Abra o espaço de trabalho no GitHub.dev.
   - É possível adicionar o URL `https://github.dev/awsdocs/amazon-eks-user-guide/blob/mainline/eks-docs.code-workspace?workspace=true` aos seus favoritos
2. (Apenas na primeira configuração) Você pode ser solicitado a criar uma bifurcação do repositório na sua própria conta do GitHub. Aceite esse prompt. Para obter mais informações, consulte Sobre bifurcações na documentação do GitHub.
3. (Apenas na primeira configuração) Aceite o prompt no canto inferior direito para instalar a extensão do AsciiDoc.
4. Acesse o conteúdo da documentação em `latest/ug`.
   - Os arquivos de documentação são organizados de acordo com suas seções principais. Por exemplo, as páginas do capítulo “Segurança” têm os arquivos de origem no diretório “security/”.
5. Para visualizar uma versão prévia de uma página de documentação, use o botão Abrir a versão prévia lateral no canto superior direito. O ícone contém uma pequena lupa.
6. Use a guia Controle de fonte à esquerda para realizar a confirmação das suas alterações. Para obter mais informações, consulte a documentação do Visual Studio Code:
   - Confirmação das alterações
   - Criar uma solicitação pull

Após criar uma solicitação pull, ela será analisada pela equipe responsável pela documentação.

Visualização dos comentários de estilo enquanto digita ao instalar a Vale localmente

É possível visualizar comentários de estilo enquanto digita. Isso auxilia na identificação de erros de digitação e de redação inadequada.



Visão geral:

- A Vale CLI carrega guias de estilo e os executa nos arquivos de origem.
- O repositório de documentação do EKS inclui um arquivo de configuração da Vale que carrega guias de estilo e regras locais.
- A extensão da Vale para o Visual Studio (VS) Code exibe os comentários diretamente no editor.

Instalação da Vale

Siga as instruções na documentação da Vale CLI para instalar a Vale com um gerenciador de pacotes.

Instalação da extensão da Vale para o VS Code

1. Abra o VS Code.
2. Clique no ícone Extensões na barra de atividades (ou pressione Ctrl+Shift+X).
3. Pesquise por “Vale”.
4. Clique em Instalar na extensão “Vale VSCode” de Chris Chinchilla.
5. Recarregue o VS Code quando solicitado.

Sincronização da Vale

A Vale usa o arquivo de configuração `.vale.ini` na raiz do seu projeto para determinar as regras de estilo a serem aplicadas.

1. Abra o VS Code.
2. Clique em Visualizar > Terminal (ou pressione Ctrl+`).
3. Navegue até o diretório raiz do seu projeto, se necessário.
4. Execute o comando :

```
vale sync
```

5. Aguarde a Vale concluir o download e a sincronização das regras de estilo

Visualize os comentários de estilo no VS Code

1. Abra um arquivo Markdown ou AsciiDoc no VS Code.
2. A extensão Vale verificará automaticamente o texto de acordo com as regras de estilo.
3. Os problemas de estilo serão sublinhados no editor.
4. Passe o mouse sobre o texto sublinhado para visualizar a sugestão de estilo específica.
5. Corrija os problemas ao seguir as sugestões ou ao consultar o guia de estilo.

Criação de uma nova página

Saiba como criar uma nova página de documentação. Neste tópico, você encontrará instruções para criar os metadados iniciais da página e adicionar a página ao sumário do guia.

Criar página

1. Acesse o diretório do capítulo. Por exemplo, se você deseja criar uma nova página na seção “Segurança”, acesse o diretório `latest/ug/security`.
2. Determine o ID da página. De acordo com a convenção, o ID da página é todo em letras minúsculas e segmentado por `-`. O ID desta página é `create-page`.
3. Crie um novo arquivo com o ID da página e a extensão `adoc`. Por exemplo, `.create-page.adoc`
4. Insira os metadados da página usando este modelo:

```
include::../attributes.txt[]

[.topic]
[#unique-page-id]
= Page title
:info_titleabbrev: Short title

[abstract]
--
Brief summary of the page's content.
--

Introduction paragraph goes here.
```

Adicionar página à navegação

1. Acesse a página principal. A página principal das seções de nível superior é `book.adoc`.
2. Na parte inferior da página principal, inclua a página secundária.

   ```
   include::${filename}[leveloffset=+1]
   ```

   Por exemplo:

   ```
   include::create-page.adoc[leveloffset=+1]
   ```

Inserir um link

O AsciiDoc oferece suporte a vários tipos de links. Usar o tipo de link correto é importante para que o link funcione corretamente em diferentes ambientes.

Criar um link para uma página ou seção no Guia do usuário do EKS

Use referências cruzadas (xref) para criar links entre páginas ou seções dentro do mesmo site de documentação, como o Guia do usuário do EKS. Elas se atualizam automaticamente caso a seção de destino mude de lugar ou seja renomeada.

Usar o título da página como texto do link

Na maioria dos casos, ao vincular a outro ID neste guia do usuário, use a abordagem a seguir para que o texto do link seja atualizado automaticamente para o título mais recente, conforme necessário.

```
For more information, see <<page-or-section-id>>.
```

Definir um texto de link personalizado

Para casos em que você precisa ter um texto de link personalizado, use o formato a seguir.

```
Here's an example of a <<page-or-section-id,link with custom text>>.
```

Link para outro guia na documentação da AWS

1. Encontre o link para a página da documentação da AWS.
2. Remova o prefixo `https://docs.aws.amazon.com/` , mantendo somente o caminho. O caminho deve começar com uma letra.
3. Crie um link conforme mostrado abaixo:

```
link:AmazonS3/latest/userguide/create-bucket-overview.html[Create a bucket,
 type="documentation"]
```

Link para uma página da web externa

Este formato cria um link padrão para uma página não hospedada pela Amazon. Por exemplo, use isso para links do GitHub.

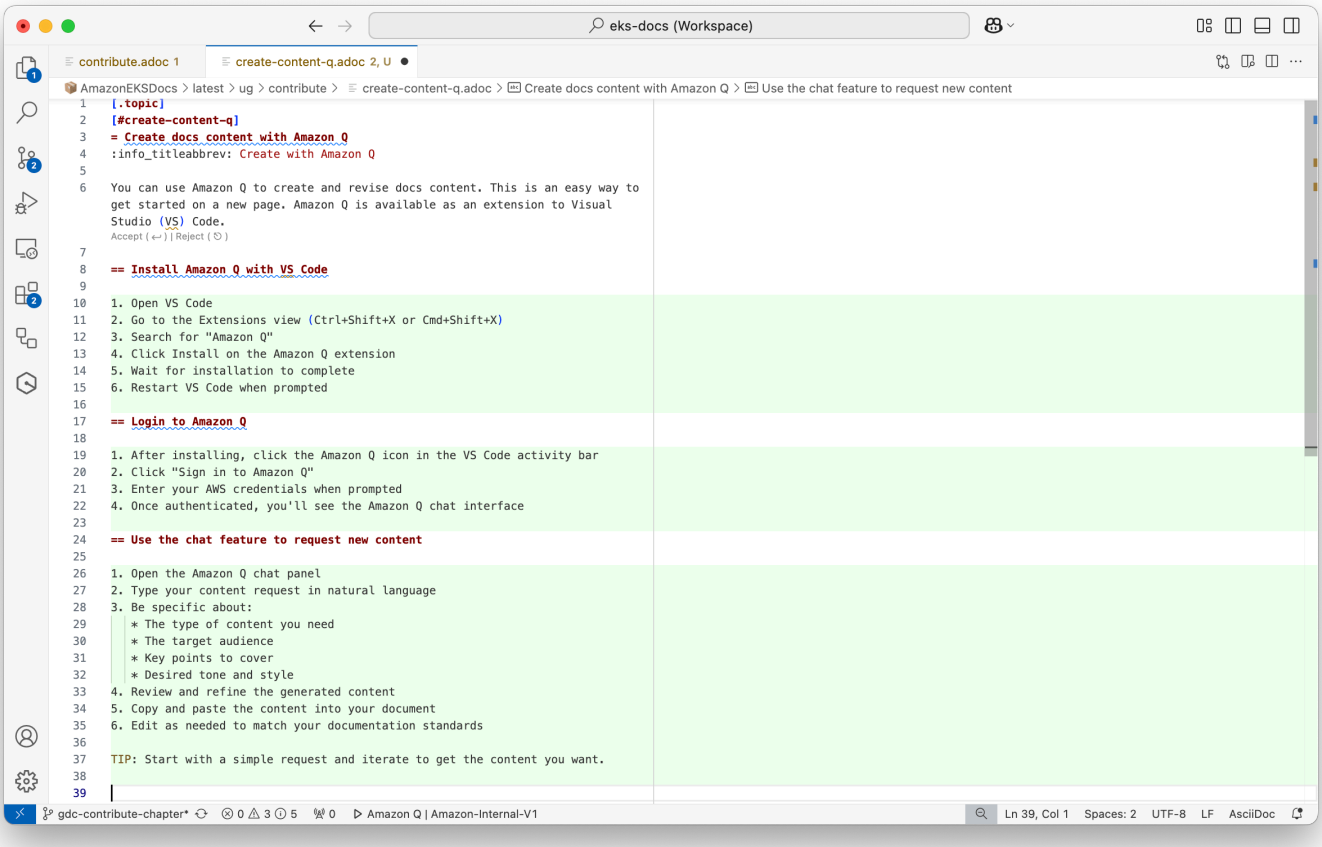
```
https://example.com[Link text]
```

> **Note**
>
> Temos uma lista de permissões para domínios externos. A lista de permissões pode ser consultada em `vale/styles/EksDocs/ExternalDomains.yml`

Criação de conteúdo para a documentação com o Amazon Q

É possível usar o Amazon Q para criar e revisar o conteúdo da documentação. Esta é uma maneira fácil de começar a usar uma nova página. O Amazon Q está disponível como uma extensão para o Visual Studio (VS) Code.

Na imagem apresentada a seguir, o Amazon Q gerou as linhas destacadas em verde.



Instalação do Amazon Q com o VS Code

1. Abra o VS Code
2. Acesse a visualização Extensões (Ctrl+Shift+X ou Cmd+Shift+X)
3. Pesquise por “Amazon Q”
4. Clique em Instalar na extensão Amazon Q
5. Aguarde a conclusão da instalação
6. Reinicie o VS Code quando solicitado

Login no Amazon Q

1. Após a instalação, escolha o ícone do Amazon Q na barra de atividades do VS Code.
2. Escolha Fazer login no Amazon Q.
3. Insira suas credenciais da AWS quando solicitado.
4. Após a autenticação, a interface de chat do Amazon Q será exibida.

Uso do Amazon Q para criar conteúdo

1. Abra o arquivo que você deseja editar no VS Code.
2. Selecione o texto a ser revisado ou o local em que deseja adicionar conteúdo novo.
3. Pressione Ctrl+I ou Cmd+I.
4. No prompt, forneça detalhes sobre:
   - O tipo de conteúdo necessário.
   - O público-alvo.
   - Os pontos principais a serem abordados.
   - O tom e o estilo desejados.
5. Analise o conteúdo gerado na pré-visualização em linha.
6. Use Enter para aceitar as alterações ou Esc para rejeitá-las.
7. Faça edições adicionais, conforme necessário.

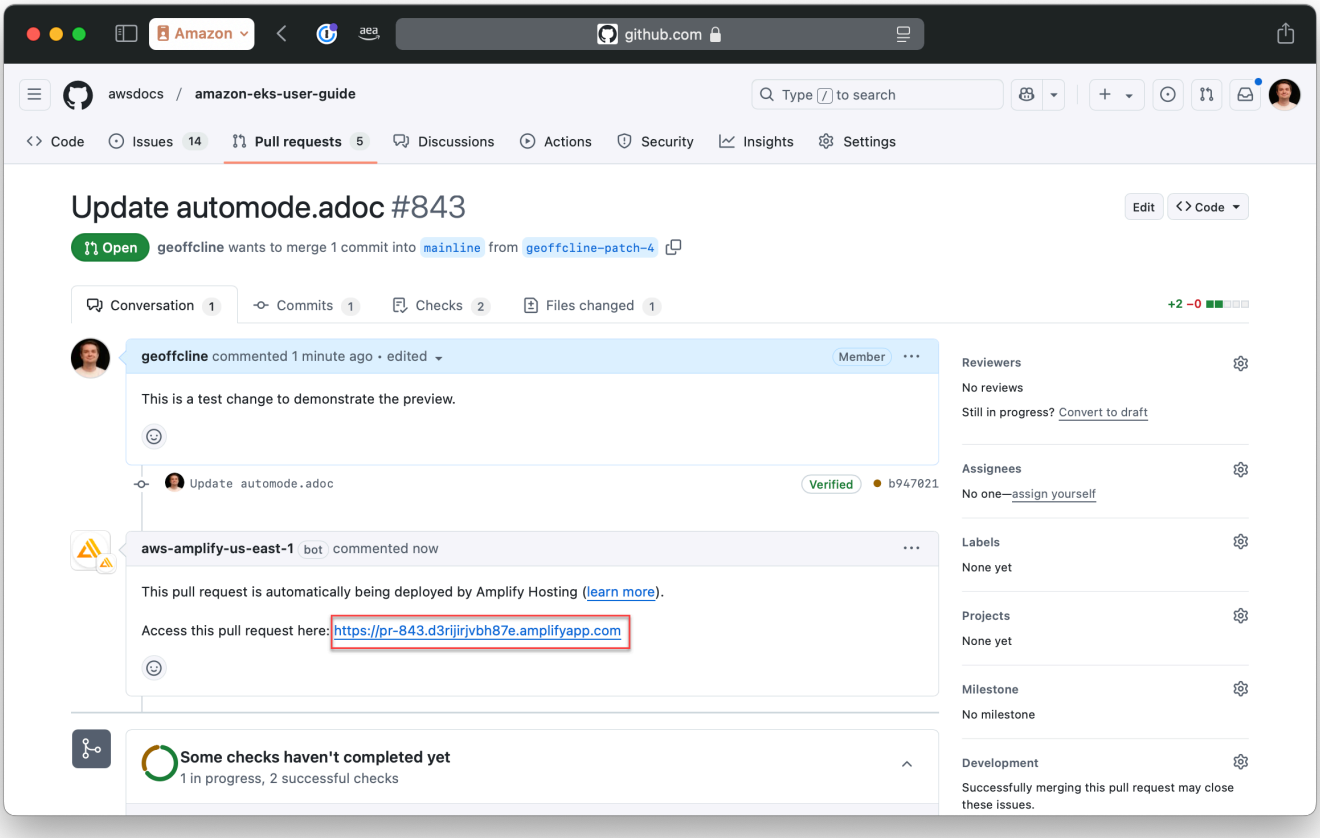
Dicas

- Comece com uma solicitação simples e itere para obter o conteúdo desejado.
- Crie um esboço inicial dos títulos das páginas e, em seguida, solicite ao Q para completá-los.
- O Amazon Q pode gerar Markdown. Isso é perfeitamente aceitável. As ferramentas do AsciiDoc são compatíveis com a maior parte da sintaxe do Markdown.

Para saber mais informações sobre o Amazon Q Developer, consulte Using Amazon Q Developer in the IDE.

Visualização de uma versão prévia do conteúdo da solicitação pull

O Guia do usuário do Amazon EKS para GitHub está configurado para desenvolver e gerar uma pré-visualização do site de documentação. Essa pré-visualização não contém o tema completo da AWS, mas verifica se o desenvolvimento do conteúdo está correto e se os links estão funcionando.



Essa pré-visualização fica hospedada em um URL temporário do AWS Amplify.

Pré-visualizar

Quando você envia uma solicitação pull, o AWS Amplify tenta criar e implantar uma pré-visualização do conteúdo.

Se a criação tiver êxito, o aws-amplify-us-east-1 adicionará um comentário à solicitação pull que tem um link para a pré-visualização. Escolha o link à direita de "Acesse esta solicitação pull aqui" (conforme destacado na captura de tela com um contorno vermelho).

Se o desenvolvimento falhar, os administradores do repositório podem visualizar os logs e fornecer comentários.

> **Note**
>
> Se você nunca contribuiu antes, pode ser necessário que um mantenedor do projeto aprove a execução do desenvolvimento.

Limitações da pré-visualização

A versão prévia é desenvolvida como um único arquivo HTML grande. Ela será exibida como várias páginas quando publicada.

O que funciona:

- Referências cruzadas (`xref`)
- Links para sites externos da Internet
- Imagens
- Conteúdo hospedado em `samples/`

O que não funciona:

- Links para outros tipos de conteúdo da AWS, usando a `type="documentation"`. Esse conteúdo não existe no ambiente de pré-visualização.
- O recurso `{aws}` não será exibido corretamente. O valor dela muda dependendo do ambiente.

Referência de sintaxe do AsciiDoc

AsciiDoc é a linguagem de marcação de preferência da documentação da AWS. Embora as ferramentas tenham suporte parcial para a sintaxe Markdown (como cabeçalhos e listas), recomendamos usar a sintaxe AsciiDoc para todo o conteúdo para garantir consistência e renderização adequada.

Este guia aborda elementos de sintaxe comuns de que você precisará ao contribuir com a documentação do Amazon EKS. Para obter uma sintaxe mais avançada e informações detalhadas, consulte a documentação do AsciiDoc.

Nova página

Cada novo documento AsciiDoc deve começar com a estrutura definida em the section called “Criar página”.

Títulos

Use cabeçalhos para organizar seu conteúdo hierarquicamente:

```
= Page/topic title (level 1)
== Section title (level 2)
=== Level 3 heading
==== Level 4 heading
===== Level 5 heading
====== Level 6 heading
```

Observação: sempre use letras maiúsculas e minúsculas de frase para cabeçalhos na documentação da AWS.

Formatação de texto

Formate o texto para enfatizar informações importantes:

```
Use *bold text* for UI labels.
Use _italic text_ for introducing terms or light emphasis.
Use `monospace text` for code, file names, and commands.
```

Listas

Listas não ordenadas

Crie listas com marcadores para itens sem uma sequência específica:

```
* First item
* Second item
** Nested item
** Another nested item
* Third item
```

Listas ordenadas

Crie listas numeradas para etapas sequenciais ou itens priorizados:

```
. First step
. Second step
.. Substep 1
.. Substep 2
. Third step
```

Links

Consulte the section called “Inserir link” para obter detalhes sobre como formatar links corretamente. Links no estilo Markdown não são compatíveis.

Exemplos de código

Código em linha

Use acentos graves para código em linha:

```
Use the `kubectl get pods` command to list all pods.
```

Blocos de código

Crie blocos de código com destaque de sintaxe e suporte para atributos (semelhantes a entidades):

```
 [source,python,subs="verbatim,attributes"]
```

```
----
def hello_world():
    print("Hello, World!")
----
```

Imagens

Insira imagens com texto alternativo para acessibilidade:

```
image::images/image-file.png[Alt text description]
```

Tabelas

Crie tabelas para organizar as informações:

```
[%header,cols="2"]
|===
|Header 1
|Header 2

|Cell 1,1
|Cell 1,2

|Cell 2,1
|Cell 2,2
|===
```

Para tabelas mais complexas, consulte a documentação da tabela do AsciiDoc.

Chamadas

Use avisos para destacar informações e advertências importantes:

```
NOTE: This is a note callout for general information.

TIP: This is a tip callout for helpful advice.

IMPORTANT: This is an important callout for critical information.
```

Versão prévia:

> **Note**
>
> Este é um aviso de observação.

Incluir outros arquivos

Inclua conteúdo de outros arquivos:

```
include::filename.adoc[]
```

Atributos (similares a entidades)

Use atributos predefinidos para manter a consistência. Em particular, você DEVE usar atributos para a AWS e `arn:aws:` .

```
{aws} provides Amazon EKS as a managed Kubernetes service.
```

```
[source,bash,subs="verbatim,attributes"]
----
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn {arn-aws}iam::aws:policy/AmazonEKSClusterPolicy
----
```

Para obter uma lista de atributos, consulte o arquivo `../attributes.txt`.

Procedimentos

Procedimentos passo a passo do formato:

```
To create an Amaozon EKS cluster. do the following steps.

. Sign in to the {aws} Management Console.
. Open the Amazon EKS console.
. Choose *Create cluster*.

...
```