Guida per l'utente

Amazon EKS



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

Amazon EKS: Guida per l’utente

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

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in qualsiasi modo che possa causare confusione tra i clienti o in qualsiasi modo che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

... xxvi
... xxvii
Che cosa è Amazon EKS? ... 1
Amazon EKS: gestione semplificata di Kubernetes ... 1
Creazione e scalabilità con Kubernetes: Amazon EKS Capabilities ... 2
Funzionalità di Amazon EKS ... 3
Servizi correlati ... 4
Prezzi di Amazon EKS ... 5
Casi di utilizzo comune ... 6
Architecture ... 7
Piano di controllo ... 7
Calcolo ... 8
Funzionalità EKS ... 10
Concetti di Kubernetes ... 11
Perché Kubernetes? ... 11
Cluster ... 16
Carichi di lavoro ... 21
Passaggi successivi ... 27
Opzioni di implementazione ... 27
Comprensione delle opzioni di implementazione di Amazon EKS ... 27
Amazon EKS nel cloud ... 28
Amazon EKS nei tuoi data center o ambienti edge ... 28
Amazon EKS Anywhere per ambienti isolati ... 29
Strumenti di Amazon EKS ... 30
Configurazione ... 31
Passaggi successivi ... 31
Configurazione di AWS CLI ... 31
Per creare una chiave di accesso ... 32
Per configurare la CLI di AWS ... 32
Per ottenere un token di sicurezza ... 33
Per verificare l'identità dell'utente ... 33
Passaggi successivi ... 34
Configura kubectl ed eksctl ... 34
Installazione o aggiornamento di `kubectl` ... 34

Fase 1: verifica se `kubectl` è installato ... 35
Fase 2: installa o aggiorna `kubectl` ... 35
Installazione di `eksctl` ... 49
Fasi successive ... 50
Guida introduttiva ... 51
In questo tutorial ... 51
Prerequisiti ... 51
Configurazione del cluster ... 52
Crea IngressClass ... 52
Distribuzione dell'applicazione di gioco 2048 esemplificativa ... 53
Dati persistenti utilizzando la modalità automatica Amazon EKS ... 56
Eliminazione ... 58
Scopri Amazon EKS ... 59
Panoramica ... 59
Workshop su Amazon EKS ... 59
Tutorial pratici per la configurazione dei cluster su Amazon EKS ... 61
Esempi di Amazon EKS ... 62
Tutorial di AWS ... 62
Workshop per sviluppatori ... 63
Workshop Terraform ... 63
Formazione su AWS Amazon EKS ... 63
Nozioni di base ... 64
Crea cluster (EKS Auto Mode) ... 64
Crea cluster (eksctl) ... 65
Prerequisiti ... 66
Passaggio 1: Creazione di cluster e nodi Amazon EKS ... 66
Passaggio 2: Visualizzazione delle risorse Kubernetes ... 67
Passaggio 3: Eliminazione del cluster e dei nodi ... 69
Passaggi successivi ... 69
Crea cluster (console e CLI) ... 70
Prerequisiti ... 70
Passaggio 1: Creazione del cluster Amazon EKS; ... 71
Passaggio 2: Configurazione del computer per comunicare con il cluster ... 74
Passaggio 3: Creazione di nodi ... 75
Passaggio 4: Visualizzazione delle risorse ... 77
Passaggio 5. Eliminazione delle risorse ... 78

Fasi successive 79
Modalità automatica EKS 80
Funzionalità 80
Componenti automatici 81
Configurazione 83
Modello di responsabilità condivisa 84
Creazione di un cluster 85
CLI di eksctl 86
AWS CLI 88
Console di gestione 96
Abilita i cluster esistenti 98
Materiale di riferimento per le migrazioni 99
Migration di volumi EBS 100
Migrazione dei bilanciatori del carico 100
Abilitazione sul cluster 101
Eseguire la migrazione da Karpenter 105
Eseguire la migrazione da MNG 108
Eseguire la migrazione da Fargate 109
Esegui carichi di lavoro 117
Implementazione di un carico di lavoro di decompressione 118
Implementa il bilanciatore del carico 121
Implementazione di un carico di lavoro stateful 126
Implementare carico di lavoro accelerato 132
Configura 139
Creazione di una classe di nodi 143
Creazione di pool di nodi 152
Pool di nodi a capacità statica 159
Creare una classe in ingresso 165
Creare servizio 174
Crea StorageClass 180
Disabilita modalità automatica di EKS 189
Aggiornamento della versione di Kubernetes 191
Riesamina i pool di nodi integrati 192
Controllare l'implementazione 194
Eseguire componenti aggiuntivi critici 195
Usare le policy di rete 197

Assegnazione di tag alle sottoreti 208
Generare report CIS. 210
Crittografare i volumi root 213
Aggiorna i controlli AMI 217
Controllare le prenotazioni della capacità 221
Usa Local Zones 225
Sicurezza avanzata 230
Come funziona 232
Istanze gestite 232
Identità e accesso 237
Rete 242
Osservabilità 246
Registri dei componenti gestiti automaticamente da EKS 246
Risoluzione dei problemi 249
Agente di monitoraggio del nodo 250
Ottieni l'output della console da un'istanza gestita da EC2 utilizzando la AWS CLI EC2 250
Ottieni i log dei nodi utilizzando i container di debug e `kubectl` CLI 251
Visualizza le risorse associate alla modalità automatica EKS nella console AWS 252
Visualizza gli errori IAM nel tuo AWS account 253
Risolvere i problemi relativi alla mancata pianificazione dei Pod sul nodo Auto Mode 253
Risolvere i problemi del nodo che non si unisce al cluster 253
Condivisione di volumi tra pod 256
Visualizzare gli eventi di Karpenter nei log del piano di controllo 256
Risolvere i problemi relativi ai controller inclusi in Auto Mode 258
Risorse correlate 213
Note di rilascio 259
2 febbraio 2026 259
19 dicembre 2025 259
19 novembre 2025 259
19 novembre 2025 259
23 ottobre 2025 260
1° ottobre 2025 260
30 settembre 2025 260
29 settembre 2025 260
10 settembre 2025 260
24 agosto 2025 260

15 agosto 2025 260
6 agosto 2025 261
30 giugno 2025 261
20 giugno 2025 261
13 giugno 2025 261
30 aprile 2025 261
18 aprile 2025 261
11 aprile 2025 262
4 aprile 2025 262
31 marzo 2025 262
21 marzo 2025 262
14 marzo 2025 262
Configurazione dei cluster 263
Creare un cluster automatico 263
Prerequisiti 51
Crea cluster - AWS console 265
Crea cluster - AWS CLI 270
Fasi successive 96
Creazione di un cluster 275
Prerequisiti 51
Passaggio 1: crea un ruolo IAM del cluster 277
Fase 2: Creare un cluster 278
Passaggio 3: aggiornamento di kubeconfig 288
Passaggio 4: configurazione del cluster 288
Fasi successive 96
Crea un piano di controllo predisposto 290
Panoramica di 94
Casi d'uso 149
Control Plane Scaling Tiers 293
Considerazioni 100
Guida introduttiva a Provisioned Control Plane 298
Approfondimenti sui cluster 305
Tipi di approfondimenti del cluster 306
Considerazioni 306
Casi d'uso 306
Inizia a usare 307

Visualizzazione degli approfondimenti sui cluster 308
Aggiornamento della versione di Kubernetes 314
Considerazioni per la modalità automatica di Amazon EKS 316
Riepilogo 316
Passaggio 1: preparazione per l'aggiornamento 317
Passaggio 2: esaminare le considerazioni sull'aggiornamento 317
Fase 3: aggiornamento del piano di controllo del cluster 319
Fase 4: aggiornamento dei componenti del cluster 321
Effettuare il downgrade della versione di Kubernetes di un cluster Amazon EKS 322
Eliminazione di un cluster 323
Considerazioni 100
Prerequisiti 323
Eliminare un cluster (eksctl) 324
Elimina cluster (AWS console) 325
Elimina cluster (AWS CLI) 326
Proteggi i cluster EKS dall'eliminazione accidentale 327
Accesso all'endpoint del cluster 328
formato dell'endpoint del cluster IPv6 328
formato dell'endpoint del cluster IPv4 329
Endpoint privato del cluster 330
Modifica dell'accesso all'endpoint del cluster 330
Accesso a un server API solo privato 333
Configurare l'accesso all'endpoint 335
Abilitazione del supporto Windows 338
Considerazioni 100
Prerequisiti 51
Abilitazione del supporto Windows 340
Implementazione di pods Windows 342
Supporto di maggiore densità di pod sui nodi di Windows 342
Disabilitare il supporto Windows 343
Cluster privati 344
Requisiti dell'architettura del cluster 344
Requisiti dei nodi 344
Requisiti del pod 346
Dimensionamento automatico 348
EKS Auto Mode 349

Ulteriori soluzioni 349
Ulteriori informazioni relative allo spostamento zonale 350
Comprensione del flusso di traffico di rete est-ovest tra i pod 350
Requisiti per lo spostamento zonale EKS 355
Domande frequenti 364
Risorse aggiuntive 367
Abilitazione dello spostamento zonale 367
Considerazioni 368
What is Amazon Application Recovery Controller? 368
Che cos'è uno spostamento zonale? 368
Cos'è lo spostamento automatico zonale automatico? 369
Cosa fa EKS durante uno spostamento automatico? 369
Registra il cluster EKS con Amazon Application Recovery Controller (ARC) (AWS console) 369
Fasi successive 96
Gestione dell'accesso 371
Attività comuni 371
Contesto 208
Considerazioni relative alla modalità automatica di EKS 323
Accesso API Kubernetes 372
Associa le identità IAM alle autorizzazioni Kubernetes 373
Set Cluster Authentication Mode 374
Voci di accesso 376
aws-auth ConfigMap 433
Collegare il provider OIDC 443
Scollegare il gestore OIDC 448
Accesso alle risorse di cluster 449
Autorizzazioni richieste 450
CloudTrail visibilità 454
Modifica le risorse del cluster 454
Autorizzazioni richieste 454
CloudTrail visibilità 454
Accedere al cluster con kubectl 455
Creazione automatica del file `kubeconfig` 456
Accesso ai carichi di lavoro a AWS 457
Token dell'account di servizio 458

Componenti aggiuntivi del cluster 459
Concedere le autorizzazioni di AWS Identity and Access Management ai carichi di lavoro sui cluster Amazon Elastic Kubernetes Service 460
Credenziali con IRSA 465
Pod Identity 489
Gestione del calcolo 525
Confronto delle opzioni di calcolo 525
Gruppi di nodi gestiti 531
Concetti sui gruppi di nodi gestiti 532
Tipi di capacità del gruppo di nodi gestiti 535
Crea 538
Aggiornamento 550
Dettagli sull'aggiornamento del comportamento 555
Modelli di avvio 559
Elimina 576
Nodi autogestiti 578
Amazon Linux 580
Portabottiglie 589
Windows 593
Ubuntu Linux 602
Metodi di aggiornamento 605
AWS Fargate 618
Considerazioni su AWS Fargate 619
Tabella di confronto Fargate 621
Nozioni di base 624
Definisci i profili 630
Eliminare profili 635
Dettagli di configurazione del pod 636
Eventi di applicazione di patch del sistema operativo 639
Raccolta dei parametri 642
Registrazione di log 644
Tipi di istanza Amazon EC2 655
Per ogni tipo di istanza Amazon EC2, Amazon EKS consiglia un numero massimo di pod ... 658
Come viene determinato MaxPods 659
Considerazioni per la modalità automatica di EKS 323
AMI ottimizzate preconfigurate 661

AL2 Deprecazione AMI 661
Amazon Linux 668
Bottlerocket 678
Ubuntu Linux 684
Windows 684
Integrità del nodo 758
Agente di monitoraggio del nodo 759
Riparazione automatica del nodo 759
Problemi di integrità dei nodi 762
Visualizzazione dello stato dei nodi 776
Ottenere i log dei nodi 779
Nodi ibridi 781
Caratteristiche 782
Limiti 783
Considerazioni 783
Risorse aggiuntive 784
Prerequisiti 784
Esegui i nodi ibridi 852
Configura 874
Come funziona 960
nodeadm dei nodi ibridi 1002
risoluzione dei problemi 1020
Archiviazione di dati per app 1042
Amazon EBS 1042
Considerazioni 1043
Prerequisiti 1043
Passaggio 1: Creazione di un ruolo IAM 1044
Fase 2: ottenere il driver CSI di Amazon EBS 1052
Fase 3: implementazione di un'applicazione di esempio 1053
Amazon EFS 1053
Considerazioni 1053
Prerequisiti 1054
Passaggio 1: Creazione di un ruolo IAM 1055
Fase 2: ottenere il driver CSI per Amazon EFS 1061
Fase 3: Creazione di un file system Amazon EFS 1061
Fase 4: implementazione di un'applicazione di esempio 1061

Amazon FSx per Lustre 1062
Implementare il driver 1062
Ottimizza (EFA) 1069
Ottimizza (non EFA) 1086
Amazon FSx per NetApp ONTAP 1093
Amazon FSx per OpenZFS 1094
Amazon File Cache 1095
Mountpoint per Amazon S3 1095
Considerazioni 1095
Implementazione del driver 1096
Rimozione del driver 1106
Controller snapshot CSI 1108
Configurazione delle reti 1110
Aggiunta di una sottorete VPC esistente a un cluster Amazon EKS dalla console di gestione . 1110
Requisiti del VPC e delle sottoreti 1110
Considerazioni e requisiti relativi al VPC 1111
Considerazioni e requisiti relativi alle sottoreti 1113
Considerazioni e requisiti relativi alle sottoreti condivisi 1119
Crea un VPC 1120
Prerequisiti 51
Sottoreti pubbliche e private 1121
Solo sottoreti pubbliche 1124
Solo sottoreti private 1125
Requisiti relativi al gruppo di sicurezza 1127
Gruppo di sicurezza del cluster predefinito 1127
Limitazione del traffico cluster 1129
Gruppi di sicurezza condivisi 1130
Gestione di componenti aggiuntivi di rete 1131
Componenti aggiuntivi integrati 1131
Componenti aggiuntivi di rete opzionali AWS 1132
CNI di Amazon VPC 1132
Plug-in CNI alternativi 1256
Multus 1258
AWS Controller Load Balancer 1259
CoreDNS 1278
kube-proxy 1296

Carichi di lavoro 1302
Implementazione esemplificativa (Linux) 1302
Prerequisiti 51
Creare uno spazio dei nomi 1303
Creazione di un'implementazione Kubernetes 1303
Crea un servizio. 1305
Esame delle risorse create 1306
Eseguire una shell su un pod 1309
Fasi successive 1310
Implementazione esemplificativa (Windows) 1310
Prerequisiti 51
Creare uno spazio dei nomi 1303
Creazione di un'implementazione Kubernetes 1303
Crea un servizio. 1305
Esame delle risorse create 1313
Eseguire una shell su un pod 1309
Fasi successive 1317
Vertical Pod Autoscaler 1317
Implementazione di Vertical Pod Autoscaler 1318
Test dell'installazione di Vertical Pod Autoscaler 1319
Horizontal Pod Autoscaler 1323
Esecuzione di un'applicazione di test di Horizontal Pod Autoscaler 1324
Bilanciamento del carico di rete 1327
Prerequisiti 51
Considerazioni 100
Creazione di un Network Load Balancer 1331
(Facoltativo) Implementare un'applicazione di esempio 1333
Bilanciamento del carico di applicazione 1337
Prerequisiti 51
Riutilizzo con Ingress Groups ALBs 1340
(Facoltativo) Implementare un'applicazione di esempio 1341
Limitazione degli IP esterni ai servizi 1344
Copia di un'immagine in un repository 1347
Visualizza i registri delle immagini di Amazon 1350
Componenti aggiuntivi Amazon EKS 1353
Considerazioni 1354

Namespace personalizzato per componenti aggiuntivi 1355
Considerazioni per la modalità automatica di Amazon EKS 1356
Supporto 1357
AWS componenti aggiuntivi 1359
Componenti aggiuntivi della community 1382
Componenti aggiuntivi di Marketplace 1388
Creare un componente aggiuntivo 1406
Aggiornamento di un componente aggiuntivo 1419
Verifica la compatibilità 1428
Rimozione di un componente aggiuntivo 1430
Ruoli IAM 1433
Campi che puoi personalizzare 1440
Verifica delle immagini di container 1444
Funzionalità EKS 1445
Funzionalità disponibili 1445
AWS Controller per Kubernetes (ACK) 1445
Argo CD 1446
kro (Kube Resource Orchestrator) 1446
Vantaggi delle funzionalità EKS 1447
Prezzi 248
Come funzionano le funzionalità EKS 1448
Casi di utilizzo comune 1449
Lavorare con le funzionalità 1450
Risorse di funzionalità EKS 1450
Comprensione dello stato delle capacità 1451
Crea funzionalità 1452
Elenca le funzionalità 1453
Descrivi una funzionalità 1454
Aggiorna la configurazione di una funzionalità 1455
Eliminare una funzionalità 1456
Fasi successive 96
risorse Kubernetes 1457
Risorse Argo CD 1458
risorse kro 1459
Risorse ACK 1460
Limiti delle risorse 1461

Fasi successive 96
Considerazioni 1462
Capability, ruoli IAM e Kubernetes RBAC 1462
Modelli di architettura multicluster 1463
Confronto tra le funzionalità EKS e le soluzioni autogestite 1465
AWS Controller per Kubernetes 1468
Come funziona ACK 1468
Vantaggi di ACK 1469
AWS Servizi supportati 1469
Integrazione con altre funzionalità gestite da EKS 1470
Guida introduttiva a ACK 1470
Crea una funzionalità ACK 1471
Concetti ACK 1481
Configurazione delle autorizzazioni 1489
Considerazioni 1496
Risoluzione dei problemi 1504
Confronto con l'autogestione 1508
Argo CD 1510
Come funziona Argo CD 1510
Vantaggi di Argo CD 1512
Integrazione con Identity Center AWS 1512
Integrazione con altre funzionalità gestite da EKS 1470
Guida introduttiva ad Argo CD 1513
Crea una funzionalità Argo CD 1513
Concetti di Argo CD 1524
Configurazione delle autorizzazioni 1530
Lavorare con Argo CD 1544
Considerazioni su Argo CD 1587
Risoluzione dei problemi 1596
Confronto con l'autogestione 1601
kro 1606
Come funziona kro 1606
Vantaggi di kro 1607
Integrazione con altre funzionalità gestite da EKS 1470
Guida introduttiva a kro 1608
Crea una capacità kro 1609

concetti kro ... 1621
Configurazione delle autorizzazioni ... 1628
Considerazioni ... 1633
Risoluzione dei problemi ... 1638
Confronto con quello autogestito ... 1645
Risoluzione dei problemi relativi alle funzionalità ... 1646
Un approccio generale alla risoluzione dei problemi ... 1646
Verifica lo stato delle funzionalità ... 1647
Stati di funzionalità comuni ... 1648
Verifica lo stato delle risorse Kubernetes ... 1648
Verifica le autorizzazioni IAM e l'accesso al cluster ... 1649
Risoluzione dei problemi specifici per funzionalità ... 1651
Problemi comuni a tutte le funzionalità ... 1651
Fasi successive ... 96
Gestione dei cluster ... 1653
Monitoraggio dei costi ... 1653
Visualizzare i costi per pod ... 1654
Installare Kubecost ... 1655
Accedere alla dashboard Kubecost ... 1657
Informazioni su Kubecost ... 1658
Metrics Server ... 1671
Considerazioni ... 100
Implementa come componente aggiuntivo della community con i componenti aggiuntivi di Amazon EKS ... 1671
Implementa con manifesto ... 1672
Implementazione di app con Helm ... 1673
Assegnazione di tag alle risorse ... 1675
Nozioni di base sui tag ... 1675
Assegnazione di tag alle risorse ... 1676
Limitazioni applicate ai tag ... 1677
Tagging delle risorse per la fatturazione ... 1677
Utilizzo di tag tramite la console ... 1678
Utilizzo di tag tramite la CLI, l'API o `eksctl` ... 1679
Quote di servizio ... 1681
Visualizza le quote dei servizi EKS nel Console di gestione AWS ... 1681
Visualizza le quote dei servizi EKS con la CLI AWS ... 1682

Service Quotas di Amazon EKS 1682
AWS Quote di servizio Fargate 1683
Sicurezza 1684
Best practice 1685
Analizzare le vulnerabilità 1685
Il Center for Internet Security (CIS) Benchmark per Amazon EKS 1686
Versioni della piattaforma Amazon EKS 1686
Elenco delle vulnerabilità del sistema operativo 1686
Rilevamento dei nodi con Amazon Inspector 1687
Rilevamento di cluster e nodi con Amazon GuardDuty 1687
Convalida della conformità 1687
Considerazioni per EKS 1688
Sicurezza dell'infrastruttura 1689
Resilienza 1693
Prevenzione del problema "confused deputy" tra servizi 1694
Considerazioni su Kubernetes 1696
Firma dei certificati 1696
Ruoli e utenti predefiniti 1699
Abilitare la crittografia dei segreti 1703
AWS Secrets Manager 1708
Crittografia a busta predefinita per tutti i dati dell'API Kubernetes 1709
Considerazioni per EKS Auto 1717
Sicurezza e autenticazione delle API 1718
Sicurezza di rete 1718
Sicurezza delle istanze gestite di EC2 1718
Gestione automatizzata delle risorse 1720
Best practice di sicurezza 1720
Considerazioni sulle funzionalità EKS 1721
Capacità, ruolo IAM 1721
Voci di accesso EKS 1722
Autorizzazioni Kubernetes aggiuntive 1723
Autorizzazioni IAM richieste per funzionalità 1725
Best practice di sicurezza 1726
Fasi successive 96
Riferimento IAM 1729
Destinatari 1729

Autenticazione con identità 1730
Gestione dell'accesso con policy 1733
Amazon EKS e IAM 1736
Policy basate sull'identità 1740
Ruoli collegati ai servizi 1748
Ruolo IAM per l'esecuzione del pod 1765
Ruolo IAM di Connector 1770
AWS politiche gestite 1774
Risoluzione dei problemi 1804
Ruolo IAM del cluster 1807
Ruolo IAM del nodo 1810
Ruolo IAM del cluster Auto Mode 1816
Ruolo IAM del nodo in Auto Mode 1819
Capacità, ruolo IAM 1822
Monitoraggio dei cluster 1829
Registrazione e monitoraggio su Amazon EKS 1829
Strumenti di registrazione e monitoraggio di Amazon EKS 1831
Dashboard di osservabilità 1835
Riepilogo 1835
Integrità del cluster 1835
Monitoraggio del piano di controllo 1835
Approfondimenti sui cluster 1838
Problemi di integrità dei nodi 1838
Funzionalità EKS 1838
Osservabilità della rete di container 1839
Casi d'uso 149
Funzionalità 80
Nozioni di base 307
Prerequisiti e note importanti 1841
Utilizzo di AWS CLI, API EKS e API NFM 1842
Utilizzo dell'infrastruttura come codice (IaC) 1844
Come funziona? 201
Considerazioni e limitazioni 1854
Parametri Prometheus 1855
Passaggio 1: attivazione delle metriche Prometheus 1856
Passaggio 2: utilizzo delle metriche Prometheus 1858

Passaggio 3: gestione degli scraper Prometheus 1858
Implementare con Helm 1858
Piano di controllo (control-plane) 1862
Amazon CloudWatch 1870
Metriche di base in Amazon CloudWatch 1870
Operatore di CloudWatch osservabilità di Amazon 1880
Registrazione dei log del piano di controllo (control-plane) 1880
Abilitare o disabilitare i log del piano di controllo (control-plane) 1882
Visualizzare i log del piano di controllo (control-plane) del cluster 1884
AWS CloudTrail 1886
Riferimenti 1886
Voci del file di log 1887
Parametri del gruppo con scalabilità automatica 1890
Operatore ADOT 1890
Pannello di controllo Amazon EKS 1891
Cos'è il pannello di controllo Amazon EKS? 1891
In che modo la dashboard utilizza AWS Organizations? 1891
Abilita la dashboard EKS utilizzando la console AWS 1893
Visualizzazione del pannello di controllo EKS 1894
Configura il pannello di controllo 1894
Disattiva la dashboard EKS utilizzando la AWS console 1895
Risoluzione dei problemi del pannello di controllo EKS 1896
Configurare le organizzazioni 1896
Uso di altri servizi 1903
AWS Backup 1903
AWS CloudFormation 1904
Modelli Amazon EKS e AWS CloudFormation 1904
Ulteriori informazioni su AWS CloudFormation 1905
AWS CodeConnections 1905
Utilizzare AWS CodeConnections con Argo CD 1905
Considerazioni sull'utilizzo con Argo CD CodeConnections 1906
Amazon Detective 1907
Uso di Amazon Detective con Amazon EKS 1907
Amazon GuardDuty 1908
AWSZone locali 1909
AWS Resilience Hub 1910

AWS Secrets Manager 1910
Usare AWS Secrets Manager con Argo CD 1910
Amazon Security Lake 1912
Vantaggi dell'utilizzo di Security Lake con Amazon EKS 1913
Abilitare Security Lake per Amazon EKS 1913
Analisi dei log EKS in Security Lake 1914
Amazon VPC Lattice 1914
Connettore Amazon EKS 1915
Considerazioni su Amazon EKS Connector 1915
Ruoli IAM richiesti per Amazon EKS Connector 1916
Connettere un cluster 1916
Considerazioni 1917
Prerequisiti 1917
Fase 1: Registrazione del cluster 1917
Passaggio 2: installazione dell'agente `eks-connector` 1920
Passaggi successivi 1922
Garantire l'accesso ai cluster 1922
Prerequisiti 1922
Annullamento della registrazione di un cluster 1924
Annullare la registrazione del cluster Kubernetes 1924
Eliminare le risorse del cluster Kubernetes 1925
Risoluzione dei problemi del connettore EKS 1926
Risoluzione dei problemi di base 1926
Problema di Helm: 403 Forbidden 1928
Errore della console: il cluster è bloccato nello stato Pending (In sospeso) 1928
Errore della console: User system:serviceaccount:eks-connector:eks-connector non può impersonare la risorsa utenti nel gruppo API nell'ambito del cluster 1929
Errore della console: […] non consentito: L'utente […] non può elencare la risorsa […] nel gruppo API nell'ambito del cluster 1929
Errore della console: Amazon EKS non è in grado di comunicare con il server API cluster Kubernetes. Per una connessione corretta, il cluster deve trovarsi nello stato ACTIVE (ATTIVO). Riprova tra qualche minuto. 1930
I pod di Amazon EKS Connector stanno eseguendo un ciclo di arresto anomalo 1930
Impossibile avviare eks-connector: InvalidActivation 1931
Nel nodo del cluster manca la connettività in uscita 1932
I pod del connettore Amazon EKS sono in stato `ImagePullBackOff` 1932

Domande frequenti 1933
Considerazioni relative alla sicurezza 1934
Responsabilità di AWS 1934
Responsabilità del cliente 1935
Amazon EKS su AWS Outposts 1936
Quando utilizzare ciascuna opzione di implementazione 1936
Confronto tra le opzioni di implementazione 1937
Esecuzione di cluster locali 1940
Regioni AWS supportate 1940
Implementazione di un cluster locale 1940
Versioni della piattaforma EKS 1951
Creazione di un VPC e delle sottoreti 1965
Preparazione per le disconnessioni 1969
Considerazioni sulla capacità 1974
Risoluzione dei problemi dei cluster 1976
Nodi 1988
`eksctl` 1990
Console di gestione AWS 1992
IA/ML su EKS 1999
Perché scegliere EKS per IA/ML? 1999
Casi d'uso principali 2000
Casi di studio 2001
Inizia a usare il machine learning su EKS 2002
Inferenza in tempo reale 2002
Crea cluster 2002
Inferenza LLM con vLLm 2039
Configurazione del cluster 2057
Usa la GPU EKS Linux AMIs 2057
Installa il plug-in del dispositivo per GPUs 2062
Configura le AMI GPU Windows 2069
Configurazione dei cluster di formazione con EFA 2075
Configura cluster di inferenza con Inferentia 2086
Gestione della capacità 2092
GPU di riserva per gruppi di nodi gestiti 2093
GPU di riserva per i nodi autogestiti 2096
Usa P6e- 00 con EKS GB2 2099

Ricette 2110
Impedire che i pod siano pianificati su nodi specifici 2110
Risorse 2112
Workshop 2112
Best practice 2113
Architetture di riferimento 2114
Tutorial 2115
Tools (Strumenti) 2117
Model Context Protocol (MCP) 2117
Panoramica di 94
Esempi di integrazione 2119
Nozioni di base 307
Nozioni di base 2120
Configurazione 2136
Tools (Strumenti) 2138
Amazon Q 2149
Come funziona 507
Utilizzo di Amazon Q per la risoluzione dei problemi 2150
Considerazioni 100
Ulteriori informazioni 2151
Controllo delle versioni 2152
Versioni di Kubernetes 2152
Versioni disponibili con supporto standard 2153
Versioni disponibili con supporto esteso 2153
Calendario di rilascio di Amazon EKS Kubernetes 2153
Ottieni informazioni sulla versione con la AWS CLI 2154
Versione Amazon EKS FAQs 2156
Supporto esteso per Amazon EKS FAQs 2158
Versioni di supporto standard 2160
Versioni di supporto esteso 2168
Visualizzazione del periodo di supporto 2171
Visualizzazione della policy di aggiornamento 2171
Abilitare il supporto esteso 2173
Disabilitare il supporto esteso 2175
Versioni della piattaforma 2176
Versione `1.35` di Kubernetes 2177

Versione 1.34 di Kubernetes 2178
Versione 1.33 di Kubernetes 2180
Versione 1.32 di Kubernetes 2183
Versione 1.31 di Kubernetes 2188
Versione 1.30 di Kubernetes 2194
Versione 1.29 di Kubernetes 2201
Ottieni versione corrente della piattaforma 2209
Modificare la versione della piattaforma 2210
Risoluzione dei problemi 2211
Capacità insufficiente 2211
Impossibile aggiungere i nodi al cluster 2211
Accesso negato o non autorizzato (kubectl) 2213
hostname doesn't match 2214
getsockopt: no route to host 2214
Instances failed to join the Kubernetes cluster 2215
Codici di errore dei gruppi di nodi gestiti 2215
Not authorized for images 2220
Il nodo è in stato NotReady 2220
Raccoglitore di log EKS 2221
Rete runtime container non pronta 2222
Timeout dell'handshake TLS 2223
InvalidClientTokenId 2224
Prima di poter aggiornare il piano di controllo, i gruppi di nodi devono corrispondere alla versione di Kubernetes 2224
All'avvio di molti nodi, si verificano errori Too Many Requests 2225
Risposta agli errori HTTP 401 Unauthorized (Autorizzazione negata) per le richieste del server API Kubernetes 2225
La versione della piattaforma Amazon EKS è più avanti di due versioni rispetto all'attuale versione della piattaforma 2226
Domande frequenti sull'integrità del cluster e codici di errore con percorsi di risoluzione 2229
Progetti relativi ad Amazon EKS 2235
Supporto per il software distribuito su EKS 2235
Strumenti di gestione 2236
eksctl 2236
AWS Controller per Kubernetes 2236
kro (Kube Resource Orchestrator) 2237

Argo CD 2237
CD Flux 2237
CDK per Kubernetes 2237
Rete 2238
Plug-in CNI di Amazon VPC per Kubernetes 2238
AWS Load Balancer Controller per Kubernetes 2238
ExternalDNS 2238
Machine learning 2238
Kubeflow 2239
Auto Scaling 2239
Cluster Autoscaler 2239
Karpenter 2239
Escalator 2239
Monitoraggio 2240
Prometheus 2240
Integrazione continua / distribuzione continua 2240
Jenkins X 2240
Nuove caratteristiche e roadmap 2241
Cronologia dei documenti 2242
Contributi 2304
Modificare una singola pagina 2304
Prerequisiti 51
Procedura 103
Panoramica della richiesta di pull 2306
Modifica i file con GitHub 2307
Prerequisiti 51
Procedura 103
Visualizzazione di un feedback sullo stile 2309
Installa Java 2309
Installa l'estensione VS Code Vale 2310
Sincronizzazione di Vale 2310
Visualizza il feedback di stile in VS Code 2310
Creazione di una pagina 2310
Creazione di una pagina 2311
Aggiungi pagina alla navigazione 2311
Inserire link 2311

Collegamento a una pagina o sezione della Guida per l'utente EKS 2312
Collegamento a un'altra guida nei AWS Documenti 2312
Collegamento a una pagina Web esterna 2312
Creare con Amazon Q 2313
Installare Amazon Q con VS Code 2313
Accedere ad Amazon Q 2314
Utilizzare Amazon Q per creare i contenuti 2314
Suggerimenti 2314
Visualizzazione dell'anteprima PR 2315
Visualizzazione dell'anteprima 2316
Limitazioni dell'anteprima 2316
Sintassi AsciiDoc 2317
Nuova pagina 2317
Intestazioni 2317
Formattazione del testo 2317
Elenchi 2318
Link 2318
Esempi di codice 2318
Immagini 2319
Tabelle 2319
Callout 2319
Includere altri file 2320
Attributi (simili alle entità) 2320
Procedure 2320

Contribuisci a migliorare questa pagina

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.

Per contribuire a questa guida per l'utente, scegli il GitHub link Modifica questa pagina nel riquadro destro di ogni pagina.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.

Che cosa è Amazon EKS?

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

Amazon EKS: gestione semplificata di Kubernetes

Amazon Elastic Kubernetes Service (EKS) fornisce un servizio Kubernetes completamente gestito che elimina la complessità del funzionamento dei cluster Kubernetes. Con EKS è possibile:

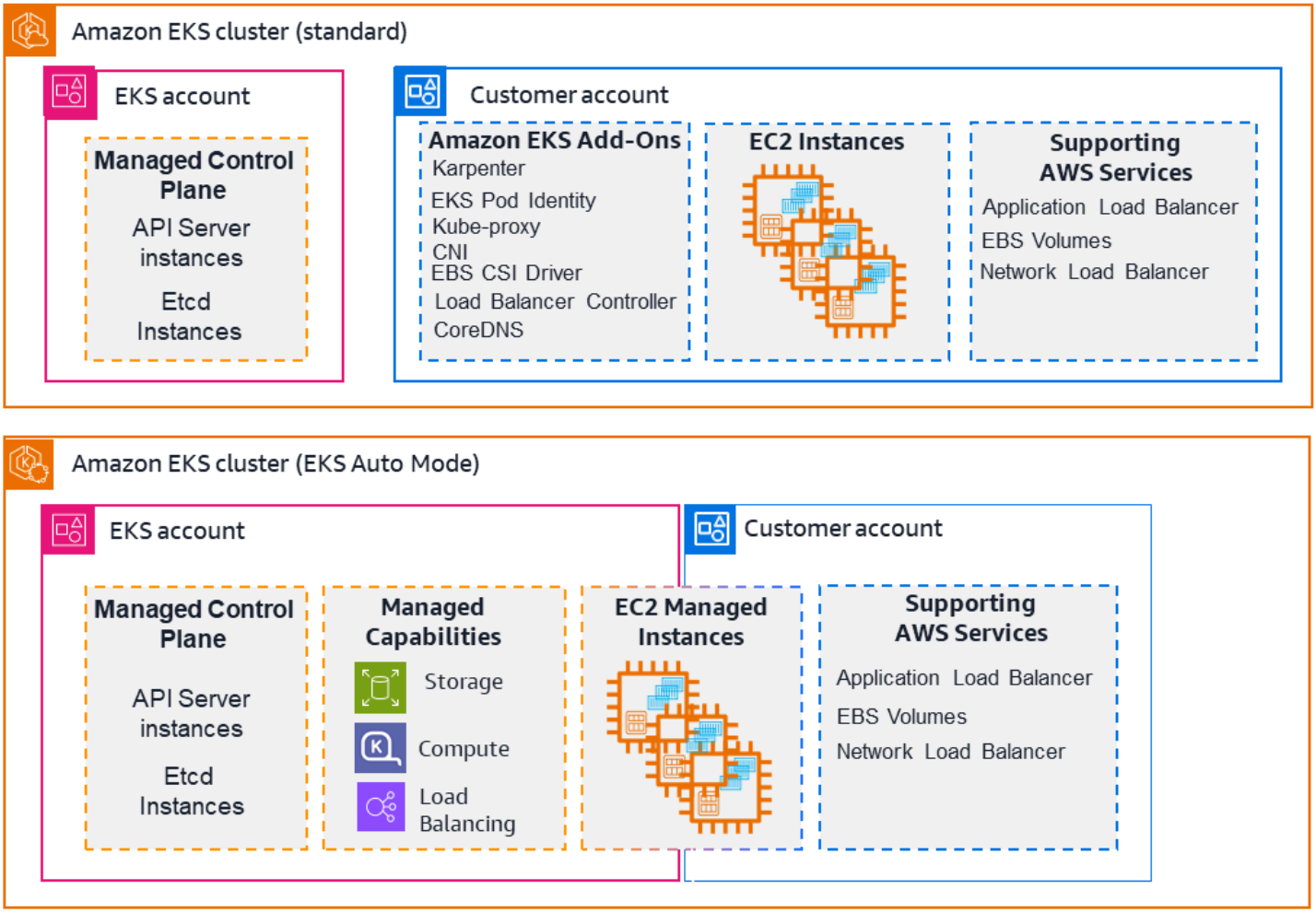
- Implementare le applicazioni più velocemente con meno spese operative
- Scalare senza problemi per soddisfare le mutevoli esigenze dei carichi di lavoro
- Migliora la sicurezza attraverso AWS l'integrazione e gli aggiornamenti automatici
- Scegliere tra la modalità automatica EKS standard o completamente automatizzata

Amazon Elastic Kubernetes Service (Amazon EKS) è la piattaforma principale per l'esecuzione di cluster Kubernetes, sia nel cloud Amazon Web Services (AWS) che nei tuoi data center (EKS Anywhere e Amazon EKS Hybrid Nodes).

Amazon EKS semplifica la creazione, la protezione e la manutenzione dei cluster Kubernetes. Fornire risorse sufficienti per soddisfare i picchi di domanda può essere più conveniente rispetto alla manutenzione dei propri data center. Due degli approcci principali all'utilizzo di Amazon EKS sono i seguenti:

- Standard EKS: AWS gestisce il piano di controllo di Kubernetes quando si crea un cluster con EKS. I componenti che gestiscono i nodi, pianificano i carichi di lavoro, si integrano con il AWS cloud e archiviano e ridimensionano le informazioni del piano di controllo per mantenere i cluster attivi e funzionanti, vengono gestiti automaticamente.
- Modalità automatica EKS: utilizzando la funzione Modalità automatica EKS, EKS estende il proprio controllo anche alla gestione dei nodi (piano dati Kubernetes). Semplifica la gestione di Kubernetes effettuando automaticamente il provisioning dell'infrastruttura, selezionando le istanze di calcolo ottimali, scalando dinamicamente le risorse, ottimizzando continuamente i costi, applicando patch ai sistemi operativi e integrandosi con i servizi di sicurezza. AWS

Il diagramma seguente illustra come Amazon EKS integra i tuoi cluster Kubernetes con il AWS cloud, a seconda del metodo di creazione del cluster scelto:



Amazon EKS ti aiuta a rimuovere gli attriti e ad accelerare i tempi di produzione, migliorare le prestazioni, la disponibilità e la resilienza e migliorare la sicurezza del sistema. Per ulteriori informazioni, consulta Amazon Elastic Kubernetes Service.

Creazione e scalabilità con Kubernetes: Amazon EKS Capabilities

Amazon EKS non solo ti aiuta a creare e gestire cluster, ma ti aiuta a creare e scalare sistemi applicativi con Kubernetes. Amazon EKS Capabilities sono servizi cluster completamente gestiti che estendono le funzionalità del cluster con strumenti nativi Kubernetes a mani libere, tra cui:

- Argo CD: Argo CD fornisce una distribuzione continua dichiarativa e GitOps basata su dichiarazioni per i carichi di lavoro, le risorse e l'infrastruttura cloud. AWS

- AWS Controller for Kubernetes (ACK): ACK consente la creazione nativa di Kubernetes e la gestione del ciclo di vita delle risorse, unificando l'orchestrazione dei carichi di lavoro e i flussi di lavoro. AWS Infrastructure-as-code
- kro (Kube Resource Orchestrator): kro estende le funzionalità native di Kubernetes per semplificare la creazione, l'orchestrazione e la composizione di risorse personalizzate, offrendoti gli strumenti per creare i tuoi elementi costitutivi cloud personalizzati.

EKS Capabilities sono risorse cloud che riducono al minimo l'onere operativo di installazione, manutenzione e scalabilità di questi componenti fondamentali della piattaforma nei cluster, consentendoti di concentrarti sulla creazione di software anziché sulle operazioni della piattaforma cluster.

Per ulteriori informazioni, consulta Funzionalità EKS.

Funzionalità di Amazon EKS

Amazon EKS offre le seguenti funzionalità di alto livello:

Interfacce di gestione

EKS offre diverse interfacce per il provisioning, la gestione e la manutenzione dei cluster, tra cui Console di gestione AWS Amazon EKS API/SDKs, CDK, AWS CLI, eksctl CLI e Terraform. AWS CloudFormation Per ulteriori informazioni, consultare Nozioni di base e Configurazione dei cluster.

Accedere agli strumenti di controllo

EKS si affida alle funzionalità di Kubernetes e AWS Identity and Access Management (AWS IAM) per gestire l'accesso da utenti e carichi di lavoro. Per ulteriori informazioni, consultare the section called “Accesso API Kubernetes” e the section called “Accesso ai carichi di lavoro a AWS ”.

Risorse di calcolo

Per quanto riguarda le risorse di calcolo, EKS consente l'intera gamma di tipi di EC2 istanze Amazon e AWS innovazioni come Nitro e Graviton con Amazon EKS per ottimizzare l'elaborazione per i tuoi carichi di lavoro. Per ulteriori informazioni, consulta Gestione del calcolo.

Storage

Modalità automatica EKS crea automaticamente classi di archiviazione utilizzando volumi EBS, Utilizzando i driver dell'interfaccia di archiviazione in container (CSI), è anche possibile

utilizzare Amazon S3, Amazon EFS, Amazon FSx e Amazon File Cache per le tue esigenze di archiviazione delle applicazioni. Per ulteriori informazioni, consulta Archiviazione di dati per app.

Sicurezza

Il modello di responsabilità condivisa è impiegato in relazione alla Sicurezza in Amazon EKS. Per ulteriori informazioni, consulta Best practice di sicurezza, Sicurezza dell'infrastruttura e Sicurezza Kubernetes.

Strumenti di monitoraggio

Utilizza la dashboard di osservabilità per monitorare i cluster Amazon EKS. Gli strumenti di monitoraggio includono Prometheus CloudWatch, Cloudtrail e ADOT Operator. Per ulteriori informazioni su dashboard, server di metriche e altri strumenti, consulta Costi del cluster EKS e Server delle metriche Kubernetes.

Funzionalità del cluster

EKS offre funzionalità di cluster gestite per l'implementazione continua, la gestione delle risorse cloud e la composizione delle risorse basate su innovazioni open source. EKS installa Kubernetes APIs nei cluster, ma i controller e gli altri componenti vengono eseguiti in EKS e sono completamente gestiti, fornendo patch, scalabilità e monitoraggio automatizzati. Per ulteriori informazioni, consulta Funzionalità EKS.

Compatibilità e supporto Kubernetes

Amazon EKS è certificato conforme a Kubernetes, quindi è possibile implementare applicazioni compatibili con Kubernetes senza rifattorizzare e utilizzare strumenti e plugin della community Kubernetes. EKS offre supporto standard e supporto esteso per Kubernetes. Per ulteriori informazioni, consulta the section called "Versioni di Kubernetes".

Servizi correlati

Servizi da utilizzare con Amazon EKS

Puoi utilizzare altri AWS servizi con i cluster che distribuisci con Amazon EKS:

Amazon EC2

Ottieni capacità di elaborazione scalabile su richiesta con Amazon. EC2

Amazon EBS

Collegare risorse di archiviazione a blocchi scalabili e ad alte prestazioni con Amazon EBS.

Amazon ECR

Archiviare le immagini dei container in modo sicuro con Amazon ECR.

Amazon CloudWatch

Monitora AWS risorse e applicazioni in tempo reale con Amazon CloudWatch.

Amazon Prometheus

Tenere traccia delle metriche per le applicazioni all'interno di container con Servizio gestito da Amazon per Prometheus.

Elastic Load Balancing

Distribuire il traffico in ingresso tra più destinazioni con Elastic Load Balancing.

Amazon GuardDuty

Rileva le minacce ai cluster EKS con Amazon GuardDuty.

AWS Hub di resilienza

Valutare la resilienza dei cluster EKS con AWS Resilience Hub.

Prezzi di Amazon EKS

Amazon EKS prevede prezzi per cluster basati sul supporto della versione del cluster Kubernetes, prezzi per modalità automatica Amazon EKS e prezzi per vCPU per Amazon EKS Hybrid Nodes.

Quando usi Amazon EKS, paghi separatamente per le AWS risorse utilizzate per eseguire le tue applicazioni sui nodi di lavoro Kubernetes. Ad esempio, se utilizzi nodi di lavoro Kubernetes come EC2 istanze Amazon con volumi e indirizzi IPv4 pubblici Amazon EBS, ti verrà addebitata la capacità dell'istanza tramite Amazon, la capacità di volume tramite EC2 Amazon EBS e l'indirizzo IPv4 tramite Amazon VPC.

Visita le rispettive pagine dei prezzi dei AWS servizi che utilizzi con le tue applicazioni Kubernetes per informazioni dettagliate sui prezzi.

- Per i prezzi di cluster Amazon EKS, Amazon EKS Auto Mode, Amazon EKS Capabilities e Amazon EKS Hybrid Nodes, consulta i prezzi di Amazon EKS.
- Per i EC2 prezzi di Amazon, consulta i prezzi di Amazon EC2 On-Demand e i prezzi Amazon EC2 Spot.

- Per i prezzi di AWS Fargate, consulta i prezzi di Fargate AWS .
- Puoi utilizzare i tuoi savings plans per le risorse di calcolo utilizzate nei cluster Amazon EKS. Per ulteriori informazioni, consulta Prezzi con Savings Plans.

Casi d'uso comuni in Amazon EKS

Amazon EKS offre solidi servizi Kubernetes gestiti su AWS, progettati per ottimizzare le applicazioni containerizzate. Di seguito sono riportati alcuni dei casi d'uso più comuni di Amazon EKS per aiutarti a sfruttarne i punti di forza per esigenze specifiche.

Implementazione di applicazioni ad alta disponibilità

Utilizzando Elastic Load Balancing, puoi assicurarti che le applicazioni siano altamente disponibili in più zone di disponibilità.

Creazione di architetture di microservizi

Utilizzare le funzionalità di rilevamento servizi di Kubernetes con AWS Cloud Map o Amazon VPC Lattice per creare sistemi resilienti.

Automazione dei processi di rilascio del software

Gestisci le pipeline di integrazione continua e distribuzione continua (CI/CD) che semplificano il processo di creazione, test e implementazione automatizzati delle applicazioni. Per una distribuzione continua dichiarativa, consulta Continuous Deployment with Argo CD.

Esecuzione di applicazioni serverless

Utilizzare AWS Fargate con Amazon EKS per eseguire applicazioni serverless. In questo modo, potrai concentrarti esclusivamente sullo sviluppo delle applicazioni mentre Amazon EKS e Fargate gestiscono l'infrastruttura sottostante.

Esecuzione di carichi di lavoro di machine learning

Amazon EKS è compatibile con i framework di machine learning più diffusi, come TensorFlow, MXNet e PyTorch. Con il supporto GPU, è possibile gestire efficacemente anche attività di machine learning complesse.

Implementazione coerente in locale e nel cloud

Per semplificare l'esecuzione di Kubernetes in ambienti on-premises, è possibile utilizzare gli stessi cluster, funzionalità e strumenti di Amazon EKS per eseguire nodi autogestiti su AWS Outposts o utilizzare Amazon EKS Hybrid Nodes con la propria infrastruttura. Per ambienti

autonomi e isolati, è possibile utilizzare Amazon EKS Anywhere per automatizzare la gestione del ciclo di vita dei cluster di Kubernetes sulla propria infrastruttura.

Esecuzione di carichi di lavoro di elaborazione in batch e big data a costi contenuti

Utilizzare le istanze spot per eseguire carichi di lavoro di elaborazione in batch e big data, come Apache Hadoop e Spark, a un costo ridotto. In questo modo puoi sfruttare la EC2 capacità inutilizzata di Amazon a prezzi scontati.

Gestione delle AWS risorse da Kubernetes

Usa AWS Controllers for Kubernetes (ACK) per creare e gestire AWS risorse direttamente dal tuo cluster Kubernetes utilizzando Kubernetes nativo. APIs

Creazione di astrazioni ingegneristiche della piattaforma

Crea Kubernetes personalizzati APIs che compongono più risorse in astrazioni di livello superiore usando kro (Kube Resource Orchestrator).

Proteggere le applicazioni e garantire la conformità

Implementa pratiche di sicurezza solide e mantieni la conformità con Amazon EKS, che si integra con servizi di AWS sicurezza come AWS Identity and Access Management (IAM), Amazon Virtual Private Cloud (Amazon VPC) AWS e Key Management Service AWS (KMS). Ciò garantisce la privacy e la protezione dei dati secondo gli standard del settore.

Architettura di Amazon EKS

Amazon EKS segue l'architettura generale dei cluster di Kubernetes. Per ulteriori informazioni, consultare la pagina Kubernetes Components nella documentazione Kubernetes. Le seguenti sezioni riassumono alcuni dettagli aggiuntivi sull'architettura di Amazon EKS.

Piano di controllo

Amazon EKS garantisce che ogni cluster abbia un proprio piano di controllo Kubernetes specifico. Questo design mantiene l'infrastruttura di ciascun cluster separata, senza sovrapposizioni tra cluster o account. AWS La configurazione include:

Componenti distribuiti

Il piano di controllo posiziona almeno due istanze del server API e tre istanze etcd in tre zone di AWS disponibilità all'interno di una regione. AWS

Prestazioni ottimali

Amazon EKS monitora e regola attivamente le istanze del piano di controllo per mantenere le massime prestazioni.

Resilienza

Se un'istanza del piano di controllo fallisce, Amazon EKS la sostituisce rapidamente, utilizzando una zona di disponibilità diversa, se necessario.

Tempo di attività costante

Eseguendo i cluster su più zone di disponibilità, si consegue l'affidabilità dell'Accordo sul livello di servizio (SLA) sulla disponibilità degli endpoint dei server API.

Amazon EKS utilizza Amazon Virtual Private Cloud (Amazon VPC) per limitare il traffico tra i componenti del piano di controllo all'interno di un singolo cluster. I componenti del cluster non possono visualizzare o ricevere comunicazioni da altri cluster o AWS account, tranne quando autorizzati dalle politiche di controllo degli accessi basato sui ruoli (RBAC) di Kubernetes.

Calcolo

Oltre al piano di controllo, un cluster Amazon EKS dispone di un set di macchine di lavoro chiamate nodi. La selezione del tipo di nodo del cluster Amazon EKS appropriato è fondamentale per soddisfare i propri requisiti specifici e ottimizzare l'utilizzo delle risorse. Amazon EKS offre i seguenti tipi di nodi primari:

Modalità automatica EKS

EKS Auto Mode estende la AWS gestione oltre il piano di controllo per includere il piano dati, automatizzando la gestione dell'infrastruttura del cluster. Integra le funzionalità di base di Kubernetes come componenti integrati, tra cui scalabilità automatica del calcolo, rete, bilanciamento del carico, DNS, archiviazione e supporto della GPU. EKS Auto Mode gestisce dinamicamente i nodi in base alle richieste del carico di lavoro, utilizzando funzionalità di sicurezza immutabili AMIs e avanzate. Automatizza aggiornamenti e upgrade rispettando i budget per le interruzioni dei pod e include componenti gestiti che altrimenti richiederebbero la gestione di componenti aggiuntivi. Questa opzione è ideale per gli utenti che desiderano sfruttare l' AWS esperienza per day-to-day le operazioni, ridurre al minimo il sovraccarico operativo e concentrarsi sullo sviluppo di applicazioni piuttosto che sulla gestione dell'infrastruttura.

AWS Fargate

Fargate è un motore di elaborazione serverless per container che elimina la necessità di gestire le istanze sottostanti. Fargate consente di specificare le esigenze di risorse dell'applicazione e di effettuare AWS automaticamente il provisioning, la scalabilità e la manutenzione dell'infrastruttura. Questa opzione è ideale per gli utenti che danno priorità ease-of-use e vogliono concentrarsi sullo sviluppo e l'implementazione delle applicazioni piuttosto che sulla gestione dell'infrastruttura.

Karpenter

Karpenter è un cluster autoscaler Kubernetes flessibile e ad alte prestazioni che aiuta a migliorare la disponibilità delle applicazioni e l'efficienza del cluster. Karpenter avvia risorse di calcolo dimensionate in modo ottimale in risposta alla modifica del carico dell'applicazione. Questa opzione può fornire risorse di just-in-time elaborazione che soddisfino i requisiti del carico di lavoro.

Gruppi di nodi gestiti

I gruppi di nodi gestiti sono una combinazione di automazione e personalizzazione per la gestione di una raccolta di EC2 istanze Amazon all'interno di un cluster Amazon EKS. AWS si occupa di attività come l'applicazione di patch, l'aggiornamento e il ridimensionamento dei nodi, semplificando gli aspetti operativi. Parallelamente, sono supportati gli argomenti `kubelet` personalizzati, che offrono la possibilità di creare policy avanzate per la gestione della CPU e della memoria. Inoltre, migliorano la sicurezza tramite i ruoli AWS Identity and Access Management (IAM) per gli account di servizio, riducendo al contempo la necessità di autorizzazioni separate per cluster.

Nodi autogestiti

I nodi autogestiti offrono il controllo completo sulle EC2 istanze Amazon all'interno di un cluster Amazon EKS. L'utente è responsabile della gestione, del dimensionamento e della manutenzione dei nodi, disponendo del controllo totale sull'infrastruttura sottostante. Questa opzione è consigliata per gli utenti che necessitano di controllo e personalizzazione granulari dei propri nodi e sono pronti a investire tempo nella gestione e nella manutenzione della propria infrastruttura.

Amazon EKS Hybrid Nodes

Con Amazon EKS Hybrid Nodes è possibile utilizzare l'infrastruttura on-premises ed edge come nodi nei cluster Amazon EKS. Amazon EKS Hybrid Nodes unifica la gestione di Kubernetes tra gli ambienti e affida la gestione del piano di controllo Kubernetes alle applicazioni locali e periferiche. AWS

Funzionalità EKS

Amazon EKS offre funzionalità di cluster completamente gestite, installando e gestendo Kubernetes APIs (con Kubernetes Custom Resource Definitions) nel cluster e gestendo controller e altri componenti in AWS un'infrastruttura di proprietà, separata dal cluster. EKS fornisce patch, scalabilità e monitoraggio automatizzati di queste funzionalità, gestendone completamente il ciclo di vita per ridurre il carico di lavoro dei servizi in cluster per l'orchestrazione dei carichi di lavoro, la gestione delle risorse e altro ancora. AWS

EKS offre i seguenti tipi di funzionalità:

AWS Controller per Kubernetes (ACK)

AWS Controllers for Kubernetes (ACK) ti consente di gestire le AWS risorse utilizzando Kubernetes APIs, consentendoti di definire bucket S3, database RDS, ruoli IAM e altre risorse come risorse personalizzate Kubernetes. AWS Puoi gestire AWS le risorse insieme ai carichi di lavoro Kubernetes utilizzando gli stessi strumenti e flussi di lavoro, con supporto per oltre 50 AWS servizi tra cui S3, RDS, DynamoDB e Lambda.

CD Argo

Argo CD GitOps implementa una distribuzione continua basata sui carichi di lavoro delle applicazioni, sulle AWS risorse e sulla configurazione del cluster, utilizzando i repository Git come fonte di verità. Argo CD sincronizza automaticamente i cluster con i repository Git e rileva eventuali deviazioni, riconciliandosi continuamente per garantire che le applicazioni e le risorse distribuite corrispondano allo stato desiderato nel controllo delle versioni. È possibile utilizzare Argo CD per gestire le applicazioni su un determinato cluster o distribuire e gestire applicazioni su più cluster da una singola risorsa Argo CD, con la distribuzione automatica dai repository Git ogni volta che vengono apportate modifiche.

kro (Kube Resource Orchestrator)

kro (Kube Resource Orchestrator) ti consente di creare Kubernetes personalizzati APIs che compongono più risorse in astrazioni di livello superiore, permettendo ai team della piattaforma di definire modelli riutilizzabili per combinazioni di risorse comuni. Ciò consente ai team della piattaforma di fornire funzionalità self-service con barriere appropriate, permettendo agli sviluppatori di fornire infrastrutture complesse utilizzando infrastrutture semplici e appositamente progettate, mantenendo al contempo gli standard organizzativi e le migliori pratiche. APIs

Concetti di Kubernetes

Amazon Elastic Kubernetes Service (Amazon EKS) è un servizio gestito da AWS basato sul progetto open source Kubernetes. Sebbene ci siano cose che devi sapere su come il servizio Amazon EKS si integra con il cloud AWS (in particolare quando crei per la prima volta un cluster Amazon EKS), una volta che è attivo e funzionante, puoi utilizzare il cluster Amazon EKS più o meno allo stesso modo di qualsiasi altro cluster Kubernetes. Quindi, per iniziare a gestire i cluster Kubernetes e distribuire carichi di lavoro, è necessaria almeno una conoscenza di base dei concetti di Kubernetes.

Questa pagina divide i concetti di Kubernetes in tre sezioni: the section called "Perché Kubernetes?", the section called "Cluster" e the section called "Carichi di lavoro". La prima sezione descrive il valore dell'esecuzione di un servizio Kubernetes, in particolare come servizio gestito quale Amazon EKS. La sezione Carichi di lavoro illustra come le applicazioni Kubernetes vengono create, archiviate, eseguite e gestite. La sezione Cluster illustra i diversi componenti che formano i cluster Kubernetes e quali sono le tue responsabilità per la creazione e la manutenzione di questi cluster.

Argomenti

- Perché Kubernetes?
- Cluster
- Carichi di lavoro
- Passaggi successivi

Durante la lettura di questo contenuto, i link ti condurranno a ulteriori descrizioni dei concetti di Kubernetes nella documentazione di Amazon EKS e Kubernetes, nel caso in cui desideri approfondire uno degli argomenti trattati qui. Per dettagli su come Amazon EKS implementa il piano di controllo e le funzionalità di calcolo di Kubernetes, consulta the section called "Architecture".

Perché Kubernetes?

Kubernetes è stato progettato per migliorare la disponibilità e la scalabilità durante l'esecuzione di applicazioni containerizzate mission-critical e di qualità produttiva. Anziché limitarsi a eseguire Kubernetes su una singola macchina (sebbene ciò sia possibile), Kubernetes raggiunge questi obiettivi consentendoti di eseguire applicazioni su set di computer che possono espandersi o contrarsi per soddisfare la domanda. Kubernetes include funzionalità che rendono più semplice:

- Distribuire applicazioni su più macchine (utilizzando container distribuiti nei pod)

- Monitorare lo stato dei container e riavviare quelli che hanno subito un errore
- Aumentare e ridurre i container in base al carico
- Aggiornare i container con le nuove versioni
- Allocare le risorse tra i container
- Bilanciare il traffico tra le macchine

L'automazione di questi tipi di attività complesse da parte di Kubernetes consente agli sviluppatori di applicazioni di concentrarsi sulla creazione e sul miglioramento dei carichi di lavoro delle applicazioni, anziché preoccuparsi dell'infrastruttura. In genere, lo sviluppatore crea file di configurazione, formattati come file YAML, che descrivono lo stato desiderato dell'applicazione. Ciò potrebbe includere i container da eseguire, i limiti delle risorse, il numero di repliche dei pod, l'allocazione della CPU/memoria, le regole di affinità e molto altro.

Attributi di Kubernetes

Per raggiungere gli obiettivi, Kubernetes ha i seguenti attributi:

- Containerizzato: Kubernetes è uno strumento di orchestrazione dei container. Per utilizzare Kubernetes, devi prima avere le tue applicazioni containerizzate. A seconda del tipo di applicazione, potrebbe trattarsi di un insieme di microservizi, di processi in batch o in altre forme. Quindi, le tue applicazioni possono trarre vantaggio da un flusso di lavoro di Kubernetes che comprende un enorme ecosistema di strumenti, in cui i container possono essere archiviati come immagini in un registro di container, distribuiti in un cluster Kubernetes ed eseguiti su un nodo disponibile. Puoi creare e testare singoli container sul tuo computer locale con Docker o un altro runtime di container prima di distribuirli nel cluster Kubernetes.
- Scalabile: se la domanda per le tue applicazioni supera la capacità delle istanze in esecuzione delle stesse, Kubernetes è in grado di aumentare verticalmente. Se necessario, Kubernetes è in grado di stabilire se le applicazioni richiedono più CPU o memoria e rispondere o espandendo automaticamente la capacità disponibile o utilizzando una maggiore capacità esistente. La scalabilità può essere eseguita a livello di pod, se è disponibile una quantità di calcolo sufficiente per eseguire solo più istanze dell'applicazione (scalabilità automatica del pod in orizzontale), o a livello di nodo, se è necessario attivare più nodi per gestire l'aumento della capacità (Cluster Autoscaler o Karpenter). Poiché la capacità non è più necessaria, questi servizi possono eliminare i pod non necessari e chiudere i nodi non necessari.
- Disponibile: se un'applicazione o un nodo non sono integri o non sono disponibili, Kubernetes può spostare i carichi di lavoro in esecuzione su un altro nodo disponibile. Puoi forzare il problema

semplicemente eliminando un'istanza in esecuzione di un carico di lavoro o di un nodo che esegue i tuoi carichi di lavoro. In conclusione, i carichi di lavoro possono essere trasferiti in altre posizioni se non possono più essere eseguiti dove si trovano.

- Dichiarativo: Kubernetes utilizza la riconciliazione attiva per verificare costantemente che lo stato dichiarato per il cluster corrisponda allo stato effettivo. Applicando oggetti Kubernetes a un cluster, in genere tramite file di configurazione in formato YAML, puoi, ad esempio, chiedere di avviare i carichi di lavoro che desideri eseguire sul cluster. Successivamente, puoi modificare le configurazioni per eseguire operazioni quali utilizzare una versione successiva di un container o allocare più memoria. Kubernetes farà ciò che deve fare per stabilire lo stato desiderato. Ciò può includere l'attivazione o la disattivazione dei nodi, l'arresto e il riavvio dei carichi di lavoro o l'estrazione di container aggiornati.
- Componibile: poiché un'applicazione è in genere composta da più componenti, è necessario essere in grado di gestire insieme un insieme di questi componenti (spesso rappresentati da più container). Sebbene Docker Compose offra un modo per farlo direttamente con Docker, il comando Kompose di Kubernetes può aiutarti a farlo con Kubernetes. Consultare Translate a Docker Compose File to Kubernetes Resources per un esempio di come eseguire questa operazione.
- Estensibile: a differenza del software proprietario, il progetto open source Kubernetes è progettato per consentirti di estendere Kubernetes in qualsiasi modo desideri per soddisfare le tue esigenze. Le API e i file di configurazione sono aperti a modifiche dirette. I terzi sono incoraggiati a scrivere i propri comandi per estendere sia l'infrastruttura che le funzionalità di Kubernetes per gli utenti finali. I webhook consentono di configurare le regole del cluster per applicare le policy e adattarsi alle condizioni mutevoli. Per altre idee su come estendere i cluster Kubernetes, consultare Extending Kubernetes.
- Portatile: molte organizzazioni hanno standardizzato le proprie operazioni su Kubernetes perché consente loro di gestire tutte le esigenze applicative nello stesso modo. Gli sviluppatori possono utilizzare le stesse pipeline per creare e archiviare applicazioni containerizzate. Queste applicazioni possono quindi essere distribuite su cluster Kubernetes che vengono eseguiti on-premises, nel cloud, sui terminali dei punti vendita nei ristoranti o su dispositivi IOT distribuiti tra i siti remoti dell'azienda. La sua natura open source consente alle persone di sviluppare queste speciali distribuzioni Kubernetes, insieme agli strumenti necessari per gestirle.

Gestione di Kubernetes

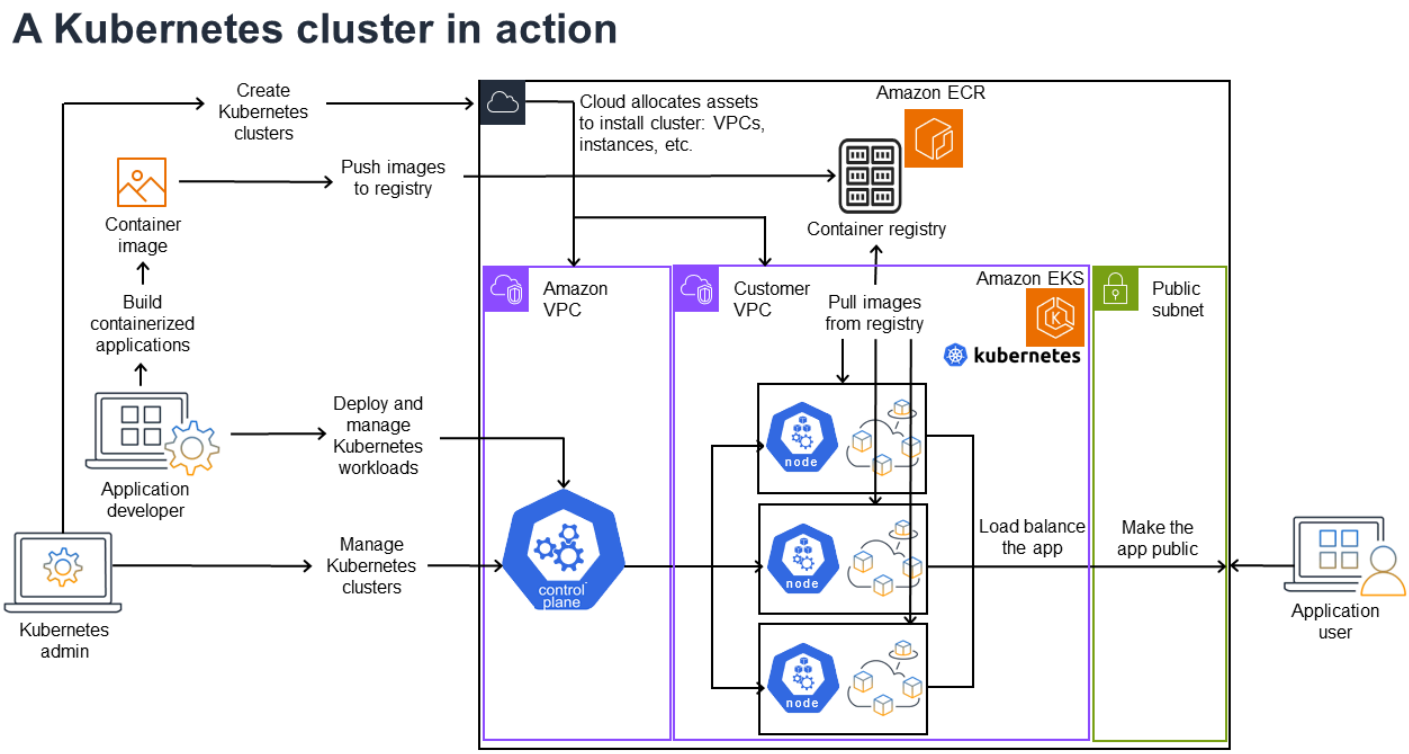
Il codice sorgente di Kubernetes è disponibile gratuitamente, quindi è possibile installare e gestire Kubernetes da soli con le proprie apparecchiature. Tuttavia, la gestione automatica di Kubernetes richiede una profonda esperienza operativa e richiede tempo e impegno per la manutenzione. Per

questi motivi, la maggior parte delle persone che implementano carichi di lavoro di produzione sceglie un provider cloud (come Amazon EKS) o un provider on-premises (come Amazon EKS Anywhere) con la propria distribuzione Kubernetes testata e il supporto di esperti Kubernetes. Ciò consente di alleggerire gran parte del carico di lavoro indifferenziato necessario per la manutenzione dei cluster, tra cui:

- Hardware: se non disponi di hardware disponibile per eseguire Kubernetes in base alle tue esigenze, un provider di servizi cloud come AWS Amazon EKS può farti risparmiare sui costi iniziali. Con Amazon EKS, ciò significa che puoi utilizzare le migliori risorse cloud offerte da AWS, tra cui istanze di computer (Amazon Elastic Compute Cloud), il tuo ambiente privato (Amazon VPC), la gestione centralizzata di identità e permessi (IAM) e lo storage (Amazon EBS). AWS gestisce i computer, le reti, i data center e tutti gli altri componenti fisici necessari per eseguire Kubernetes. Allo stesso modo, non è necessario pianificare il data center per gestire la capacità massima nei giorni con una maggiore richiesta. Per Amazon EKS Anywhere o altri cluster Kubernetes on-premises, sei responsabile della gestione dell'infrastruttura utilizzata nelle implementazioni Kubernetes, ma puoi comunque contare sull'aiuto di AWS per mantenere Kubernetes aggiornato.
- Gestione del piano di controllo: Amazon EKS gestisce la sicurezza e la disponibilità del piano di controllo Kubernetes ospitato da AWS, responsabile della pianificazione dei container, della gestione della disponibilità delle applicazioni e di altre attività chiave, così che tu possa concentrarti sui carichi di lavoro delle applicazioni. In caso di guasto del cluster, AWS dovrebbe avere a disposizione i mezzi per ripristinarlo in uno stato di esecuzione. Per Amazon EKS Anywhere, il piano di controllo verrebbe gestito dall'utente stesso.
- Upgrade testati: quando aggiorni i tuoi cluster, puoi fare affidamento su Amazon EKS o Amazon EKS Anywhere per fornire versioni testate delle distribuzioni Kubernetes.
- Componenti aggiuntivi: esistono centinaia di progetti creati per estendere e utilizzare Kubernetes che puoi aggiungere all'infrastruttura del cluster o utilizzare per facilitare l'esecuzione dei tuoi carichi di lavoro. Invece di creare e gestire per proprio conto questi componenti aggiuntivi, AWS fornisce the section called "Componenti aggiuntivi Amazon EKS" che puoi utilizzare coi cluster. Amazon EKS Anywhere fornisce pacchetti curati che includono build di molti progetti open source popolari. Quindi non è necessario creare il software o gestire patch di sicurezza, correzioni di bug o aggiornamenti critici per proprio conto. Allo stesso modo, se le impostazioni predefinite soddisfano le tue esigenze, è normale che sia necessaria una configurazione minima di tali componenti aggiuntivi. Consultare the section called "Estendere i cluster" per i dettagli sull'estensione del cluster con componenti aggiuntivi.

Kubernetes in azione

Il diagramma seguente mostra le attività chiave che potresti svolgere in qualità di amministratore o sviluppatore di applicazioni Kubernetes per creare e utilizzare un cluster Kubernetes. Nel processo, illustra come i componenti di Kubernetes interagiscono tra loro, utilizzando il cloud AWS come esempio del provider cloud sottostante.



Un amministratore Kubernetes crea il cluster Kubernetes utilizzando uno strumento specifico per il tipo di provider su cui verrà creato il cluster. Questo esempio utilizza il cloud AWS come provider, che offre il servizio Kubernetes gestito chiamato Amazon EKS. Il servizio gestito alloca automaticamente le risorse necessarie per creare il cluster, inclusa la creazione di due nuovi cloud privati virtuali (Amazon VPC) per il cluster, la configurazione della rete e la mappatura delle autorizzazioni Kubernetes direttamente nei nuovi VPC per la gestione delle risorse cloud. Il servizio gestito verifica inoltre che i servizi del piano di controllo abbiano luoghi in cui essere eseguiti e alloca zero o più istanze Amazon EC2 come nodi Kubernetes per l'esecuzione di carichi di lavoro. AWS gestisce un Amazon VPC per il piano di controllo, mentre l'altro Amazon VPC contiene i nodi cliente che eseguono i carichi di lavoro.

Molte delle attività future dell'amministratore di Kubernetes vengono eseguite utilizzando strumenti Kubernetes come `kubectl`. Questo strumento invia le richieste di servizi direttamente al piano di

controllo del cluster. I modi in cui le query e le modifiche vengono apportate al cluster sono quindi molto simili a quelli in cui verrebbero eseguite in qualsiasi cluster Kubernetes.

Uno sviluppatore di applicazioni che desidera distribuire carichi di lavoro in questo cluster può eseguire diverse attività. Lo sviluppatore deve creare l'applicazione in una o più immagini di container, quindi inviare tali immagini a un registro di contenitori accessibile al cluster Kubernetes. A tale scopo, AWS offre Amazon Elastic Container Registry (Amazon ECR).

Per eseguire l'applicazione, lo sviluppatore può creare file di configurazione in formato YAML che indicano al cluster come eseguire l'applicazione, compresi quali container estrarre dal registro e come avvolgerli nei pod. Il piano di controllo (scheduler) pianifica i container su uno o più nodi e il runtime del container su ciascun nodo recupera ed esegue effettivamente i container necessari. Lo sviluppatore può anche configurare un Application Load Balancer per bilanciare il traffico verso i container disponibili in esecuzione su ciascun nodo ed esporre l'applicazione al mondo esterno in modo che sia disponibile su una rete pubblica. Fatto ciò, qualcuno che desidera utilizzare l'applicazione può connettersi al suo endpoint per accedervi.

Le sezioni seguenti illustrano i dettagli di ciascuna di queste funzionalità, dal punto di vista dei cluster e dei carichi di lavoro Kubernetes.

Cluster

Se il tuo compito è avviare e gestire i cluster Kubernetes, devi sapere come vengono creati, migliorati, gestiti ed eliminati. Devi anche sapere quali sono i componenti che formano un cluster e cosa fare per mantenerli.

Gli strumenti per la gestione dei cluster si occupano della sovrapposizione tra i servizi Kubernetes e il provider hardware sottostante. Per questo motivo, l'automazione di queste attività tende ad essere eseguita dal provider Kubernetes (come Amazon EKS o Amazon EKS Anywhere) utilizzando strumenti specifici per il provider. Ad esempio, per avviare un cluster Amazon EKS si può utilizzare `eksctl create cluster`, mentre per Amazon EKS Anywhere è possibile utilizzare `eksctl anywhere create cluster`. Nota che, sebbene questi comandi creino un cluster Kubernetes, sono specifici del provider e non fanno parte del progetto Kubernetes stesso.

Strumenti per la creazione e la gestione dei cluster

Il progetto Kubernetes offre strumenti per creare manualmente un cluster Kubernetes. Quindi, se vuoi installare Kubernetes su una singola macchina o eseguire il piano di controllo su una macchina e aggiungere nodi manualmente, puoi utilizzare strumenti CLI come kind, minikube o kubeadm elencati in Install Tools di Kubernetes. Per semplificare e automatizzare l'intero ciclo di vita della creazione

e della gestione dei cluster, è molto più semplice utilizzare gli strumenti supportati da un provider Kubernetes affermato, come Amazon EKS o Amazon EKS Anywhere.

Nel cloud AWS, puoi creare cluster Amazon EKS utilizzando strumenti CLI, come eksctl, o strumenti più dichiarativi, come Terraform (consultare Amazon EKS Blueprints for Terraform). Puoi creare un cluster da Console di gestione AWS. Consultare Amazon EKS features per un elenco di ciò che si ottiene con Amazon EKS. Le responsabilità di Kubernetes che Amazon EKS si assume per te includono:

- Piano di controllo gestito: AWS assicura che il cluster Amazon EKS sia disponibile e scalabile perché gestisce il piano di controllo per te e lo rende disponibile in tutte le zone di disponibilità AWS.
- Gestione dei nodi: invece di aggiungere i nodi manualmente, puoi fare in modo che Amazon EKS crei i nodi in automatico a seconda delle necessità, utilizzando Gruppi di nodi gestiti (vedere the section called "Gruppi di nodi gestiti") o Karpenter. I gruppi di nodi gestiti presentano integrazioni con Cluster Autoscaling di Kubernetes. Utilizzando gli strumenti di gestione dei nodi, è possibile trarre vantaggio dai risparmi sui costi, come le istanze spot e il consolidamento e la disponibilità dei nodi, utilizzando le funzionalità di pianificazione per impostare la modalità di distribuzione dei carichi di lavoro e la selezione dei nodi.
- Rete di cluster: utilizzando i modelli CloudFormation, `eksctl` configura la rete tra i componenti del piano di controllo e del piano dati (nodo) nel cluster Kubernetes. Inoltre, configura gli endpoint attraverso i quali possono avvenire le comunicazioni interne ed esterne. Per i dettagli, consultare De-mystifying cluster networking for Amazon EKS worker nodes. Le comunicazioni tra i pod in Amazon EKS vengono effettuate utilizzando Pod Identity di Amazon EKS (consultare the section called "Pod Identity"), che fornisce un mezzo per consentire ai pod di attingere ai metodi cloud AWS per la gestione delle credenziali e delle autorizzazioni.
- Componenti aggiuntivi: con Amazon EKS non c'è bisogno di creare e aggiungere componenti software comunemente usati per supportare i cluster Kubernetes. Ad esempio, quando crei un cluster Amazon EKS da Console di gestione AWS, questo aggiunge automaticamente i componenti aggiuntivi Amazon EKS kube-proxy (the section called "kube-proxy"), il plug-in Amazon VPC CNI per Kubernetes (the section called "CNI di Amazon VPC") e CoreDNS (the section called "CoreDNS"). Consultare the section called "Componenti aggiuntivi Amazon EKS" per ulteriori elementi aggiuntivi, incluso un elenco di quali sono disponibili.

Per eseguire i cluster su computer e reti on-premises, Amazon offre Amazon EKS Anywhere. Invece del cloud AWS come provider, puoi scegliere di eseguire Amazon EKS Anywhere su piattaforme

VMware vSphere, bare metal (provider Tinkerbell), Snow, CloudStack o Nutanix utilizzando le tue apparecchiature.

Amazon EKS Anywhere si basa sullo stesso software Amazon EKS Distro utilizzato da Amazon EKS. Tuttavia, Amazon EKS Anywhere si affida a diverse implementazioni dell'interfaccia API del cluster Kubernetes (CAPI) per gestire l'intero ciclo di vita delle macchine in un cluster Amazon EKS Anywhere (come CAPV per vSphere e CAPC per CloudStack). Poiché l'intero cluster è in esecuzione sulle tue apparecchiature, ti assumi la responsabilità aggiuntiva della gestione del piano di controllo e del backup dei relativi dati (consulta `etcd` più avanti in questo documento).

Componenti del cluster

I componenti del cluster Kubernetes sono suddivisi in due aree principali: piano di controllo e nodi worker. I componenti del piano di controllo gestiscono il cluster e forniscono l'accesso alle relative API. I nodi worker (a volte denominati semplicemente nodi) forniscono i luoghi in cui vengono eseguiti i carichi di lavoro effettivi. I componenti del nodo sono costituiti da servizi eseguiti su ciascun nodo per comunicare con il piano di controllo ed eseguire i container. Il set di nodi worker per il cluster è denominato Piano dati.

Piano di controllo

Il piano di controllo è costituito da un insieme di servizi che gestiscono il cluster. Questi servizi possono essere eseguiti tutti su un singolo computer o possono essere distribuiti su più computer. Internamente, queste sono denominate Istanze del piano di controllo (CPI). Il modo in cui vengono eseguite le CPI dipende dalla dimensione del cluster e dai requisiti di elevata disponibilità. Con l'aumento della domanda nel cluster, un servizio del piano di controllo può scalare per fornire più istanze di quel servizio, utilizzando le richieste come bilanciatore del carico tra le istanze.

Le attività eseguite dai componenti del piano di controllo Kubernetes includono:

- Comunicazione con i componenti del cluster (server API): il server API (kube-apiserver) espone l'API Kubernetes in modo che le richieste al cluster possano essere effettuate sia dall'interno che dall'esterno del cluster. In altre parole, le richieste di aggiunta o modifica degli oggetti di un cluster (pod, servizi, nodi e così via) possono provenire da comandi esterni, come le richieste di esecuzione da parte di `kubectl` di eseguire un pod. Allo stesso modo, è possibile effettuare richieste dal server API ai componenti all'interno del cluster, ad esempio una richiesta al servizio `kubelet` per lo stato di un pod.
- Archivia dati sul cluster (archivio di valori delle chiavi **`etcd`**): il servizio `etcd` svolge il ruolo fondamentale di tenere traccia dello stato attuale del cluster. Se il servizio `etcd` diventasse

inaccessibile, non sarebbe possibile aggiornare lo stato del cluster o eseguire query su di esso, anche se i carichi di lavoro continuerebbero a funzionare per un po'. Per questo motivo, i cluster critici in genere dispongono di più istanze di bilanciatore del carico del servizio `etcd` in esecuzione contemporaneamente ed effettuano backup periodici dell'archivio di valori delle chiavi `etcd` in caso di perdita o danneggiamento dei dati. Tieni presente che, in Amazon EKS, tutto questo viene gestito automaticamente per impostazione predefinita. Amazon EKS Anywhere fornisce istruzioni per etcd backup and restore. Consulta etcd Data Model per scoprire come `etcd` gestisce i dati.

- Programma i pod ai nodi (Scheduler): le richieste di avvio o arresto di un pod in Kubernetes vengono indirizzate a Kubernetes Scheduler (kube-scheduler). Poiché un cluster può avere più nodi in grado di eseguire il pod, spetta allo Scheduler scegliere su quale nodo (o nodi, nel caso delle repliche) eseguire il pod. Se la capacità disponibile non è sufficiente per eseguire il pod richiesto su un nodo esistente, la richiesta avrà esito negativo, a meno che non siano state prese altre disposizioni. Tali disposizioni potrebbero includere l'abilitazione di servizi come Gruppi di nodi gestiti (the section called "Gruppi di nodi gestiti") o Karpenter in grado di avviare automaticamente nuovi nodi per gestire i carichi di lavoro.
- Mantieni i componenti nello stato desiderato (Gestore di controller): il Gestore controller di Kubernetes viene eseguito come processo daemon (kube-controller-manager) per monitorare lo stato del cluster e apportare modifiche al cluster per ristabilire gli stati previsti. In particolare, esistono diversi controller che controllano diversi oggetti Kubernetes, tra cui `statefulset-controller`, `endpoint-controller`, `cronjob-controller`, `node-controller` e altri.
- Gestisci le risorse cloud (Gestore di controller cloud): le interazioni tra Kubernetes e il provider cloud che esegue le richieste per le risorse del data center sottostanti vengono gestite dal Gestore di controller cloud (cloud-controller-manager). I controller gestiti da Gestore di controller cloud possono includere un controller di percorso (per configurare i percorsi di rete cloud), un controller di servizio (per l'utilizzo dei servizi di bilanciatore del carico nel cloud) e un controller del ciclo di vita dei nodi (per mantenere i nodi sincronizzati con Kubernetes per tutto il loro ciclo di vita).

Nodi di lavoro (piano dati)

Per un cluster Kubernetes a nodo singolo, i carichi di lavoro vengono eseguiti sulla stessa macchina del piano di controllo. Tuttavia, una configurazione più standard prevede uno o più sistemi informatici separati (nodi) dedicati all'esecuzione dei carichi di lavoro Kubernetes.

Quando crei per la prima volta un cluster Kubernetes, alcuni strumenti per la creazione di cluster ti consentono di configurare un certo numero di nodi da aggiungere al cluster (identificando i sistemi

informatici esistenti o chiedendo al provider di crearne di nuovi). Prima di aggiungere qualsiasi carico di lavoro a tali sistemi, vengono aggiunti servizi a ciascun nodo per implementare queste funzionalità:

- Gestisci ogni nodo (**`kubelet`**): il server API comunica con il servizio kubelet in esecuzione su ciascun nodo per assicurarsi che il nodo sia registrato correttamente e che i pod richiesti dallo Scheduler siano in esecuzione. Il kubelet può leggere i manifest dei pod e configurare volumi di archiviazione o altre funzionalità necessarie ai pod sul sistema locale. Può anche verificare lo stato dei container in esecuzione locale.
- Esegui i container su un nodo (runtime del container): il Runtime del container su ciascun nodo gestisce i container richiesti per ogni pod assegnato al nodo. Ciò significa che può estrarre le immagini del container dal registro appropriato, eseguire il container, interromperlo e rispondere alle query sul container. Il runtime predefinito del container è containerd. A partire da Kubernetes 1.24, la speciale integrazione di Docker (`dockershim`) che poteva essere utilizzata come runtime del container è stata eliminata da Kubernetes. Sebbene sia ancora possibile utilizzare Docker per testare ed eseguire container sul sistema locale, per utilizzare Docker con Kubernetes ora è necessario installare il motore Docker su ogni nodo per utilizzarlo con Kubernetes.
- Gestisci la rete tra container (**`kube-proxy`**): per supportare la comunicazione tra i pod, Kubernetes utilizza una funzionalità denominata Servizio per configurare le reti pod che tengono traccia degli indirizzi IP e delle porte associate a tali pod. Il servizio kube-proxy viene eseguito su ogni nodo per consentire la comunicazione tra i pod.

Estendere i cluster

Esistono alcuni servizi che puoi aggiungere a Kubernetes per supportare il cluster, ma non vengono eseguiti nel piano di controllo. Questi servizi spesso vengono eseguiti direttamente sui nodi del namespace kube-system o nel relativo namespace (come spesso accade con fornitori di servizi di terzi). Un esempio comune è il servizio CoreDNS, che fornisce servizi DNS al cluster. Fai riferimento a Discovering builtin services per informazioni su come vedere quali servizi cluster sono in esecuzione in kube-system sul tuo cluster.

Esistono diversi tipi di componenti aggiuntivi che puoi considerare di aggiungere ai tuoi cluster. Per mantenere integri i cluster, puoi aggiungere funzionalità di osservabilità (consultare Monitoraggio dei cluster) che consentono di eseguire operazioni come la registrazione, il controllo e le metriche. Con queste informazioni, puoi risolvere i problemi che si verificano, spesso tramite le stesse interfacce di osservabilità. Esempi di questi tipi di servizi includono Amazon GuardDuty, CloudWatch (consultare the section called "Amazon CloudWatch"), AWSDistro for OpenTelemetry, il plug-in Amazon VPC CNI per Kubernetes (consultare the section called "CNI di Amazon VPC") e Grafana Kubernetes

Monitoring. Per lo storage (consulta Archiviazione di dati per app), i componenti aggiuntivi di Amazon EKS includono Driver CSI per Amazon Elastic Block Store (consulta the section called "Amazon EBS"), Driver CSI per Amazon Elastic File System (consulta the section called "Amazon EFS") e diversi componenti aggiuntivi di storage di terzi come Amazon FSx per NetApp ONTAP (driver CSI the section called "Amazon FSx per NetApp ONTAP").

Per un elenco più completo dei componenti aggiuntivi di Amazon EKS, consultare the section called "Componenti aggiuntivi Amazon EKS".

Carichi di lavoro

Kubernetes definisce un carico di lavoro come "un'applicazione in esecuzione su Kubernetes". Tale applicazione può essere costituita da un set di micro servizi eseguiti come container in pod oppure può essere eseguita come processo batch o un altro tipo di applicazioni. Il compito di Kubernetes è assicurarsi che le richieste effettuate per la configurazione o la distribuzione di tali oggetti vengano eseguite. Chi distribuisce le applicazioni dovrebbe imparare come vengono creati i container, come vengono definiti i pod e quali metodi è possibile utilizzare per distribuirli.

Container

L'elemento più basilare del carico di lavoro di un'applicazione che distribuisci e gestisci in Kubernetes è un pod . Un pod rappresenta un modo per contenere i componenti di un'applicazione e per definire le specifiche che descrivono gli attributi del pod. Confrontalo con qualcosa come un pacchetto RPM o Deb, che raggruppa il software per un sistema Linux, ma non funziona come un'entità.

Poiché il pod è l'unità implementabile più piccola, in genere contiene un singolo container. Tuttavia, in un pod possono essere presenti più container nei casi in cui questi siano strettamente collegati. Ad esempio, un container di server Web potrebbe essere impacchettato in un pod con un tipo di container secondario che può fornire la registrazione, il monitoraggio o altri servizi strettamente legati al container del server Web. In questo caso, la presenza nello stesso pod garantisce che per ogni istanza in esecuzione del pod, entrambi i container vengano sempre eseguiti sullo stesso nodo. Allo stesso modo, tutti i container in un pod condividono lo stesso ambiente, con i container in un pod che funzionano come se si trovassero nello stesso host isolato. L'effetto è che i container condividono un unico indirizzo IP che fornisce l'accesso al pod e possono comunicare tra loro come se fossero in esecuzione sul proprio host locale.

Le specifiche del pod (PodSpec) definiscono lo stato desiderato del pod. Puoi implementare uno o più pod utilizzando le risorse del carico di lavoro per gestire i modelli di pod. Le risorse del carico di lavoro includono implementazioni (per gestire più repliche di pod), StatefulSets (per implementare

pod che devono essere unici, come i pod del database) e DaemonSet (in cui un pod deve essere eseguito continuamente su ogni nodo). Ne parleremo più avanti.

Sebbene un pod sia l'unità più piccola implementabile, un container è l'unità più piccola da costruire e gestire.

Costruire container

Il pod è in realtà solo una struttura attorno a uno o più container, dove ognuno di essi contiene il file system, gli eseguibili, i file di configurazione, le librerie e altri componenti per eseguire effettivamente l'applicazione. Poiché i container sono diventati popolari grazie a una società chiamata Docker Inc., alcune persone si riferiscono ai container come container Docker. Tuttavia, da allora l'Open Container Initiative ha definito i runtime, le immagini e i metodi di distribuzione dei container per il settore. A ciò si aggiunge il fatto che i container sono stati creati partendo da molte funzionalità Linux esistenti, per cui spesso le persone si riferiscono ai contenitori come container OCI, container Linux o semplicemente container.

Quando si crea un container, in genere si inizia con un Dockerfile (chiamato letteralmente così). All'interno di quel Dockerfile, si identifica:

- Un'immagine di base: un'immagine container di base è un container che viene in genere creato da una versione minima del file system di un sistema operativo (come Red Hat Enterprise Linux o Ubuntu) o da un sistema minimale migliorato per fornire software al fine di eseguire tipi specifici di applicazioni (come app nodejs o python).
- Software applicativo: puoi aggiungere il tuo software applicativo al container più o meno nello stesso modo in cui lo aggiungeresti a un sistema Linux. Ad esempio, nel tuo Dockerfile puoi eseguire `npm` e `yarn` per installare un'applicazione Java o `yum` e `dnf` per installare pacchetti RPM. In altre parole, utilizzando un comando RUN in un Dockerfile, è possibile eseguire qualsiasi comando disponibile nel file system dell'immagine di base per installare software o configurare software all'interno dell'immagine del container risultante.
- Istruzioni: il Dockerfile reference descrive le istruzioni che è possibile aggiungere a un Dockerfile durante la configurazione. Queste includono le istruzioni utilizzate per creare ciò che è contenuto nel container stesso (file `ADD` o `COPY` dal sistema locale), identificare i comandi da eseguire quando il container viene eseguito (`CMD` o `ENTRYPOINT`) e connettere il container al sistema su cui viene eseguito (identificando `USER` eseguire come, un `VOLUME` locale da montare o le porte su `EXPOSE`).

Sebbene il comando e il servizio `docker` siano stati tradizionalmente utilizzati per creare container (`docker build`), altri strumenti disponibili per creare immagini di contenitori includono podman e

nerdctl. Consultare Creare immagini di container migliori o Panoramica di Docker Build per saperne di più sulla creazione di container.

Archiviazione dei container

Dopo aver creato l'immagine del container, puoi archiviarla in un registro di distribuzione dei container sulla tua workstation o in un registro pubblico dei container. L'esecuzione di un registro di container privato sulla workstation consente di archiviare le immagini dei container in locale, rendendole immediatamente disponibili.

Per archiviare le immagini dei container in modo più pubblico, puoi inviarle a un registro di container pubblico. I registri pubblici dei container forniscono una posizione centrale per l'archiviazione e la distribuzione delle immagini dei container. Esempi di registri di container pubblici includono Amazon Elastic Container Registry, il registro Red Hat Quay e il registro Docker Hub.

Quando esegui carichi di lavoro containerizzati su Amazon Elastic Kubernetes Service (Amazon EKS), consigliamo di estrarre copie delle immagini ufficiali Docker archiviate in Amazon Elastic Container Registry. Amazon ECR archivia queste immagini dal 2021. Puoi cercare le immagini dei container più comuni nella Galleria pubblica di Amazon ECR e, in particolare, per le immagini di Docker Hub, puoi cercare nella Galleria docker di Amazon ECR.

Esecuzione dei container

Poiché i container sono creati in un formato standard, un container può essere eseguito su qualsiasi macchina in grado di eseguire un runtime di container (come Docker) e il cui contenuto corrisponda all'architettura della macchina locale (come `x86_64` o `arm`). Per testare un container o semplicemente eseguirlo sul desktop locale, puoi utilizzare i comandi `docker run` o `podman run` per avviare un container su host locale. Per Kubernetes, tuttavia, ogni nodo worker dispone di un runtime di container implementato e spetta a Kubernetes richiedere che un nodo esegua un container.

Una volta che un container è stato assegnato per l'esecuzione su un nodo, quest'ultimo verifica se la versione richiesta dell'immagine del container esiste già sul nodo. In caso contrario, Kubernetes indica al runtime del container di estrarlo dal registro di container appropriato, quindi di eseguirlo localmente. Tieni presente che un'immagine del container si riferisce al pacchetto software che viene spostato tra il laptop, il registro dei container e i nodi Kubernetes. Un container si riferisce a un'istanza in esecuzione di quell'immagine.

Pod

Una volta che i container sono pronti, l'utilizzo dei pod include la sua configurazione, implementazione e accessibilità.

Configurazione dei pod

Quando definisci un pod, gli assegni una serie di attributi. Questi attributi devono includere almeno il nome del pod e l'immagine del container da eseguire. Tuttavia, ci sono molte altre cose che si possono configurare anche con le definizioni dei pod (consulta la pagina PodSpec per i dettagli su cosa può essere inserito in un pod). Ciò include:

- Archiviazione: quando un container in esecuzione viene interrotto ed eliminato, l'archiviazione di dati in quel container scompare, a meno che non si configuri uno spazio di archiviazione più permanente. Kubernetes supporta molti tipi di storage diversi e li riassume sotto il termine Volumi. I tipi di storage includono CephFS, NFS, iSCSI e altri. È anche possibile utilizzare un dispositivo a blocchi locale dal computer locale. Con uno di questi tipi di archiviazione disponibili nel cluster, è possibile montare il volume di archiviazione su un punto di montaggio selezionato nel file system del container. Un volume persistente è quello che continua a esistere dopo l'eliminazione del pod, mentre un volume temporaneo viene eliminato quando il pod viene eliminato. Se l'amministratore del cluster ha creato diverse classi di archiviazione per il cluster, potrebbe essere possibile scegliere gli attributi dello storage da utilizzare, ad esempio se il volume viene eliminato o recuperato dopo l'uso, se si espanderà se è necessario più spazio e anche se soddisfa determinati requisiti di prestazioni.
- Segreti: rendendo disponibile Segreti ai container nelle specifiche del pod, puoi fornire le autorizzazioni necessarie a tali container per accedere a file system, database o altre risorse protette. Chiavi, password e token sono tra gli elementi che possono essere archiviati come segreti. L'uso dei segreti consente di non dover memorizzare queste informazioni nelle immagini dei container, ma è sufficiente rendere i segreti disponibili ai container in esecuzione. ConfigMaps sono simili a Segreti. `ConfigMap` tende a contenere informazioni meno critiche, come coppie chiave-valore per la configurazione di un servizio.
- Risorse del container: gli oggetti per un'ulteriore configurazione dei container possono assumere la forma di configurazione delle risorse. Per ogni container, puoi richiedere la quantità di memoria e CPU che può utilizzare, nonché porre limiti alla quantità totale di tali risorse che il container può utilizzare. Per degli esempi, consultare Resource Management for Pods and Containers.
- Interruzioni: i pod possono essere interrotti involontariamente (un nodo non funziona) o volontariamente (è necessario un aggiornamento). Configurando un budget per le interruzioni dei

pod, puoi esercitare un certo controllo sulla disponibilità dell'applicazione in caso di interruzioni. Per degli esempi, consultare Specifica di un budget di interruzioni per la tua applicazione.

- Namespace: Kubernetes offre diversi modi per isolare i componenti e i carichi di lavoro di Kubernetes gli uni dagli altri. L'esecuzione di tutti i pod per una particolare applicazione nello stesso namespace è un modo comune per proteggere e gestire insieme tali pod. Puoi creare i tuoi namespace da utilizzare o scegliere di non indicare un namespace (il che fa sì che Kubernetes utilizzi il namespace `default`). I componenti del piano di controllo di Kubernetes vengono in genere eseguiti nel namespace kube-system.

La configurazione appena descritta viene in genere raccolta in un file YAML da applicare al cluster Kubernetes. Per i cluster Kubernetes personali, è possibile archiviare semplicemente questi file YAML sul proprio sistema locale. Tuttavia, con cluster e carichi di lavoro più critici, GitOps è un metodo popolare per automatizzare lo storage e gli aggiornamenti sia del carico di lavoro che delle risorse dell'infrastruttura Kubernetes.

Gli oggetti utilizzati per raccogliere e implementare le informazioni sui pod sono definiti da uno dei seguenti metodi di implementazione.

Implementazione di pod

Il metodo da scegliere per implementare i pod dipende dal tipo di applicazione che intendi eseguire con gli stessi. Ecco alcune delle scelte:

- Applicazioni stateless: un'applicazione stateless non salva i dati della sessione di un client, quindi un'altra sessione non deve fare riferimento a ciò che è accaduto a una sessione precedente. In questo modo è più semplice sostituire i pod con altri nuovi se non funzionano bene o spostarli senza salvarne lo stato. Se si esegue un'applicazione stateless (come un server Web), è possibile utilizzare un'Implementazione per implementare pod e ReplicaSet. Un ReplicaSet definisce quante istanze di un pod si desidera eseguire contemporaneamente. Sebbene sia possibile eseguire direttamente un ReplicaSet, è comune eseguire le repliche direttamente all'interno di un'Implementazione, per definire quante repliche di un pod devono essere eseguite contemporaneamente.
- Applicazioni stateful: un'applicazione stateful è un'applicazione in cui l'identità del pod e l'ordine in cui vengono avviati i pod sono importanti. Queste applicazioni necessitano di uno storage persistente che sia stabile e che debba essere distribuito e scalato in modo coerente. Per distribuire un'applicazione stateful in Kubernetes, puoi utilizzare StatefulSets. Un esempio di applicazione che viene in genere eseguita come StatefulSet è un database. All'interno di uno

StatefulSet, è possibile definire le repliche, il pod e i relativi container, i volumi di archiviazione da montare e le posizioni nel contenitore in cui sono archiviati i dati. Consultare Run a Replicated Stateful Application per un esempio di database implementato come ReplicaSet.

- Applicazioni per nodo: a volte è necessario eseguire un'applicazione su ogni nodo del cluster Kubernetes. Ad esempio, il tuo data center potrebbe richiedere che ogni computer esegua un'applicazione di monitoraggio o un particolare servizio di accesso remoto. Per Kubernetes, puoi utilizzare un DaemonSet per garantire che l'applicazione selezionata venga eseguita su ogni nodo del cluster.
- Applicazioni eseguite fino al completamento: ci sono alcune applicazioni che desideri eseguire per completare un'attività particolare. Ciò potrebbe includerne una che esegue report mensili sullo stato o elimina i vecchi dati. Un oggetto Job può essere utilizzato per configurare un'applicazione per l'avvio e l'esecuzione, quindi uscire al termine dell'attività. Un oggetto CronJob consente di configurare un'applicazione in modo che venga eseguita a un'ora, un minuto, un giorno del mese, un mese o un giorno della settimana specifici, utilizzando una struttura definita dal formato crontab di Linux.

Rendere le applicazioni accessibili dalla rete

Poiché le applicazioni venivano spesso implementate come un insieme di micro servizi che si spostavano in luoghi diversi, Kubernetes aveva bisogno di un modo per consentire a tali micro servizi di trovarsi a vicenda. Inoltre, per consentire ad altri di accedere a un'applicazione esterna al cluster Kubernetes, quest'ultimo aveva bisogno di un modo per esporre tale applicazione su indirizzi e porte esterni. Queste funzionalità relative alla rete vengono eseguite rispettivamente con oggetti Servizi e In ingresso:

- Servizi: poiché un pod può spostarsi su nodi e indirizzi diversi, un altro pod che deve comunicare con il primo pod potrebbe avere difficoltà a localizzarlo. Per risolvere questo problema, Kubernetes ti consente di rappresentare un'applicazione come servizio. Con un servizio, puoi identificare un pod o un set di pod con un nome particolare, quindi indicare quale porta espone il servizio di quell'applicazione dal pod e quali porte potrebbe utilizzare un'altra applicazione per contattare quel servizio. Un altro pod all'interno di un cluster può semplicemente richiedere un servizio per nome e Kubernetes indirizzerà la richiesta alla porta appropriata per un'istanza del pod che esegue quel servizio.
- In ingresso: In ingresso è ciò che può rendere disponibili le applicazioni rappresentate dai servizi Kubernetes ai client esterni al cluster. Le funzionalità di base di In ingresso includono un sistema

di bilanciatore del carico (gestito da In ingresso), il controller In ingresso e regole per il routing delle richieste dal controller al servizio. Con Kubernetes, puoi scegliere tra diversi controller in ingresso.

Passaggi successivi

Comprendere i concetti di base di Kubernetes e il modo in cui si relazionano ad Amazon EKS ti aiuterà a navigare sia la Documentazione su Amazon Elastic Kubernetes Service che la Documentazione di Kubernetes per trovare le informazioni necessarie per gestire i cluster Amazon EKS e implementare i carichi di lavoro su tali cluster. Per iniziare a utilizzare Amazon EKS, scegli tra i seguenti:

- the section called "Crea cluster (eksctl)"
- the section called "Creazione di un cluster "
- the section called "Implementazione esemplificativa (Linux)"
- Gestione dei cluster

Implementazione di cluster Amazon EKS in ambienti cloud e on-premises

Comprensione delle opzioni di implementazione di Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) è un servizio Kubernetes completamente gestito che semplifica l'uso di Kubernetes nel cloud e negli ambienti on-premises.

Nel cloud, Amazon EKS automatizza la gestione dell'infrastruttura del cluster Kubernetes per il piano di controllo (control-plane) e i nodi di Kubernetes. Ciò è essenziale per la pianificazione dei container, la gestione della disponibilità delle applicazioni, il dimensionamento dinamica delle risorse, l'ottimizzazione dell'elaborazione, l'archiviazione dei dati del cluster e l'esecuzione di altre funzioni critiche. Con Amazon EKS, ottieni solide prestazioni, dimensionamento, affidabilità e disponibilità dell'infrastruttura AWS, oltre a integrazioni native con servizi AWS di rete, sicurezza, archiviazione e osservabilità.

Per semplificare l'esecuzione di Kubernetes nei tuoi ambienti on-premises, puoi utilizzare gli stessi cluster, funzionalità e strumenti di Amazon EKS in the section called "Nodi" o Amazon EKS Hybrid Nodes sulla tua infrastruttura, oppure puoi usare Amazon EKS Anywhere per ambienti isolati autonomi.

Amazon EKS nel cloud

Puoi utilizzare Amazon EKS con elaborazione in Regioni AWS, AWS Local Zones e Zone Wavelength AWS. Con Amazon EKS nel cloud, la sicurezza, il dimensionamento e la disponibilità del piano di controllo (control-plane) di Kubernetes sono completamente gestiti da AWS nella Regione AWS. Quando esegui applicazioni con elaborazione in Regioni AWS, ottieni l'intera gamma di funzionalità di AWS e Amazon EKS, incluso Amazon EKS Auto Mode, che automatizza completamente la gestione dell'infrastruttura del cluster Kubernetes per elaborazione, l'archiviazione e la rete su AWS con un solo clic. Quando esegui applicazioni con elaborazione in AWS Local Zones e Zone Wavelength AWS, puoi utilizzare i nodi autogestiti di Amazon EKS per connettere le istanze Amazon EC2 per il cluster computing e utilizzare gli altri servizi AWS disponibili in AWS Local Zones e Zone Wavelength AWS. Per ulteriori informazioni, consulta AWS Local Zones features and AWS Wavelength Zones features.

	Amazon EKS nelle Regioni AWS	Amazon EKS nelle Local Zone/Zone Wavelength
Gestione del piano di controllo (control-plane) di Kubernetes	Gestito da AWS	Gestito da AWS
Posizione del piano di controllo Kubernetes	Regioni AWS	Regioni AWS
Piano dati di Kubernetes	• Amazon EKS Auto Mode • Gruppi di nodi gestiti Amazon EKS • Nodi di Amazon EC2 autogestiti • AWS Fargate	• Gruppi di nodi gestiti di Amazon EKS (solo Local Zones) • Nodi di Amazon EC2 autogestiti
Posizione del piano dati Kubernetes	Regioni AWS	AWS Local Zones o Zone Wavelength

Amazon EKS nei tuoi data center o ambienti edge

Se devi eseguire applicazioni nei tuoi data center o ambienti edge, puoi utilizzare Amazon EKS su AWS Outposts o Amazon EKS Hybrid Nodes. Puoi utilizzare nodi autogestiti con istanze Amazon

EC2 su AWS Outposts per il cluster computing oppure puoi usare Amazon EKS Hybrid Nodes con la tua infrastruttura on-premises o edge per il cluster computing. AWS Outposts è un'infrastruttura gestita da AWS che esegui nei data center o nelle strutture di co-location, mentre Amazon EKS Hybrid Nodes funziona su macchine fisiche o virtuali gestite in ambienti on-premises o edge. Amazon EKS su AWS Outposts e Amazon EKS Hybrid Nodes richiedono una connessione affidabile dagli ambienti on-premises a una Regione AWS e puoi utilizzare gli stessi cluster, funzionalità e strumenti Amazon EKS che usi per eseguire applicazioni nel cloud. In alternativa, quando esegui su AWS Outposts, puoi implementare l'intero cluster Kubernetes su AWS Outposts con i cluster locali Amazon EKS su AWS Outposts.

	Amazon EKS Hybrid Nodes	Amazon EKS su AWS Outposts
Gestione del piano di controllo (control-plane) di Kubernetes	Gestito da AWS	Gestito da AWS
Posizione del piano di controllo Kubernetes	Regioni AWS	Regioni AWS o AWS Outposts
Piano dati di Kubernetes	Macchine fisiche o virtuali gestite dal cliente	Nodi di Amazon EC2 autogesti ti
Posizione del piano dati Kubernetes	Data center o ambiente edge del cliente	Data center o ambiente edge del cliente

Amazon EKS Anywhere per ambienti isolati

Amazon EKS Anywhere semplifica la gestione dei cluster Kubernetes attraverso l'automazione di operazioni indifferenziate come la configurazione dell'infrastruttura e le operazioni del ciclo di vita del cluster Kubernetes in ambienti on-premises ed edge. A differenza di Amazon EKS, Amazon EKS Anywhere è un prodotto gestito dai clienti e questi ultimi sono responsabili delle operazioni del ciclo di vita dei cluster e della manutenzione dei cluster Amazon EKS Anywhere. Amazon EKS Anywhere è basato sul sottoprogetto dell'API cluster (CAPI) Kubernetes e supporta una vasta gamma di infrastrutture tra cui VMware vSphere, bare metal, Nutanix, Apache CloudStack e AWS Snow. Amazon EKS Anywhere può essere eseguito in ambienti isolati e offre integrazioni opzionali con servizi AWS regionali per l'osservabilità e la gestione delle identità. Per ricevere supporto per

Amazon EKS Anywhere e accedere ai componenti aggiuntivi Kubernetes forniti da AWS, puoi acquistare gli abbonamenti Amazon EKS Anywhere Enterprise.

	Amazon EKS Anywhere
Gestione del piano di controllo (control-plane) di Kubernetes	Gestito dal cliente
Posizione del piano di controllo Kubernetes	Data center o ambiente edge del cliente
Piano dati di Kubernetes	Macchine fisiche o virtuali gestite dal cliente
Posizione del piano dati Kubernetes	Data center o ambiente edge del cliente

Strumenti di Amazon EKS

Puoi utilizzare Amazon EKS Connector per registrare e connettere qualsiasi cluster Kubernetes conforme ad AWS e visualizzarlo nella console Amazon EKS. Una volta connesso, è possibile visualizzare lo stato, la configurazione e i carichi di lavoro del cluster nella console Amazon EKS. Puoi utilizzare questa funzionalità per visualizzare i cluster connessi nella console Amazon EKS, ma Amazon EKS Connector non semplifica la gestione o le operazioni di modifica dei cluster connessi tramite tale console.

Amazon EKS Distro è la distribuzione AWS dei componenti Kubernetes sottostanti che alimentano tutte le offerte di Amazon EKS. Include i componenti principali necessari per un cluster Kubernetes funzionante, come i componenti del piano di controllo (control-plane) di Kubernetes (etcd, kube-apiserver, kube-scheduler, kube-controller-manager) e i componenti di rete (CoreDNS, kube-proxy, plug-in della Container Network Interface (CNI)). Amazon EKS Distro può essere utilizzato per gestire autonomamente i cluster Kubernetes con strumenti a tua scelta. Le implementazioni di Amazon EKS Distro non sono coperte dai piani di supporto AWS.

Configurazione per l'utilizzo di Amazon EKS

Per prepararti per la gestione a riga di comando dei cluster Amazon EKS, devi installare molti strumenti. Utilizza quanto segue per configurare le credenziali, creare e modificare i cluster e lavorare con i cluster una volta che sono in esecuzione:

- Configura AWS CLI: ottieni AWS CLI per configurare e gestire i servizi necessari per lavorare con i cluster Amazon EKS. In particolare, è necessario disporre di AWS CLI per configurare le credenziali, ma è necessaria anche con altri servizi AWS.
- Configura kubectl ed eksctl: `eksctl` CLI interagisce con AWS per creare, modificare ed eliminare i cluster Amazon EKS. Una volta che un cluster è attivo, utilizza il comando `kubectl` open-source per gestire gli oggetti Kubernetes all'interno dei cluster Amazon EKS.
- Configura un ambiente di sviluppo (opzionale): valuta la possibilità di aggiungere i seguenti strumenti:
  - Strumento di implementazione locale: se non hai esperienza con Kubernetes, valuta la possibilità di installare uno strumento di implementazione locale come minikube o kind. Questi strumenti ti consentono di gestire un cluster Amazon EKS sulla macchina locale per testare le applicazioni.
  - Gestore pacchetti: Helm è un celebre gestore di pacchetti per Kubernetes che semplifica l'installazione e la gestione di pacchetti complessi. Grazie a Helm è più facile installare e gestire pacchetti, come AWS Load Balancer Controller, sul cluster Amazon EKS.

Passaggi successivi

- Configurazione di AWS CLI
- Configura kubectl ed eksctl
- Avvio rapido: implementa un'app Web e archivia i dati

Configurazione di AWS CLI

AWS CLI è uno strumento a riga di comando per usare i servizi AWS, tra cui Amazon EKS. Viene anche utilizzata per autenticare utenti o ruoli IAM per l'accesso al cluster Amazon EKS e ad altre risorse AWS dalla macchina locale. Per effettuare il provisioning di risorse in AWS dalla riga di comando, è necessario ottenere un ID della chiave di accesso AWS e una chiave segreta da utilizzare nella riga di comando. Quindi è necessario configurare queste credenziali nella AWS CLI.

Se non hai già installato AWS CLI, consulta Install or update the latest version of the AWS CLI nella Guida per l'utente dell'Interfaccia a riga di comando AWS.

Per creare una chiave di accesso

1. Accedi alla Console di gestione AWS.
2. Per account con utente singolo o multiplo:
   - Account con utente singolo: in alto a destra, scegli il tuo nome utente AWS per aprire il menu di navigazione. Ad esempio, scegli **`webadmin`** .
   - Account multiutente: scegli IAM dall'elenco dei servizi. Dalla dashboard IAM, seleziona Utenti e scegli il nome dell'utente.
3. Scegli Security Credentials (Credenziali di sicurezza).
4. In Chiavi di accesso, seleziona Crea chiave di accesso.
5. Scegli Interfaccia a riga di comando (CLI), quindi scegli Avanti.
6. Selezionare Create access key (Crea chiave di accesso).
7. Scegli Scarica file .csv.

Per configurare la CLI di AWS

Dopo aver installato AWS CLI, segui le fasi di seguito per installarla e configurarla. Per ulteriori informazioni, consulta Configure the AWS CLI nella Guida per l'utente dell'interfaccia a riga di comando AWS.

1. In una finestra del terminale, digita il comando riportato qui sotto:

   ```
   aws configure
   ```

   Facoltativamente, puoi configurare un profilo con un nome, ad esempio `--profile cluster-admin`. Se configuri un profilo con un nome nella AWS CLI, è necessario passare sempre questo flag nei comandi successivi.
2. Inserisci le credenziali AWS. Per esempio:

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

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

Per ottenere un token di sicurezza

Se necessario, esegui il seguente comando per ottenere un nuovo token di sicurezza per AWS CLI. Per ulteriori informazioni, consulta get-session-token nella Guida di riferimento dei comandi di CLI AWS.

Per impostazione predefinita, il token è valido per 15 minuti. Per modificare il timeout di sessione predefinito, passa il flag `--duration-seconds`. Per esempio:

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

Questo comando restituisce le credenziali di sicurezza temporanee per una sessione di AWS CLI. Dovresti vedere il seguente messaggio di risposta:

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

Per verificare l'identità dell'utente

Se necessario, esegui il comando seguente per verificare le credenziali AWS dell'identità utente IAM (ad esempio *`ClusterAdmin`*) per la sessione del terminale.

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

Questo comando restituisce il nome della risorsa Amazon (ARN) dell'entità IAM configurata per la AWS CLI. Dovresti vedere il seguente esempio di risposta:

```
{
    "UserId": "AKIAIOSFODNN7EXAMPLE",
```

```
    "Account": "01234567890",
    "Arn": "arn:aws:iam::01234567890:user/ClusterAdmin"
}
```

Passaggi successivi

- Configura kubectl ed eksctl
- Avvio rapido: implementa un'app Web e archivia i dati

Impostazione di **`kubectl`** e **`eksctl`**

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

Una volta installata la AWS CLI, ci sono altri due strumenti da installare per creare e gestire i cluster Kubernetes:

- `kubectl`: lo strumento a riga di comando `kubectl` è lo strumento principale per gestire le risorse all'interno del tuo cluster Kubernetes. Questa pagina descrive come scaricare e configurare il file binario `kubectl` che corrisponde alla versione del cluster Kubernetes. Consultare Install or update kubectl.
- `eksctl`: Lo strumento a riga di `eksctl` comando è progettato per creare cluster EKS nel AWS cloud o in locale (con EKS Anywhere), nonché per modificare ed eliminare tali cluster. Consultare Installare eksctl.

Installazione o aggiornamento di **`kubectl`**

In questa sezione viene descritto come scaricare e installare o aggiornare il file binario `kubectl` sul tuo dispositivo. Il file binario è identico alle versioni della community upstream. Il file binario non è esclusivo di Amazon EKS o AWS. Segui i passaggi seguenti per ottenere la versione specifica di `kubectl` cui hai bisogno, anche se molti builder si limitano a eseguire `brew install kubectl` per installarla.

> **Note**
>
> Utilizza la versione secondaria `kubectl` immediatamente precedente a quella del piano di controllo del cluster Amazon EKS. Ad esempio, un client `1.32` `kubectl` deve funzionare con cluster Kubernetes `1.31`, `1.32` e `1.33`.

Fase 1: verifica se **`kubectl`** è installato

Determina se `kubectl` è già installato sul tuo dispositivo.

```
kubectl version --client
```

Se `kubectl` è già installato nel percorso del tuo dispositivo, l'output di esempio include informazioni simili alle seguenti. Se desideri aggiornare la versione correntemente installata con una versione più recente, completa la fase successiva assicurandoti di installare la nuova versione nella stessa posizione in cui si trova la versione corrente.

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

Se non ricevi alcun output, allora `kubectl` non è installato o è installato in una posizione che non si trova nel percorso del dispositivo.

Fase 2: installa o aggiorna **`kubectl`**

Installare o aggiornare `kubectl` su uno dei seguenti sistemi operativi:

- the section called "macOS"
- the section called "Linux (amd64)"
- the section called "Linux (arm64)"
- the section called "Windows"

> **Note**
>
> Se i download AWS nella tua regione sono lenti a causa delle AWS aree utilizzate in questa sezione, valuta la possibilità di CloudFront configurare il contenuto in primo piano. Per ulteriori informazioni, consulta Guida introduttiva a una CloudFront distribuzione di base.

macOS

Segui la seguente procedura per installare `kubectl` su macOS. Queste fasi includono:

- Scegli e scarica il file binario per la versione di Kubernetes desiderata.
- Facoltativamente, verifica il checksum del file binario.
- Aggiungi l'esecuzione alle autorizzazioni del file binario.
- Copia il file binario in una cartella nel tuo PATH.
- Facoltativamente, aggiungi la directory del file binario al tuo PATH.

Procedura:

1. Scarica il file binario per la versione del cluster di Kubernetes da 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. (Facoltativo) Verifica il file binario scaricato con il relativo checksum `SHA-256`.
   a. Scarica il checksum di `SHA-256` per la versione del tuo cluster di Kubernetes.
      - 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. Controlla il checksum `SHA-256` del file binario scaricato.

```
openssl sha1 -sha256 kubectl
```

c. Assicurati che la somma di controllo generata nell'output corrisponda alla somma di controllo nel file scaricato `kubectl.sha256`.

3. Applica le autorizzazioni di esecuzione al file binario.

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

4. Copia il file binario in una cartella nel tuo `PATH`. Se disponi già di una versione installata di `kubectl`, consigliamo di creare un `$HOME/bin/kubectl` e verificare che `$HOME/bin` venga per primo in `$PATH`.

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

5. (Facoltativo) Aggiungi il percorso `$HOME/bin` al file di inizializzazione dello shell (interprete di comandi), in modo che sia configurato all'apertura di quest'ultimo.

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

Linux (amd64)

Segui la seguente procedura per installare `kubectl` su Linux (amd64). Queste fasi includono:

- Scegli e scarica il file binario per la versione di Kubernetes desiderata.
- Facoltativamente, verifica il checksum del file binario.
- Aggiungi l'esecuzione alle autorizzazioni del file binario.
- Copia il file binario in una cartella nel tuo PATH.
- Facoltativamente, aggiungi la directory del file binario al tuo PATH.

Procedura:

1. Scarica il file binario `kubectl` per la versione del cluster di Kubernetes da 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. (Facoltativo) Verifica il file binario scaricato con il relativo checksum `SHA-256`.
   a. Scarica il checksum di `SHA-256` per la versione del cluster di Kubernetes da Amazon S3 utilizzando il comando per la tua piattaforma hardware.
      - 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. Controlla la somma di controllo SHA-256 del file binario scaricato con uno dei seguenti comandi.

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

or

```
openssl sha1 -sha256 kubectl
```

c. Come prima cosa, dovresti vedere `kubectl: OK`, quindi puoi verificare che il checksum generato nell'output corrisponda al checksum nel file scaricato `kubectl.sha256`.

3. Applica le autorizzazioni di esecuzione al file binario.

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

4. Copia il file binario in una cartella nel tuo PATH. Se disponi già di una versione installata di `kubectl`, consigliamo di creare un `$HOME/bin/kubectl` e verificare che `$HOME/bin` venga per primo in `$PATH`.

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

5. (Facoltativo) Aggiungi il percorso `$HOME/bin` al file di inizializzazione dello shell (interprete di comandi), in modo che sia configurato all'apertura di quest'ultimo.

> **Note**
>
> Questa fase prevede l'utilizzo dello shell (interprete di comandi) Bash; se stai utilizzando un altro shell, modifica il comando per utilizzare il file di inizializzazione dello shell in uso.

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

Linux (arm64)

Segui la seguente procedura per installare `kubectl` su Linux (arm64). Queste fasi includono:

- Scegli e scarica il file binario per la versione di Kubernetes desiderata.
- Facoltativamente, verifica il checksum del file binario.
- Aggiungi l'esecuzione alle autorizzazioni del file binario.
- Copia il file binario in una cartella nel tuo PATH.
- Facoltativamente, aggiungi la directory del file binario al tuo PATH.

Procedura:

1. Scarica il file binario `kubectl` per la versione del cluster di Kubernetes da 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. (Facoltativo) Verifica il file binario scaricato con il relativo checksum `SHA-256`.
   a. Scarica il checksum di `SHA-256` per la versione del cluster di Kubernetes da Amazon S3 utilizzando il comando per la tua piattaforma hardware.
      - 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. Controlla la somma di controllo `SHA-256` del file binario scaricato con uno dei seguenti comandi.

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

or

```
openssl sha1 -sha256 kubectl
```

c. Come prima cosa, dovresti vedere `kubectl: OK`, quindi puoi verificare che il checksum generato nell'output corrisponda al checksum nel file scaricato `kubectl.sha256`.

3. Applica le autorizzazioni di esecuzione al file binario.

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

4. Copia il file binario in una cartella nel tuo `PATH`. Se disponi già di una versione installata di `kubectl`, consigliamo di creare un `$HOME/bin/kubectl` e verificare che `$HOME/bin` venga per primo in `$PATH`.

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

5. (Facoltativo) Aggiungi il percorso `$HOME/bin` al file di inizializzazione dello shell (interprete di comandi), in modo che sia configurato all'apertura di quest'ultimo.

> **Note**
>
> Questa fase prevede l'utilizzo dello shell (interprete di comandi) Bash; se stai utilizzando un altro shell, modifica il comando per utilizzare il file di inizializzazione dello shell in uso.

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

Windows

Segui la seguente procedura per installare `kubectl` su Windows. Queste fasi includono:

- Scegli e scarica il file binario per la versione di Kubernetes desiderata.
- Facoltativamente, verifica il checksum del file binario.
- Copia il file binario in una cartella nel tuo PATH.
- Facoltativamente, aggiungi la directory del file binario al tuo PATH.

Procedura:

1. Aprire un terminale. PowerShell
2. Scarica il file binario `kubectl` per la versione del cluster di Kubernetes da 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. (Facoltativo) Verifica il file binario scaricato con il relativo checksum `SHA-256`.

   a. Scarica il checksum di `SHA-256` per la versione per Windows del tuo cluster di Kubernetes.

   - 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. Controlla il checksum SHA-256 del file binario scaricato.

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

c. Assicurati che il checksum generato nell'output corrisponda al checksum nel file `kubectl.sha256` scaricato. L' PowerShell output deve essere una stringa di caratteri equivalente in maiuscolo.

4. Copia il file binario in una cartella nel tuo `PATH`. Se disponi di una directory esistente nel `PATH`dedicata alle utility a riga di comando, copia il file binario in questa directory. In alternativa, completa la procedura seguente.
   a. Crea una nuova directory per i file binari della riga di comando, ad esempio `C:\bin`.
   b. Copia il file binario `kubectl.exe` nella nuova directory.
   c. Modifica la variabile di ambiente `PATH` dell'utente o del sistema per aggiungere la nuova directory a `PATH`.
   d. Chiudi il PowerShell terminale e apritene uno nuovo per raccogliere la nuova `PATH` variabile.
5. Dopo l'installazione di `kubectl`, puoi verificarne la versione.

```
kubectl version --client
```

6. Alla prima installazione, `kubectl` non è ancora configurato per comunicare con alcun server. Tratteremo questa configurazione, se necessario, in altre procedure. Per aggiornare la configurazione per comunicare con un particolare cluster, è possibile eseguire il comando seguente. Sostituiscila *`region-code`* con la AWS regione in cui si trova il cluster. Sostituisci *`my-cluster`* con il nome del cluster.

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

7. Prendi in considerazione la configurazione del completamento automatico, che consente di utilizzare il tasto TAB per completare i sottocomandi `kubectl` dopo aver digitato le prime lettere. Per i dettagli, consultare Kubectl autocomplete nella documentazione di Kubernetes.

Installazione di **`eksctl`**

`eksctl` CLI viene utilizzata per lavorare con i cluster EKS. Automatizza molte attività individuali. Per istruzioni sull'installazione di `eksctl`, consultare Installation nella documentazione di `eksctl`. Per Linux, usa le istruzioni UNIX.

Quando utilizzi `eksctl` il principale di sicurezza IAM che stai utilizzando, devi disporre delle autorizzazioni per lavorare con i ruoli IAM di Amazon EKS, i ruoli collegati ai servizi AWS CloudFormation, un VPC e le risorse correlate. Per ulteriori informazioni, consulta le pagine Actions e Using service-linked roles nella Guida per l'utente IAM. È necessario che tutti i passaggi di questa guida siano completati dallo stesso utente. Esegui il comando seguente per controllare l'utente corrente:

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

Fasi successive

- Avvio rapido: implementa un'app Web e archivia i dati

Avvio rapido: implementazione di un'app Web e archiviazione dei dati

Questo tutorial di avvio rapido ti guida attraverso i passaggi per distribuire l'applicazione di esempio di gioco 2048 e mantenere i suoi dati su un cluster Amazon EKS Auto Mode utilizzando eksctl.

La modalità automatica di Amazon EKS semplifica la gestione dei cluster automatizzando attività di routine come lo storage a blocchi, il networking, il bilanciamento del carico e la scalabilità automatica del calcolo. Durante la configurazione, gestisce la creazione di nodi con istanze EC2 gestite, sistemi di bilanciamento del carico delle applicazioni e volumi EBS.

In sintesi, distribuirai un carico di lavoro di esempio con le annotazioni personalizzate necessarie per una perfetta integrazione con i servizi. AWS

In questo tutorial

Utilizzando il modello di cluster `eksctl` che segue, creerai un cluster con modalità automatica EKS per il provisioning automatico dei nodi.

- Configurazione VPC: quando si utilizza il modello di cluster eksctl che segue, eksctl crea automaticamente un Virtual Private Cloud (IPv4 VPC) per il cluster. Per impostazione predefinita, eksctl configura un VPC che soddisfa tutti i requisiti di rete, oltre a creare endpoint pubblici e privati.
- Gestione delle istanze: EKS Auto Mode aggiunge o rimuove dinamicamente i nodi nel cluster EKS in base alle esigenze delle applicazioni Kubernetes.
- Persistenza dei dati: utilizzate la funzionalità di archiviazione a blocchi di EKS Auto Mode per garantire la persistenza dei dati delle applicazioni, anche in scenari che prevedono riavvii o guasti dei pod.
- Accesso esterno alle app: utilizza la funzionalità di bilanciamento del carico di EKS Auto Mode per effettuare il provisioning dinamico di un Application Load Balancer (ALB).

Prerequisiti

Prima di iniziare, esegui le seguenti attività:

- Configura il tuo ambiente per Amazon EKS

- Installa la versione più recente di eksctl

Configurazione del cluster

In questa sezione creerai un cluster utilizzando la modalità automatica EKS per il provisioning dinamico dei nodi.

Crea un file `cluster-config.yaml` e incollaci il seguente contenuto. Sostituisci `region-code` con una regione valida (ad es.). `us-east-1`

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

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

autoModeConfig:
  enabled: true
```

Ora siamo pronti per creare il cluster.

Crea il cluster EKS utilizzando`cluster-config.yaml`:

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

> **Important**
>
> Se non usi eksctl per creare il cluster, devi etichettare manualmente le sottoreti VPC.

Crea IngressClass

Crea un Kubernetes `IngressClass` per la modalità automatica EKS. IngressClass Definisce come EKS Auto Mode gestisce le risorse di Ingress. Questo passaggio configura la capacità di bilanciamento del carico della modalità automatica EKS. Quando create risorse Ingress per le vostre applicazioni, EKS Auto Mode le utilizza per fornire e IngressClass gestire automaticamente i sistemi di bilanciamento del carico, integrando le applicazioni Kubernetes con i servizi di bilanciamento del carico. AWS

Salva il seguente file yaml come `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
```

Applica il al tuo cluster: IngressClass

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

Distribuzione dell'applicazione di gioco 2048 esemplificativa

In questa sezione, ti illustreremo i passaggi per implementare il popolare "gioco 2048" come applicazione esemplificativa all'interno del cluster. Il manifesto fornito include annotazioni personalizzate per l'Application Load Balancer (ALB). Queste annotazioni si integrano e indicano a EKS di gestire il traffico HTTP in entrata come «rivolto a Internet» e di indirizzarlo al servizio appropriato nel `game-2048` namespace utilizzando il tipo di destinazione «ip».

> **Note**
>
> L'immagine `docker-2048` nell'esempio è un'immagine di container `x86_64` e non sarà eseguita su altre architetture.

1. Crea un namespace Kubernetes chiamato `game-2048` con il flag `--save-config`.

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

   Dovresti vedere il seguente messaggio di risposta:

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

2. Implementare l'applicazione di gioco 2048 esemplificativa.

```
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
```

Questo manifesto configura l'implementazione, il servizio e l'entrata Kubernetes per il namespace `game-2048`, creando le risorse necessarie per implementare ed esporre l'applicazione `game-2048` all'interno del cluster. Include la creazione di un servizio denominato `service-2048` che espone l'implementazione sulla porta `80` e una risorsa di entrata denominata `ingress-2048` che definisce le regole di routing per il traffico HTTP in entrata e le annotazioni per un Application Load Balancer (ALB) con accesso a Internet. Dovresti vedere il seguente messaggio di risposta:

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

3. Esegui il seguente comando per ottenere la risorsa di ingresso per il namespace `game-2048`.

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

Dovresti vedere il seguente messaggio di risposta:

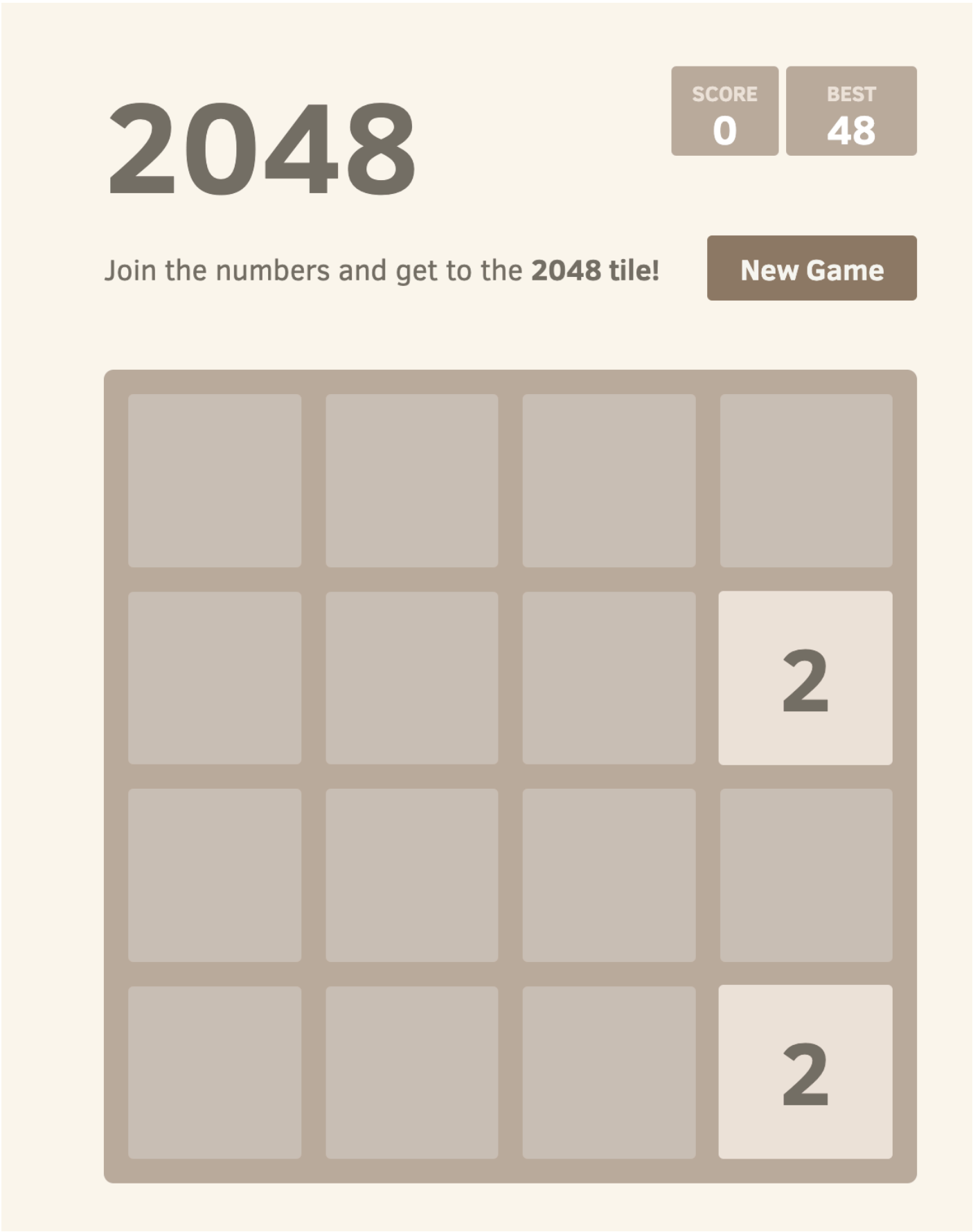
```
NAME           CLASS   HOSTS   ADDRESS
                      PORTS   AGE
ingress-2048   alb     *       k8s-game2048-ingress2-eb379a0f83-378466616.region-
code.elb.amazonaws.com   80      31s
```

Dovrai attendere alcuni minuti per effettuare il provisioning dell'Application Load Balancer (ALB) prima di iniziare con i passaggi seguenti.

4. Apri un browser Web e inserisci `ADDRESS` dal passaggio precedente per accedere all'applicazione Web. Esempio:

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

Il gioco 2048 dovrebbe essere visualizzato nel browser in uso. Gioca!



Distribuzione dell'applicazione di gioco 2048 esemplificativa

Dati persistenti utilizzando la modalità automatica Amazon EKS

Ora che il gioco 2048 è attivo e funzionante sul tuo cluster Amazon EKS, è il momento di garantire che i tuoi dati di gioco siano conservati in modo sicuro, utilizzando la funzionalità di archiviazione a blocchi della modalità automatica Amazon EKS.

1. Creare un file denominato `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. Applica `StorageClass`:

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

3. Crea una richiesta di volumi persistenti (PVC) per richiedere lo spazio di archiviazione per i tuoi dati di gioco. Crea un file denominato `ebs-pvc.yaml` che contenga quando segue e aggiungilo a .

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

4. Applica PVC al cluster:

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

Dovresti vedere il seguente messaggio di risposta:

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

5. Ora devi aggiornare l'implementazione del gioco 2048 per utilizzare questo PVC per archiviare i dati. La seguente implementazione è configurata per utilizzare PVC per l'archiviazione dei dati di gioco. Crea un file denominato `ebs-deployment.yaml` e aggiungilo a esso:

```
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. Applica l'implementazione aggiornata:

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

Dovresti vedere il seguente messaggio di risposta:

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

Con questi passaggi, il gioco 2048 sul cluster è ora configurato per rendere persistenti i dati, utilizzando la funzionalità di archiviazione a blocchi della modalità automatica Amazon EKS. Ciò garantisce che i progressi e i dati di gioco siano al sicuro anche in caso di guasti ai pod o ai nodi.

Se ti è piaciuto questo tutorial, faccelo sapere fornendo un feedback, così saremo in grado di fornirti altri tutorial rapidi di utilizzo specifici per ogni caso, come questo.

Eliminazione

Per evitare di incorrere in addebiti futuri, è necessario eliminare manualmente lo CloudFormation stack associato per eliminare tutte le risorse create durante questa guida, inclusa la rete VPC.

Eliminare lo stack: CloudFormation

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

Scopri Amazon EKS con l'esempio

Panoramica

Questa guida per l'utente di Amazon EKS contiene procedure generiche per creare il tuo primo cluster EKS dalla riga di comando o Console di gestione AWS e un solido riferimento per tutti i principali componenti di Amazon EKS. Tuttavia, in qualità di amministratore o sviluppatore di cluster Amazon EKS, puoi acquisire una comprensione più approfondita di Amazon EKS seguendo i percorsi di apprendimento presenti in siti esterni a questa guida. Questi siti possono aiutarti a:

- Configurare tipi di cluster specifici. I tipi di cluster specifici possono essere basati sui tipi di carico di lavoro o sui requisiti di sicurezza. Ad esempio, potresti voler ottimizzare un cluster per eseguire carichi di lavoro in batch, di machine learning o con elaborazione intensiva.
- Migliorare i tuoi cluster. Puoi aggiungere funzionalità avanzate al tuo cluster per fornire funzionalità come osservabilità, archiviazione flessibile, dimensionamento automatico o reti di cluster specializzate.
- Automatizzare gli aggiornamenti. Utilizzando funzionalità come GitOps, puoi configurare il provisioning automatico dell'infrastruttura e dei carichi di lavoro del cluster, in base alle modifiche apportate a tali componenti nei tuoi repository Git.
- Utilizzare strumenti avanzati di configurazione del cluster. Anche se `eksctl` rappresenta un modo rapido per creare un cluster, esistono altri strumenti che possono semplificare la configurazione e l'aggiornamento di cluster più complessi. Tra questi ci sono strumenti come Terraform e CloudFormation.

Per iniziare il percorso di apprendimento su Amazon EKS, consiglio di visitare alcuni dei siti descritti in questa pagina. Se riscontri problemi lungo il percorso, sono disponibili anche delle risorse per aiutarti a risolverli. Ad esempio, il Re:post Knowledge Center consente di cercare nel database di assistenza problemi di supporto relativi ad Amazon EKS. Inoltre, la Amazon EKS Best Practices Guide offre suggerimenti sui modi migliori per configurare cluster di livello di produzione.

Workshop su Amazon EKS

Partendo da una conoscenza di base di Kubernetes e dei container, Amazon EKS workshop è una piattaforma di apprendimento per insegnare agli amministratori dei cluster le funzionalità importanti di Amazon EKS. Ecco alcuni modi per partecipare al workshop Amazon EKS:

- Amazon EKS Basics: guarda il video nella pagina di introduzione per scoprire come Amazon EKS implementa le funzionalità di Kubernetes sul cloud AWS. Se hai bisogno di conoscere meglio le basi di Kubernetes, guarda il video What is Kubernetes.
- Configurazione Amazon EKS: se disponi di un account AWS, la sezione Configurazione ti aiuta a configurare un ambiente CloudShell per creare un cluster. Infatti, offre una scelta tra eksctl (una semplice riga di comando per la creazione di cluster) e Terraform (un approccio più da infrastruttura come codice alla creazione di un cluster) per creare un cluster Amazon EKS.
- Amazon EKS Getting started: prova un semplice web store nella sezione Sample application. Puoi utilizzarlo durante gli altri esercizi. In questa sezione, puoi anche scoprire come creare pacchetti con le immagini dei container e come vengono gestiti i microservizi utilizzando pod, implementazioni, servizi, StatefulSets e Namespace Kubernetes. Quindi utilizza Kustomize per implementare le modifiche ai manifesti di Kubernetes.
- Amazon EKS Fundamentals: utilizzando funzionalità AWS come AWS Load Balancer Controller, il workshop mostra come esporre le applicazioni al mondo esterno. Per l'archiviazione, il workshop mostra come utilizzare Amazon EBS per l'archiviazione a blocchi, Amazon EFS per l'archiviazione dei file system e Amazon FSx per NetApp ONTAP per gestire i file system ONTAP in AWS. Per la gestione dei nodi, il workshop ti aiuta a configurare i gruppi di nodi gestiti.
- Funzionalità avanzate di Amazon EKS: le funzionalità più avanzate offerte tramite il workshop Amazon EKS includono laboratori per la configurazione di:
  - Dimensionamento automatico: include il dimensionamento automatico dei nodi (con Cluster Autoscaler o Karpenter) e quello dei carichi di lavoro (con Horizontal Pod Autoscaler e Cluster Proportional Autoscaler).
  - Osservabilità: scopri di più su registrazione, OpenSearch, Container Insights su Amazon EKS e la visibilità dei costi con Kubecost in una serie di laboratori sull'osservabilità.
  - Sicurezza: questo set di laboratori di sicurezza consente di esplorare la gestione dei segreti, Amazon GuardDuty, gli standard di Pod Security e la gestione delle policy di Kyverno.
  - Rete: scopri le funzionalità di rete per Amazon EKS dai laboratori sulla rete che includono CNI di Amazon VPC (plug-in della rete di supporto) e Amazon VPC Lattice (per la configurazione di cluster tra VC e account utente).
  - Automazione: laboratori su automazione per mostrare i metodi GitOps per gestire cluster e progetti come AWS Controllers per Kubernetes e Crossplane per la gestione dei piani di controllo di Amazon EKS.

Tutorial pratici per la configurazione dei cluster su Amazon EKS

Una serie di tutorial sulla configurazione dei cluster Amazon EKS sul sito della community AWS può aiutarti a creare cluster Amazon EKS speciali e a migliorarli in vari modi. I tutorial sono suddivisi in tre tipi diversi:

Creazione di cluster

Questi tutorial aiutano a creare cluster che possono essere utilizzati per scopi speciali. Tra questi, c'è la possibilità di eseguire:

- Applicazioni scalabili a livello globale basate su IPv6
- Attività in batch asincrone
- Microservizi ad alto traffico
- Dimensionamento automatico con Karpenter su Fargate
- Carichi di lavoro finanziari
- Gruppi di nodi gestiti da Windows

Potenziamento dei cluster

Una volta che disponi di un cluster esistente, puoi estenderlo e migliorarlo in modo da consentirgli di eseguire carichi di lavoro specializzati e di potenziare in altro modo i cluster. Questi tutorial includono modi per:

- Fornire soluzioni di archiviazione con CSI EFS
- Fornire un'archiviazione dinamica del database con CSI EBS
- Esporre le applicazioni su cluster IPv4 utilizzando il Load Balancer Controller di AWS
- Esporre le applicazioni su cluster IPv6 utilizzando il Load Balancer Controller di AWS

Ottimizzazione dei servizi AWS

Utilizzando questi tutorial, è possibile integrare meglio i cluster con i servizi AWS. Questi tutorial includono quelli che aiutano a:

- Gestire i record DNS per i microservizi con ExternalDNS
- Monitorare le applicazioni con CloudWatch

- Gestire le attività asincrone con l'archiviazione SQS ed EFS
- Utilizzare i segreti di Gestione dei segreti AWS dai carichi di lavoro
- Configurare mTLs con Fargate, NGINX e ACM PCA

Esempi di Amazon EKS

Il repository degli esempi di Amazon EKS memorizza i manifesti da utilizzare con Amazon EKS. Questi manifesti offrono l'opportunità di provare diversi tipi di applicazioni in Amazon EKS o creare tipi specifici di cluster Amazon EKS. Tra gli esempi troviamo manifesti per:

- Creare un cluster Fargate di AWS Amazon EKS
- Creare un cluster con un ruolo IAM esistente
- Aggiungere un gruppo di nodi gestito da Ubuntu a un cluster
- Eseguire il backup e ripristinare l'archiviazione dei pod con snapshot di volume
- Recuperare i volumi EBS montati come PVC con più account
- Abilitare il protocollo proxy per NGINX Ingress Controller con Classic Load Balancer
- Configurare la registrazione su Fargate in AWS OpenSearch
- Eseguire SDK Python con un provider di identità federato web
- Implementare un'app di esempio su un controller CSI NFS
- Usare snapshot di volume per StatefulSets
- Distribuire i pod tra nodi in zone di disponibilità diverse

Tieni presente che questi esempi sono solo a scopo di apprendimento e test e non sono destinati all'utilizzo in produzione.

Tutorial di AWS

Il sito AWS Tutorials pubblica alcuni tutorial di Amazon EKS, ma offre anche uno strumento di ricerca per trovarne altri pubblicati sui siti AWS (come il sito della community di AWS). I tutorial di Amazon EKS pubblicati direttamente su questo sito includono:

- Distribuzione di un'app web per container su Amazon EKS
- Esecuzione di cluster Kubernetes a un prezzo inferiore (istanze Amazon EKS e Spot)
- Come ottimizzare i costi dei lavori Jenkins su Kubernetes

Workshop per sviluppatori

Se sei uno sviluppatore di software e desideri creare o rifattorizzare applicazioni da eseguire su Amazon EKS, Amazon EKS Developers workshop è un buon punto di partenza. Questo workshop non aiuta solo a creare applicazioni containerizzate, ma anche a distribuire tali container in un registro di container (ECR) e da lì in un cluster Amazon EKS.

Inizia con Amazon EKS Python Workshop per completare il processo di rifattorizzazione di un'applicazione Python, quindi configura il tuo ambiente di sviluppo per prepararti alla distribuzione dell'applicazione. Consulta le sezioni relative a container, Kubernetes e Amazon EKS per prepararti a eseguire le applicazioni containerizzate in quegli ambienti.

Workshop Terraform

Anche se `eksctl` è uno strumento semplice per la creazione di un cluster, per tipi di implementazioni Amazon EKS di tipo infrastruttura come codice più complessi, Terraform è un popolare strumento di creazione e gestione di cluster Amazon EKS. Terraform Amazon EKS Workshop insegna come usare Terraform per creare un AWS VPC e i cluster Amazon EKS e per aggiungere miglioramenti opzionali al cluster. In particolare, esiste una sezione per la creazione di un cluster Amazon EKS privato

Formazione su AWS Amazon EKS

AWS offre una formazione formale per conoscere Amazon EKS. Un corso di formazione di tre giorni intitolato Running Containers on Amazon Elastic Kubernetes Service insegna:

- Nozioni di base su Kubernetes e Amazon EKS
- Come creare cluster Amazon EKS
- Proteggere Amazon EKS con l'autorizzazione AWS RBAC IAM e Kubernetes
- Strumenti di automazione GitOps
- Strumenti di monitoraggio
- Tecniche per migliorare costi, efficienza e resilienza

Nozioni di base su Amazon EKS

Prima di consultare le guide introduttive, verifica di aver completato la configurazione necessaria per utilizzare Amazon EKS. Per ulteriori informazioni, consulta Configurazione.

Sono disponibili due guide di istruzioni rapide per la creazione di un nuovo cluster Kubernetes con nodi in Amazon EKS:

- Nozioni di base su Amazon EKS – eksctl: questa guida alle operazioni di base consente di installare tutte le risorse necessarie per iniziare a usare Amazon EKS attraverso `eksctl`, una semplice utility a riga di comando per la creazione e la gestione di cluster di Kubernetes su Amazon EKS. Al termine del tutorial, sarà creato un cluster Amazon EKS in esecuzione a cui implementare applicazioni. Questo è il modo più rapido e semplice per iniziare a utilizzare Amazon EKS.
- Nozioni di base su Amazon EKS – Console di gestione AWS e AWS CLI: questa guida alle operazioni di base ti consente di creare tutte le risorse necessarie per iniziare a utilizzare Amazon EKS attraverso Console di gestione AWS e AWS CLI. Al termine del tutorial, sarà creato un cluster Amazon EKS in esecuzione a cui implementare applicazioni. Attraverso questa guida è possibile creare manualmente ogni risorsa richiesta per un cluster Amazon EKS. Le procedure offrono una visibilità completa sul modo in cui ogni risorsa viene creata e sul modo in cui interagiscono tra loro.

Offriamo anche i seguenti riferimenti:

- Per una raccolta di tutorial pratici, consulta EKS Cluster Setup su AWSCommunity.
- Per esempi di codice, consulta Code examples for Amazon EKS using AWS SDKs.

Nozioni di base su Amazon EKS: EKS Auto Mode

Come altre esperienze introduttive con EKS, la creazione del primo cluster con EKS Auto Mode delega la gestione del cluster stesso a AWS. Tuttavia, EKS Auto Mode estende l'automazione EKS affidando la responsabilità di molti servizi essenziali necessari per configurare l'infrastruttura dei carichi di lavoro (nodi, reti e vari servizi), semplificando la gestione dei nodi e la scalabilità per soddisfare le richieste del carico di lavoro.

Scegli uno dei seguenti modi per creare un cluster con EKS Auto Mode:

- the section called " AWS CLI": utilizza l'interfaccia a riga di comando `aws` per creare un cluster.

- the section called "Console di gestione": utilizza Console di gestione AWS per creare un cluster.
- the section called "CLI di eksctl": utilizza l'interfaccia a riga di comando `eksctl` per creare un cluster.

Se stai confrontando diversi approcci alla creazione del vostro primo cluster EKS, devi sapere che EKS Auto Mode AWS si è assunta ulteriori responsabilità di gestione dei cluster, tra cui la configurazione di componenti per:

- Avvio e scalabilità dei nodi man mano che la domanda di carico di lavoro aumenta e diminuisce.
- Aggiornare regolarmente il cluster stesso (piano di controllo), i sistemi operativi dei nodi e i servizi in esecuzione sui nodi.
- Scelta delle impostazioni predefinite che determinano elementi come la dimensione e la velocità dell'archivio dei nodi e la configurazione della rete Pod.

Per informazioni sui vantaggi ottenibili con i cluster EKS Auto Mode, consulta Modalità automatica EKS.

Nozioni di base su Amazon EKS: **`eksctl`**

> **Note**
>
> Questo argomento illustra come iniziare senza EKS Auto Mode.
> EKS Auto Mode automatizza le attività di routine per il cluster computing, l'archiviazione e la rete. Learn how to get started with Amazon EKS Auto Mode.

Questa guida consente di creare tutte le risorse necessarie per iniziare a usare Amazon Elastic Kubernetes Service (Amazon EKS) con `eksctl`, una semplice utility a riga di comando per la creazione e la gestione di cluster di Kubernetes su Amazon EKS. Al termine di questo tutorial, sarà creato un cluster Amazon EKS in esecuzione a cui implementare applicazioni.

Le procedure descritte in questa guida consentono di creare automaticamente diverse risorse che è necessario creare manualmente quando si crea il cluster utilizzando il comando Console di gestione AWS. Se si preferisce creare manualmente la maggior parte delle risorse e capire meglio come interagiscono tra loro, utilizzare la Console di gestione AWS per creare i cluster e calcolare. Per ulteriori informazioni, consulta the section called "Crea cluster (console e CLI)".

Prerequisiti

Prima di iniziare questo tutorial, è necessario installare e configurare gli strumenti AWS CLI, kubectl ed eksctl come descritto in Set up to use Amazon EKS.

Passaggio 1: Creazione di cluster e nodi Amazon EKS

> **Important**
>
> Per iniziare nel modo più semplice e rapido possibile, in questo argomento sono inclusi i passaggi per creare un cluster e nodi con impostazioni predefinite. Prima di creare un cluster e i nodi da utilizzare in produzione, è consigliabile acquisire familiarità con tutte le impostazioni e implementare il cluster e nodi con le impostazioni che soddisfano i requisiti. Per ulteriori informazioni, consulta the section called "Creazione di un cluster " e Gestione del calcolo. Alcune impostazioni possono essere abilitate solo quando crei cluster e nodi.

È possibile creare un cluster con uno dei seguenti tipi di nodo. Per ulteriori informazioni sui tipi di nodi, consultare Gestione del calcolo. Dopo aver implementato il cluster, puoi aggiungere altri tipi di nodo.

- Fargate – Linux: selezionare questo tipo di nodo se si desidera eseguire applicazioni Linux su the section called " AWS Fargate". Fargate è un motore di elaborazione serverless che consente di implementare pod di Kubernetes senza gestire le istanze Amazon EC2.
- Nodi gestiti – Linux – Selezionare questo tipo di nodo se si desidera eseguire applicazioni Amazon Linux su istanze Amazon EC2. Sebbene non sia trattato in questa guida, è anche possibile aggiungere nodi autogestiti di Windows e Bottlerocket al cluster.

Creare il cluster Amazon EKS con il seguente comando. Puoi sostituire *`my-cluster`* con il tuo valore. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della Regione AWS e dell'account AWS in cui stai creando il cluster. Sostituisci *`region-code`* con qualsiasi Regione AWS supportata da Amazon EKS. Per un elenco delle Regioni AWS supportate, consultare la pagina Amazon EKS endpoints and quotas nella AWS General Reference Guide.

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
```

La creazione di cluster richiede diversi minuti. Durante la creazione, vengono generate diverse righe di output. L'ultima riga di output è simile alla seguente riga di esempio.

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

`eksctl` ha creato un file di configurazione `kubectl` in `~/.kube/config` o ha aggiunto la configurazione del nuovo cluster all'interno di un file di configurazione esistente in `~/.kube/config` sul tuo computer.

Al termine della creazione del cluster, per visualizzare tutte le risorse create, visualizzare la pila CloudFormation AWS denominata `eksctl-`*`my-cluster`*`-cluster` nella console AWS CloudFormation.

Passaggio 2: Visualizzazione delle risorse Kubernetes

1. Visualizzare i nodi del cluster.

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

   Di seguito viene riportato un output di esempio:

   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
```

Per ulteriori informazioni sugli elementi visualizzati nell'output, consultare the section called "Accesso alle risorse di cluster".

2. Visualizzazione dei carichi di lavoro in esecuzione nel cluster.

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

Di seguito viene riportato un output di esempio:

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>
```

Per ulteriori informazioni sugli elementi visualizzati nell'output, consultare the section called "Accesso alle risorse di cluster".

Passaggio 3: Eliminazione del cluster e dei nodi

Dopo aver creato il cluster e i nodi per questo tutorial, è consigliabile eseguire la pulizia mediante l'eliminazione del cluster e dei nodi con il seguente comando. Per eseguire altre operazioni con questo cluster prima di eseguire la pulizia, consultare the section called "Passaggi successivi".

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

Passaggi successivi

I seguenti argomenti della documentazione consentono di estendere la funzionalità del cluster.

- Implementare un'applicazione di esempio al cluster.
- Il principale IAM in grado di creare il cluster è l'unico principale che può effettuare chiamate al server API Kubernetes tramite `kubectl` o la Console di gestione AWS. Se si desidera che altri principali IAM abbiano accesso al cluster, è necessario aggiungerli. Per ulteriori informazioni, consulta the section called "Accesso API Kubernetes" e the section called "Autorizzazioni richieste".

- Prima di implementare un cluster da utilizzare in produzione, ti consigliamo di acquisire familiarità con tutte le impostazioni di cluster e nodi. Alcune impostazioni, come l'abilitazione dell'accesso SSH ai nodi Amazon EC2, devono essere definite al momento della creazione del cluster.
- Per aumentare la sicurezza del cluster, configura il plugin Amazon VPC Container Networking Interface per utilizzare i ruoli IAM per gli account di servizio.

Inizia a usare Amazon EKS Console di gestione AWS e AWS CLI

> **Note**
>
> Questo argomento illustra come iniziare senza modalità automatica di EKS. Utilizza i Gruppi di nodi gestiti per distribuire i nodi.
> La modalità automatica di EKS automatizza le attività di routine per il cluster computing, l'archiviazione e la rete. Learn how to get started with Amazon EKS Auto Mode. La modalità automatica di EKS è il metodo preferito per distribuire i nodi.

Questa guida ti aiuta a creare tutte le risorse necessarie per iniziare a usare Amazon Elastic Kubernetes Service (Amazon EKS) utilizzando Console di gestione AWS e la CLI. AWS Con questa guida è possibile creare manualmente ogni risorsa. Al termine di questo tutorial, sarà creato un cluster Amazon EKS in esecuzione a cui implementare applicazioni.

Le procedure descritte in questa guida ti consentono di visualizzare in modo completo il modo in cui ogni risorsa viene creata e il modo in cui le risorse interagiscono tra loro. Se preferisci creare automaticamente la maggior parte delle risorse, utilizza la CLI `eksctl` per creare il cluster e i nodi. Per ulteriori informazioni, consulta the section called "Crea cluster (eksctl)".

Prerequisiti

Prima di iniziare questo tutorial, è necessario installare e configurare i seguenti strumenti e risorse necessarie per creare e gestire un cluster Amazon EKS.

- AWS CLI: uno strumento da riga di comando per lavorare con AWS servizi, incluso Amazon EKS. Per ulteriori informazioni, consulta la Guida per l'utente all'installazione nell'interfaccia a riga di AWS comando. Dopo aver installato la AWS CLI, ti consigliamo di configurarla anche. Per ulteriori informazioni, consulta Configurazione rapida con aws configure nella Guida per l'utente

dell'interfaccia a riga di AWS comando. Nota che la AWS CLI v2 è necessaria per utilizzare l'opzione update-kubeconfig mostrata in questa pagina.

- **`kubectl`**— Uno strumento da riga di comando per lavorare con i cluster Kubernetes. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".
- Autorizzazioni IAM richieste: il responsabile della sicurezza IAM che stai utilizzando deve disporre delle autorizzazioni per lavorare con i ruoli IAM di Amazon EKS, i ruoli collegati ai servizi AWS CloudFormation, un VPC e le risorse correlate. Per ulteriori informazioni, consulta le pagine Actions e Using service-linked roles nella Guida per l'utente IAM. È necessario che tutti i passaggi di questa guida siano completati dallo stesso utente. Esegui il comando seguente per controllare l'utente corrente:

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

Ti consigliamo di completare la procedura descritta in questo argomento in una shell Bash. In alternativa, puoi apportare alcune modifiche alla tua shell per alcuni comandi di script, come i caratteri di continuazione della riga, e per il modo in cui le variabili vengono impostate e utilizzate. Inoltre, le regole di escape e di utilizzo delle virgolette per la shell (interprete di comandi) potrebbero essere diverse. Per ulteriori informazioni, consulta Uso delle virgolette con le stringhe nella AWS CLI nella Guida per l'utente dell'interfaccia a riga di AWS comando.

Passaggio 1: Creazione del cluster Amazon EKS;

> **Important**
>
> Per iniziare nel modo più semplice e rapido possibile, in questo argomento sono inclusi i passaggi per creare un cluster con impostazioni predefinite. Prima di creare un cluster da utilizzare in produzione, è consigliabile acquisire familiarità con tutte le impostazioni e implementare un cluster con le impostazioni che soddisfano i tuoi requisiti. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ". Alcune impostazioni possono essere abilitate solo quando crei cluster.

1. Crea un Amazon VPC con sottoreti pubbliche e private che soddisfino i requisiti Amazon EKS. Sostituisci *`region-code`* con qualsiasi AWS regione supportata da Amazon EKS. Per un elenco di AWS regioni, consulta gli endpoint e le quote di Amazon EKS nella guida di riferimento AWS generale. Puoi sostituire *`my-eks-vpc-stack`* con un nome a tua scelta.

```
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**
>
> Per un elenco di tutte le risorse create dal comando precedente, apri la AWS CloudFormation console all'indirizzo. https://console.aws.amazon.com/cloudformation/ Scegli lo stack *my-eks-vpc-stack*, quindi la scheda Resources (Risorse).

2. Crea un ruolo IAM del cluster e collegalo alla policy gestita IAM richiesta dall'AMI per Amazon EKS. I cluster Kubernetes gestiti da Amazon EKS effettuano chiamate ad altri AWS servizi per tuo conto per gestire le risorse che utilizzi con il servizio.

   a. Copiare i seguenti contenuti in un file denominato *eks-cluster-role-trust-policy.json*.

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

   b. Creare il ruolo.

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

   c. Allega la policy IAM gestita da Amazon EKS richiesta al ruolo.

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

3. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/home#/clusters.

   Assicurati che la AWS regione mostrata in alto a destra della console sia la AWS regione in cui desideri creare il cluster. In caso contrario, scegli il menu a discesa accanto al nome AWS della regione e scegli la AWS regione che desideri utilizzare.

4. Scegli Crea cluster. Se non viene visualizzata questa opzione, scegli Cluster nel pannello di navigazione a sinistra.

5. Nella pagina Configura cluster, completa le seguenti operazioni:

   a. Seleziona Configurazione personalizzata e disabilita Usa modalità automatica di EKS (se preferisci un cluster della modalità automatica di EKS, consulta invece the section called "Console di gestione").

   b. Immettere un nome per il cluster, ad esempio *my-cluster*. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.

   c. Per Cluster Service Role, scegli*myAmazonEKSClusterRole*.

   d. Lascia le impostazioni rimanenti sui valori di default e seleziona Successivo.

6. Nella pagina Specifica rete, procedi come segue:

   a. Scegli l'ID del VPC creato in un passaggio precedente nell'elenco a discesa VPC. È qualcosa del genere** | my-eks-vpc-stack-VPC*.

   b. Seleziona le sottoreti create in un passaggio precedente nell'elenco a discesa Sottoreti. Le sottoreti saranno qualcosa del genere. ** | my-eks-vpc-stack-**

   c. Seleziona il gruppo di sicurezza creato in un passaggio precedente nell'elenco a discesa Gruppi di sicurezza aggiuntivi. È qualcosa di simile. ** | my-eks-vpc-stack-ControlPlaneSecurityGroup-**

   d. Lascia le impostazioni rimanenti sui valori di default e seleziona Successivo.

7. Nella pagina Configura osservabilità, scegli Avanti.

8. Nella pagina Seleziona componenti aggiuntivi, scegli Successivo.

Per ulteriori informazioni sui componenti aggiuntivi, consulta la pagina the section called "Componenti aggiuntivi Amazon EKS".

9. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, scegli Successivo.
10. Nella pagina Rivedi e crea, scegli Crea.

    A destra del nome del cluster, lo stato del cluster è Creazione in corso per alcuni minuti fino al termine del processo di provisioning del cluster. Non proseguire con il passaggio successivo finché lo stato non è Attivo.

> **Note**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

Passaggio 2: Configurazione del computer per comunicare con il cluster

In questa sezione, imparerai a creare un file `kubeconfig` per il cluster. Le impostazioni in questo file abilitano la CLI `kubectl` per comunicare con il cluster.

Prima di procedere, assicurati che la creazione del cluster sia stata completata correttamente nel passaggio 1.

1. Crea o aggiorna un file `kubeconfig` per il cluster. Sostituisci *`region-code`* con la AWS regione in cui hai creato il cluster. Sostituisci *`my-cluster`* con il nome del cluster.

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

   Per impostazione predefinita, il file `config` viene creato in `~/.kube` o la configurazione del nuovo cluster viene aggiunta a un file `config` esistente in `~/.kube`.

2. Prova la configurazione.

```
kubectl get svc
```

> **Note**
>
> Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso negato o non autorizzato (`kubectl`)" nell'argomento relativo alla risoluzione dei problemi.

Di seguito viene riportato un output di esempio:

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

Passaggio 3: Creazione di nodi

> **Important**
>
> Per iniziare nel modo più semplice e rapido possibile, in questo argomento sono inclusi i passaggi per creare nodi con la maggior parte delle impostazioni predefinite. Prima di creare i nodi da utilizzare in produzione, è consigliabile acquisire familiarità con tutte le impostazioni e implementare i nodi con le impostazioni che soddisfano i tuoi requisiti. Per ulteriori informazioni, consulta Gestione del calcolo. Alcune impostazioni possono essere abilitate solo quando crei i nodi.

Questa procedura configura il cluster per utilizzare gruppi di nodi gestiti per creare nodi, specificando le sottoreti e il ruolo IAM del nodo creato nei passaggi precedenti. Consente di eseguire applicazioni Amazon Linux su EC2 istanze Amazon.

Per ulteriori informazioni sui diversi modi di configurare i nodi in EKS, consulta Gestione del calcolo. Dopo aver implementato il cluster, puoi aggiungere altri tipi di nodo. Sebbene non sia trattato in questa guida, è anche possibile aggiungere nodi autogestiti di Windows e Bottlerocket al cluster.

Per creare il tuo gruppo di nodi gestiti da EC2 Linux

1. Creare un ruolo IAM del nodo e allegarvi la policy gestita IAM di Amazon EKS richiesta. Il `kubelet` daemon del nodo Amazon EKS effettua chiamate per tuo AWS APIs conto. I nodi ricevono le autorizzazioni per queste chiamate API attraverso un profilo dell'istanza IAM e le policy associate.

   a. Copia i seguenti contenuti in un file denominato `node-role-trust-policy.json`.

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

   b. Crea il ruolo IAM del nodo.

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

   c. Allega al ruolo le policy gestite IAM richieste.

   ```
   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. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/home#/clusters.

   e. Scegli il nome del cluster che hai creato nella Fase 1: Crea il tuo cluster Amazon EKS, ad esempio*my-cluster*.

   f. Nella ***my-cluster***pagina, procedi come segue:

g. Scegli la scheda Calcolo.

h. Scegli Aggiungi gruppo di nodi.

2. Nella pagina Configure Node Group (Configura gruppo di nodi), effettuare le seguenti operazioni:

a. Per Nome, immettere un nome univoco per il gruppo di nodi gestiti, ad esempio *my-nodegroup*. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura.

b. Per il nome del ruolo Node IAM, scegli il *myAmazonEKSNodeRole* ruolo creato in un passaggio precedente. Consigliamo che ciascun gruppo di nodi utilizzi il proprio ruolo IAM univoco.

c. Scegli Next (Successivo).

3. Sulla pagina Set compute and scaling configuration (Impostazione della configurazione di calcolo e dimensionamento) accettare i valori di default e selezionare Next (Successivo).

4. Nella pagina Specify networking (Specifica rete), accettare i valori di default e selezionare Next (Successivo).

5. Nella pagina Rivedi e crea, controlla la configurazione del gruppo di nodi gestiti e scegli Crea.

6. Dopo alcuni minuti, lo Stato nella Configurazione del gruppo di nodi cambierà da In Passaggio di creazione a Attivo. Non proseguire con il passaggio successivo finché lo stato non è Attivo.

Passaggio 4: Visualizzazione delle risorse

È possibile visualizzare i nodi e i carichi di lavoro Kubernetes.

1. Nel pannello di navigazione a sinistra, seleziona Cluster. Nell'elenco dei cluster, scegli il nome del cluster che hai creato, ad esempio*my-cluster*.

2. Nella ***my-cluster***pagina, scegli quanto segue:

a. Scheda Calcolo: verrà visualizzato l'elenco di nodi che sono stati implementati per il cluster. È possibile scegliere il nome di un nodo per visualizzare ulteriori informazioni su di esso.

b. Scheda Risorse: verranno visualizzate tutte le risorse Kubernetes che sono state implementate per impostazione predefinita in un cluster Amazon EKS. Seleziona qualsiasi tipo di risorsa nella console per avere ulteriori informazioni in merito.

Passaggio 5. Eliminazione delle risorse

Dopo aver creato il cluster e i nodi per questo tutorial, devi eliminare le risorse create. Per eseguire altre operazioni con questo cluster prima di eliminare le risorse, consultare the section called "Fasi successive".

1. Eliminare qualsiasi gruppo di nodi o i profili creati.
   a. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/home#/clusters.
   b. Nel pannello di navigazione a sinistra, seleziona Cluster. Nell'elenco dei cluster, scegli *my-cluster*.
   c. Scegli la scheda Calcolo.
   d. Se hai creato un gruppo di nodi, scegli il gruppo di ***my-nodegroup*** nodi, quindi scegli Elimina. Immettere *my-nodegroup*, quindi scegliere Delete (Elimina).
   e. Non continuare fino a quando il gruppo di nodi o i profili non saranno stati eliminati.
2. Elimina il cluster.
   a. Nel pannello di navigazione a sinistra, seleziona Cluster. Nell'elenco dei cluster, scegli *my-cluster*.
   b. Scegli Delete cluster (Elimina cluster).
   c. Immetti *my-cluster*, quindi scegli Elimina. Non continuare fino a quando il cluster non sarà stato eliminato.
3. Elimina lo AWS CloudFormation stack VPC che hai creato.
   a. Apri la CloudFormation console all'indirizzo. https://console.aws.amazon.com/cloudformation/
   b. Scegli lo stack *my-eks-vpc-stack*, quindi seleziona Delete (Elimina).
   c. Nella finestra di dialogo di ***my-eks-vpc-stack*** conferma dell'eliminazione, scegli Elimina pila.
4. Elimina i ruoli IAM creati.
   a. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
   b. Nel pannello di navigazione a sinistra, seleziona Ruoli.
   c. Seleziona ogni ruolo che hai creato dall'elenco (***myAmazonEKSClusterRole***e anche*myAmazonEKSNodeRole*). Scegli Elimina, inserisci il testo di conferma richiesto, quindi scegli Elimina.

Fasi successive

I seguenti argomenti della documentazione consentono di estendere la funzionalità del cluster.

- Il principale IAM in grado di creare il cluster è l'unico principale che può effettuare chiamate al server API Kubernetes tramite `kubectl` o la Console di gestione AWS. Se si desidera che altri principali IAM abbiano accesso al cluster, è necessario aggiungerli. Per ulteriori informazioni, consulta the section called "Accesso API Kubernetes" e the section called "Autorizzazioni richieste".
- Implementare un'applicazione di esempio al cluster.
- Prima di implementare un cluster da utilizzare in produzione, ti consigliamo di acquisire familiarità con tutte le impostazioni di cluster e nodi. Alcune impostazioni (come l'abilitazione dell'accesso SSH ai EC2 nodi Amazon) devono essere effettuate al momento della creazione del cluster.
- Per aumentare la sicurezza del cluster, configura il plugin Amazon VPC Container Networking Interface per utilizzare i ruoli IAM per gli account di servizio.

Automate cluster infrastructure with EKS Auto Mode

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS Auto Mode.

EKS Auto Mode estende la AWS gestione dei cluster Kubernetes oltre il cluster stesso, AWS per consentire anche la configurazione e la gestione dell'infrastruttura che consente il corretto funzionamento dei carichi di lavoro. Puoi delegare le decisioni chiave sull'infrastruttura e sfruttare l'esperienza di for operations. AWS day-to-day L'infrastruttura cluster gestita da AWS include molte funzionalità di Kubernetes come componenti principali, al contrario dei componenti aggiuntivi, come la scalabilità automatica delle risorse di elaborazione, la rete di pod e servizi, il bilanciamento del carico delle applicazioni, il DNS del cluster, lo storage a blocchi e il supporto per GPU.

Per iniziare, è possibile implementare un nuovo cluster con modalità automatica EKS o abilitare la modalità automatica EKS su un cluster esistente. Puoi distribuire, aggiornare o modificare i tuoi cluster EKS Auto Mode usando eksctl, la AWS CLI, EKS APIs o i tuoi strumenti Console di gestione AWS preferiti. infrastructure-as-code

Con la modalità automatica EKS, puoi continuare a usare gli strumenti compatibili con Kubernetes che preferisci. EKS Auto Mode si integra con AWS servizi come Amazon EC2, Amazon EBS ed ELB, AWS sfruttando risorse cloud che seguono le migliori pratiche. Queste risorse vengono dimensionate automaticamente, ottimizzate in termini di costi e aggiornate regolarmente per contribuire a ridurre al minimo i costi operativi e le spese generali.

Funzionalità

modalità automatica di EKS offre le seguenti funzionalità di alto livello:

Ottimizzazione della gestione dei cluster Kubernetes: la modalità automatica EKS semplifica la gestione di EKS fornendo cluster pronti per la produzione con un sovraccarico operativo minimo. Con la modalità automatica EKS, puoi eseguire carichi di lavoro impegnativi e dinamici in sicurezza, senza bisogno di una profonda conoscenza di EKS.

Disponibilità delle applicazioni: modalità automatica di EKS aggiunge o rimuove dinamicamente i nodi nel cluster EKS in base alle esigenze delle applicazioni Kubernetes. Questo riduce al minimo la necessità di pianificare manualmente la capacità e garantisce la disponibilità delle applicazioni.

Efficienza: EKS Auto Mode è progettata per ottimizzare i costi di elaborazione rispettando al contempo la flessibilità definita dai requisiti dell'utente e del carico di lavoro. NodePool Inoltre, termina le istanze non utilizzate e consolida i carichi di lavoro su altri nodi per migliorare l'efficienza in termini di costi.

Sicurezza: EKS Auto Mode utilizza AMIs dati considerati immutabili per i nodi. Questi applicano AMIs software bloccati, abilitano i controlli di accesso SELinux obbligatori e forniscono file system root di sola lettura. Inoltre, i nodi avviati dalla modalità automatica EKS hanno una durata massima di 21 giorni (che può essere ridotta), dopodiché vengono sostituiti automaticamente con nuovi nodi. Questo approccio migliora il livello di sicurezza grazie al riciclo regolare dei nodi, in linea con le best practice già adottate da molti clienti.

Aggiornamenti automatici: EKS Auto Mode mantiene il cluster Kubernetes, i nodi e i relativi componenti aggiornati con le patch più recenti, rispettando al contempo i Pod Disruption Budgets () e Disruption Budgets () configurati. PDBs NodePool NDBs Fino alla durata massima di 21 giorni, potrebbe essere necessario intervenire se il blocco o altre configurazioni impediscono gli aggiornamenti. PDBs

Componenti gestiti: EKS Auto Mode include Kubernetes e funzionalità AWS cloud come componenti principali che altrimenti dovrebbero essere gestiti come componenti aggiuntivi. Sono inclusi il supporto integrato per l'assegnazione degli indirizzi IP dei pod, le policy di rete dei pod, i servizi DNS locali, i componenti aggiuntivi GPU, i controlli di integrità e lo storage EBS CSI.

Personalizzabile NodePools e NodeClasses: se il carico di lavoro richiede modifiche alle configurazioni di archiviazione, elaborazione o rete, è possibile creare configurazioni personalizzate utilizzando la modalità automatica EKS. NodePools NodeClasses Sebbene non sia necessario modificare le impostazioni predefinite NodePools NodeClasses, è possibile aggiungere nuove configurazioni personalizzate NodePools o NodeClasses affiancarle a quelle predefinite per soddisfare i requisiti specifici.

Componenti automatici

La modalità automatica EKS semplifica il funzionamento dei cluster Amazon EKS automatizzando i componenti chiave dell'infrastruttura. L'attivazione della modalità automatica EKS riduce ulteriormente le attività di gestione dei cluster EKS.

Di seguito è riportata una lista di componenti del piano dati che sono automatizzati:

- Calcolo: per molti carichi di lavoro, con la modalità automatica EKS puoi dimenticarti di molti aspetti relativi al calcolo per i cluster EKS. Ciò include:
  - Nodi: i nodi della modalità automatica EKS sono progettati per essere trattati come dispositivi. La modalità automatica EKS esegue le seguenti operazioni:
    - Sceglie un'AMI appropriata configurata con molti servizi necessari per eseguire i carichi di lavoro senza bisogno di intervento.
    - Blocca l'accesso ai file sull'AMI utilizzando la modalità di SELinux applicazione e un file system root di sola lettura.
    - Impedisce l'accesso diretto ai nodi disabilitando l'accesso SSH o SSM.
    - Include il supporto GPU, con driver del kernel e plugin separati per NVIDIA e Neuron, che consente carichi di lavoro ad alte prestazioni. GPUs
    - Gestisce automaticamente gli avvisi di interruzione delle istanze Spot EC2 e gli eventi di integrità delle istanze EC2
  - Dimensionamento automatico: basandosi sul dimensionamento automatico di Karpenter, la modalità automatica EKS monitora i Pod non programmabili e consente l'implementazione di nuovi nodi per eseguire tali pod. Quando i carichi di lavoro vengono terminati, la modalità automatica EKS interrompe e termina in modo dinamico i nodi quando non sono più necessari, ottimizzando l'uso delle risorse.
  - Aggiornamenti: il controllo dei nodi semplifica la capacità della modalità automatica EKS di offrire patch di sicurezza e aggiornamenti del sistema operativo e dei componenti in base alle esigenze. Tali aggiornamenti sono progettati per ridurre al minimo le interruzioni dei carichi di lavoro. EKS Auto Mode impone una durata massima dei nodi di 21 giorni per garantire software e. up-to-date APIs
- Bilanciamento del carico: modalità automatica di EKS semplifica il bilanciamento del carico integrandosi con il servizio Elastic Load Balancing di Amazon, automatizzando il provisioning e la configurazione dei bilanciatori di carico per i servizi Kubernetes e le risorse Ingress. Supporta funzionalità avanzate per Application e Network Load Balancer, ne gestisce il ciclo di vita e le dimensiona in base alle esigenze del cluster. Questa integrazione fornisce una soluzione di bilanciamento del carico pronta per la produzione conforme alle AWS migliori pratiche, che consente di concentrarsi sulle applicazioni anziché sulla gestione dell'infrastruttura.
- Storage: modalità automatica di EKS configura automaticamente lo storage temporaneo impostando i tipi di volume, le dimensioni dei volumi, le policy di crittografia e le policy di eliminazione alla chiusura del nodo.

- Rete: modalità automatica di EKS automatizza le attività di rete critiche per quanto riguarda la connettività dei pod e dei servizi. Ciò include IPv4/IPv6 support e l'uso di blocchi CIDR secondari per estendere gli spazi degli indirizzi IP.
- Identity and Access Management: non è necessario installare l'agente EKS Pod Identity sui cluster in modalità automatica di EKS.

Per ulteriori informazioni su questi componenti, consulta the section called "Come funziona".

Configurazione

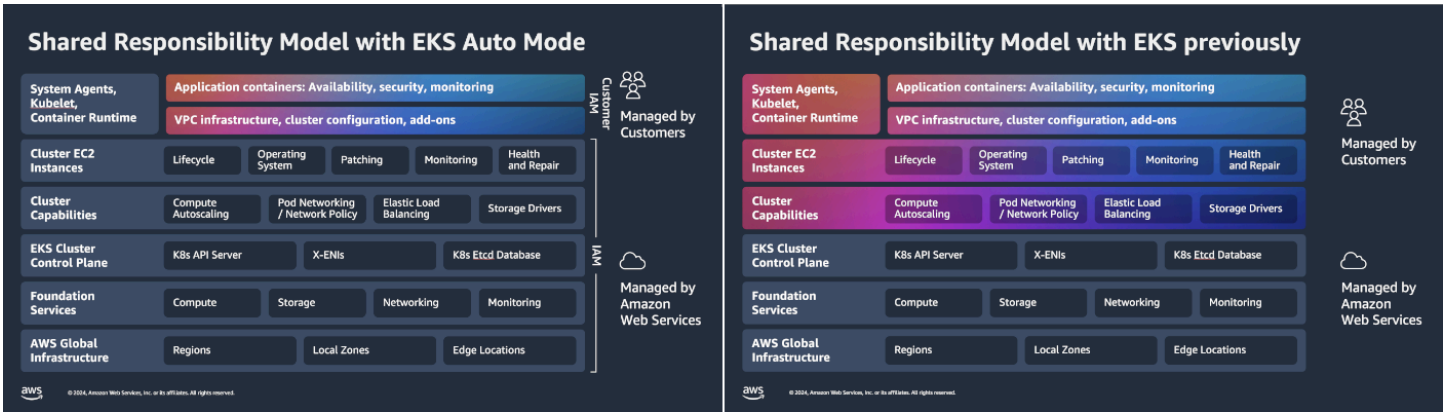
Sebbene modalità automatica di EKS gestisca in modo efficace la maggior parte dei servizi del piano dati senza alcun intervento da parte tua, potrebbero esserci casi in cui desideri modificare il comportamento di alcuni di questi servizi. Puoi modificare la configurazione dei cluster della modalità automatica di EKS nei seguenti modi:

- Kubernetes DaemonSets: anziché modificare i servizi installati sui nodi, puoi invece utilizzare i daemonset Kubernetes. I daemonset sono progettati per essere gestiti da Kubernetes, ma vengono eseguiti su ogni nodo del cluster. In questo modo, puoi aggiungere servizi speciali per il monitoraggio o la sorveglianza dei nodi.
- Personalizzati NodePools e NodeClasses: sono predefiniti e NodeClasses sono configurati da EKS Auto Mode NodePools e non è necessario modificarli. Per personalizzare il comportamento dei nodi, puoi creare casi NodeClasses aggiuntivi NodePools o da utilizzare come:
  - La selezione di tipi di istanze specifici (ad esempio, processori accelerati o istanze Spot EC2).
  - L'isolamento dei carichi di lavoro per scopi di sicurezza o di monitoraggio dei costi.
  - La configurazione di impostazioni di storage temporaneo come IOPS, dimensioni e throughput.
- Bilanciamento del carico: alcuni servizi, come il bilanciamento del carico, che modalità automatica di EKS esegue come oggetti Kubernetes, possono essere configurati direttamente sui cluster della modalità automatica di EKS.

Per ulteriori informazioni sulle opzioni di configurazione della modalità automatica di EKS, consulta the section called "Configura".

Modello di responsabilità condivisa

Il modello di responsabilità AWS condivisa definisce le responsabilità di sicurezza AWS e conformità tra i clienti. Le immagini e il testo seguenti mettono a confronto e contrappongono AWS le differenze tra cliente e responsabilità tra la modalità EKS Auto e la modalità standard EKS.



La modalità automatica di EKS trasferisce gran parte della responsabilità condivisa per l'infrastruttura Kubernetes dai clienti ad AWS. Con EKS Auto Mode, AWS si assume maggiori responsabilità per la sicurezza del cloud, che una volta era responsabilità del cliente e ora è condivisa. I clienti possono ora concentrarsi maggiormente sulle proprie applicazioni AWS gestendo al contempo l'infrastruttura sottostante.

Responsabilità del cliente

Con la modalità automatica di EKS, i clienti continuano a mantenere la responsabilità dei container delle applicazioni, inclusi disponibilità, sicurezza e monitoraggio. Mantengono inoltre il controllo sull'infrastruttura VPC e sulla configurazione del cluster EKS. Questo modello consente ai clienti di concentrarsi su problemi specifici delle applicazioni delegando al contempo la gestione dell'infrastruttura del cluster a. AWS Le funzionalità opzionali per nodo possono essere incluse nei cluster tramite componenti aggiuntivi. AWS

AWS responsabilità

Con EKS Auto Mode, AWS amplia la propria responsabilità includendo la gestione di diversi componenti critici aggiuntivi rispetto a quelli già gestiti nei cluster EKS che non utilizzano la modalità automatica. In particolare, modalità automatica di EKS si occupa della configurazione, della gestione, della sicurezza e del dimensionamento delle istanze EC2 avviate, nonché delle funzionalità del cluster per il bilanciamento del carico, la gestione degli indirizzi IP, la policy di rete e lo storage a blocchi. I seguenti componenti sono gestiti da AWS EKS Auto Mode:

- Istanze EC2 con lancio automatico in modalità: AWS gestisce l'intero ciclo di vita dei nodi sfruttando le istanze gestite di Amazon EC2. Le istanze gestite da EC2 si assumono la responsabilità della configurazione del sistema operativo, dell'applicazione di patch, del monitoraggio e del mantenimento dell'integrità. In questo modello, sia l'istanza stessa che il sistema operativo guest in esecuzione su di essa sono di responsabilità di AWS. I nodi utilizzano varianti di Bottlerocket ottimizzate per l'esecuzione di contenitori. AMIs I Bottlerocket AMIs dispongono di software bloccato, file system root immutabili e accesso sicuro alla rete (per impedire comunicazioni dirette tramite SSH o SSM).
- Funzionalità del cluster: AWS gestisce la scalabilità automatica dell'elaborazione, il networking Pod con l'applicazione delle policy di rete, l'integrazione di Elastic Load Balancing e la configurazione dei driver di archiviazione.
- Cluster Control Plane: AWS continua a gestire il server API Kubernetes, il cross-account e il database etcd, come con EKS ENIs standard.
- Foundation Services and Global Infrastructure: AWS è responsabile dei servizi di elaborazione, archiviazione, rete e monitoraggio sottostanti, nonché dell'infrastruttura globale di regioni, zone locali e edge location.

Creare un cluster con la modalità automatica Amazon EKS

Questo capitolo spiega come creare un cluster Amazon EKS con la modalità automatica abilitata utilizzando vari strumenti e interfacce. La modalità automatica semplifica la creazione di cluster grazie alla configurazione e alla gestione automatiche dell'infrastruttura di calcolo, della rete e dell'archiviazione del cluster. Si imparerà come creare un cluster con la modalità automatica utilizzando la AWS CLI, la Console di gestione AWS o lo strumento a riga di comando eksctl.

> **Note**
>
> La modalità automatica EKS richiede la versione 1.29 o successive di Kubernetes.

Scegliere lo strumento preferito in base alle esigenze: la Console di gestione AWS offre un'interfaccia visiva ideale per scoprire le funzionalità della modalità automatica EKS e creare singoli cluster. La AWS CLI è la soluzione ideale per attività di scripting e automazione, in particolare quando si integra la creazione di cluster in flussi di lavoro o pipeline CI/CD esistenti. Lo strumento CLI eksctl offre un'esperienza nativa per Kubernetes ed è consigliato agli utenti che hanno familiarità con gli

strumenti Kubernetes e che desiderano operazioni a riga di comando semplificate con impostazioni predefinite ragionevoli.

Prima di iniziare, assicurarsi di aver installato e configurato i prerequisiti necessari, incluse le autorizzazioni IAM adeguate per creare cluster EKS. Per ulteriori informazioni su come installare strumenti CLI come `kubectl`, `aws` e `eksctl`, consultare Configurazione.

È possibile utilizzare la AWS CLI, la Console di gestione AWS o lo strumento CLI eksctl per creare un cluster con la modalità automatica Amazon EKS.

Argomenti

- Crea un cluster di EKS Auto Mode con la CLI di eksctl
- Crea un cluster EKS Auto Mode con la AWS CLI
- Crea un cluster di EKS Auto Mode con la Console di gestione AWS

Crea un cluster di EKS Auto Mode con la CLI di eksctl

Questo argomento ti mostra come creare un cluster di Amazon EKS Auto Mode con l'interfaccia a riga di comando (CLI) di eksctl. Puoi creare un cluster Auto Mode eseguendo un singolo comando CLI o applicando un file di configurazione YAML. Entrambi i metodi offrono le stesse funzionalità, con l'approccio YAML che offre un controllo più granulare sulle impostazioni del cluster.

La CLI di eksctl semplifica il processo di creazione e gestione dei cluster di EKS Auto Mode gestendo la creazione e la configurazione delle risorse AWS sottostanti. Prima di procedere, assicurati di avere le credenziali AWS e le autorizzazioni necessarie configurate sul computer locale. Questa guida presuppone che tu abbia una certa familiarità con i concetti base di Amazon EKS e che gli strumenti CLI richiesti siano già stati installati.

> **Note**
>
> Devi installare la versione `0.195.0` o una versione successiva di eksctl. Per ulteriori informazioni, consulta eksctl releases su GitHub.

Crea un cluster di EKS Auto Mode con un comando CLI

Devi avere l'installazione degli strumenti `aws` ed `eksctl`. Devi aver effettuato l'accesso ad AWS CLI con autorizzazioni sufficienti per gestire le risorse AWS tra cui: istanze EC2, reti EC2, cluster EKS e ruoli IAM. Per ulteriori informazioni, consulta Configurazione.

Esegui il seguente comando per creare un nuovo cluster di EKS Auto Mode con

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

Crea un cluster di EKS Auto Mode con un file YAML

Devi avere l'installazione degli strumenti `aws` ed `eksctl`. Devi aver effettuato l'accesso ad AWS CLI con autorizzazioni sufficienti per gestire le risorse AWS tra cui: istanze EC2, reti EC2, cluster EKS e ruoli IAM. Per ulteriori informazioni, consulta Configurazione.

Esamina le opzioni di configurazione di EKS Auto Mode nella risorsa ClusterConfig di esempio qui sotto. Per la specifica completa di ClusterConfig, consulta eksctl documentation.

AWS suggerisce di abilitare EKS Auto Mode. Se è la prima volta che crei un cluster di EKS Auto Mode, lascia il `nodeRoleARN` non precisato per creare un ruolo IAM del nodo per EKS Auto Mode. Se disponi già di un ruolo IAM del nodo nell'account AWS, AWS suggerisce di riutilizzarlo.

AWS suggerisce di non specificare alcun valore per `nodePools`. EKS Auto Mode creerà pool di nodi predefiniti. Puoi utilizzare l'API Kubernetes per creare pool di nodi aggiuntivi.

```
# 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
```

Salva il file `ClusterConfig` con nome `cluster.yaml` e usa il seguente comando per creare il cluster:

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

Crea un cluster EKS Auto Mode con la AWS CLI

I cluster della modalità automatica di EKS automatizzano le attività di routine di gestione del cluster l'elaborazione, l'archiviazione e la rete. Ad esempio, i cluster EKS Auto Mode rilevano automaticamente quando sono necessari nodi aggiuntivi e forniscono nuove EC2 istanze per soddisfare le richieste del carico di lavoro.

Questo argomento illustra la creazione di un nuovo cluster EKS Auto Mode utilizzando la AWS CLI e, facoltativamente, la distribuzione di un carico di lavoro di esempio.

Prerequisiti

- La versione più recente dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo. Per verificare la versione attuale, usa `aws --version`. Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando.
  - Accedi alla CLI con autorizzazioni IAM sufficienti per creare AWS risorse tra cui politiche IAM, ruoli IAM e cluster EKS.
- Lo strumento da riga di comando kubectl installato sul tuo dispositivo. AWS suggerisce di utilizzare la stessa versione di kubectl della versione Kubernetes del cluster EKS. Per installare o aggiornare kubectl, consulta the section called "Configura kubectl ed eksctl".

Specifica le sottoreti VPC

La modalità automatica di Amazon EKS implementa i nodi nelle sottoreti VPC. Quando si crea un cluster EKS, devi specificare le sottoreti VPC in cui verranno implementati i nodi. Puoi utilizzare le sottoreti VPC predefinite nel tuo AWS account o creare un VPC dedicato per carichi di lavoro critici.

- AWS suggerisce di creare un VPC dedicato per il cluster. Informazioni su come the section called "Crea un VPC".
- La console EKS aiuta a creare un nuovo VPC. Informazioni su come the section called "Console di gestione".
- In alternativa, puoi utilizzare il VPC predefinito del tuo AWS account. Usa le seguenti istruzioni per trovare la IDs sottorete.

Per trovare la sottorete IDs del tuo VPC predefinito

Utilizzo della AWS CLI:

1. Esegui il comando seguente per elencare il VPC di default e le relative sottoreti:

```
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. Salva l'output e annota la IDssottorete.

   Output di esempio:

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

Ruoli IAM per cluster della modalità automatica di EKS

Ruolo IAM del cluster

EKS Auto Mode richiede un ruolo Cluster IAM per eseguire azioni nell' AWS account, come il provisioning di nuove EC2 istanze. È necessario creare questo ruolo per concedere a EKS le autorizzazioni necessarie. AWS consiglia di allegare le seguenti politiche AWS gestite al ruolo Cluster IAM:

- EKSComputePolitica di Amazon
- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- EKSNetworkingPolitica di Amazon
- EKSClusterPolitica di Amazon

Ruolo IAM del nodo

Quando creai un cluster della modalità automatica di EKS, specifichi un ruolo IAM del nodo. Quando EKS Auto Mode crea nodi per elaborare carichi di lavoro in sospeso, a ogni nuovo nodo di EC2 istanza viene assegnato il ruolo Node IAM. Questo ruolo consente al nodo di comunicare con EKS, ma in genere non è accessibile ai carichi di lavoro in esecuzione sul nodo.

Se desideri concedere le autorizzazioni ai carichi di lavoro in esecuzione su un nodo, usa EKS Pod Identity. Per ulteriori informazioni, consulta the section called “Pod Identity”.

È necessario creare questo ruolo e allegare la seguente politica AWS gestita:

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

Ruolo collegato ai servizi

La modalità automatica di EKS richiede anche un ruolo collegato ai servizi, che viene creato e configurato automaticamente da AWS. Per ulteriori informazioni, consulta AWSServiceRoleForAmazonEKS.

Creazione di un ruolo IAM per un cluster della modalità automatica di EKS

Passaggio 1: creazione della policy di attendibilità

Crea una policy di attendibilità che consenta al servizio Amazon EKS di assumere tale ruolo. Salva la policy come `trust-policy.json`:

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

Passaggio 2: creazione del ruolo IAM

Usa la policy di attendibilità per creare il ruolo IAM del cluster:

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

Passaggio 3: annotazione del ruolo ARN

Recupera e salva l'ARN del nuovo ruolo per usarlo nelle fasi successive:

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

Passaggio 4: collegamento delle policy obbligatorie

Allega le seguenti politiche AWS gestite al ruolo Cluster IAM per concedere le autorizzazioni necessarie:

EKSClusterPolitica di Amazon:

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

EKSComputePolitica di Amazon:

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

Amazon EKSBlock StoragePolicy:

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

Amazon EKSLoad BalancingPolicy:

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

EKSNetworkingPolitica di Amazon:

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

Creazione di un ruolo IAM per un nodo modalità automatica di EKS

Passaggio 1: creazione della policy di attendibilità

Crea una policy di attendibilità che consenta al servizio Amazon EKS di assumere tale ruolo. Salva la policy come `node-trust-policy.json`:

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

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

Passaggio 2: creare il ruolo IAM del nodo

Utilizza il node-trust-policyfile.json del passaggio precedente per definire quali entità possono assumere il ruolo. Per creare il ruolo IAM del nodo, esegui il seguente comando:

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

Passaggio 3: annotazione del ruolo ARN

Una volta creato il ruolo, recupera e salva l'ARN del ruolo IAM del nodo. Sarà necessario usare questo ARN nei passaggi successivi. Usa il seguente comando per ottenere l'ARN:

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

Passaggio 4: collegamento delle policy obbligatorie

Allega le seguenti policy AWS gestite al ruolo Node IAM per fornire le autorizzazioni necessarie:

Amazon EKSWorker NodeMinimalPolicy:

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

Amazon EC2 ContainerRegistryPullOnly:

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

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

Creazione di un cluster della modalità automatica di EKS

Panoramica di

Per creare un cluster EKS Auto Mode utilizzando la AWS CLI, sono necessari i seguenti parametri:

- `cluster-name`: il nome del cluster.
- `k8s-version`: la versione di Kubernetes (ad esempio, 1.31).
- `subnet-ids`: sottorete IDs identificata nei passaggi precedenti.
- `cluster-role-arn`: ARN del ruolo IAM del cluster.
- `node-role-arn`: ARN del ruolo IAM del nodo.

Configurazioni predefinite del cluster

Esaminare questi valori e funzionalità predefiniti prima di creare il cluster:

- `nodePools`: modalità automatica di EKS include pool di nodi generici e predefiniti di sistema. Maggiori informazioni sui Node Pools.

Nota: i pool di nodi in modalità automatica di EKS differiscono dai gruppi di nodi gestiti di Amazon EKS ma possono coesistere nello stesso cluster.

- `computeConfig.enabled`: automatizza le attività di calcolo di routine, come la creazione e l'eliminazione di istanze. EC2
- `kubernetesNetworkConfig.elasticLoadBalancing.enabled`: automatizza le attività di bilanciamento del carico, includendo la creazione e l'eliminazione dei bilanciatori del carico elastico.
- `storageConfig.blockStorage.enabled`: automatizza le attività di archiviazione, come la creazione e l'eliminazione di volumi Amazon EBS.
- `accessConfig.authenticationMode`: richiede voci di accesso EKS. Maggiori informazioni su EKS authentication modes.

Esegui il comando

Usa il comando seguente per creare il 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\"
      }
    }
```

Verifica lo stato del cluster

Fase 1: verificare la creazione del cluster

Esegui il comando seguente per verificare lo stato del cluster. La creazione del cluster richiede in genere 15 minuti.

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

Fase 2: aggiornare kubeconfig

Una volta che il cluster è pronto, aggiorna il file kubeconfig locale per consentire a `kubectl` di comunicare con il cluster. Questa configurazione utilizza la AWS CLI per l'autenticazione.

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

Fase 3: verificare i pool di nodi

Elenca i pool di nodi nel cluster con il comando seguente:

```
kubectl get nodepools
```

Fasi successive

- Informazioni su deploy a sample workload nel nuovo cluster della modalità automatica di EKS.

Crea un cluster di EKS Auto Mode con la Console di gestione AWS

La creazione di un cluster di EKS Auto Mode nella Console di gestione AWS richiede meno configurazione rispetto ad altre opzioni. EKS si integra con AWS IAM e le reti VPC per aiutarti a creare le risorse associate a un cluster EKS.

Hai due opzioni per creare un cluster nella console:

- Configurazione rapida (con EKS Auto Mode)
- Configurazione personalizzata

Questo argomento illustra come creare un cluster di EKS Auto Mode utilizzando l'opzione di configurazione rapida.

Crea un EKS Auto Mode utilizzando l'opzione di configurazione rapida

Deve aver effettuato l'accesso a Console di gestione AWS con autorizzazioni sufficienti per gestire risorse AWS tra cui: istanze EC2, reti EC2, cluster EKS e ruoli IAM.

1. Passa alla console EKS
2. Fai clic su Crea cluster.
3. Conferma che l'opzione di configurazione rapida è selezionata

4. Determina i seguenti valori o usare i valori predefiniti per un cluster di test.
   - Nome del cluster
   - Versione di Kubernetes
5. Seleziona il ruolo IAM del cluster. Se è la prima volta che hai creato un cluster di EKS Auto Mode, utilizza l'opzione Crea ruolo consigliato.
   - Facoltativamente, puoi riutilizzare un singolo ruolo IAM del cluster nell'account AWS per tutti i cluster di EKS Auto Mode.
   - Il ruolo IAM del cluster include le autorizzazioni necessarie affinché EKS Auto Mode gestisca le risorse, tra cui istanze EC2, volumi EBS e bilanciatori del carico EC2.
   - L'opzione Crea ruolo consigliato precompila tutti i campi con i valori consigliati. Seleziona Avanti e poi Crea. Il ruolo utilizzerà il nome `AmazonEKSAutoClusterRole` suggerito.
   - Se recentemente hai creato un nuovo ruolo, usa l'icona Aggiorna per ricaricare il menu a discesa di selezione del ruolo.
6. Seleziona il ruolo IAM del nodo. Se è la prima volta che hai creato un cluster di EKS Auto Mode, utilizza l'opzione Crea ruolo consigliato.
   - Facoltativamente, puoi riutilizzare un singolo ruolo IAM del nodo nell'account AWS per tutti i cluster di EKS Auto Mode.
   - Il ruolo IAM del nodo include le autorizzazioni necessarie per la connessione dei nodi Auto Mode al cluster. Il ruolo IAM del nodo deve includere le autorizzazioni per recuperare le immagini ECR per i container.
   - L'opzione Crea ruolo consigliato precompila tutti i campi con i valori consigliati. Seleziona Avanti e poi Crea. Il ruolo utilizzerà il nome `AmazonEKSAutoNodeRole` suggerito.
   - Se recentemente hai creato un nuovo ruolo, usa l'icona Aggiorna per ricaricare il menu a discesa di selezione del ruolo.
7. Seleziona il VPC per il cluster di EKS Auto Mode. Scegli Crea VPC per creare un nuovo VPC per EKS oppure scegli un VPC creato in precedenza per EKS.
   - Se utilizzi la console VPC per creare un nuovo VPC, AWS suggerisce di creare almeno un gateway NAT per zona di disponibilità. In caso contrario, puoi usare tutte le altre impostazioni predefinite.
   - Per ulteriori informazioni e dettagli sui requisiti del cluster IPv6, consulta the section called "Crea un VPC".
8. (opzionale) EKS Auto Mode popola automaticamente le sottoreti private per il VPC selezionato. Puoi rimuovere le sottoreti indesiderate.

- EKS seleziona automaticamente le sottoreti private dal VPC seguendo le migliori pratiche. Facoltativamente, puoi selezionare sottoreti aggiuntive dal VPC, come quelle pubbliche.

9. (facoltativo) Seleziona Visualizza impostazioni predefinite di configurazione rapida per esaminare tutti i valori di configurazione per il nuovo cluster. La tabella indica che alcuni valori non sono modificabili dopo la creazione del cluster.
10. Selezionare Creazione di un cluster. Il completamento della creazione del cluster potrebbe richiedere quindici minuti.

Fasi successive

- Informazioni su Deploy a Sample Workload to your EKS Auto Mode cluster

Abilita EKS Auto Mode sui cluster EKS esistenti

Puoi abilitare EKS Auto Mode sui cluster EKS esistenti.

AWS supporta le seguenti migrazioni:

- Migrazione da Karpenter ai nodi EKS Auto Mode. Per ulteriori informazioni, consulta the section called "Eseguire la migrazione da Karpenter".
- Migrazione dai gruppi di nodi gestiti EKS ai nodi EKS Auto Mode. Per ulteriori informazioni, consulta the section called "Eseguire la migrazione da MNG".
- Migrazione da EKS Fargate alla EKS Auto Mode. Per ulteriori informazioni, consulta the section called "Eseguire la migrazione da Fargate".

AWS non supporta le seguenti funzionalità:

- Migrazione di volumi dal controller CSI EBS (utilizzando il componente aggiuntivo Amazon EKS) al controller CSI EBS di EKS Auto Mode (gestito da EKS Auto Mode). I PVC realizzati con uno non possono essere montati con l'altro, perché utilizzano due diversi provider di volumi Kubernetes.
  - Il `eks-auto-mode-ebs-migration-tool` (progetto AWS Labs) consente la migrazione tra StorageClass CSI EBS standard (`ebs.csi.aws.com`) ed EKS StorageClass CSI EBS automatica di EKS (`ebs.csi.eks.amazonaws.com`). Tieni presente che la migrazione richiede l'eliminazione e la ricreazione delle risorse PersistentVolumeClaim/PersistentVolume esistenti, pertanto la convalida in un ambiente non di produzione è essenziale prima dell'implementazione.

- Migrazione dei sistemi di bilanciamento del carico da AWS Load Balancer Controller a EKS Auto Mode

  Puoi installare AWS Load Balancer Controller su un cluster Amazon EKS Auto Mode. Utilizza le opzioni `IngressClass` o `loadBalancerClass` per associare le risorse Service e Ingress al Load Balancer Controller o a EKS Auto Mode.
- Migrazione di cluster EKS con CNI alternativi o altre configurazioni di rete non supportate

Materiale di riferimento per le migrazioni

Utilizza il seguente riferimento sulla migrazione per configurare le risorse Kubernetes in modo che siano di proprietà di controller autogestiti o EKS Auto Mode.

Funzionalità	Risorsa	Campo	Autogestito	EKS Auto Mode
Storage a blocchi	`StorageClass`	`provisioner`	`ebs.csi.aws.com`	`ebs.csi.eks.amazonaws.com`
Sistema di bilanciamento del carico	`Service`	`loadBalancerClass`	`service.k8s.aws/nlb`	`eks.amazonaws.com/nlb`
Sistema di bilanciamento del carico	`IngressClass`	`controller`	`ingress.k8s.aws/alb`	`eks.amazonaws.com/alb`
Sistema di bilanciamento del carico	`IngressClassParams`	`apiversion`	`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`
Sistema di bilanciamento del carico	`TargetGroupBinding`	`apiversion`	`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`
Calcolo	`NodeClass`	`apiVersion`	`karpenter.sh/v1`	`eks.amazonaws.com/v1`

Migration di volumi EBS

Durante la migrazione dei carichi di lavoro a EKS Auto Mode, è necessario gestire la migrazione dei volumi EBS a causa di diversi provisioner di driver CSI:

- Provisioner EKS Auto Mode: `ebs.csi.eks.amazonaws.com`
- Provisioner CSI EBS open source: `ebs.csi.aws.com`

Segui questi passaggi per migrare i volumi persistenti:

1. Modifica la policy di conservazione dei volumi: modifica la versione della piattaforma esistente `persistentVolumeReclaimPolicy` in `Retain` per garantire che il volume EBS sottostante non venga eliminato.
2. Rimuovi la versione della piattaforma da Kubernetes: elimina la vecchia risorsa della versione mantenendo intatto il volume EBS effettivo.
3. Crea una nuova versione della piattaforma con provisioning statico: crea una nuova versione che faccia riferimento allo stesso volume EBS ma funzioni con il driver CSI di destinazione.
4. Associa a un nuovo PVC: crea un nuovo PVC che faccia riferimento in modo specifico alla versione della tua piattaforma utilizzando il campo `volumeName`.

Considerazioni

- Assicurati che le tue applicazioni siano interrotte prima di iniziare questa migrazione.
- Esegui il backup dei dati prima di iniziare il processo di migrazione.
- Questo processo deve essere eseguito per ogni volume persistente.
- Il carico di lavoro deve essere aggiornato per utilizzare il nuovo PVC.

Migrazione dei bilanciatori del carico

Non è possibile trasferire direttamente i bilanciatori del carico esistenti da AWS Load Balancer Controller autogestito a EKS Auto Mode. Devi invece implementare una strategia di implementazione blu/verde. Questo comporta il mantenimento della configurazione del bilanciatore del carico esistente e la creazione di nuovi sistemi di bilanciamento del carico sotto il controller gestito.

Per ridurre al minimo le interruzioni del servizio, consigliamo un approccio di trasferimento del traffico basato su DNS. Innanzitutto, crea nuovi sistemi di bilanciatore del carico utilizzando EKS Auto Mode

e mantenendo operativa la configurazione esistente. Quindi, utilizza il routing DNS (come Route 53) per spostare gradualmente il traffico dai vecchi sistemi di bilanciatore del carico a quelli nuovi. Una volta completata la migrazione del traffico e verificata la nuova configurazione, puoi disattivare i vecchi sistemi di bilanciatore del carico e il controller autogestito.

Abilita modalità automatica di EKS su un cluster esistente

In questo argomento viene descritto come abilitare la modalità automatica di Amazon EKS sui cluster Amazon EKS esistenti. L'attivazione di Auto Mode su un cluster esistente richiede l'aggiornamento delle autorizzazioni IAM e la configurazione delle impostazioni principali della modalità automatica di EKS. Dopo l'abilitazione, puoi iniziare a eseguire la migrazione dei carichi di lavoro di elaborazione esistenti per sfruttare le operazioni semplificate e la gestione automatizzata dell'infrastruttura di Auto Mode.

> **Important**
>
> Verifica di aver installato la versione minima richiesta di alcuni componenti aggiuntivi di Amazon EKS prima di abilitare la modalità automatica di EKS. Per ulteriori informazioni, consulta the section called "Versioni richieste del componente aggiuntivo".

Prima di iniziare, assicurati di disporre dell'accesso da amministratore al cluster Amazon EKS e delle autorizzazioni per modificare i ruoli IAM. I passaggi descritti in questo argomento illustrano come abilitare la modalità automatica utilizzando Console di gestione AWS o la AWS CLI.

Console di gestione AWS

È necessario accedere alla AWS console con l'autorizzazione a gestire IAM, EKS e EC2 le risorse.

> **Note**
>
> Il ruolo IAM del cluster di un cluster EKS non può essere modificato dopo la creazione del cluster. modalità automatica di EKS richiede autorizzazioni aggiuntive per questo ruolo. Devi allegare policy aggiuntive al ruolo attuale.

Aggiorna il ruolo IAM del cluster

1. Apri la pagina di panoramica del cluster in Console di gestione AWS.

2. In ARN del ruolo IAM del cluster, seleziona Visualizza in IAM.
3. Dall'elenco a discesa Aggiungi autorizzazioni, scegli Allega policy.
4. Utilizza la casella Ricerca per trovare e selezionare le seguenti policy:
   - `AmazonEKSComputePolicy`
   - `AmazonEKSBlockStoragePolicy`
   - `AmazonEKSLoadBalancingPolicy`
   - `AmazonEKSNetworkingPolicy`
   - `AmazonEKSClusterPolicy`
5. Seleziona Aggiungi autorizzazioni.
6. Dalla scheda Relazioni di attendibilità, seleziona Modifica policy di attendibilità
7. Inserisci la seguente policy di attendibilità del ruolo IAM del cluster e seleziona Aggiorna policy

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

Abilita modalità automatica di EKS

1. Apri la pagina di panoramica del cluster in Console di gestione AWS.
2. In modalità automatica di EKS, seleziona Gestisci
3. Attiva modalità automatica di EKS.
4. Dal menu a discesa Pool di nodi EKS, seleziona i pool di nodi predefiniti da creare.
   - Maggiori informazioni sui pool di nodi in modalità automatica di EKS. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

5. Se in precedenza hai creato un ruolo IAM EKS Auto Mode Node su questo AWS account, selezionalo nel menu a discesa Node IAM Role. Se non hai mai creato questo ruolo in precedenza, seleziona Crea ruolo consigliato e seguire i passaggi.

AWS CLI

Prerequisiti

- Il ruolo IAM del cluster EKS esistente deve includere autorizzazioni sufficienti per la modalità automatica di EKS, come le seguenti policy:
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicy`
  - `AmazonEKSLoadBalancingPolicy`
  - `AmazonEKSNetworkingPolicy`
  - `AmazonEKSClusterPolicy`
- Il ruolo IAM del cluster deve avere una policy di attendibilità aggiornata che includa l'operazione `sts:TagSession`. Per ulteriori informazioni sulla creazione di un ruolo IAM del cluster, consulta the section called " AWS CLI".
- `aws` CLI installata, con accesso effettuato e una versione sufficiente. È necessario disporre dell'autorizzazione per gestire IAM, EKS e EC2 risorse. Per ulteriori informazioni, consulta Configurazione.

Procedura

Utilizza i seguenti comandi per abilitare modalità automatica di EKS su un cluster esistente.

> **Note**
>
> Le funzionalità di calcolo, archiviazione a blocchi e bilanciamento del carico devono essere tutte abilitate o disabilitate nella stessa richiesta.

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

```
  --storage-config '{"blockStorage":{"enabled": true}}'
```

Versioni richieste del componente aggiuntivo

Se intendi abilitare modalità automatica di EKS su un cluster esistente, potrebbe dover aggiornare alcuni componenti aggiuntivi. Nota:

- Questo vale solo per i cluster esistenti che passano alla modalità automatica di EKS .
- I nuovi cluster creati con la modalità automatica di EKS abilitata non richiedono questi aggiornamenti.

Se hai installato uno dei seguenti componenti aggiuntivi, assicurati che abbia almeno la versione minima specificata:

Nome del componente aggiuntivo	Versione minima richiesta
Plug-in della CNI di Amazon VPC per 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 CSI per Amazon EBS	v1.37.0-eksbuild.1
Controller di snapshot CSI	v8.1.0-eksbuild.2
EKS Pod Identity Agent	v1.3.4-eksbuild.1

Per ulteriori informazioni, consulta the section called "Aggiornamento di un componente aggiuntivo".

Fasi successive

- Per eseguire la migrazione dei carichi di lavoro di Manage Node Group, consulta the section called "Eseguire la migrazione da MNG".
- Per eseguire la migrazione da Self-Managed Karpenter, consulta the section called "Eseguire la migrazione da Karpenter".

Eseguire la migrazione da Karpenter alla modalità automatica di EKS utilizzando kubectl

Questo argomento illustra il processo di migrazione dei carichi di lavoro da Karpenter alla modalità automatica di Amazon EKS utilizzando kubectl. La migrazione può essere eseguita gradualmente, consentendo di spostare i carichi di lavoro al proprio ritmo, mantenendo al contempo la stabilità del cluster e la disponibilità delle applicazioni durante la transizione.

L' step-by-stepapproccio descritto di seguito consente di eseguire Karpenter ed EKS Auto Mode fianco a fianco durante il periodo di migrazione. Questa strategia a doppio funzionamento aiuta a garantire una transizione fluida consentendoti di convalidare il comportamento del carico di lavoro in modalità automatica di EKS prima di disattivare completamente Karpenter. Puoi eseguire la migrazione delle applicazioni singolarmente o in gruppo, garantendo la flessibilità necessaria per soddisfare i requisiti operativi specifici e la tolleranza al rischio.

Prerequisiti

Prima di iniziare la migrazione, assicurati di avere:

- Karpenter v1.1 o versioni successive deve essere installato sul cluster. Per ulteriori informazioni, consulta Upgrading to 1.1.0+ nei documenti di Karpenter.
- `kubectl` installato e connesso al cluster. Per ulteriori informazioni, consulta Configurazione.

Questo argomento presuppone che tu abbia familiarità con Karpenter e. NodePools Per ulteriori informazioni, consulta Karpenter Documentation.

Fase 1: abilitare modalità automatica di EKS sul cluster

Abilita la modalità automatica EKS sul tuo cluster esistente utilizzando la AWS CLI o la console di gestione. Per ulteriori informazioni, consulta the section called "Abilitazione sul cluster".

> **Note**
>
> Durante l'abilitazione della modalità automatica di EKS, non abilitare il nodepool `general purpose` in questa fase durante la transizione. Questo pool di nodi non è selettivo.
> Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".

Fase 2: Creare una modalità automatica EKS alterata NodePool

Crea una nuova modalità NodePool per EKS Auto con una macchia. Ciò garantisce che i pod esistenti non vengano programmati automaticamente sui nuovi nodi della modalità automatica di EKS. Questo pool di nodi utilizza `NodeClass` integrato `default` in modalità automatica di EKS. Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

Esempio di pool di nodi con 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"
```

Aggiorna i requisiti per il pool di nodi in modo tale che corrispondano alla configurazione di Karpenter da cui stai eseguendo la migrazione. Devi soddisfare almeno un requisito.

Fase 3: aggiornare i carichi di lavoro per la migrazione

Identifica e aggiorna i carichi di lavoro che desideri migrare alla modalità automatica di EKS . Aggiungi sia le tolleranze che i selettori di nodi a questi carichi di lavoro:

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

Questa modifica consente di pianificare il carico di lavoro sui nuovi nodella modalità automatica di EKS.

La modalità automatica di EKS utilizza etichette diverse rispetto a Karpenter. Le etichette relative alle istanze EC2 gestite iniziano con. `eks.amazonaws.com` Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

Fase 4: migrare gradualmente i carichi di lavoro

Ripeti la fase 3 per ogni carico di lavoro di cui desideri eseguire la migrazione. Ciò consente di spostare i carichi di lavoro individualmente o in gruppo, in base ai requisiti e alla tolleranza al rischio.

Passaggio 5: rimuovere il Karpenter originale NodePool

Una volta migrati tutti i carichi di lavoro, puoi rimuovere il Karpenter originale: NodePool

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

Passaggio 6: rimuovere la contaminazione dalla modalità automatica EKS (opzionale) NodePool

Se desideri che la modalità automatica EKS diventi l'impostazione predefinita per i nuovi carichi di lavoro, puoi rimuovere la contaminazione dalla modalità automatica EKS: NodePool

```
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
```

Fase 7: rimuovere i selettori di nodi dai carichi di lavoro (facoltativo)

Se hai rimosso la macchia dalla modalità automatica EKS NodePool, puoi facoltativamente rimuovere i selettori dei nodi dai tuoi carichi di lavoro, poiché la modalità automatica EKS è ora l'impostazione predefinita:

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      # Remove the nodeSelector section
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
```

Fase 8: disinstallare Karpenter dal cluster

Le fasi per rimuovere Karpenter dipendono da come l'hai installato. Per ulteriori informazioni, consulta Karpenter install instructions.

Eseguire la migrazione dai gruppi di nodi gestiti EKS a EKS Auto Mode

Quando si esegue la transizione del cluster Amazon EKS per utilizzare EKS Auto Mode, puoi eseguire la migrazione senza problemi i carichi di lavoro esistenti dai gruppi di nodi gestiti (MNG) utilizzando lo strumento CLI eksctl. Questo processo garantisce la disponibilità continua delle applicazioni mentre EKS Auto Mode ottimizza le risorse di calcolo. La migrazione può essere eseguita con interruzioni minime delle applicazioni in esecuzione.

Questo argomento illustra le fasi per drenare in sicurezza i pod dai gruppi di nodi gestiti esistenti e consentire a EKS Auto Mode di riprogrammarli sulle nuove istanze di cui è appena stato effettuato il provisioning. Seguendo questa procedura, puoi sfruttare il consolidamento intelligente del carico di lavoro di EKS Auto Mode mantenendo al contempo la disponibilità dell'applicazione durante la migrazione.

Prerequisiti

- Cluster con EKS Auto Mode abilitata
- `eksctl` CLI installato e connesso al cluster. Per ulteriori informazioni, consulta Configurazione.
- Karpenter non è installato nel cluster.

Procedura

Utilizza il seguente comando `eksctl` CLI per avviare i pod di drenaggio dalle istanze del gruppo di nodi gestiti esistente. EKS Auto Mode creerà nuovi nodi per supportare i pod spostati.

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

Dovrai eseguire questo comando per ogni gruppo di nodi gestiti nel cluster.

Per ulteriori informazioni su questo comando, consulta Deleting and draining nodegroups nei documenti di eksctl.

Eseguire la migrazione da EKS Fargate alla modalità automatica di EKS

Questo argomento descrive il processo di migrazione dei carichi di lavoro da EKS Fargate alla modalità automatica di Amazon EKS utilizzando `kubectl`. La migrazione può essere eseguita gradualmente, consentendo di spostare i carichi di lavoro al proprio ritmo, mantenendo al contempo la stabilità del cluster e la disponibilità delle applicazioni durante la transizione.

L' step-by-stepapproccio descritto di seguito consente di eseguire EKS Fargate e EKS Auto Mode fianco a fianco durante il periodo di migrazione. Questa strategia a doppio funzionamento aiuta a garantire una transizione fluida consentendo di convalidare il comportamento del carico di lavoro in modalità automatica di EKS prima di disattivare completamente EKS Fargate. Puoi eseguire la migrazione delle applicazioni singolarmente o in gruppo, garantendo la flessibilità necessaria per soddisfare i requisiti operativi specifici e la tolleranza al rischio.

Confronto tra Amazon EKS Auto Mode ed EKS e AWS Fargate

Amazon EKS con AWS Fargate rimane un'opzione per i clienti che desiderano utilizzare EKS, ma Amazon EKS Auto Mode è l'approccio consigliato in futuro. La modalità automatica di EKS è completamente conforme a Kubernetes e supporta tutte le primitive Kubernetes a monte e gli strumenti di piattaforma come Istio, che Fargate non è in grado di supportare. Inoltre, modalità automatica di EKS supporta completamente tutte le opzioni di acquisto del runtime EC2, incluse GPU e istanze Spot, consentendo ai clienti di sfruttare gli sconti EC2 negoziati e altri meccanismi di risparmio. Queste funzionalità non sono disponibili quando si utilizza EKS con Fargate.

Inoltre, modalità automatica di EKS consente ai clienti di ottenere lo stesso modello di isolamento di Fargate, utilizzando le funzionalità di pianificazione di Kubernetes standard per garantire che ogni istanza EC2 esegua un singolo container di applicazioni. Adottando la modalità Auto di Amazon EKS, i clienti possono sfruttare tutti i vantaggi dell'esecuzione di Kubernetes su AWS una piattaforma completamente conforme a Kubernetes che offre la flessibilità necessaria per sfruttare l'intera gamma di opzioni di acquisto e EC2, pur mantenendo la facilità d'uso e l'astrazione dalla gestione dell'infrastruttura fornite da Fargate.

Raggiungere un isolamento simile a quello di Fargate in modalità automatica EKS

Per replicare il modello di isolamento dei pod di Fargate, in cui ogni pod viene eseguito su una propria istanza dedicata, puoi utilizzare i vincoli di diffusione della topologia di Kubernetes. Questo è l'approccio consigliato per controllare la distribuzione dei pod tra i nodi:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: isolated-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: isolated-app
  template:
    metadata:
      labels:
        app: isolated-app
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      topologySpreadConstraints:
```

```
      - maxSkew: 1
        topologyKey: kubernetes.io/hostname
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchLabels:
            app: isolated-app
        minDomains: 1
      containers:
      - name: app
        image: nginx
        ports:
        - containerPort: 80
```

In questa configurazione:

- `maxSkew: 1`assicura che la differenza nel numero di pod tra due nodi qualsiasi sia al massimo 1, distribuendo efficacemente un pod per nodo
- `topologyKey: kubernetes.io/hostname`definisce il nodo come dominio della topologia
- `whenUnsatisfiable: DoNotSchedule`impedisce la pianificazione se il vincolo non può essere soddisfatto
- `minDomains: 1`assicura l'esistenza di almeno un dominio (nodo) prima della pianificazione

EKS Auto Mode eseguirà automaticamente il provisioning di nuove istanze EC2 secondo necessità per soddisfare questo vincolo, fornendo lo stesso modello di isolamento di Fargate e dandoti accesso all'intera gamma di tipi di istanze EC2 e opzioni di acquisto.

In alternativa, puoi utilizzare le regole anti-affinità dei pod per un isolamento più rigoroso:

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

```
    annotations:
      eks.amazonaws.com/compute-type: ec2
  spec:
    affinity:
      podAntiAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - isolated-app
          topologyKey: kubernetes.io/hostname
    containers:
    - name: app
      image: nginx
      ports:
      - containerPort: 80
```

La `podAntiAffinity` regola `requiredDuringSchedulingIgnoredDuringExecution` garantisce che non sia `app: isolated-app` possibile programmare due pod con l'etichetta sullo stesso nodo. Questo approccio offre garanzie di isolamento rigide simili a quelle di Fargate.

Prerequisiti

Prima di iniziare la migrazione, assicurarti di aver:

- configurato un cluster con Fargate. Per ulteriori informazioni, consulta the section called "Nozioni di base";
- installato e connesso `kubectl` al cluster. Per ulteriori informazioni, consulta Configurazione.

Fase 1: Verificare il cluster Fargate

1. Verifica se il cluster EKS con Fargate è in esecuzione:

```
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. Verifica i pod in esecuzione:

```
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. Create un'implementazione in un file chiamato `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. Applica l'implementazione:

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

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

5. Verifica i pod e le implementazioni:

```
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. Verifica il nodo:

```
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
```

Fase 2: Abilitare modalità automatica di EKS sul cluster

1. Abilita la modalità automatica EKS sul tuo cluster esistente utilizzando la AWS CLI o la console di gestione. Per ulteriori informazioni, consulta the section called “Abilitazione sul cluster”.
2. Verifica il nodepool:

```
kubectl get nodepool
```

```
NAME              NODECLASS   NODES   READY   AGE
general-purpose   default     1       True    6m58s
```

```
system           default     0       True    3d14h
```

Fase 3: Aggiornare i carichi di lavoro per la migrazione

Identifica e aggiorna i carichi di lavoro che desideri migrare alla modalità automatica di EKS .

Per eseguire la migrazione di un carico di lavoro da Fargate alla modalità automatica di EKS , applica l'annotazione `eks.amazonaws.com/compute-type: ec2`. Ciò garantisce che il carico di lavoro non venga programmato da Fargate, nonostante il profilo Fargate, e venga interrotto dalla modalità automatica EKS. NodePool Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

1. Modifica le implementazioni (ad esempio, il file `deployment_fargate.yaml`) per cambiare il tipo di elaborazione in `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. Applica l'implementazione. Questa modifica consente di pianificare il carico di lavoro sui nuovi nodella modalità automatica di EKS:

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

3. Verifica che l'implementazione sia in esecuzione nel cluster della modalità automatica di 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. Verifica che non sia in esecuzione alcun nodo Fargate e che l'implementazione sia in esecuzione nei nodi gestiti dalla modalità automatica di 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
```

Fase 4: Migrare gradualmente i carichi di lavoro

Ripeti la fase 3 per ogni carico di lavoro di cui desideri eseguire la migrazione. Ciò consente di spostare i carichi di lavoro individualmente o in gruppo, in base ai requisiti e alla tolleranza al rischio.

Fase 5: Rimuovere il profilo Fargate originale

Una volta eseguita la migrazione di tutti i carichi di lavoro, puoi rimuovere il profilo `fargate` originale. Sostituisci *`<fargate profile name>`* con il nome del tuo profilo Fargate:

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

Fase 6: Ridurre verticalmente CoreDNS

Poiché modalità automatica di EKS gestisce CoreDNS, puoi ridurre verticalmente l'implementazione `coredns` fino a 0:

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

Esegui carichi di lavoro di esempio nei cluster della modalità automatica di EKS

Questo capitolo fornisce esempi di come implementare diversi tipi di carichi di lavoro nei cluster Amazon EKS in esecuzione in Auto Mode. Gli esempi illustrano i principali schemi di carico di lavoro, tra cui applicazioni di esempio, applicazioni web con bilanciamento del carico, carichi di lavoro stateful che utilizzano l'archiviazione persistente, e carichi di lavoro con requisiti di posizionamento dei nodi specifici. Ogni esempio include manifesti completi e istruzioni di step-by-step distribuzione che è possibile utilizzare come modelli per le proprie applicazioni.

Prima di procedere con gli esempi, assicurati di avere un cluster EKS in esecuzione in modalità automatica e di aver installato la AWS CLI e kubectl. Per ulteriori informazioni, consulta Configurazione. Gli esempi presuppongono una conoscenza base dei concetti di Kubernetes e dei comandi kubectl.

Puoi utilizzare questi esempi basati su casi d'uso per eseguire carichi di lavoro in cluster della modalità automatica di EKS.

the section called "Implementazione di un carico di lavoro di decompressione"

> Mostra come implementare un carico di lavoro di esempio in un cluster della modalità automatica di EKS utilizzando i comandi `kubectl`.

the section called "Implementa il bilanciatore del carico"

> Mostra come implementare una versione containerizzata del gioco 2048 su Amazon EKS.

the section called "Implementazione di un carico di lavoro stateful"

> Mostra come implementare un'applicazione stateful di esempio in un cluster della modalità automatica di EKS.

the section called "Implementare carico di lavoro accelerato"

Mostra come distribuire carichi di lavoro con accelerazione hardware nei nodi gestiti da EKS Auto Mode.

the section called "Controllare l'implementazione"

Mostra come utilizzare un'annotazione per controllare se un carico di lavoro viene implementato nei nodi gestiti dalla modalità automatica di EKS.

Implementazione di un esempio di carico di lavoro di decompressione in un cluster con modalità automatica di Amazon EKS

In questo tutorial imparerai come implementare un carico di lavoro di esempio su un cluster con modalità automatica EKS e osservare come esegue automaticamente il provisioning delle risorse di calcolo necessarie. Utilizzerai `kubectl` i comandi per osservare il comportamento del cluster e vedere in prima persona come la modalità automatica semplifica le operazioni di Kubernetes su. AWS Al termine di questo tutorial, capirai come la modalità automatica EKS risponde alle distribuzioni dei carichi di lavoro gestendo automaticamente le risorse di calcolo sottostanti, senza richiedere la configurazione manuale dei gruppi di nodi.

Prerequisiti

- Un cluster della modalità automatica di Amazon EKS. Annota il nome e la regione del cluster. AWS
- Un principale IAM, come ad esempio un utente o un ruolo, con autorizzazioni sufficienti per gestire le risorse di rete, di elaborazione ed EKS.
  - Per ulteriori informazioni, consulta Creating roles and attaching policies in the IAM User Guide nella Guida per l'utente IAM.
- `aws` CLI installata e configurata con un'identità IAM.
- `kubectl` CLI installata e connessa al cluster.
  - Per ulteriori informazioni, consulta Configurazione.

Passaggio 1: rivedere le risorse di elaborazione esistenti (facoltativo)

Innanzitutto, usa `kubectl` per elencare i pool di nodi sul cluster.

```
kubectl get nodepools
```

Output di esempio:

```
general-purpose
```

In questo tutorial, implementeremo un carico di lavoro configurato per usare il pool di nodi `general-purpose`. Questo pool di nodi è integrato nella modalità automatica EKS e include impostazioni predefinite ragionevoli per carichi di lavoro generici, come microservizi e applicazioni web. Puoi creare un pool di nodi personalizzato. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

Poi, utilizza `kubectl` per elencare i nodi collegati al tuo cluster.

```
kubectl get nodes
```

Se hai appena creato un cluster con modalità automatica EKS, non avrai nodi.

In questo tutorial implementerai un carico di lavoro di esempio. Se non ci sono nodi o il carico di lavoro non può essere gestito dai nodi esistenti, la modalità automatica EKS provvederà a fornire un nuovo nodo.

Passaggio 2: implementare un'applicazione di esempio sul cluster

Esamina la seguente implementazione di Kubernetes e salvala come `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
```

Il selettore `eks.amazonaws.com/compute-type: auto` richiede che il carico di lavoro sia implementato su un nodo della modalità automatica di Amazon EKS.

Applica l'implementazione al cluster.

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

Passaggio 3: vedere gli eventi di Kubernetes

Utilizza il seguente comando per vedere gli eventi di Kubernetes, inclusa la creazione di un nuovo nodo. Usa `ctrl+c` per interrompere la visione degli eventi.

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

Utilizza `kubectl` per elencare nuovamente i nodi collegati al tuo cluster. Prendi nota del nodo appena creato.

```
kubectl get nodes
```

Passaggio 4: Visualizza i nodi e le istanze nella console AWS

È possibile visualizzare i nodi EKS Auto Mode nella console EKS e le EC2 istanze associate nella EC2 console.

EC2 Le istanze distribuite da EKS Auto Mode sono limitate. Non puoi eseguire comandi arbitrari sui nodi della modalità automatica EKS.

Passaggio 5: eliminare l'implementazione

Per eliminare un'implementazione, usa `kubectl`

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

Se non ci sono altri carichi di lavoro implementati nel cluster, il nodo creato dalla modalità automatica EKS sarà vuoto.

Nella configurazione predefinita, la modalità automatica EKS rileva i nodi che rimangono vuoti per trenta secondi e li termina.

Utilizza `kubectl` o la EC2 console per confermare che l'istanza associata è stata eliminata.

Implementa un carico di lavoro del bilanciatore del carico esemplificativo in EKS Auto Mode

Questa guida mostra come implementare una versione containerizzata del gioco 2048 su Amazon EKS, completa di bilanciamento del carico e accessibilità a Internet.

Prerequisiti

- Un cluster EKS Auto Mode
- `kubectl` configurato per interagire con il cluster
- Autorizzazioni IAM appropriate per la creazione di risorse ALB

Fase 1: Creazione del namespace

Innanzitutto, crea un namespace dedicato per l'applicazione di gioco 2048.

Creare un file denominato `01-namespace.yaml`:

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

Applica la configurazione del namespace:

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

Fase 2: Implementazione dell'applicazione

L'applicazione esegue più repliche del container del gioco 2048.

Creare un file denominato `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 ricevi un errore durante il caricamento dell'immagine `public.ecr.aws/l6m2t8p7/docker-2048:latest`, conferma che il ruolo IAM del nodo dispone delle autorizzazioni sufficienti per estrarre immagini da ECR. Per ulteriori informazioni, consulta the section called "Ruolo IAM del nodo". Inoltre, l'immagine `docker-2048` nell'esempio è un'immagine `x86_64` e non sarà eseguita su altre architetture.

Componenti chiave:

- Implementa 5 repliche dell'applicazione
- Utilizza un'immagine ECR pubblica
- Richiede 0,5 core CPU per pod
- Espone la porta 80 per il traffico HTTP

Applica l'implementazione:

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

Fase 3: Creazione del servizio

Il servizio espone l'implementazione alla rete di cluster.

Creare un file denominato `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
```

Componenti chiave:

- Crea un servizio NodePort
- Mappa la porta 80 alla porta 80 del container
- Utilizza il selettore di etichette per trovare i pod

Applica il servizio:

```
kubectl apply -f 03-service.yaml
```

Fase 4: Configurazione del bilanciamento del carico

Configurerai un ingresso per esporre l'applicazione a Internet.

Innanzitutto, crea la `IngressClass`. Creare un file denominato `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**
>
> EKS Auto Mode richiede i tag di sottorete per identificare le sottoreti pubbliche e private. Se hai creato il cluster con `eksctl`, disponi già di questi tag.
> Informazioni su come the section called "Assegnazione di tag alle sottoreti".

Poi, crea la risorsa Ingress. Creare un file denominato `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
```

Componenti chiave:

- Crea un ALB connesso a Internet
- Utilizza il tipo di destinazione IP per l'instradamento diretto al pod
- Instrada tutto il traffico (/) al servizio di gioco

Applica le configurazioni di ingresso:

```
kubectl apply -f 04-ingressclass.yaml
kubectl apply -f 05-ingress.yaml
```

Fase 5: Verifica dell'implementazione

1. Controlla che tutti i pod siano in esecuzione:

   ```
   kubectl get pods -n game-2048
   ```

2. Verifica che il servizio sia stato creato:

   ```
   kubectl get svc -n game-2048
   ```

3. Ottieni l'endpoint di ALB:

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

Il campo ADDRESS nell'output di ingresso mostrerà l'endpoint di ALB. Attendi 2-3 minuti affinché l'ALB effettui il provisioning e registri tutti gli obiettivi.

Fase 6: Accesso al gioco

Apri il browser web e accedi all'URL dell'endpoint di ALB della fase precedente. Dovresti vedere l'interfaccia di gioco 2048.

Fase 7: Eliminazione

Per rimuovere tutte le risorse create in questo tutorial:

```
kubectl delete namespace game-2048
```

Questo eliminerà tutte le risorse nel namespace, incluse le risorse di implementazione, di servizio e di ingresso.

Cosa succede dietro le quinte

1. L'implementazione crea 5 pod che eseguono il gioco 2048
2. Il servizio fornisce un accesso stabile alla rete a questi pod
3. EKS Auto Mode:
   - Crea un Application Load Balancer in AWS
   - Configura i gruppi di destinazione per i pod
   - Imposta le regole di instradamento per indirizzare il traffico al servizio

Risoluzione dei problemi

Se il gioco non si carica:

- Assicurati che tutti i pod siano in esecuzione: `kubectl get pods -n game-2048`
- Verifica lo stato di ingresso: `kubectl describe ingress -n game-2048`
- Verifica i controlli dell'integrità di ALB: Verificare lo stato del gruppo di destinazione nella Console AWS

Implementazione di un carico di lavoro stateful esemplificativo in modalità automatica EKS

Questo tutorial ti guiderà nella distribuzione di un'applicazione stateful esemplificativa nel tuo cluster di modalità automatica EKS. L'applicazione scrive marche temporali su un volume persistente, dimostrando le funzionalità automatiche di provisioning e persistenza dei volumi EBS di modalità automatica EKS.

Prerequisiti

- Un cluster in modalità automatica EKS
- AWS CLI configurata con le autorizzazioni appropriate
- Installazione e configurazione di `kubectl`.
  - Per ulteriori informazioni, consulta Configurazione.

Passaggio 1: configurazione dell'ambiente

1. Imposta le variabili di ambiente:

```
export CLUSTER_NAME=my-auto-cluster
export AWS_REGION="us-west-2"
```

2. Aggiorna kubeconfig:

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

Passaggio 2: creazione della classe di archiviazione

`StorageClass` definisce in che modo la modalità automatica EKS fornirà i volumi EBS.

Modalità automatica EKS non crea un `StorageClass` per te. Devi creare una `StorageClass` che faccia riferimento a `ebs.csi.eks.amazonaws.com` per utilizzare la funzionalità di archiviazione di modalità automatica EKS.

1. Creare un file denominato `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. Applica: `StorageClass`

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

componenti chiave

- `provisioner: ebs.csi.eks.amazonaws.com`: utilizza modalità automatica EKS
- `volumeBindingMode: WaitForFirstConsumer`: ritarda la creazione del volume fino a quando un pod non ne ha bisogno
- `type: gp3`: specifica il tipo di volume EBS
- `encrypted: "true"`: EBS utilizzerà la chiave `aws/ebs` predefinita per crittografare i volumi creati con questa classe. Questo passaggio è facoltativo, ma è consigliato.
- `storageclass.kubernetes.io/is-default-class: "true"`: Kubernetes userà questa classe di archiviazione per impostazione predefinita, a meno che non sia specificata una classe di volume diversa su una dichiarazione di volume persistente. Fai attenzione quando imposti questo valore se stai migrando da un altro controller di archiviazione (facoltativo).

Passaggio 3: creazione della dichiarazione di volume persistente

Il PVC richiede l'archiviazione da `StorageClass`.

1. Creare un file denominato `pvc.yaml`:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-ebs-claim
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: auto-ebs-sc
  resources:
    requests:
      storage: 8Gi
```

2. Applica il PVC:

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

Componenti chiave:

- `accessModes: ReadWriteOnce`: il volume può essere montato da un nodo alla volta
- `storage: 8Gi`: richiede un volume da 8 GiB
- `storageClassName: auto-ebs-sc`: riferimenti su `StorageClass` abbiamo creato

Passaggio 4: implementare un'applicazione

L'implementazione esegue un container che scrive marche temporali nel volume persistente.

1. Creare un file denominato `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. Applica l'implementazione:

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

Componenti chiave:

- Semplice container bash che scrive una marca temporale su un file
- Monta il PVC su `/data`
- Richiede 1 core CPU
- Utilizza il selettore di nodi per i nodi gestiti da EKS

Passaggio 5: verificare la configurazione

1. Verifica che il pod sia in esecuzione:

```
kubectl get pods -l app=inflate-stateful
```

2. Verifica che il PVC sia legato:

```
kubectl get pvc auto-ebs-claim
```

3. Controlla il volume 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. Verifica che i dati siano scritti:

```
kubectl exec "$(kubectl get pods -l app=inflate-stateful \
  -o=jsonpath='{.items[0].metadata.name}')" -- \
  cat /data/out.txt
```

Passaggio 6: pulizia

Eseguire il comando riportato di seguito per rimuovere tutte le risorse create in questo tutorial:

```
# Delete all resources in one command
kubectl delete deployment/inflate-stateful pvc/auto-ebs-claim storageclass/auto-ebs-sc
```

Cosa succede dietro le quinte

1. Il PVC richiede l'archiviazione da `StorageClass`
2. Quando è il pod è pianificato:
   a. La modalità automatica EKS fornisce un volume EBS
   b. Crea PersistentVolume
   c. Collega il volume al nodo
3. Il pod monta il volume e inizia a scrivere le marche temporali

Controller di snapshot

La modalità automatica EKS è compatibile con lo Snapshotter di CSI Kubernetes, noto anche come controller di snapshot. Tuttavia, la modalità automatica EKS non include il controller di snapshot. L'utente è responsabile dell'installazione e della configurazione del controller di snapshot. Per ulteriori informazioni, consulta the section called "Controller snapshot CSI".

Leggi quanto segue `VolumeSnapshotClass` che fa riferimento alla capacità di archiviazione della modalità automatica EKS.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: auto-ebs-vsclass
driver: ebs.csi.eks.amazonaws.com
deletionPolicy: Delete
```

Scopri di più su Snapshotter di CSI Kubernetes.

Implementare un carico di lavoro accelerato

Questo tutorial dimostra come la modalità automatica di Amazon EKS semplifica l'avvio di carichi di lavoro con accelerazione hardware. La modalità automatica di Amazon EKS semplifica le operazioni oltre il cluster stesso automatizzando i componenti chiave dell'infrastruttura che forniscono funzionalità di elaborazione, rete, bilanciamento del carico, storage e accesso e gestione delle identità pronte all'uso.

La modalità Auto di Amazon EKS include i driver e i plug-in di dispositivo necessari per determinati tipi di istanze, come i driver NVIDIA e AWS Neuron. Non è necessario installare o aggiornare questi componenti.

modalità automatica di EKS gestisce automaticamente i driver per questi acceleratori:

- AWS Trainium
- AWS Inferenza
- Istanze EC2 accelerate NVIDIA GPUs su Amazon

> **Note**
>
> La modalità automatica di EKS include il plug-in di dispositivo NVIDIA per Kubernetes. Questo plug-in viene eseguito automaticamente e non è visibile come daemon impostato nel cluster.

Supporto di reti avanzate:

- Elastic Fabric Adapter (EFA)

La modalità automatica di Amazon EKS elimina la difficoltà della gestione dei driver dell'acceleratore e dei plug-in di dispositivo.

Puoi anche trarre vantaggio dai risparmi sui costi ridimensionando il cluster fino a zero. Puoi configurare modalità automatica di EKS per terminare le istanze quando non è in esecuzione alcun carico di lavoro. Ciò è utile per i carichi di lavoro di inferenza basati su batch.

Di seguito viene fornito un esempio di come avviare carichi di lavoro accelerati con la modalità automatica di Amazon EKS.

Prerequisiti

- Un cluster Kubernetes con la modalità automatica di Amazon EKS configurata.
- Una classe del nodo `default` EKS creata quando i pool di nodi gestiti da `general-purpose` o `system` sono abilitati.

Fase 1: Implementare un carico di lavoro GPU

In questo esempio, creerai un file NodePool per carichi di lavoro basati su NVIDIA che richiede 45 GB di memoria GPU. Con la modalità automatica di EKS, utilizzi i vincoli di pianificazione di Kubernetes per definire i requisiti delle istanze.

Per distribuire la modalità automatica di Amazon EKS `NodePool` e l'esempio`workload`, esamina quanto segue NodePool e la definizione di Pod e salva con nome `nodepool-gpu.yaml` e `pod.yaml` nome:

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
```

Il selettore `eks.amazonaws.com/compute-type: auto` richiede che il carico di lavoro sia implementato su un nodo della modalità automatica di Amazon EKS. NodePool Inoltre, imposta una caratteristica che consente di programmare solo pod con tolleranze per GPUs Nvidia.

Applica il carico di lavoro NodePool and al tuo cluster.

```
kubectl apply -f nodepool-gpu.yaml
kubectl apply -f pod.yaml
```

Verrà visualizzato l'output seguente:

```
nodepool.karpenter.sh/gpu configured created
pod/nvidia-smi created
```

Attendi qualche secondo e controlla i nodi del cluster. Ora dovresti vedere un nuovo nodo fornito nel cluster della modalità automatica di 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
```

Fase 2: Convalida

Puoi vedere che la modalità automatica di Amazon EKS ha lanciato un `g6e.2xlarge` anziché un `g6.2xlarge` in quanto il carico di lavoro richiedeva un'istanza con l40s GPU, in base ai seguenti vincoli di pianificazione di 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
```

Ora, analizzare i log dei container, eseguendo il seguente comando:

```
kubectl logs nvidia-smi
```

Output di esempio:

```
+-----------------------------------------------------------------------------------------
+
| 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
 |
+-----------------------------------------------------------------------------------------
+
```

Puoi vedere che il container ha rilevato che è in esecuzione su un'istanza con una GPU `NVIDIA` e che non hai dovuto installare alcun driver di dispositivo, poiché questa è gestita dalla modalità automatica di Amazon EKS.

Fase 3: Pulizia

Per rimuovere tutti gli oggetti creati, usa `kubectl` per eliminare la distribuzione di esempio e NodePool quindi il nodo viene terminato:

```
kubectl delete -f nodepool-gpu.yaml
kubectl delete -f pod.yaml
```

Esempio NodePools di riferimento

Crea un NVIDIA NodePool

Quanto segue NodePool definisce:

- Solo istanze di avvio della famiglia `g6e` e `g6`
- Consolidare i nodi quando sono vuoti per 1 ora
  - Il valore di 1 ora per `consolodateAfter` supporta carichi di lavoro con picchi e riduce il tasso di abbandono dei nodi. Puoi eseguire l'ottimizzazione di `consolidateAfter` in base ai requisiti del carico di lavoro.

Esempio NodePool con famiglia di istanze GPU e consolidamento

```
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
```

Invece di impostare il `eks.amazonaws.com/instance-gpu-name`, puoi utilizzare `eks.amazonaws.com/instance-family` per specificare la famiglia di istanze. Per altre etichette note che influenzano la revisione della pianificazione, consultare the section called "Etichette supportate da EKS Auto Mode".

Se hai requisiti di storage specifici, puoi ottimizzare lo storage `iops` temporaneo dei nodi `size` e `throughput` crearne uno personalizzato a cui fare riferimento nel. NodeClass NodePool Scopri di più sulle opzioni configurabili. NodeClass

Esempio di configurazione di archiviazione per NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu
spec:
  ephemeralStorage:
    iops: 3000
    size: 80Gi
    throughput: 125
```

Definisci AWS Trainium e AWS Inferentia NodePool

Di seguito NodePool è riportato un `eks.amazonaws.com/instance-category` set che indica solo le istanze di avvio delle famiglie Inferentia e Trainium:

```
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values:
            - inf
            - trn
```

Configurazione delle impostazioni della modalità automatica EKS

Questo capitolo descrive come configurare aspetti specifici dei cluster in modalità automatica Amazon Elastic Kubernetes Service (EKS). Sebbene la modalità automatica EKS gestisca automaticamente la maggior parte dei componenti dell'infrastruttura, è possibile personalizzare alcune funzionalità per soddisfare i requisiti del tuo carico di lavoro.

Utilizzando le opzioni di configurazione descritte in questo argomento, è possibile modificare le impostazioni di rete, le risorse di calcolo e i comportamenti di bilanciamento del carico, mantenendo al contempo i vantaggi della gestione automatizzata dell'infrastruttura. Prima di apportare modifiche alla configurazione, esamina le opzioni disponibili nelle sezioni seguenti per determinare l'approccio più adatto alle tue esigenze.

Quali funzionalità vuoi configurare?	Opzione di configurazione
Rete e archiviazione dei nodi • Configura il posizionamento dei nodi tra sottoreti pubbliche e private • Definisci gruppi di sicurezza personalizzati per il controllo degli accessi ai nodi • Personalizza le policy di traslazione degli indirizzi di rete (SNAT) • Abilita le policy di rete Kubernetes, la registrazione e il monitoraggio dettagliati delle policy di rete dettagliate • Imposta parametri di archiviazione temporanei (dimensioni, IOPS, throughput) • Configura l'archiviazione temporanea crittografata con chiavi KMS personalizzate	the section called "Creazione di una classe di nodi"

Quali funzionalità vuoi configurare?	Opzione di configurazione
• Isola il traffico dei pod in sottoreti separate dai nodi	
Risorse di calcolo dei nodi • Seleziona tipi e famiglie di EC2 istanze specifici • Definisci le architetture della CPU (x86_64,) ARM64 • Configura i tipi di capacità (su richiesta, spot) • Specifica zone di disponibilità • Configura i taint e le etichette dei nodi • Imposta limiti di risorse per l'utilizzo di CPU e memoria	the section called "Creazione di pool di nodi"
Pool di nodi a capacità statica • Mantieni un numero fisso di nodi indipende ntemente dalla richiesta del pod • Configura la capacità prevedibile per le istanze riservate • Imposta un numero di repliche specifico per un ingombro dell'infrastruttura coerente • Ridimensiona manualmente i pool di nodi statici • Monitora lo stato del pool di nodi con capacità statica	the section called "Pool di nodi a capacità statica"

Quali funzionalità vuoi configurare?	Opzione di configurazione
Impostazioni Application Load Balancer • Implementa bilanciatori del carico interni o connessi a Internet • Configura il load balancer tra zone • Imposta periodi di timeout di inattività • Abilita HTTP/2 e supporta WebSocket • Configura i parametri del controllo dell'integrità • Specifica le impostazioni del certificato TLS • Definisci gli attributi dei gruppi di destinazione • Imposta il tipo di indirizzo IP (IPv4dual-stack)	the section called "Creare una classe in ingresso"
Impostazioni Network Load Balancer • Configura il routing IP diretto del pod • Abilita il load balancer tra zone • Imposta un timeout di inattività della connessione • Configura i parametri del controllo dell'integrità • Specifica il posizionamento della sottorete • Imposta il tipo di indirizzo IP (IPv4dual-stack) • Configura l'IP di conservazione dell'origine del client • Definisci gli attributi dei gruppi di destinazione	the section called "Creare servizio"

Quali funzionalità vuoi configurare?	Opzione di configurazione
Impostazioni della classe di archiviazione • Definisci i tipi di volume EBS (gp3, io1, io2, eccetera) • Configura la crittografia dei volumi e l'utilizzo delle chiavi KMS • Imposta i parametri IOPS e di throughput • Imposta come classe di archiviazione predefinita • Definisci tag personalizzati per i volumi assegnati	the section called "Crea StorageClass"
Controlla l'utilizzo dell'ODCR • Configura la distribuzione del carico di lavoro nelle prenotazioni di capacità su richiesta EC2 • Seleziona esplicitamente specifico in base all'ID per l' ODCRs utilizzo mirato della capacità • Usa la selezione basata su tag per scegliere come target gruppi di ODCRs • Filtra in ODCRs base al proprietario AWS dell'account per scenari che coinvolgono più account • Controlla se i carichi di lavoro utilizzano automaticamente open ODCRs	the section called "Controllare le prenotazioni della capacità"
Sicurezza avanzata del nodo • Abilita i moduli crittografici convalidati FIPS 140-2 • Configura la modalità di sicurezza Kernel Lockdown	the section called "Sicurezza avanzata"

Creazione di una classe di nodi per Amazon EKS

Le classi di nodi Amazon EKS sono modelli che forniscono un controllo granulare sulla configurazione dei nodi gestiti dalla modalità automatica di EKS. Una classe di nodi definisce le impostazioni a livello di infrastruttura che si applicano ai gruppi di nodi del cluster EKS, tra cui la configurazione di rete, le impostazioni di storage e il tagging delle risorse. Questo argomento mostra come creare e configurare una classe di nodi per soddisfare i tuoi requisiti operativi specifici.

Quando è necessario personalizzare le modalità di provisioning e configurazione delle istanze EC2 da parte della modalità automatica di EKS oltre le impostazioni predefinite, la creazione di una classe di nodi fornisce un controllo preciso sui parametri critici dell'infrastruttura. Ad esempio, è possibile specificare il posizionamento di sottoreti private per una maggiore sicurezza, configurare lo storage temporaneo delle istanze per carichi di lavoro sensibili alle prestazioni o applicare tag personalizzati per l'allocazione dei costi.

Creazione di una classe di nodi

Per creare una `NodeClass`, segui questi passaggi:

1. Crea un file YAML (ad esempio, `nodeclass.yaml`) con la configurazione della classe di nodi
2. Applica la configurazione al cluster utilizzando `kubectl`
3. Fai riferimento alla classe di nodi nella configurazione del tuo pool di nodi. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

`kubectl` deve essere installato e configurato. Per ulteriori informazioni, consulta Configurazione.

Esempio di classe di nodi di base

Ecco un esempio di classe di nodi:

```
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"
```

Ciò NodeClass aumenta la quantità di storage temporaneo sul nodo.

Applica questa configurazione usando:

```
kubectl apply -f nodeclass.yaml
```

Poi fai riferimento alla classe di nodi nella configurazione del pool di nodi. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

Creazione di una voce di accesso alla classe di nodi

Se crei una classe di nodi personalizzata, devi creare una voce di accesso EKS per consentire ai nodi di unirsi al cluster. Quando si utilizzano la classe di nodi e i pool di nodi integrati, EKS crea automaticamente le voci di accesso.

Per informazioni sul funzionamento delle voci di accesso, consulta the section called "Voci di accesso".

Quando si creano voci di accesso per le classi di nodi della modalità automatica di EKS, è necessario usare il tipo di voce di accesso `EC2`.

Creazione di un accesso con CLI

Per creare una voce di accesso per i nodi EC2 e associare la policy dei nodi di EKS Auto:

Aggiorna i comandi CLI seguenti con il nome del cluster e l'ARN del ruolo del nodo. Il ruolo del nodo ARN viene specificato nella classe di nodi YAML.

```
# 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
```

Crea un ingresso di accesso con CloudFormation

Per creare una voce di accesso per i nodi EC2 e associare la policy dei nodi di EKS Auto:

Aggiorna quanto segue CloudFormation con il nome del cluster e l'ARN del ruolo del nodo. Il ruolo del nodo ARN viene specificato nella classe di nodi YAML.

```
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
```

Per informazioni sulla distribuzione degli CloudFormation stack, consulta Guida introduttiva a CloudFormation

Specifiche della classe di nodi

```
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
  # 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
    # ipv4PrefixSize is default to Auto which is prefix and fallback to secondary IP.
 "32" is the secondary IP mode.
    ipv4PrefixSize: Auto # or "32"

    # enableV4Egress is default to true. Setting it to false when using network policy
 or blocking IPv4 traffic in IPv6 clusters
    enableV4Egress: false

  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
```

Considerazioni

- Se vuoi verificare la quantità di storage locale di cui dispone un'istanza, puoi descrivere il nodo per visualizzare la risorsa di storage temporaneo.
- Crittografia dei volumi: EKS utilizza la chiave KMS personalizzata configurata per crittografare il volume root di sola lettura dell'istanza e il volume di dati. read/write
- Sostituisci il ruolo IAM del nodo: se modifichi il ruolo IAM del nodo associato a `NodeClass`, dovrai creare una nuova voce d'accesso. EKS crea automaticamente una voce d'accesso per il ruolo IAM del nodo durante la creazione del cluster. Il ruolo IAM del nodo richiede la policy di accesso di `AmazonEKSAutoNodePolicy` EKS. Per ulteriori informazioni, consulta the section called "Voci di accesso".
- densità massima di pod: EKS limita il numero massimo di pod su un nodo a 110. Questo limite viene applicato dopo il calcolo del numero massimo di pod attuale. Per ulteriori informazioni, consulta the section called "Tipi di istanza Amazon EC2".
- Tag: se vuoi propagare i tag da Kubernetes a EC2, devi configurare autorizzazioni IAM aggiuntive. Per ulteriori informazioni, consulta the section called "Identità e accesso".
- Classe di nodi predefinita: non assegnare un nome alla classe di nodi `default` personalizzata. Questo perché la modalità automatica di EKS include una `NodeClass` denominata `default` che viene fornita automaticamente quando si abilita almeno un `NodePool` integrato. Per informazioni sull'abilitazione dei `NodePools`, consulta the section called "Riesamina i pool di nodi integrati".

- comportamento di **`subnetSelectorTerms`** con più sottoreti: se esistono più sottoreti che soddisfano le condizioni di `subnetSelectorTerms` o che vengono fornite tramite ID, la modalità automatica di EKS crea nodi distribuiti tra le sottoreti.
  - Se le sottoreti si trovano in zone di disponibilità diverse (AZs), puoi utilizzare le funzionalità di Kubernetes come i vincoli di diffusione della topologia Pod e il Topology Aware Routing per distribuire rispettivamente i pod e il traffico tra le zone.
  - Se nella stessa AZ sono presenti più sottoreti che corrispondono al `subnetSelectorTerms`, la modalità automatica di EKS crea pod su ciascun nodo distribuito tra le sottoreti in quella AZ. La modalità automatica di EKS crea interfacce di rete secondarie su ogni nodo nelle altre sottoreti della stessa AZ. Decide in base al numero di indirizzi IP disponibili in ciascuna sottorete, per usare le sottoreti in modo più efficiente. Non è tuttavia possibile specificare quale sottorete viene usata dalla modalità automatica di EKS per ogni Pod; se hai bisogno che i Pod funzionino in sottoreti specifiche, usa the section called "Selezione della sottorete per i pod".

Selezione della sottorete per i pod

I campi `podSubnetSelectorTerms` e `podSecurityGroupSelectorTerms` permettono configurazioni di rete avanzate consentendo ai Pod di essere eseguiti in sottoreti diverse rispetto ai relativi nodi. Questa separazione offre maggiore controllo sul routing del traffico di rete e sulle policy di sicurezza. Nota che `podSecurityGroupSelectorTerms` è richiesto con `podSubnetSelectorTerms`.

Casi d'uso

Usa `podSubnetSelectorTerms` quando devi:

- Separa il traffico dell'infrastruttura (comunicazione) dal traffico delle applicazioni (comunicazione) node-to-node Pod-to-Pod
- Applicare configurazioni di rete diverse alle sottoreti dei nodi rispetto alle sottoreti Pod.
- Implementare policy di sicurezza o regole di routing diverse per nodi e pod.
- Configurare i proxy inversi o il filtraggio di rete in modo specifico per il traffico dei nodi senza influire sul traffico dei Pod. Utilizza `advancedNetworking` e `certificateBundles` per definire il tuo proxy inverso e qualsiasi certificato autofirmato o privato per il proxy.

Configurazione di esempio

```
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"
```

Considerazioni sui selettori di sottorete per i pod

- Densità dei pod ridotta: Quando si usa `podSubnetSelectorTerms`, è possibile eseguire un numero inferiore di pod su ciascun nodo, poiché l'interfaccia di rete primaria del nodo si trova nella sottorete del nodo e non può essere usata per i pod nella sottorete dei pod.
- Limitazioni del selettore della sottorete: le configurazioni standard `subnetSelectorTerms` e `securityGroupSelectorTerms` non si applicano alla selezione della sottorete dei pod.
- Pianificazione della rete: assicurati che lo spazio degli indirizzi IP sia adeguato sia nella sottorete del nodo che in quella del pod per supportare i requisiti del tuo carico di lavoro.
- Configurazione del routing: verifica che la tabella di routing e la lista di controllo degli accessi (ACL) delle sottoreti del pod siano configurate correttamente per la comunicazione tra il nodo e le sottoreti del pod.
- Zone di disponibilità: verifica di aver creato sottoreti Pod su più sottoreti. AZs Se usi una sottorete del pod specifica, questa deve trovarsi nella stessa AZ della sottorete del nodo AZ.

Modalità IP secondaria per i pod

I `ipv4PrefixSize` campi consentono configurazioni di rete avanzate consentendo solo l'allocazione di indirizzi IP secondari ai nodi. Questa funzionalità non alloca il prefisso (/28) ai nodi e mantiene un solo IP secondario come minimo. IPTarget

Casi d'uso

Usa `ipv4PrefixSize` quando devi:

- Utilizzo IP ridotto: in ogni nodo verrà riscaldato solo un indirizzo IP.
- Riduzione della velocità di produzione dei pod: la velocità di creazione dei pod non è un problema importante.
- Nessuna frammentazione del prefisso: la frammentazione causata dal prefisso è una delle principali preoccupazioni o ostacola l'utilizzo della modalità Auto.

Configurazione di esempio

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  advancedNetworking:
    ipv4PrefixSize: "32"
```

Considerazioni sulla modalità IP secondaria

- Velocità di creazione dei pod ridotta: poiché viene riscaldato solo un IP secondario, il servizio IPAM richiede più tempo per eseguire il provisioning IPs sulla creazione di più pod.

Disabilita l' IPv4 uscita dai pod nei cluster. IPv6 IPv6

I `enableV4Egress` campi sono veri per impostazione predefinita. Per IPv6 i cluster Auto Mode, la funzione può essere disabilitata e quindi Auto Mode non creerà un' IPv4 interfaccia di sola uscita per i IPv6 pod. Questo è importante perché l'interfaccia IPv4 di rete non sarà protetta dalla funzionalità

Network policy. Le politiche di rete verranno applicate solo sull'interfaccia principale del Pod (ad esempio, eth0).

Casi d'uso

Usa `enableV4Egress` quando devi:

- Usa IPv6 Cluster: il traffico IPv4 in uscita è consentito per impostazione predefinita.
- Usa la politica di rete: attualmente la politica di rete EKS non supporta il dual stack. La disabilitazione di V4egress può proteggere il traffico dei pod in uscita dai pod in modo imprevisto.

Configurazione di esempio

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  advancedNetworking:
    enableV4Egress: false
```

Considerazioni sulla disabilitazione di EnableV4egress

- Politica di rete nel cluster: i IPv6 cluster consentono il traffico per impostazione predefinita. IPv6 IPv4 L'impostazione `enableV4Egress: false` blocca il traffico in IPv4 uscita, fornendo una maggiore sicurezza, specialmente se utilizzata con le politiche di rete.

Crea un pool di nodi per EKS Auto Mode

I pool di nodi Amazon EKS forniscono un modo flessibile per gestire le risorse di calcolo nel cluster Kubernetes. Questo argomento mostra come creare e configurare pool di nodi usando Karpenter, uno strumento di provisioning dei nodi che aiuta a ottimizzare il dimensionamento dei cluster e l'utilizzo delle risorse. Con la risorsa NodePool di Karpenter, puoi definire requisiti specifici per le tue risorse di calcolo, inclusi i tipi di istanze, le zone di disponibilità, le architetture e i tipi di capacità.

Non è possibile modificare il `system` integrato e i pool di nodi `general-purpose`. È solo possibile abilitarli o disabilitarli. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".

La specifica NodePool permette un controllo granulare delle risorse di elaborazione del cluster EKS tramite diverse etichette e requisiti supportati. Queste comprendono opzioni per specificare le categorie di istanze EC2, le configurazioni della CPU, le zone di disponibilità, le architetture (ARM64/AMD64) e i tipi di capacità (spot o on-demand). È inoltre possibile impostare limiti di risorse per l'utilizzo della CPU e della memoria, garantendo che il cluster rimanga entro i limiti operativi richiesti.

EKS Auto Mode sfrutta le note etichette Kubernetes per offrire metodi coerenti e standardizzati di identificazione delle caratteristiche dei nodi. Queste etichette, come `topology.kubernetes.io/zone` per le zone di disponibilità e `kubernetes.io/arch` per l'architettura della CPU, seguono le convenzioni stabilite da Kubernetes. Inoltre, le etichette specifiche EKS (precedute dal prefisso `eks.amazonaws.com/`) estendono questa funzionalità con attributi specifici AWS quali tipi di istanza, produttori di CPU, capacità GPU e specifiche di rete. Questo sistema di etichettatura standardizzato consente un'integrazione ottimale con gli strumenti Kubernetes esistenti, fornendo al tempo stesso una profonda integrazione dell'infrastruttura AWS.

Creazione di NodePool

Segui i seguenti passaggi per creare un NodePool per il cluster Amazon EKS:

1. Crea un file YAML denominato `nodepool.yaml` con la configurazione NodePool richiesta. Puoi utilizzare la configurazione di esempio seguente.
2. Applica il NodePool al cluster:

```
kubectl apply -f nodepool.yaml
```

3. Verificare che il NodePool sia stato creato correttamente:

```
kubectl get nodepools
```

4. (Facoltativo) Monitora lo stato del NodePool:

```
kubectl describe nodepool default
```

Assicurati che il NodePool faccia riferimento a una NodeClass valida ed esistente nel tuo cluster. NodeClass definisce configurazioni specifiche AWS per le risorse di calcolo. Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

NodePool di esempio

```
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
```

Etichette supportate da EKS Auto Mode

EKS Auto Mode supporta le seguenti etichette note.

> **Note**
>
> EKS Auto Mode utilizza etichette diverse rispetto a Karpenter. Le etichette relative alle istanze gestite da EC2 iniziano con `eks.amazonaws.com`.

Etichetta	Esempio	Descrizione
topology.kubernetes.io/zone	us-east-2a	Regione AWS
node.kubernetes.io/instance-type	g4dn.8xlarge	Tipo di istanza AWS
kubernetes.io/arch	amd64	Le architetture sono definite dai valori GOARCH sull'istanza
karpenter.sh/capacity-type	spot	I tipi di capacità includono `spot`, `on-demand`
eks.amazonaws.com/instance-hypervisor	nitro	Tipi di istanze che usano un hypervisor specifico
eks.amazonaws.com/compute-type	auto	Identifica i nodi gestiti da EKS Auto Mode
eks.amazonaws.com/instance-encryption-in-transit-supported	true	Tipi di istanza che supportano (o non supportano) la crittografia in transito
eks.amazonaws.com/instance-category	g	Tipi di istanza della stessa categoria, in genere la stringa che precede il numero di generazione
eks.amazonaws.com/instance-generation	4	Numero di generazione del tipo di istanza in una categoria di istanza
eks.amazonaws.com/instance-family	g4dn	Tipi di istanza con proprietà simili ma quantità di risorse differenti
eks.amazonaws.com/instance-size	8xlarge	Tipi di istanza con quantità di risorse simili ma proprietà differenti

Etichetta	Esempio	Descrizione
eks.amazonaws.com/instance-cpu	32	Numero di CPU sull'istanza
eks.amazonaws.com/instance-cpu-manufacturer	aws	Nome del produttore della CPU
eks.amazonaws.com/instance-memory	131072	Numero di mebibyte di memoria sull'istanza
eks.amazonaws.com/instance-ebs-bandwidth	9500	Numero massimo di megabit di EBS disponibili sull'istanza
eks.amazonaws.com/instance-network-bandwidth	131072	Numero di megabit di base disponibili sull'istanza
eks.amazonaws.com/instance-gpu-name	t4	Nome della GPU nell'istanza, se disponibile
eks.amazonaws.com/instance-gpu-manufacturer	nvidia	Nome del produttore della GPU
eks.amazonaws.com/instance-gpu-count	1	Numero di GPU nell'istanza
eks.amazonaws.com/instance-gpu-memory	16384	Numero di mebibyte di memoria sulla GPU
eks.amazonaws.com/instance-local-nvme	900	Gibibyte di storage nvme locale nell'istanza

Note

EKS Auto Mode supporta solo certe istanze e prevede dei requisiti minimi di dimensione. Per ulteriori informazioni, consulta the section called "Riferimento all'istanza supportata dalla modalità automatica EKS".

Etichette non supportate da EKS Auto Mode

EKS Auto Mode non supporta le seguenti etichette.

- EKS Auto Mode supporta soltanto Linux
  - `node.kubernetes.io/windows-build`
  - `kubernetes.io/os`

Disattivare i pool di nodi integrati

Se crei pool di nodi personalizzati, puoi disattivare i pool di nodi integrati. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".

Cluster senza pool di nodi integrati

Puoi creare un cluster senza pool di nodi integrati. È utile quando l'organizzazione ha creato pool di nodi personalizzati.

> **Note**
>
> Quando crei un cluster senza pool di nodi integrati, il provisioning di NodeClass `default` non viene eseguito automaticamente. Sarà necessario creare una NodeClass personalizzata. Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

Panoramica:

1. Crea un cluster EKS con i valori `nodePools` e `nodeRoleArn` vuoti.
   - Esempio eksctl `autoModeConfig`:

     ```
     autoModeConfig:
       enabled: true
       nodePools: []
       # Do not set a nodeRoleARN
     ```

     Per ulteriori informazioni, consulta the section called "CLI di eksctl"
2. Crea una classe di nodi personalizzata con un ruolo del nodo ARN
   - Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi"

3. Crea una voce di accesso per la classe di nodi personalizzata
   - Per ulteriori informazioni, consulta the section called "Creazione di una voce di accesso alla classe di nodi"
4. Crea un pool di nodi personalizzato come descritto sopra.

Interruzione

Puoi configurare EKS Auto Mode in modo che interrompa i nodi tramite NodePool in diversi modi. Puoi usare `spec.disruption.consolidationPolicy`, `spec.disruption.consolidateAfter` o `spec.template.spec.expireAfter`. Puoi anche limitare l'interruzione di EKS Auto Mode tramite il `spec.disruption.budgets` del NodePool. Puoi inoltre controllare le finestre temporali e il numero di nodi simultanei interrotti. Per istruzioni sulla configurazione di questo comportamento, consulta la pagina Disruption nella documentazione di Karpenter.

Puoi configurare l'interruzione per i pool di nodi per:

- Identifica le istanze sottoutilizzate e consolida i carichi di lavoro.
- Crea un budget per le interruzioni dei pool di nodi per limitare le interruzioni dei nodi causate da deviazioni, vuoto e consolidamento.

Per impostazione predefinita, EKS Auto Mode:

- Consolida le istanze sottoutilizzate.
- Termina le istanze dopo 336 ore.
- Imposta un budget unico per le interruzioni pari al 10% dei nodi.
- Permette la sostituzione dei nodi a causa della deriva quando viene rilasciata una nuova AMI in Auto Mode, cosa che avviene circa una volta alla settimana.

Periodo di tolleranza della terminazione

Quando un `terminationGracePeriod` non è definito esplicitamente su un EKS Auto NodePool, il sistema applica automaticamente un periodo di tolleranza della terminazione predefinito di 24 ore al NodeClaim associato. Sebbene i clienti EKS Auto non vedranno un termine `terminationGracePeriod` predefinito nelle loro configurazioni NodePool personalizzate, osserveranno questo valore predefinito su NodeClaim. La funzionalità rimane coerente sia che

il periodo di tolleranza sia impostato esplicitamente sul NodePool sia che sia predefinito sul NodeClaim, garantendo così un comportamento prevedibile di terminazione dei nodi in tutto il cluster.

Pool di nodi a capacità statica in modalità automatica EKS

La modalità automatica di Amazon EKS supporta pool di nodi a capacità statica che mantengono un numero fisso di nodi indipendentemente dalla domanda del pod. I pool di nodi con capacità statica sono utili per carichi di lavoro che richiedono capacità prevedibile, istanze riservate o requisiti di conformità specifici in cui è necessario mantenere un ingombro dell'infrastruttura coerente.

A differenza dei pool di nodi dinamici che si ridimensionano in base alle richieste di pianificazione dei pod, i pool di nodi con capacità statica mantengono il numero di nodi configurati.

Esempio di base

Ecco un semplice pool di nodi con capacità statica che mantiene 5 nodi:

```
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
```

Configurare un pool di nodi con capacità statica

Per creare un pool di nodi con capacità statica, imposta il `replicas` campo nelle tue specifiche. NodePool Il `replicas` campo definisce il numero esatto di nodi che il pool di nodi manterrà.

Vincoli del pool di nodi con capacità statica

I pool di nodi con capacità statica presentano diversi vincoli e comportamenti importanti:

Vincoli di configurazione:

- Impossibile cambiare modalità: una volta impostato `replicas` un pool di nodi, non è possibile rimuoverlo. Il pool di nodi non può passare dalla modalità statica a quella dinamica.
- Limiti di risorse limitati: nella sezione dei limiti è supportato solo il `limits.nodes` campo. I limiti di CPU e memoria non sono applicabili.
- Nessun campo di peso: il `weight` campo non può essere impostato su pool di nodi con capacità statica poiché la selezione dei nodi non si basa sulla priorità.

Comportamento operativo:

- Nessun consolidamento: i nodi nei pool di capacità statica non vengono presi in considerazione per il consolidamento in base all'utilizzo.
- Operazioni di scalabilità: le operazioni di scalabilità evitano i budget relativi alle interruzioni dei nodi, ma rispettano comunque i requisiti. PodDisruptionBudgets
- Sostituzione dei nodi: i nodi vengono comunque sostituiti per eventuali variazioni (come gli aggiornamenti AMI) e per scadenza in base alla configurazione.

Ridimensiona un pool di nodi con capacità statica

È possibile modificare il numero di repliche in un pool di nodi con capacità statica utilizzando il comando: `kubectl scale`

```
# Scale down to 5 nodes
kubectl scale nodepool static-nodepool --replicas=5
```

In caso di ridimensionamento, EKS Auto Mode terminerà i nodi in modo corretto, rispettando PodDisruptionBudgets e consentendo la riprogrammazione dei pod in esecuzione sui nodi rimanenti.

Monitora i pool di nodi con capacità statica

Usa i seguenti comandi per monitorare i tuoi pool di nodi a capacità statica:

```
# 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}'
```

Il `status.nodes` campo mostra il numero attuale di nodi gestiti dal pool di nodi, che dovrebbe corrispondere al `replicas` numero desiderato in condizioni normali.

Configurazioni di esempio

Pool di nodi con capacità statica di base

```
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
```

Capacità statica con tipi di istanze specifici

```
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%
```

Pool di nodi con capacità statica multizona

```
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%
```

Best practice

Pianificazione della capacità:

- Impostato su un `limits.nodes` valore superiore `replicas` a per consentire il ridimensionamento temporaneo durante le operazioni di sostituzione dei nodi.
- Quando imposti i limiti, considera la capacità massima necessaria durante la deriva dei nodi o gli aggiornamenti AMI.

Selezione dell'istanza:

- Utilizza tipi di istanze specifici quando hai istanze riservate o requisiti hardware specifici.
- Evita requisiti eccessivamente restrittivi che potrebbero limitare la disponibilità delle istanze durante la scalabilità.

Gestione delle interruzioni:

- Configura budget adeguati per le interruzioni per bilanciare la disponibilità con le operazioni di manutenzione.
- Considerate la tolleranza della vostra applicazione per la sostituzione dei nodi quando impostate le percentuali di budget.

Monitoraggio:

- Monitora regolarmente il `status.nodes` campo per garantire il mantenimento della capacità desiderata.
- Imposta avvisi per quando il numero effettivo di nodi si discosta dalle repliche desiderate.

Distribuzione delle zone:

- Per un'elevata disponibilità, distribuisci la capacità statica su più zone di disponibilità.
- Quando si crea un pool di nodi a capacità statica che si estende su più zone di disponibilità, EKS Auto Mode distribuisce i nodi tra le zone specificate, ma non è garantito che la distribuzione sia uniforme.
- Per una distribuzione prevedibile e uniforme tra le zone di disponibilità, crea pool di nodi con capacità statica separati, ciascuno collegato a una zona di disponibilità specifica utilizzando il requisito. `topology.kubernetes.io/zone`
- Se hai bisogno di 12 nodi distribuiti uniformemente su tre zone, crea tre pool di nodi con 4 repliche ciascuno, anziché un pool di nodi con 12 repliche su tre zone.

risoluzione dei problemi

I nodi non raggiungono le repliche desiderate:

- Controlla se il `limits.nodes` valore è sufficiente
- Verifica che i requisiti non limitino eccessivamente la selezione delle istanze
- Controlla le quote AWS di servizio per i tipi di istanze e le regioni che stai utilizzando

La sostituzione dei nodi richiede troppo tempo:

- Modifica i budget per le interruzioni per consentire più sostituzioni simultanee
- Controlla se PodDisruptionBudgets stanno impedendo la chiusura del nodo

Chiusura imprevista del nodo:

- Rivedi le impostazioni `expireAfter` e `terminationGracePeriod`
- Verifica la presenza di interruzioni manuali dei nodi o di eventi AWS di manutenzione

Creare e IngressClass configurare un Application Load Balancer

La modalità automatica di EKS automatizza le attività di routine per il bilanciamento del carico, inclusa l'esposizione delle app del cluster a Internet.

AWS suggerisce di utilizzare Application Load Balancers (ALB) per servire il traffico HTTP e HTTPS. Gli Application Load Balancer possono instradare le richieste in base al contenuto della richiesta. Per ulteriori informazioni sugli Application Load Balancer, consulta What is Elastic Load Balancing?

EKS Auto Mode crea e configura Application Load Balancer (). ALBs Ad esempio, modalità automatica di EKS crea un bilanciatore del carico quando crei un oggetto Kubernetes `Ingress` e lo configura per indirizzare il traffico verso il carico di lavoro del cluster.

Panoramica

1. Crea un carico di lavoro da esporre a Internet.
2. Crea una `IngressClassParams` risorsa, specificando valori di configurazione AWS specifici come il certificato da utilizzare per le SSL/TLS sottoreti VPC.

3. Creare una risorsa `IngressClass`, specificando che modalità automatica di EKS sarà il controller della risorsa.
4. Crea una risorsa `Ingress` da associare a un percorso e una porta HTTP a un carico di lavoro del cluster.

La modalità automatica di EKS creerà un Application Load Balancer che punta al carico di lavoro specificato nella risorsa `Ingress`, utilizzando la configurazione del bilanciatore del carico specificata nella risorsa `IngressClassParams`.

Prerequisiti

- La modalità automatica di EKS su un cluster Amazon EKS
- Kubectl configurato per connettersi al cluster
  - Puoi usare `kubectl apply -f <filename>` per applicare i file di configurazione di esempio YAML riportati di seguito al cluster.

> **Note**
>
> La modalità automatica di EKS richiede i tag di sottorete per identificare le sottoreti pubbliche e private.
> Se hai creato il cluster con `eksctl`, disponi già di questi tag.
> Informazioni su come the section called "Assegnazione di tag alle sottoreti".

Fase 1: Creare un carico di lavoro

Per iniziare, crea un carico di lavoro da esporre a Internet. Può trattarsi di qualsiasi risorsa di Kubernetes che serve il traffico HTTP, ad esempio un'implementazione o un servizio.

Questo esempio utilizza un semplice servizio HTTP chiamato `service-2048` che ascolta sulla porta `80`. Creare questo servizio e la sua implementazione applicando il seguente 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
```

Applica la configurazione al cluster:

```
kubectl apply -f 2048-deployment-service.yaml
```

Le risorse sopra elencate verranno create nel namespace predefinito. Puoi verificarlo eseguendo il comando seguente:

```
kubectl get all -n default
```

Fase 2: Creare IngressClassParams

Crea un `IngressClassParams` oggetto per specificare opzioni di configurazione AWS specifiche per l'Application Load Balancer. In questo esempio, creiamo una risorsa `IngressClassParams` denominata `alb` (che verrà utilizzata nella fase successiva) che specifica lo schema del bilanciatore del carico come `internet-facing` in un file chiamato `alb-ingressclassparams.yaml`.

```
apiVersion: eks.amazonaws.com/v1
kind: IngressClassParams
metadata:
  name: alb
spec:
  scheme: internet-facing
```

Applica la configurazione al cluster:

```
kubectl apply -f alb-ingressclassparams.yaml
```

Fase 3: Creare IngressClass

Crea un file `IngressClass` che faccia riferimento ai valori di configurazione AWS specifici impostati nella `IngressClassParams` risorsa in un file denominato`alb-ingressclass.yaml`. Annotare il nome della `IngressClass`. In questo esempio, sia `IngressClass` che `IngressClassParams` sono denominati `alb`.

Utilizzare l'annotazione `is-default-class` per controllare se le risorse `Ingress` devono utilizzare questa classe per impostazione predefinita.

```
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
```

Per ulteriori informazioni sulle opzioni di configurazione, consultare the section called "IngressClassParams Riferimento".

Applica la configurazione al cluster:

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

Fase 4: Creare Ingress

Creare una risorsa `Ingress` in un file denominato `alb-ingress.yaml`. Lo scopo di questa risorsa è associare percorsi e porte sull'Application Load Balancer ai carichi di lavoro del cluster. Per questo esempio, creiamo una risorsa `Ingress` denominata `2048-ingress` che instrada il traffico verso un servizio denominato `service-2048` sulla porta 80.

Per ulteriori informazioni sulla configurazione di questa risorsa, consultare Ingress nella documentazione su 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
```

Applica la configurazione al cluster:

```
kubectl apply -f alb-ingress.yaml
```

Fase 5: Controllare lo stato

Utilizza `kubectl` per trovare lo stato di `Ingress`. Prima che il bilanciatore del carico sia disponibile, potrebbero essere necessari alcuni minuti.

Utilizza il nome della risorsa `Ingress` impostata nella fase precedente. Esempio:

```
kubectl get ingress 2048-ingress
```

Una volta che la risorsa è pronta, recuperare il nome di dominio del bilanciatore del carico.

```
kubectl get ingress 2048-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}'
```

Per visualizzare il servizio in un browser web, esamina la porta e il percorso specificati nel file di ripristino `Ingress`.

Fase 6: Eliminazione

Per eliminare il bilanciatore del carico, utilizza il seguente comando:

```
kubectl delete ingress 2048-ingress
kubectl delete ingressclass alb
kubectl delete ingressclassparams alb
```

EKS Auto Mode eliminerà automaticamente il sistema di bilanciamento del carico associato nel tuo AWS account.

IngressClassParams Riferimento

La tabella seguente è un riferimento rapido per le opzioni di configurazione più comunemente utilizzate.

Campo	Description	Valore di esempio
`scheme`	Definisce se l'ALB è interno o esposto a Internet	`internet-facing`

Campo	Description	Valore di esempio
`namespaceSelector`	Limita quali namespace possono utilizzarlo IngressCl ass	`environment: prod`
`group.name`	Raggruppa più ingressi per condividere un singolo ALB	`retail-apps`
`ipAddressType`	Imposta il tipo di indirizzo IP per ALB	`dualstack`
`subnets.ids`	Elenco di sottoreti per la distribuzione ALB IDs	`subnet-xxxx, subnet-yyyy`
`subnets.tags`	Filtri di tag per selezionare le sottoreti per ALB	`Environment: prod`
`certificateARNs`	ARNs di certificati SSL da utilizzare	`arn:aws: acm:region:account:certificate/id`
`tags`	Tag personalizzati per le risorse AWS	`Environment: prod, Team: platform`
`loadBalancerAttributes`	Attributi specifici del bilanciatore del carico	`idle_timeout.timeout_seconds: 60`

Considerazioni

- Non è possibile utilizzare Annotations su un PC IngressClass per configurare i sistemi di bilanciamento del carico con EKS Auto Mode.
- Non è possibile impostare ListenerAttributecon la modalità automatica EKS.
- È necessario aggiornare il ruolo Cluster IAM per abilitare la propagazione dei tag dalle risorse Kubernetes alle risorse Load AWS Balancer. Per ulteriori informazioni, consulta the section called "Tag personalizzati per AWS le risorse EKS Auto".

- Per informazioni sull'associazione di risorse a EKS Auto Mode o al Load AWS Balancer Controller autogestito, vedere. the section called "Materiale di riferimento per le migrazioni"
- Per informazioni sulla risoluzione dei problemi con i bilanciatori del carico, consulta the section called "Risoluzione dei problemi".
- Per ulteriori considerazioni sull'utilizzo della funzionalità di bilanciamento del carico della modalità automatica di EKS, consulta the section called "Sistema di bilanciamento del carico".

Le tabelle seguenti forniscono un confronto dettagliato delle modifiche IngressClassParams, delle annotazioni di Ingress e delle TargetGroupBinding configurazioni per EKS Auto Mode. Queste tabelle evidenziano le principali differenze tra la capacità di bilanciamento del carico della modalità automatica di EKS e il controller del bilanciatore del carico open source, comprese le modifiche alla versione dell'API, le funzionalità obsolete e i nomi dei parametri aggiornati.

IngressClassParams

Precedente	Novità	Description
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Modifica alla versione dell'API
`spec.certificateArn`	`spec.certificateARNs`	Support per più certificati ARNs
`spec.subnets.tags`	`spec.subnets.matchTags`	Schema di corrispondenza delle sottoreti modificato
`spec.listeners.listenerAttributes`	Non supportata	Non ancora supportato dalla modalità automatica di EKS

Annotazioni di ingresso

Precedente	Novità	Description
`kubernetes.io/ingress.class`	Non supportata	Utilizzare `spec.ingressClassName` su oggetti di ingresso

Precedente	Novità	Description
`alb.ingress.kubernetes.io/group.name`	Non supportata	Specificate i gruppi IngressClass solo in
`alb.ingress.kubernetes.io/waf-acl-id`	Non supportata	Usare invece WAF v2
`alb.ingress.kubernetes.io/web-acl-id`	Non supportata	Usare invece WAF v2
`alb.ingress.kubernetes.io/shield-advanced-protection`	Non supportata	Integrazione Shield disattivata
`alb.ingress.kubernetes.io/auth-type: oidc`	Non supportata	Il tipo di autenticazione OIDC non è attualmente supportato

TargetGroupBinding

Precedente	Novità	Description
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Modifica alla versione dell'API
`spec.targetType` facoltativo	`spec.targetType` obbligatorio	Specifica esplicita del tipo di destinazione
`spec.networking.ingress.from`	Non supportata	Non supporta più NLB senza gruppi di sicurezza

Per utilizzare la TargetGroupBinding funzionalità personalizzata, devi etichettare il gruppo target con il tag eks: eks-cluster-name tag with cluster name per concedere al controller le autorizzazioni IAM necessarie. Tieni presente che il controller eliminerà il gruppo di destinazione quando la TargetGroupBinding risorsa o il cluster vengono eliminati.

Usa Annotazioni del servizio per configurare Network Load Balancer

Scopri come configurare Network Load Balancer (NLB) in Amazon EKS utilizzando le annotazioni del servizio di Kubernetes. Questo argomento spiega le annotazioni supportate da EKS Auto Mode per personalizzare il comportamento NLB, tra cui accessibilità Internet, controlli di integrità, SSL/TLS terminazione e modalità di targeting IP.

Quando crei un servizio Kubernetes di tipo `LoadBalancer` in modalità automatica EKS, EKS effettua automaticamente il provisioning e configura un AWS Network Load Balancer in base alle annotazioni specificate. Questo approccio dichiarativo ti consente di gestire le configurazioni del bilanciatore del carico direttamente tramite i manifesti di Kubernetes, mantenendo l'infrastruttura come pratiche di codice.

EKS Auto Mode gestisce il provisioning di Network Load Balancer per impostazione predefinita per tutti i servizi di tipo LoadBalancer : non è richiesta alcuna installazione o configurazione di controller aggiuntivi. La specifica `loadBalancerClass: eks.amazonaws.com/nlb` viene impostata automaticamente come impostazione predefinita del cluster, semplificando il processo di implementazione e mantenendo la compatibilità con i carichi di lavoro di Kubernetes esistenti.

> **Note**
>
> La modalità automatica di EKS richiede i tag di sottorete per identificare le sottoreti pubbliche e private.
> Se hai creato il cluster con `eksctl`, disponi già di questi tag.
> Informazioni su come the section called "Assegnazione di tag alle sottoreti".

Servizio di esempio

Per ulteriori informazioni sulla risorsa `Service` di Kubernetes, consulta Kubernetes Documentation.

Consulta la risorsa di esempio `Service` riportata di seguito:

```
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
```

Annotazioni di uso comune

La tabella seguente elenca le annotazioni di uso comune supportate dalla modalità automatica di EKS. La modalità automatica di EKS potrebbe non supportare tutte le annotazioni.

> **Tip**
>
> Tutte le seguenti annotazioni devono avere il prefisso `service.beta.kubernetes.io/`

Campo	Description	Esempio
`aws-load-balancer-type`	Specifica il tipo di bilanciatore del carico. Utilizza `external` per nuove implementazioni.	`external`
`aws-load-balancer-nlb-target-type`	Speciifica se indirizzare il traffico verso le istanze del nodo o direttamente verso il pod. IPs Utilizza `instance` per implementazioni standard o `ip` per l'instradamento diretto ai pod.	`instance`

Campo	Description	Esempio
`aws-load-balancer-scheme`	Controlla se il bilanciatore del carico è interno o esposto a Internet.	`internet-facing`
`aws-load-balancer-healthcheck-protocol`	Protocollo di controllo dell'integrità per il gruppo di destinazione. Le opzioni più comuni sono `TCP` (impostazione predefinita) o `HTTP`.	`HTTP`
`aws-load-balancer-healthcheck-path`	Il percorso HTTP per i controlli di integrità quando si utilizza HTTP/HTTPS il protocollo.	`/healthz`
`aws-load-balancer-healthcheck-port`	Porta utilizzata per i controlli dell'integrità. Può essere un numero di porta specifico o `traffic-port` .	`traffic-port`
`aws-load-balancer-subnets`	Specifica in quali sottoreti creare il bilanciatore del carico. Può usare sottoreti IDs o nomi.	`subnet-xxxx, subnet-yyyy`
`aws-load-balancer-ssl-cert`	ARN del certificato SSL di Certificate Manager for AWS HTTPS/TLS.	`arn:aws: acm:region:account:certificate/cert-id`
`aws-load-balancer-ssl-ports`	Specifica quali porte devono utilizzare SSL/TLS.	`443, 8443`
`load-balancer-source-ranges`	Intervalli CIDR autorizzati ad accedere al bilanciatore del carico.	`10.0.0.0/24, 192.168.1.0/24`

Campo	Description	Esempio
`aws-load-balancer-additional-resource-tags`	AWS Tag aggiuntivi da applicare al sistema di bilanciamento del carico e alle risorse correlate.	`Environment=prod,Team=platform`
`aws-load-balancer-ip-address-type`	Speciifica se il load balancer utilizza IPv4 o dual-stack (+). IPv4 IPv6	`ipv4` o `dualstack`

Considerazioni

- È necessario aggiornare il ruolo Cluster IAM per abilitare la propagazione dei tag dalle risorse Kubernetes alle risorse Load AWS Balancer. Per ulteriori informazioni, consulta the section called "Tag personalizzati per AWS le risorse EKS Auto".
- Per informazioni sull'associazione di risorse a EKS Auto Mode o al Load AWS Balancer Controller autogestito, vedere. the section called "Materiale di riferimento per le migrazioni"
- Per informazioni sulla risoluzione dei problemi con i bilanciatori del carico, consulta the section called "Risoluzione dei problemi".
- Per ulteriori considerazioni sull'utilizzo della funzionalità di bilanciamento del carico della modalità automatica di EKS, consulta the section called "Sistema di bilanciamento del carico".

Durante la migrazione alla modalità automatica di EKS per il bilanciamento del carico, sono necessarie diverse modifiche nelle annotazioni dei servizi e nelle configurazioni delle risorse. Le tabelle seguenti descrivono le differenze principali tra le implementazioni precedenti e quelle nuove, includendo le opzioni non supportate e le alternative consigliate.

Annotazioni dei servizi

Precedente	Novità	Description
`service.beta.kubernetes.io/load-balancer-source-ranges`	Non supportata	Usa `spec.loadBalancerSourceRanges` sul servizio

Precedente	Novità	Description
`service.beta.kubernetes.io/aws-load-balancer-type`	Non supportata	Usa `spec.loadBalancerClass` sul servizio
`service.beta.kubernetes.io/aws-load-balancer-internal`	Non supportata	Utilizzare `service.beta.kubernetes.io/aws-load-balancer-scheme`
`service.beta.kubernetes.io/aws-load-balancer-proxy-protocol`	Non supportata	Usa invece `service.beta.kubernetes.io/aws-load-balancer-target-group-attributes`
Vari attributi del bilanciatore del carico	Non supportata	Utilizzare `service.beta.kubernetes.io/aws-load-balancer-attributes`
`service.beta.kubernetes.io/aws-load-balancer-access-log-enabled`	Non supportata	Usa invece `service.beta.kubernetes.io/aws-load-balancer-attributes`
`service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-name`	Non supportata	Usa invece `service.beta.kubernetes.io/aws-load-balancer-attributes`
`service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-prefix`	Non supportata	Usa invece `service.beta.kubernetes.io/aws-load-balancer-attributes`

Precedente	Novità	Description
`service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled`	Non supportata	Usa invece `service.beta.kubernetes.io/aws-load-balancer-attributes`

Per eseguire la migrazione dalle annotazioni obsolete degli attributi del bilanciatore del carico, consolida queste impostazioni nell'annotazione `service.beta.kubernetes.io/aws-load-balancer-attributes`. Questa annotazione accetta un elenco di coppie chiave-valore separato da virgole per vari attributi del bilanciatore del carico. Ad esempio, per specificare la registrazione degli accessi e il bilanciamento del carico tra zone, utilizzate il seguente 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
```

Questo formato consolidato offre un modo più coerente e flessibile per configurare gli attributi del bilanciatore del carico riducendo al contempo il numero di singole annotazioni necessarie. Rivedi le configurazioni del servizio esistenti e aggiornale per utilizzare questo formato consolidato.

TargetGroupBinding

Precedente	Novità	Description
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Modifica alla versione dell'API
`spec.targetType` facoltativo	`spec.targetType` obbligatorio	Specifica esplicita del tipo di destinazione
`spec.networking.ingress.from`	Non supportata	Non supporta più NLB senza gruppi di sicurezza

Nota: per utilizzare la TargetGroupBinding funzionalità personalizzata, è necessario etichettare il gruppo di destinazione con il tag con il `eks:eks-cluster-name` nome del cluster per concedere

al controller le autorizzazioni IAM necessarie. Tieni presente che il controller eliminerà il gruppo di destinazione quando la TargetGroupBinding risorsa o il cluster vengono eliminati.

Crea una classe di archiviazione

Una `StorageClass` in Amazon EKS Auto Mode definisce come i volumi Amazon EBS vengono forniti automaticamente quando le applicazioni richiedono uno storage persistente. In questa pagina viene spiegato come creare e configurare una `StorageClass` che funzioni con Amazon EKS Auto Mode per il provisioning dei volumi EBS.

Configurando una `StorageClass`, è possibile specificare le impostazioni predefinite per i volumi EBS, inclusi il tipo di volume, la crittografia, gli IOPS e altri parametri di storage. Puoi anche configurare l'utilizzo delle `StorageClass` chiavi AWS KMS per la gestione della crittografia.

EKS Auto Mode non crea una `StorageClass` per te. Devi creare una `StorageClass` che faccia riferimento a `ebs.csi.eks.amazonaws.com` per usare la funzionalità di storage di EKS Auto Mode.

Per prima cosa, crea un file denominato `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"
```

Poi applica la classe di archiviazione al cluster.

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

Componenti chiave:

- `provisioner: ebs.csi.eks.amazonaws.com`: usa EKS Auto Mode
- `allowedTopologies`: specificando `matchLabelExpressions` in modo che corrisponda con `eks.amazonaws.com/compute-type:auto`, se i pod richiedono un volume su cui effettuare il provisioning automatico usando Auto Mode, i pod non verranno pianificati su nodi non automatici.
- `volumeBindingMode: WaitForFirstConsumer`: ritarda la creazione del volume fino a quando un pod non ne ha bisogno
- `type: gp3`: specifica il tipo di volume EBS
- `encrypted: "true"`: EBS crittograferà tutti i volumi creati usando la `StorageClass`. EBS userà l'alias chiave predefinito `aws/ebs`. Per ulteriori informazioni, consulta How Amazon EBS encryption works nella Guida per l'utente di Amazon EBS. Questo valore è opzionale ma consigliato.
- `storageclass.kubernetes.io/is-default-class: "true"`: Kubernetes userà questa classe di archiviazione per impostazione predefinita, a meno che non venga specificata una classe di volume diversa su una dichiarazione di volume persistente. Questo valore è facoltativo. Presta attenzione quando imposti questo valore se stai effettuando la migrazione da un controller di storage diverso.

Usa la chiave KMS autogestita per crittografare i volumi EBS

Per usare una chiave KMS autogestita per crittografare i volumi EBS automatizzati da EKS Auto Mode, bisogna:

1. Creare una chiave KMS gestita dal cliente.
   - Per ulteriori informazioni, consulta Create a symmetric encryption KMS key o How Amazon Elastic Block Store (Amazon EBS) uses KMS nella Guida per gli utenti di KMS.
2. Creare una nuova policy che consenta l'accesso alla chiave KMS.
   - Usa la policy IAM di esempio riportata di seguito per creare la policy. Inserisci l'ARN della nuova chiave KMS autogestita. Per ulteriori informazioni, consulta Creazione di ruoli e collegamento di politiche (console) nella Guida per l'utente AWS IAM.
3. Collegare la policy al ruolo del cluster EKS.
   - Usa la AWS console per trovare l'ARN dell'EKS Cluster Role. Le informazioni relative al ruolo sono visibili nella sezione Panoramica. Per ulteriori informazioni, consulta the section called "Ruolo IAM del cluster".

4. Aggiornare la `StorageClass` per fare riferimento all'ID della chiave KMS nel campo `parameters.kmsKeyId`.

Esempio di policy IAM KMS autogestita

Aggiorna i valori seguenti nella policy riportata di seguito:

- `<account-id>`— L'ID AWS del tuo account, ad esempio `111122223333`
- `<aws-region>`— La AWS regione del cluster, ad esempio `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"
            }
        }
    }
  ]
}
```

Esempio di **`StorageClass`** KMS gestita dal cliente

```
parameters:
  type: gp3
  encrypted: "true"
  kmsKeyId: <custom-key-arn>
```

Guida di riferimento per i parametri **`StorageClass`**

Per informazioni generali sulle risorse `StorageClass` di Kubernetes, consulta Storage Classes nella documentazione di Kubernetes.

THe `parameters`la sezione della `StorageClass` risorsa è specifica per AWS. Utilizza la tabella seguente per esaminare le opzioni disponibili.

Parameters	Valori	Predefinita	Description
"csi.storage.k8s.io/fstype"	xfs, ext2, ext3, ext4	ext4	Tipo di file system che verrà formattato al momento della creazione del volume. Questo parametro prevede la distinzione tra maiuscole e minuscole.
"type"	io1, io2, gp2, gp3, sc1, st1, standard, sbp1, sbg1	gp3	Tipo di volume EBS.
"iopsPerGB"			Operazioni di I/O al secondo per GiB. Può

Parameters	Valori	Predefinita	Description
			essere specificato per IO1 IO2, e GP3 volumi.
«Consenti automatic amente IOPSPerGB Increase»	true, false	false	Quando`"true"`, il driver CSI aumenta gli IOPS per un volume troppo basso per rientrare nell'inte rvallo IOPS supportat o da. `iopsPerGB * <volume size>` AWS Ciò consente al provisioning dinamico di avere sempre esito positivo, anche quando l'utente specifica una capacità PVC o un valore `iopsPerGB` troppo bassi. D'altra parte, ciò potrebbe comportare costi aggiuntivi, poiché tali volumi hanno un IOPS superiore a quello richiesto in `iopsPerGB` .
"iops"			Operazioni di I/O al secondo. Può essere specificato per IO1 IO2, e volumi. GP3

Parameters	Valori	Predefinita	Description
“throughput”		125	Velocità di trasmissione effettiva in MiB/S. Efficace solo quando è specificato il tipo di volume gp3.
“encrypted”	true, false	false	Se il volume deve essere crittografato o no. I valori validi sono “true” o “false”.
“blockExpress”	true, false	false	Permette la creazione di Volumi io2 Block Express.
"kmsKeyId"			L'ARN completo della chiave da usare per la crittografia del volume. Se non specificato, AWS utilizzerà la chiave KMS predefinita per la regione in cui si trova il volume. Si tratta di una chiave generata automaticamente chiamata `/aws/ebs` se non viene modificata.

Parameters	Valori	Predefinita	Description
“blockSize”			La dimensione del blocco da utilizzar e per la formattaz ione del filesystem sottostante. Supportat a solo sui nodi linux e con fstype `ext2`, `ext3`, `ext4` o `xfs`.
“inodeSize”			La dimensione dell'inode da utilizzar e per la formattaz ione del filesystem sottostante. Supportat a solo sui nodi linux e con fstype `ext2`, `ext3`, `ext4` o `xfs`.
"bytesPerInode"			`bytes-per-inode` da utilizzare per la formattazione del filesystem sottostan te. Supportata solo sui nodi linux e con fstype `ext2`, `ext3`, `ext4`.
"numberOfInodes"			`number-of-inodes` da utilizzar e per la formattaz ione del filesystem sottostante. Supportat a solo sui nodi linux e con fstype `ext2`, `ext3`, `ext4`.

Parameters	Valori	Predefinita	Description
«BigAllocext4"	true, false	false	Modifica il filesyste m `ext4` per usare l'allocazione di blocchi in cluster abilitando l'opzione di formattazione `bigalloc`. Attenzion e: `bigalloc` potrebbe non essere completamente supportato dal kernel Linux del nodo.
«ClusterSizeext4»			La dimensione del cluster da usare per formattare un filesystem `ext4` quando la funzionalità `bigalloc` è abilitata . Nota: il parametro `ext4BigAlloc` deve essere impostato su true

Per ulteriori informazioni, consulta AWS EBS CSI Driver su. GitHub

Considerazioni

> **Note**
>
> È possibile distribuire carichi di lavoro solo in base alla modalità EKS Auto Mode sui nodi EKS Auto Mode StorageClasses . Se hai un cluster con tipi misti di nodi, devi configurare i carichi di lavoro in modo che vengano eseguiti solo sui nodi EKS Auto Mode. Per ulteriori informazioni, consulta the section called "Controllare l'implementazione".

La funzionalità di storage a blocchi di EKS Auto Mode è diversa da quella del driver CSI EBS.

- Provisioning statico
  - Se desideri utilizzare volumi EBS creati esternamente con EKS Auto Mode, devi aggiungere manualmente un AWS tag con la chiave `eks:eks-cluster-name` e il valore del nome del cluster.
- Taint startup del nodo
  - Non è possibile usare la funzionalità taint startup del nodo per impedire la pianificazione dei pod prima che la capacità di storage sia pronta
- Tag personalizzati su volumi con provisioning dinamico
  - Non è possibile usare il flag CLI extra-tag per configurare tag personalizzati su volumi EBS con provisioning dinamico
  - Puoi usare l'assegnazione di tag `StorageClass` per aggiungere tag personalizzati. EKS Auto Mode aggiungerà tag alle risorse associate. AWS Dovrai aggiornare il ruolo del IAM cluster per i tag personalizzati. Per ulteriori informazioni, consulta the section called "Tag personalizzati per AWS le risorse EKS Auto".
- Metriche sulle prestazioni dettagliate di EBS
  - Non puoi accedere alle metriche di Prometheus per le prestazioni dettagliate di EBS

Installa il componente aggiuntivo CSI Snapshot Controller

EKS Auto Mode è compatibile con il componente aggiuntivo CSI Snapshot Controller di Amazon EKS.

AWS suggerisce di configurare questo componente aggiuntivo per l'esecuzione nel pool di `system` nodi integrato.

Per ulteriori informazioni, consulta:

- the section called "Eseguire componenti aggiuntivi critici"
- the section called "Riesamina i pool di nodi integrati"
- the section called "Controller snapshot CSI"

Per installare il controller di snapshot nel pool di nodi di sistema

1. Apri il tuo cluster EKS nella console AWS

2. Nella scheda Componenti aggiuntivi, seleziona Ottieni altri componenti aggiuntivi
3. Seleziona il controller CSI Snapshot e poi Avanti
4. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona Impostazioni di configurazione opzionali per visualizzare lo schema di configurazione dei componenti aggiuntivi
   a. Inserisci il seguente yaml per associare il controller di snapshot al pool di nodi `system`. Il controller di snapshot include una tolleranza per il taint `CriticalAddonsOnly`.

```
{
        "nodeSelector": {
            "karpenter.sh/nodepool": "system"
        }
}
```

   b. Seleziona Avanti
5. Esamina la configurazione del componente aggiuntivo, poi seleziona Crea

Disabilita modalità automatica di EKS

Puoi disabilita modalità automatica di EKS su un cluster EKS esistente. Si tratta di un'operazione distruttiva.

- EKS interromperà tutte le EC2 istanze gestite da EKS Auto Mode.
- EKS eliminerà tutti i bilanciatori del carico gestiti dalla modalità automatica di EKS.
- EKS non eliminerà i volumi EBS forniti dalla modalità automatica di EKS.

modalità automatica di EKS è progettato per gestire completamente le risorse che crea. Gli interventi manuali potrebbero impedire alla modalità automatica di EKS di ripulire completamente tali risorse quando è disabilitata. Ad esempio, se hai fatto un riferimento a un gruppo di sicurezza gestito da regole esterne del gruppo di sicurezza e ti dimentica di rimuovere quel riferimento prima di disabilitare modalità automatica di EKS per un cluster, il gruppo di sicurezza gestito verrà divulgato (non verrà eliminato). Le fasi seguenti descrivono come rimuovere un gruppo di sicurezza trapelato se ciò dovesse accadere.

Disattiva la modalità automatica EKS (console)AWS

1. Apri la pagina di panoramica del cluster in Console di gestione AWS.

2. In modalità automatica di EKS, seleziona Gestisci
3. Attiva modalità automatica di EKS su `off`.

Se un gruppo di sicurezza gestito non viene eliminato al termine di questo processo, puoi eliminarlo manualmente utilizzando le descrizioni in Elimina un gruppo di sicurezza.

Disattiva la modalità automatica EKS (AWS CLI)

Utilizza il seguente comando per disabilitare modalità automatica di EKS su un cluster esistente.

Devi aver installato `aws` CLI ed essere connesso con autorizzazioni sufficienti per gestire i cluster EKS. Per ulteriori informazioni, consulta Configurazione.

> **Note**
>
> Le funzionalità di calcolo, archiviazione a blocchi e bilanciamento del carico devono essere tutte abilitate o disabilitate nella stessa richiesta.

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

Puoi verificare se non è stato possibile eliminare un gruppo di sicurezza della modalità automatica di EKS divulgato dopo aver disabilitato modalità automatica di EKS nel modo seguente:

```
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]"
```

Poi, per eliminare il gruppo di sicurezza:

```
aws ec2 delete-security-group --group-name=<sg-name>
```

Aggiornare la versione Kubernetes di un cluster di Amazon EKS Auto Mode

Questo argomento ti spiega come aggiornare la versione di Kubernetes per il cluster Auto Mode. Auto Mode semplifica il processo di aggiornamento delle versioni gestendo il coordinamento degli aggiornamenti del piano di controllo con la sostituzione dei nodi, mantenendo al contempo la disponibilità del carico di lavoro attraverso i budget relativi all'interruzione dei pod.

Quando si aggiorna un cluster Auto Mode, molti componenti che tradizionalmente richiedevano aggiornamenti manuali vengono ora gestiti come parte del servizio. Comprendere gli aspetti automatizzati del processo di aggiornamento e le proprie responsabilità aiuta a garantire una transizione fluida delle versioni per il cluster.

Informazioni sugli aggiornamenti con EKS Auto Mode

Dopo che hai avviato un aggiornamento del piano di controllo, EKS Auto Mode aggiornerà i nodi del cluster. Alla scadenza dei nodi, EKS Auto Mode li sostituirà con nuovi nodi. I nuovi nodi hanno la nuova versione di Kubernetes corrispondente. EKS Auto Mode rispetta i budget relativi alle interruzioni dei pod durante l'aggiornamento dei nodi.

Inoltre, non dovrai più aggiornare componenti come:

- CNI di Amazon VPC
- Controller del load balancerAWS
- CoreDNS
- `kube-proxy`
- Karpenter
- AWS Driver CSI per EBS

EKS Auto Mode sostituisce questi componenti con funzionalità di servizio.

L'utente è ancora responsabile dell'aggiornamento di:

- App e carichi di lavoro implementati nel cluster
- Componenti aggiuntivi e controller autogestiti
- Componenti aggiuntivi di Amazon EKS
  - Informazioni su come the section called "Aggiornamento di un componente aggiuntivo"

Informazioni su Best Practices for Cluster Upgrades

Avviare l'aggiornamento del cluster

Per avviare un aggiornamento del cluster, consulta the section called "Aggiornamento della versione di Kubernetes".

Attivazione o disattivazione della funzionalità integrata NodePools

La modalità automatica EKS ne ha due integrate NodePools. Puoi abilitarli o disabilitarli NodePools utilizzando la AWS console, la CLI o l'API.

Riferimento integrato NodePool

- `system`
  - Questo NodePool ha una `CriticalAddonsOnly` macchia. Molti componenti aggiuntivi EKS, come CoredNS, tollerano questo taint. Utilizza questo pool di nodi di sistema per separare le applicazioni critiche per il cluster.
  - Supporta entrambe le architetture `amd64` e `arm64`.
- `general-purpose`
  - Ciò NodePool fornisce supporto per l'avvio di nodi per carichi di lavoro generici nel cluster.
  - Utilizza solo l'architettura `amd64`.

Entrambi integrati: NodePools

- Usa l'EKS predefinito NodeClass
- Utilizza solo la capacità su richiesta EC2
- Usa le famiglie di EC2 istanze C, M e R
- Richiede istanze di quinta generazione o successive EC2

> **Note**
>
> NodePool È necessario abilitare almeno una funzionalità integrata per consentire a EKS di eseguire il provisioning delle impostazioni «predefinite». NodeClass Se disabiliti tutte le funzionalità integrate NodePools, dovrai crearne una personalizzata NodeClass e

configurarne una NodePool per utilizzarla. Per ulteriori informazioni su NodeClasses, consultathe section called "Creazione di una classe di nodi".

Procedura

Prerequisiti

- La versione più recente dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo. Per verificare la versione attuale, usa `aws --version`. Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando.
  - Accedi alla CLI con autorizzazioni IAM sufficienti per creare AWS risorse tra cui politiche IAM, ruoli IAM e cluster EKS.

Abilita con AWS CLI

Usa il seguente comando per abilitare entrambi i componenti integrati NodePools:

```
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}
  }'
```

È possibile modificare il comando per abilitare selettivamente il NodePools.

Disabilita con AWS CLI

Usa il seguente comando per disabilitare entrambi i componenti integrati NodePools:

```
aws eks update-cluster-config \
```

```
--name <cluster-name> \
--compute-config '{
"enabled": true,
"nodePools": []
}' \
--kubernetes-network-config '{
"elasticLoadBalancing":{"enabled": true}}' \
--storage-config '{
"blockStorage":{"enabled": true}
}'
```

Controllare se un carico di lavoro viene implementato sui nodi di EKS Auto Mode

Quando si eseguono carichi di lavoro in un cluster EKS con EKS Auto Mode, potresti dover controllare se carichi di lavoro specifici vengono eseguiti su nodi EKS Auto Mode o altri tipi di elaborazione. Questo argomento descrive come utilizzare i selettori di nodi e le regole di affinità per garantire che i carichi di lavoro siano pianificati sull'infrastruttura di elaborazione prevista.

Gli esempi in questo argomento mostrano come utilizzare l'etichetta `eks.amazonaws.com/compute-type` per richiedere o impedire l'implementazione del carico di lavoro sui nodi di EKS Auto Mode. Ciò è particolarmente utile nei cluster a modalità mista in cui è in esecuzione sia EKS Auto Mode che altri tipi di elaborazione, come i provisioner Karpenter autogestiti o i gruppi di nodi gestiti da EKS.

I nodi di EKS Auto Mode hanno impostato il valore dell'etichetta da `eks.amazonaws.com/compute-type` a `auto`. Puoi utilizzare questa etichetta per controllare se un carico di lavoro viene implementato nei nodi gestiti da EKS Auto Mode.

Richiedono che un carico di lavoro venga implementato ai nodi di EKS Auto Mode

> **Note**
>
> Questo valore `nodeSelector` non è richiesto per EKS Auto Mode. Questo valore `nodeSelector` è rilevante solo se si esegue un cluster in modalità mista, con tipi di nodi non gestiti da EKS Auto Mode. Ad esempio, puoi avere una capacità di elaborazione statica implementata nel cluster con o gruppi di nodi gestiti da EKS e avere una capacità di elaborazione dinamica gestita da EKS Auto Mode.

Puoi aggiungere questo `nodeSelector` alle implementazioni o ad altri carichi di lavoro per richiedere che Kubernetes li pianifichi sui nodi di EKS Auto Mode.

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    nodeSelector:
      eks.amazonaws.com/compute-type: auto
```

Richiedono che un carico di lavoro non venga implementato ai nodi di EKS Auto Mode

Puoi aggiungere questo `nodeAffinity` alle implementazioni o ad altri carichi di lavoro per richiedere che Kubernetes non li pianifichi sui nodi di EKS Auto Mode.

```
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - auto
```

Eseguire componenti aggiuntivi critici su istanze dedicate

Questo argomento illustra come implementare un carico di lavoro con una tolleranza `CriticalAddonsOnly` in modo che la modalità automatica EK lo pianifichi nel pool di nodi `system`.

Il pool di nodi integrato `system` della modalità automatica EKS è progettato per eseguire componenti aggiuntivi critici su istanze dedicate. Questa segregazione garantisce che i componenti essenziali dispongano di risorse dedicate e siano isolati dai carichi di lavoro generali, migliorando la stabilità e le prestazioni complessive del cluster.

Questa guida illustra come implementare componenti aggiuntivi nel pool di nodi `system` utilizzando la tolleranza `CriticalAddonsOnly` e i selettori dei nodi adeguati. Seguendo questa procedura, è possibile garantire che le applicazioni critiche siano pianificate sui nodi `system` dedicati, sfruttando i vantaggi dell'isolamento e dell'allocazione delle risorse offerti dalla struttura specializzata dei pool di nodi della modalità automatica EKS.

La modalità automatica EKS dispone di due pool di nodi integrati: `general-purpose` e `system`. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".

Lo scopo del pool di nodi `system` è segregare i componenti aggiuntivi critici su nodi diversi. I nodi con provisioning del pool di nodi `system` hanno un taint Kubernetes `CriticalAddonsOnly`. Kubernetes pianificherà i pod su questi nodi solo se dispongono di una tolleranza corrispondente. Per ulteriori informazioni, consultare Taints and Tolerations nella documentazione di Kubernetes.

Prerequisiti

- Cluster di modalità automatica EKS con il pool di nodi integrato `system` abilitato. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati"
- `kubectl` installato e configurato. Per ulteriori informazioni, consulta Configurazione.

Procedura

Rivedere l'esempio yaml riportato di seguito. Tenere presente le configurazioni seguenti:

- `nodeSelector`: questa configurazione associa il carico di lavoro al pool di nodi integrato `system`. Questo pool di nodi deve essere abilitato con l'API AWS. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".
- `tolerations`: questa tolleranza supera il taint `CriticalAddonsOnly` sui nodi del pool di nodi `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"
```

Per aggiornare un carico di lavoro affinché venga eseguito sul pool di nodi `system` è necessario:

1. Aggiornare il carico di lavoro esistenti affinché aggiunga le seguenti configurazioni descritte in precedenza:
   - `nodeSelector`
   - `tolerations`
2. Implementare il carico di lavoro aggiornato nel cluster con `kubectl apply`.

Una volta aggiornato, il carico di lavoro sarà eseguito su nodi dedicati.

Usa le policy di rete con la modalità automatica di EKS

Panoramica di

Man mano che i clienti scalano i propri ambienti applicativi utilizzando EKS, l'isolamento del traffico di rete diventa sempre più fondamentale per prevenire l'accesso non autorizzato alle risorse all'interno e all'esterno del cluster. Ciò è particolarmente importante in un ambiente multi-tenant con più carichi di lavoro non correlati che vengono eseguiti fianco a fianco nel cluster. Le policy di rete Kubernetes consentono di migliorare il livello di sicurezza della rete per i carichi di lavoro Kubernetes e le relative integrazioni con endpoint esterni al cluster. EKS Auto Mode supporta diversi tipi di politiche di rete.

Isolamento di livello 3 e 4

Le policy di rete Kubernetes standard operano ai livelli 3 e 4 del modello di rete OSI e consentono di controllare il flusso di traffico a livello di indirizzo IP o di porta all'interno del cluster Amazon EKS.

Casi d'uso

- Segmenta il traffico di rete tra i carichi di lavoro per garantire che solo le applicazioni correlate possano comunicare tra loro.
- Isola i tenant a livello di namespace utilizzando policy per imporre la separazione della rete.

Applicazione basata sul DNS

I clienti in genere implementano carichi di lavoro in EKS che fanno parte di un ambiente distribuito più ampio, alcuni dei quali devono comunicare con sistemi e servizi esterni al cluster (traffico in direzione nord). Questi sistemi e servizi possono essere nel AWS cloud o all'esterno. AWS Le policy basate sul Domain Name System (DNS) consentono di rafforzare il livello di sicurezza adottando un approccio più stabile e prevedibile per impedire l'accesso non autorizzato dai pod a risorse o endpoint esterni al cluster. Questo meccanismo elimina la necessità di tracciare e consentire manualmente l'elenco di indirizzi IP specifici. Proteggendo le risorse con un approccio basato sul DNS, si ha anche una maggiore flessibilità nell'aggiornare l'infrastruttura esterna senza dover allentare il livello di sicurezza o modificare le politiche di rete a causa delle modifiche apportate ai server e agli host upstream. È possibile filtrare il traffico in uscita verso endpoint esterni utilizzando un nome di dominio completo (FQDN) o uno schema corrispondente per un nome di dominio DNS. Ciò offre la maggiore flessibilità di estendere l'accesso a più sottodomini associati a un particolare endpoint esterno al cluster.

Casi d'uso

- Standardizza un approccio basato su DNS per filtrare l'accesso da un ambiente Kubernetes agli endpoint esterni al cluster.
- Accesso AWS sicuro ai servizi in un ambiente multi-tenant.
- Gestisci l'accesso alla rete dai pod ai carichi di lavoro on-premise nei tuoi ambienti cloud ibridi.

Regole amministrative (o con ambito cluster)

In alcuni casi, come negli scenari multi-tenant, i clienti potrebbero avere l'esigenza di applicare uno standard di sicurezza di rete che si applichi all'intero cluster. Invece di definire e mantenere ripetutamente una policy distinta per ogni namespace, è possibile utilizzare un'unica policy per gestire centralmente i controlli di accesso alla rete per i diversi carichi di lavoro del cluster, indipendentemente dal relativo namespace. Questi tipi di policy consentono di estendere l'ambito di applicazione delle regole di filtraggio di rete applicate a livello 3, livello 4 e quando si utilizzano regole DNS.

Casi d'uso

- Gestisci centralmente i controlli di accesso alla rete per tutti (o un sottoinsieme di) carichi di lavoro nel tuo cluster EKS.
- Definisci una posizione di sicurezza di rete predefinita in tutto il cluster.
- Estendi gli standard di sicurezza organizzativi all'ambito del cluster in un modo più efficiente dal punto di vista operativo.

Nozioni di base

Prerequisiti

- Un cluster Amazon EKS con la modalità automatica di EKS abilitata
- kubectl configurato per connettersi al cluster

Fase 1: abilitare il controller della policy di rete

Per utilizzare le politiche di rete con EKS Auto Mode, devi prima abilitare il Network Policy Controller applicando un ConfigMap al tuo cluster.

1. Crea un file denominato `enable-network-policy.yaml` con i seguenti contenuti:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-network-policy-controller: "true"
```

2. Applica il ConfigMap al tuo cluster:

```
kubectl apply -f enable-network-policy.yaml
```

Fase 2: Creare e testare le politiche di rete

Il cluster della modalità automatica di EKS è ora configurato per supportare le policy di rete di Kubernetes. Puoi testarlo con la the section called "Demo policy Stars".

Fase 3: Modifica della configurazione di Network Policy Agent in Node Class (opzionale)

Facoltativamente, è possibile creare una nuova classe di nodi per modificare il comportamento predefinito del Network Policy Agent sui nodi o abilitare la registrazione degli eventi di Network Policy. A tale scopo, seguire queste fasi:

1. Crea o modifica un file YAML di classe del nodo (ad esempio, `nodeclass-network-policy.yaml`) con i seguenti contenuti:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: network-policy-config
spec:
  # Optional: Changes default network policy behavior
  networkPolicy: DefaultAllow
  # Optional: Enables logging for network policy events
  networkPolicyEventLogs: Enabled
  # Include other Node Class configurations as needed
```

2. Applica la configurazione di classe del nodo al cluster:

```
kubectl apply -f nodeclass-network-policy.yaml
```

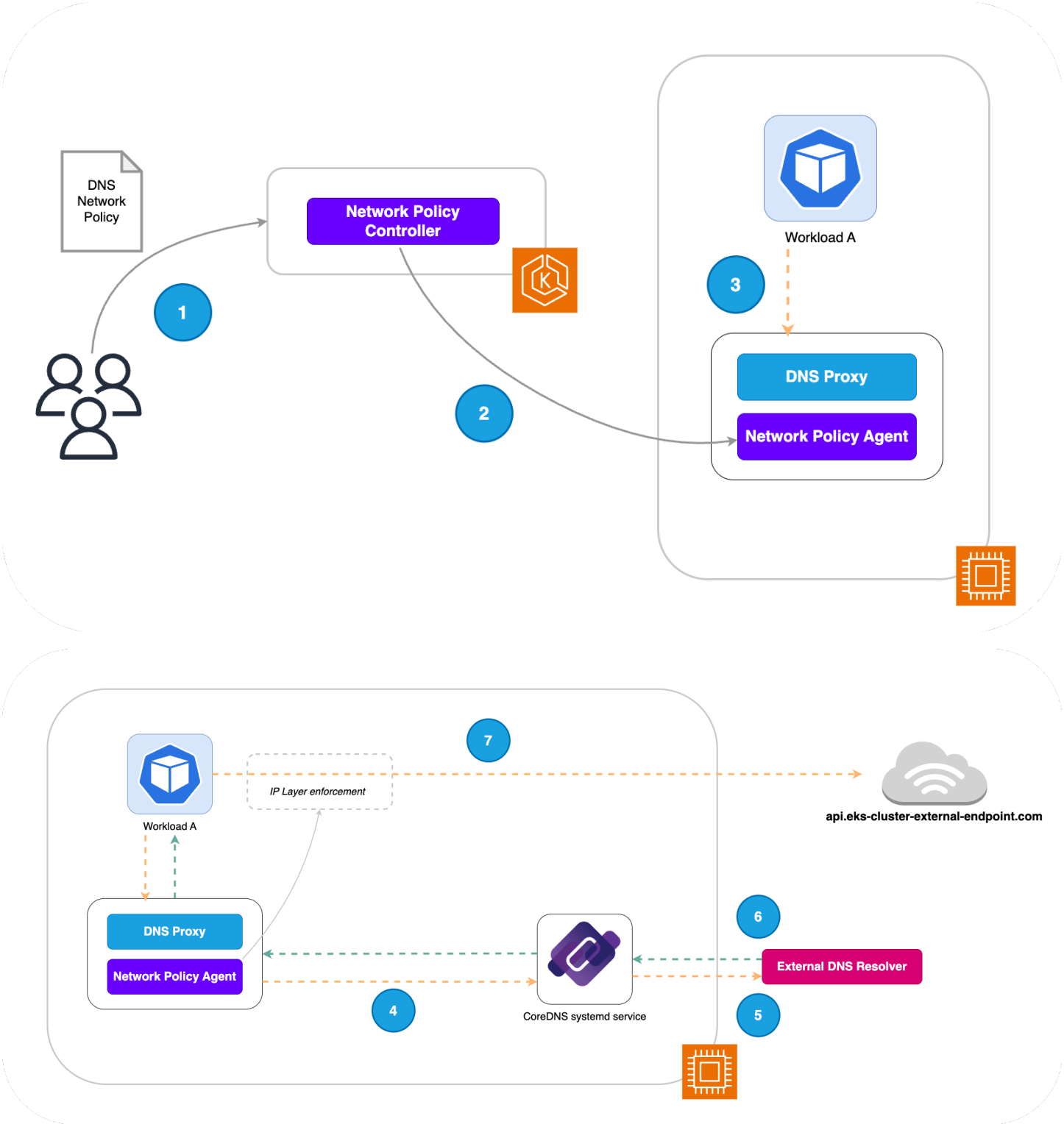
3. Verifica che la classe del nodo sia stata creata:

```
kubectl get nodeclass network-policy-config
```

4. Aggiorna il pool di nodi per utilizzare questa classe di nodi. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

Come funziona?

Politica di rete basata su DNS



1. Il team della piattaforma applica una politica basata su DNS al cluster EKS.

2. Il Network Policy Controller è responsabile del monitoraggio della creazione di politiche all'interno del cluster e della successiva riconciliazione degli endpoint delle politiche. In questo caso d'uso, il network policy controller ordina all'agente del nodo di filtrare le richieste DNS in base ai domini consentiti elencati nella policy creata. I nomi di dominio vengono elencati nell'elenco consentito utilizzando l'FQDN o un nome di dominio che corrisponde a uno schema definito nella configurazione delle risorse Kubernetes.
3. Workload A tenta di risolvere l'IP per un endpoint esterno al cluster. La richiesta DNS passa innanzitutto attraverso un proxy che filtra tali richieste in base all'elenco di autorizzazioni applicato tramite la politica di rete.
4. Una volta che la richiesta DNS passa attraverso l'elenco dei filtri DNS consentiti, viene inoltrata tramite proxy a CoredNS,
5. CoreDNS a sua volta invia la richiesta al Resolver DNS esterno (Amazon Route 53 Resolver) per ottenere l'elenco degli indirizzi IP dietro il nome di dominio.
6. I dati risolti IPs con TTL vengono restituiti nella risposta alla richiesta DNS. Questi IPs vengono quindi scritti in una mappa eBPF che viene utilizzata nella fase successiva per l'applicazione del livello IP.
7. Le sonde eBPF collegate all'interfaccia Pod veth filtreranno quindi il traffico in uscita dal carico di lavoro A all'endpoint esterno del cluster in base alle regole in vigore. Ciò garantisce che i pod possano inviare traffico esterno al cluster solo verso i domini consentiti elencati. IPs La validità di questi si IPs basa sul TTL recuperato dal Resolver DNS esterno (Amazon Route 53 Resolver).

Utilizzo della policy di rete dell'applicazione

`ApplicationNetworkPolicy`Combina le funzionalità delle politiche di rete Kubernetes standard con il filtraggio basato su DNS a livello di namespace utilizzando un'unica Custom Resource Definition (CRD). Pertanto, può essere utilizzato per: `ApplicationNetworkPolicy`

1. Definizione delle restrizioni ai livelli 3 e 4 dello stack di rete utilizzando blocchi IP e numeri di porta.
2. Definizione di regole che operano al livello 7 dello stack di rete e consentono di filtrare il traffico in base a. FQDNs

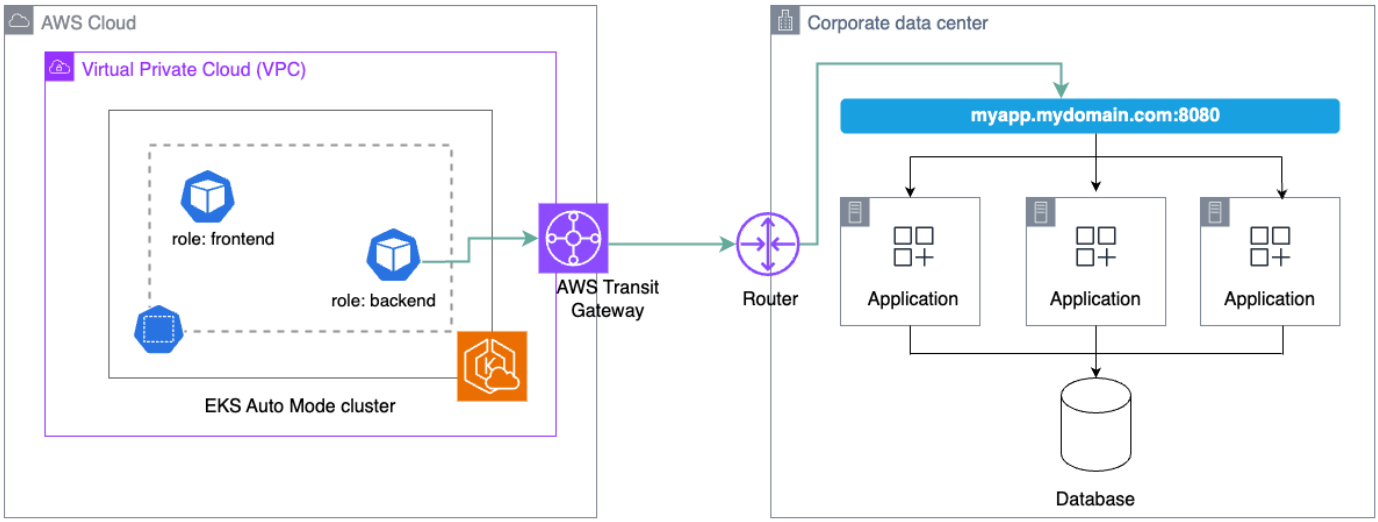
Nota importante: le regole basate sul DNS definite utilizzando il `ApplicationNetworkPolicy` sono applicabili solo ai carichi di lavoro in esecuzione su istanze lanciate in modalità automatica EKS. EC2

Esempio

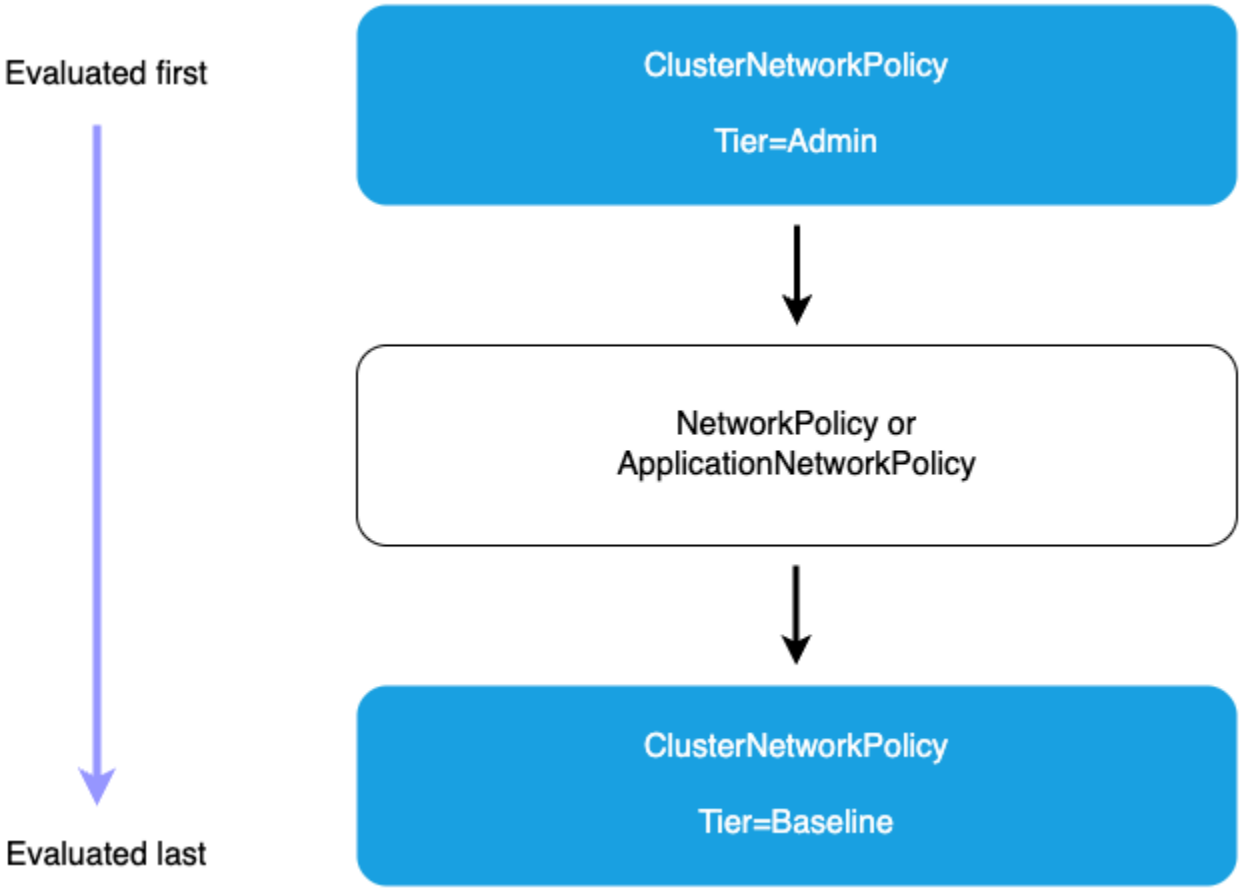
Nel cluster EKS Auto Mode è presente un carico di lavoro che deve comunicare con un'applicazione locale basata su un sistema di bilanciamento del carico con un nome DNS. È possibile ottenere ciò utilizzando la seguente politica di rete:

```
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
```

A livello di rete Kubernetes, ciò consentirebbe l'uscita da qualsiasi pod nello spazio dei nomi «galaxy» etichettato con **`role: backend`** per connettersi al nome di dominio myapp.mydomain.com sulla porta TCP 8080. Inoltre, è necessario configurare la connettività di rete per il traffico in uscita dal VPC al data center aziendale.



Politica di rete di amministrazione (o cluster)



Utilizzo della politica di rete del cluster

Quando si utilizza a`ClusterNetworkPolicy`, le politiche del livello di amministrazione vengono valutate per prime e non possono essere sostituite. Una volta valutate le politiche di livello amministratore, le politiche standard con ambito dei namespace vengono utilizzate per eseguire le regole di segmentazione della rete applicate. Questa operazione può essere eseguita utilizzando uno dei due o. `ApplicationNetworkPolicy` `NetworkPolicy` Infine, verranno applicate le regole del livello di base che definiscono le restrizioni di rete predefinite per i carichi di lavoro dei cluster. Queste regole del livello di base possono essere sostituite dalle policy con ambito dei namespace, se necessario.

Esempio

Nel cluster è presente un'applicazione che si desidera isolare dagli altri carichi di lavoro dei tenant. È possibile bloccare in modo esplicito il traffico del cluster proveniente da altri namespace per impedire l'accesso di rete allo spazio dei nomi riservato del carico di lavoro.

```
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
```

Considerazioni

Comprendi l'ordine di valutazione delle politiche

Le funzionalità relative alle politiche di rete supportate in EKS vengono valutate in un ordine specifico per garantire una gestione del traffico prevedibile e sicura. Pertanto, è importante comprendere il flusso di valutazione per progettare una posizione di sicurezza di rete efficace per il proprio ambiente.

1. Politiche del livello di amministrazione (valutate per prime): tutte le politiche di livello amministratore ClusterNetworkPolicies vengono valutate prima di qualsiasi altra politica. All'interno del livello di amministrazione, le politiche vengono elaborate in ordine di priorità (prima il numero di priorità più basso). Il tipo di azione determina cosa succede dopo.
   - Nega azione (priorità massima): quando una politica di amministrazione con un'azione Rifiuta corrisponde al traffico, tale traffico viene immediatamente bloccato indipendentemente da qualsiasi altra politica. Non vengono elaborate ClusterNetworkPolicy ulteriori NetworkPolicy regole. Ciò garantisce che i controlli di sicurezza a livello di organizzazione non possano essere sostituiti da politiche a livello di namespace.
   - Consenti azioni: dopo la valutazione delle regole Deny, le politiche di amministrazione con le azioni Consenti vengono elaborate in ordine di priorità (inizia con il numero di priorità più basso). Quando un'azione Consenti corrisponde, il traffico viene accettato e non viene effettuata alcuna ulteriore valutazione delle politiche. Queste politiche possono concedere l'accesso su più namespace in base a selettori di etichette, fornendo un controllo centralizzato su quali carichi di lavoro possono accedere a risorse specifiche.
   - Pass action: le azioni Pass Action nelle policy del livello di amministrazione delegano il processo decisionale ai livelli inferiori. Quando il traffico corrisponde a una regola Pass, la valutazione salta tutte le regole rimanenti del livello di amministratore per quel traffico e passa direttamente al livello. NetworkPolicy Ciò consente agli amministratori di delegare in modo esplicito il controllo di determinati modelli di traffico ai team applicativi. Ad esempio, è possibile utilizzare le regole Pass per delegare la gestione del traffico all'interno del namespace agli amministratori del namespace, mantenendo al contempo controlli rigorosi sull'accesso esterno.
2. Livello di policy di rete: se nessuna politica del livello di amministratore corrisponde a Deny o Allow, o se è stata soddisfatta un'azione Pass, vengono successivamente valutate le risorse tradizionali e con ambito namespace. ApplicationNetworkPolicy NetworkPolicy Queste policy forniscono un controllo granulare all'interno dei singoli namespace e sono gestite dai team applicativi. Le politiche con ambito namespace possono essere solo più restrittive delle politiche di amministrazione. Non possono ignorare la decisione di rifiuto di una politica di amministrazione, ma possono limitare ulteriormente il traffico consentito o approvato dalle politiche di amministrazione.
3. Politiche di amministrazione di livello base: se nessuna politica di amministrazione o con ambito del namespace corrisponde al traffico, viene valutato il livello di base. ClusterNetworkPolicies Queste forniscono posizioni di sicurezza predefinite che possono essere sostituite da policy con ambito namespace, consentendo agli amministratori di impostare impostazioni predefinite a livello di organizzazione e offrendo al contempo ai team la flessibilità necessaria per personalizzarle in

base alle esigenze. Le politiche di base vengono valutate in ordine di priorità (per primo il numero di priorità più basso).

4. Negazione predefinita (se nessuna policy corrisponde): questo deny-by-default comportamento garantisce che siano consentite solo le connessioni esplicitamente consentite, mantenendo un solido livello di sicurezza.

Applicazione del principio del privilegio minimo

- Inizia con politiche restrittive e aggiungi gradualmente le autorizzazioni secondo necessità. Inizia implementando le deny-by-default politiche a livello di cluster, quindi aggiungi in modo incrementale le regole di autorizzazione man mano che convalidi i requisiti di connettività legittimi. Questo approccio obbliga i team a giustificare esplicitamente ogni connessione esterna, creando un ambiente più sicuro e verificabile.
- Verifica e rimuovi regolarmente le regole di policy non utilizzate: le policy di rete possono accumularsi nel tempo man mano che le applicazioni si evolvono, lasciando dietro di sé regole obsolete che ampliano inutilmente la superficie di attacco. Implementate un processo di revisione regolare per identificare e rimuovere le regole di policy che non sono più necessarie, assicurando che il vostro livello di sicurezza rimanga rigoroso e gestibile.
- Se possibile, utilizza nomi di dominio specifici anziché schemi generici. Se da un lato i modelli wildcard `*.amazonaws.com` offrono maggiore praticità, dall'altro garantiscono l'accesso a un'ampia gamma di servizi. Ove possibile, specifica nomi di dominio esatti, `s3.us-west-2.amazonaws.com` ad esempio per limitare l'accesso solo ai servizi specifici richiesti dalle applicazioni, riducendo il rischio di spostamenti laterali se un carico di lavoro è compromesso.

Utilizzo di policy basate su DNS in EKS

- Le regole basate su DNS definite utilizzando `ApplicationNetworkPolicy` sono applicabili solo ai carichi di lavoro in esecuzione su istanze lanciate in modalità automatica EKS. EC2 Se si utilizza un cluster in modalità mista (composto da nodi di lavoro EKS Auto e non EKS Auto), le regole basate sul DNS sono efficaci solo nei nodi di lavoro in modalità EKS Auto (istanze gestite). EC2

Convalida delle politiche DNS

- Utilizzate cluster di staging che rispecchiano la topologia della rete di produzione per i test: l'ambiente di staging deve replicare l'architettura di rete, le dipendenze esterne e i modelli di connettività di produzione per garantire un test accurato delle politiche. Ciò include configurazioni

VPC corrispondenti, comportamento di risoluzione DNS e accesso agli stessi servizi esterni richiesti dai carichi di lavoro di produzione.

- Implementa test automatizzati per percorsi di rete critici: crea test automatizzati che convalidano la connettività ai servizi esterni essenziali come parte della tua pipeline. CI/CD Questi test devono verificare che i flussi di traffico legittimi siano consentiti mentre le connessioni non autorizzate vengono bloccate, in modo da confermare continuamente che le politiche di rete mantengano il corretto livello di sicurezza man mano che l'infrastruttura si evolve.
- Monitora il comportamento delle applicazioni dopo le modifiche alle policy: dopo aver implementato policy di rete nuove o modificate nella produzione, monitorate attentamente i log delle applicazioni, i tassi di errore e le metriche prestazionali per identificare rapidamente eventuali problemi di connettività. Stabilisci procedure di rollback chiare in modo da poter ripristinare rapidamente le modifiche alle policy se causano comportamenti imprevisti delle applicazioni o interruzioni del servizio.

Interazione con il firewall DNS di Amazon Route 53

Le politiche di amministrazione e di rete di EKS vengono valutate prima a livello di pod quando viene avviato il traffico. Se una policy di rete EKS consente l'uscita verso un dominio specifico, il pod esegue quindi una query DNS che raggiunge il Route 53 Resolver. A questo punto, vengono valutate le regole del firewall DNS Route 53. Se DNS Firewall blocca la query del dominio, la risoluzione DNS fallisce e la connessione non può essere stabilita, anche se la politica di rete EKS lo ha consentito. Ciò crea livelli di sicurezza complementari: le policy di rete basate su EKS DNS forniscono il controllo dell'uscita a livello di pod per i requisiti di accesso specifici delle applicazioni e i limiti di sicurezza multi-tenant, mentre DNS Firewall offre una protezione a livello di VPC contro domini dannosi noti e applica le blocklist a livello di organizzazione.

Assegnazione di tag alle sottoreti per la modalità automatica EKS

Se utilizzi la funzionalità di bilanciamento del carico di EKS Auto Mode, devi aggiungere AWS tag alle sottoreti VPC.

Contesto

Questi tag identificano le sottoreti associate al cluster e, soprattutto, se la sottorete è pubblica o privata.

Le sottoreti pubbliche dispongono di un accesso diretto a Internet tramite un gateway Internet. Sono utilizzate per risorse che devono essere accessibili al pubblico, come i bilanciatori del carico.

Le sottoreti private non dispongono di un accesso diretto a Internet e utilizzano gateway NAT per il traffico in uscita. Vengono utilizzati per risorse interne come i nodi EKS che non necessitano di risorse pubbliche. IPs

Per ulteriori informazioni sui gateway NAT e sui gateway Internet, consulta Connetti il tuo VPC ad altre reti nella Guida per l'utente del cloud privato virtuale (VPC) di Amazon.

Requisito

Al momento, le sottoreti utilizzate per il bilanciamento del carico tramite la modalità automatica EKS devono avere uno dei seguenti tag.

Sottoreti pubbliche

Le sottoreti pubbliche sono utilizzate per i bilanciatori del carico connesso a Internet. Queste sottoreti devono avere i tag seguenti:

Chiave	Valore
`kubernetes.io/role/elb`	`1` o ``

Sottoreti private

Le sottoreti private sono utilizzate per i bilanciatori del carico interni. Queste sottoreti devono avere i tag seguenti:

Chiave	Valore
`kubernetes.io/role/internal-elb`	`1` o ``

Procedura

Prima di iniziare, identifica quali sottoreti sono pubbliche (con accesso a gateway Internet) e quali sono private (tramite gateway NAT). Avrai bisogno delle autorizzazioni per modificare le risorse VPC.

Console di gestione AWS

1. Apri la console Amazon VPC e accedi alle sottoreti.
2. Seleziona la sottorete da etichettare.

3. Scegli la scheda Tag e seleziona Aggiungi tag.
4. Aggiungi il tag appropriato:
   - Per le sottoreti pubbliche: Chiave=`kubernetes.io/role/elb`
   - Per sottoreti private: Chiave=`kubernetes.io/role/internal-elb`
5. Imposta Valore su 1 o lascia vuoto.
6. Salva e ripeti per le sottoreti rimanenti.

AWS CLI

Per le sottoreti pubbliche:

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/elb,Value=1
```

Per le sottoreti private:

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/internal-elb,Value=1
```

Sostituisci `subnet-ID` con l'ID della sottorete corrente.

Generare report di conformità CIS dai nodi Kubernetes utilizzando kubectl debug

Questo argomento descrive come generare report di conformità CIS (Center for Internet Security) per i nodi Amazon EKS utilizzando il comando `kubectl debug`. Il comando ti consente di creare temporaneamente un container di debug su un nodo di Kubernetes ed eseguire controlli di conformità CIS utilizzando lo strumento `apiclient`. Lo strumento `apiclient` fa parte di Bottlerocket OS, il sistema operativo utilizzato dai nodi della modalità automatica di EKS.

Prerequisiti

Prima di iniziare, assicurati di disporre di:

- Accesso a un cluster Amazon EKS con `kubectl` configurato (la versione deve essere almeno v1.32.0; digitare `kubectl version` per verificare).

- Le autorizzazioni IAM appropriate per il debug dei nodi.
- Un profilo valido che consente operazioni di debug (ad esempio, `sysadmin`).

Per ulteriori informazioni sull'utilizzo dei profili di debug con `kubectl`, consultare Debugging a Pod or Node while applying a profile nella documentazione di Kubernetes.

Procedura

1. Determina l'ID AWS istanza del nodo su cui vuoi eseguire il rapporto. Utilizzare il seguente comando per elencare tutti i nodi nel cluster. L'ID dell'istanza si trova nella colonna del nome e inizia con `i-`:

```
kubectl get nodes
```

```
NAME                  STATUS   ROLES    AGE   VERSION
i-0ea0ba0f8ef9ad609   Ready    <none>   62s   v1.30.10-eks-1a9dacd
```

2. Esegui il comando seguente, sostituendo `<instance-id>` con l'ID di istanza del nodo su cui si desidera eseguire la query:

```
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"
```

   I componenti di questo comando includono:

   - `kubectl debug node/<instance-id>`: crea una sessione di debug sull'ID di istanza EC2 specificato.
   - `-it`: assegna un TTY (shell a riga di comando) e mantiene aperto stdin per un utilizzo interattivo.
   - `--profile=sysadmin`: utilizza il profilo `kubectl` specificato con le autorizzazioni appropriate.
   - `--image=public.ecr.aws/amazonlinux/amazonlinux:2023`: utilizza `amazonlinux:2023` come immagine del container per il debug.
   - `bash -c "…"` : esegue i seguenti comandi in una shell bash:
     - `yum install -q -y util-linux-core`: installa in background il pacchetto di utilità richiesto.
     - `nsenter -t 1 -m`: esegue `nsenter` per inserire il namespace del processo host (PID 1).

- `apiclient report cis --level 1 --format text`: esegue il report di conformità CIS a livello 1 con output di testo.

3. Esaminare l'output del testo del report.

Interpretazione dell'output

Il comando genera un report basato su testo che mostra lo stato di conformità di vari controlli CIS. L'output include:

- Controllo CIS individuale IDs
- Descrizione di ciascun controllo
- Lo stato Pass, Fail o Skip per ogni controllo
- Dettagli che spiegano eventuali problemi di conformità

Ecco un esempio di output del report eseguito su un'istanza 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
```

Per informazioni sul benchmark, consultare Kubernetes Benchmark dal Center for Internet Security (CIS).

Risorse correlate

- Bottlerocket CIS Benchmark nella documentazione del sistema operativo Bottlerocket.
- Debug Running Pods nella documentazione di Kubernetes.
- Kubernetes Benchmark dal Center for Internet Security (CIS)

Abilitare la crittografia dei volumi EBS con chiavi KMS gestite dal cliente per la modalità automatica di EKS

Puoi crittografare il volume root temporaneo per le istanze della modalità automatica di EKS con una chiave KMS gestita dal cliente.

La modalità automatica di Amazon EKS utilizza ruoli collegati ai servizi per delegare le autorizzazioni ad altri AWS servizi durante la gestione di volumi EBS crittografati per i cluster Kubernetes. In questo argomento viene descritto come impostare le policy della chiave necessarie quando si specifica una chiave gestita dal cliente per la crittografia Amazon EBS con la modalità automatica di EKS.

Considerazioni:

- EKS Auto Mode non richiede un'autorizzazione aggiuntiva per utilizzare la chiave AWS gestita predefinita per proteggere i volumi crittografati nel tuo account.
- Questo argomento tratta la crittografia dei volumi effimeri, i volumi root per le istanze. EC2 Per ulteriori informazioni sulla crittografia dei volumi di dati utilizzati per i carichi di lavoro, consulta the section called "Crea StorageClass".

Panoramica di

Le seguenti chiavi AWS KMS possono essere utilizzate per la crittografia dei volumi root di Amazon EBS quando EKS Auto Mode avvia le istanze:

- Chiave gestita da AWS : una chiave di crittografia nell'account che Amazon EBS crea, possiede e gestisce. Questa è la chiave di crittografia di default per un nuovo account.
- Chiave gestita dal cliente: una chiave di crittografia personalizzata che l'utente crea, possiede e gestisce.

> **Note**
>
> La chiave deve essere simmetrica. Amazon EBS non supporta le chiavi gestite dal cliente asimmetriche.

Fase 1: Configurare la policy della chiave

Le chiavi KMS devono avere una policy della chiave che permetta alla modalità automatica di EKS di avviare istanze con volumi Amazon EBS crittografati con una chiave gestita dal cliente.

Configura la policy della chiave con la seguente struttura:

> **Note**
>
> Questa policy include solo le autorizzazioni per la modalità automatica di EKS. La policy della chiave potrebbe richiedere autorizzazioni aggiuntive se altre identità devono utilizzare la chiave o gestire le concessioni.

```
{
    "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"
                }
            }
        }
    ]
}
```

Assicurati di sostituirlo <account-id> con l'ID dell'account effettivo. AWS

Durante la configurazione della policy della chiave:

- Per eseguire le operazioni di crittografia, il `ClusterServiceRole` deve disporre delle autorizzazioni IAM necessarie per usare la chiave KMS.
- La `kms:GrantIsForAWSResource` condizione garantisce che le sovvenzioni possano essere create solo per i servizi AWS

Fase 2: Configura NodeClass con la chiave gestita dal cliente

Dopo aver configurato la politica chiave, fai riferimento alla chiave KMS nella configurazione della modalità NodeClass automatica 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>"
```

Sostituisci i valori di segnaposto con i valori effettivi:

- `<region>`con la tua regione AWS
- `<account-id>`con l'ID AWS del tuo account
- `<key-id>` con ID della chiave KMS

Puoi specificare la chiave KMS utilizzando uno dei seguenti formati:

- ID della chiave KMS: `1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- ARN della chiave KMS: `arn:aws: kms:us-west-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- Nome alias della chiave: `alias/eks-auto-mode-key`
- ARN alias della chiave: `arn:aws: kms:us-west-2:111122223333:alias/eks-auto-mode-key`

Applica la NodeClass configurazione usando kubectl:

```
kubectl apply -f nodeclass.yaml
```

Risorse correlate

- Creazione di una classe di nodi per Amazon EKS
- Visualizza ulteriori informazioni nella AWS Key Management Service Developer Guide
  - Autorizzazioni per AWS i servizi nelle politiche chiave
  - Modificare una policy della chiave
  - Sovvenzioni in KMS AWS

Aggiorna i controlli organizzativi per la modalità automatica di EKS

Alcuni controlli organizzativi possono impedire il corretto funzionamento della modalità automatica di EKS. In tal caso, devi aggiornare questi controlli per consentire alla modalità automatica di EKS di disporre delle autorizzazioni necessarie per gestire le istanze EC2 per conto dell'utente.

modalità automatica di EKS utilizza un ruolo di servizio per l'avvio delle istanze EC2 che supportano i nodi della modalità automatica di EKS. Un ruolo di servizio è un ruolo IAM creato nell'account che un servizio assume per eseguire operazioni nell'account. Le policy di controllo del servizio (SCPs) si applicano sempre alle azioni eseguite con i ruoli di servizio. Ciò consente a una SCP di inibire le operazioni di Auto Mode. L'evento più comune si verifica quando viene utilizzato un SCP per limitare le Amazon Machine Images (AMIs) che possono essere lanciate. Per consentire il funzionamento di EKS Auto Mode, modifica l'SCP per consentire l'avvio AMIs dagli account EKS Auto Mode.

Puoi anche utilizzare la AMIs funzione EC2 Allowed per limitare la visibilità di altri account AMIs . Se utilizzi questa funzionalità, devi espandere i criteri dell'immagine per includere anche gli account AMI della modalità automatica di EKS nelle regioni di interesse.

Esempio: SCP per bloccare tutti AMIs tranne EKS Auto Mode AMIs

La SCP riportata di seguito impedisce di chiamare `ec2:RunInstances` a meno che l'AMI non appartenga all'account AMI della modalità automatica di EKS per us-west-2 o us-east-1.

> **Note**
>
> È importante non usare la chiave di contesto `ec2:Owner`. Amazon possiede gli account AMI della modalità automatica di EKS e il valore di questa chiave sarà sempre `amazon`. La creazione di un SCP che consenta l'avvio AMIs se `ec2:Owner` è `amazon` consentirà il lancio di qualsiasi AMI di proprietà di Amazon, non solo quelle per EKS Auto Mode. *

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAMI",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:*:ec2:*::image/ami-*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "767397842682",
            "992382739861"
          ]
        }
      }
    }
  ]
}
```

Account AMI della modalità automatica di EKS

AWS gli account che variano in base alla regione ospitano EKS Auto Mode public AMIs.

AWS Regione	Account
af-south-1	471112993317
ap-east-1	590183728416
ap-east-2	381492200852
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	058264376476
ap-southeast-3	471112941769
ap-southeast-4	590183863144
ap-southeast-5	654654202513
ap-southeast-6	905418310314
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
us-gov-east-1	446077414359
us-gov-west-1	446098668741

Associare un indirizzo IP pubblico

Quando viene chiamato `ec2:RunInstances`, il campo `AssociatePublicIpAddress` per l'avvio di un'istanza viene determinato automaticamente dal tipo di sottorete in cui viene avviata l'istanza. È possibile utilizzare un SCP per far sì che questo valore sia impostato esplicitamente su false, indipendentemente dal tipo di sottorete in cui viene avviato. In questo caso il NodeClass campo `spec.advancedNetworking.associatePublicIPAddress` può anche essere impostato su false per soddisfare i requisiti dell'SCP.

```
{
      "Sid": "DenyPublicEC2IPAddesses",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
          "BoolIfExists": {
              "ec2:AssociatePublicIpAddress": "true"
          }
      }
  }
```

Controlla l'implementazione dei carichi di lavoro in Prenotazioni della capacità con la modalità automatica di EKS

Puoi controllare l'implementazione dei carichi di lavoro su Prenotazioni della capacità. La modalità automatica EKS supporta le prenotazioni di capacità EC2 su richiesta (ODCRs) e i blocchi di EC2 capacità per il machine learning.

> **Tip**
>
> Per impostazione predefinita, EKS Auto Mode si avvia automaticamente nei blocchi di capacità aperti ODCRs e ML. Quando viene utilizzata `capacityReservationSelectorTerms` nella NodeClass definizione, EKS Auto Mode non utilizzerà più automaticamente alcuna riserva di capacità aperta.

EC2 Prenotazioni di capacità su richiesta () ODCRs

EC2 Le prenotazioni di capacità su richiesta (ODCRs) ti consentono di riservare la capacità di calcolo per le tue EC2 istanze Amazon in una zona di disponibilità specifica per qualsiasi durata. Quando si utilizza modalità automatica di EKS, potresti voler controllare se i carichi di lavoro di Kubernetes vengono implementati su queste istanze riservate per massimizzare l'utilizzo della capacità pre-acquistata o per garantire che i carichi di lavoro critici abbiano accesso a risorse garantite.

Per impostazione predefinita, EKS Auto Mode si apre automaticamente. ODCRs Tuttavia, configurando `capacityReservationSelectorTerms` su a NodeClass, puoi controllare in modo esplicito l'utilizzo dei ODCRs tuoi carichi di lavoro. I nodi forniti utilizzando configure ODCRs avranno e avranno priorità rispetto a on-demand `karpenter.sh/capacity-type: reserved` e spot. Una volta abilitata questa funzionalità, EKS Auto Mode non utilizzerà più automaticamente open ODCRs: devono essere selezionati esplicitamente da un NodeClass, in modo da consentire un controllo preciso sull'utilizzo della prenotazione della capacità in tutto il cluster.

> **Warning**
>
> Se si esegue la configurazione `capacityReservationSelectorTerms` all' NodeClass interno di un cluster, EKS Auto Mode non utilizzerà più automaticamente open ODCRs per nessuno NodeClass dei componenti del cluster.

Esempio NodeClass

```
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"
```

Questo esempio NodeClass illustra due approcci per la selezione ODCRs. Il primo metodo fa riferimento direttamente a una ODCR specifica tramite il relativo ID (`cr-56fac701cc1951b03`). Il secondo metodo utilizza la selezione basata su tag, mirando ODCRs con il tag. `Name: "targeted-odcr"` Facoltativamente, puoi anche filtrare in base all' AWS account proprietario della prenotazione, il che è particolarmente utile negli scenari tra più account o quando lavori con prenotazioni di capacità condivise.

EC2 Capacity Blocks per ML

I blocchi di capacità per ML prenotano istanze a calcolo accelerato basate su GPU in una data futura per supportare carichi di lavoro di machine learning (ML) di breve durata. Le istanze eseguite all'interno di un Capacity Block vengono automaticamente posizionate vicine tra loro all'interno di Amazon EC2 UltraClusters, per una rete a bassa latenza, su scala petabit e non bloccante.

Per ulteriori informazioni sulle piattaforme e sui tipi di istanze supportati, consulta Capacity Blocks for ML nella Guida per l'utente. EC2

È possibile creare una modalità automatica EKS NodeClass che utilizza un Capacity Block per ML, simile a un ODCR (descritto in precedenza).

Le seguenti definizioni di esempio creano tre risorse:

1. A NodeClass che fa riferimento alla tua prenotazione Capacity Block
2. A NodePool che utilizza NodeClass e applica una macchia
3. Una specifica per Pod che tollera il taint e richiede risorse GPU

Esempio NodeClass

Questo NodeClass fa riferimento a uno specifico Capacity Block for ML tramite il relativo ID di prenotazione. È possibile ottenere questo ID dalla EC2 console.

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu
spec:
  # Specify your Capacity Block reservation ID
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
```

Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

Esempio NodePool

Questo NodePool fa riferimento `gpu` NodeClass e specifica una configurazione importante:

- Utilizza solo la capacità riservata impostando `karpenter.sh/capacity-type: reserved`
- Richiede famiglie di istanze GPU specifiche adatte ai carichi di lavoro ML
- Applica un taint `nvidia.com/gpu` per garantire che solo i carichi di lavoro GPU siano pianificati su questi nodi

```
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
```

Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

Pod di esempio

Questo pod di esempio dimostra come configurare un carico di lavoro da eseguire sui nodi del blocco di capacità:

- Utilizza un NodeSelector per indirizzare tipi di GPU specifici (in questo caso, H200) GPUs
- Include una tolleranza per la macchia applicata dal `nvidia.com/gpu` NodePool
- Richiede esplicitamente le risorse GPU che utilizzano il tipo di risorsa `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
```

Per ulteriori informazioni, consulta Pods nella documentazione Kubernetes.

Risorse correlate

- Blocchi di capacità per il machine learning nella Amazon EC2 User Guide
- Trova e acquista Capacity Blocks nella Amazon EC2 User Guide
- Gestisci le risorse di calcolo per i AI/ML carichi di lavoro su Amazon EKS
- GPU Resource Optimization and Cost Management nella Guida alle best practice di EKS

Implementazione dei nodi EKS Auto Mode su Local Zones

EKS Auto Mode offre una gestione semplificata dei cluster con il provisioning automatico dei nodi. AWS Local Zones estendono l' AWS infrastruttura alle aree geografiche più vicine agli utenti finali, riducendo la latenza per le applicazioni sensibili alla latenza. Questa guida illustra il processo di implementazione dei nodi EKS Auto Mode su AWS Local Zones, consentendoti di eseguire applicazioni containerizzate con latenza inferiore per gli utenti in aree geografiche specifiche.

Questa guida dimostra anche come utilizzare i limiti e le tolleranze di Kubernetes per garantire che sui nodi della zona locale vengano eseguiti solo carichi di lavoro specifici, aiutandoti a controllare i costi e ottimizzare l'utilizzo delle risorse.

Prerequisiti

Prima di iniziare a distribuire i nodi EKS Auto Mode su Local Zones, assicurati di avere i seguenti prerequisiti:

- Un cluster EKS Auto Mode esistente
- Hai effettuato l'accesso alla zona locale del tuo account AWS

Fase 1: Creare una sottorete di zona locale

Il primo passo per implementare i nodi EKS Auto Mode in una zona locale consiste nella creazione di una sottorete in quella zona locale. Questa sottorete fornisce l'infrastruttura di rete per i nodi e consente loro di comunicare con il resto del VPC. Segui le istruzioni Create a Local Zone subnet (nella AWS Local Zones User Guide) per creare una sottorete nella Local Zone prescelta.

> **Tip**
>
> Prendi nota del nome della sottorete della zona locale.

Fase 2: Creazione NodeClass per la sottorete della zona locale

Dopo aver creato la sottorete Local Zone, è necessario definire una sottorete NodeClass che faccia riferimento a questa sottorete. NodeClass Si tratta di una risorsa personalizzata Kubernetes che specifica gli attributi di infrastruttura per i nodi, incluse le sottoreti, i gruppi di sicurezza e le configurazioni di archiviazione da utilizzare. Nell'esempio seguente, creiamo una NodeClass chiamata «zona locale» che ha come target una sottorete di zona locale in base al suo nome. È inoltre possibile utilizzare l'ID della sottorete. Dovrai adattare questa configurazione per indirizzare la tua sottorete Local Zone.

Per ulteriori informazioni, consulta the section called “Creazione di una classe di nodi”.

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: local-zone
spec:
  subnetSelectorTerms:
```

```
    - id: <local-subnet-id>
```

Fase 3: Crea NodePool con NodeClass e Taint

Una volta NodeClass configurato, ora devi crearne uno NodePool che lo NodeClass utilizzi. A NodePool definisce le caratteristiche di calcolo dei nodi, inclusi i tipi di istanza. NodePool utilizza il NodeClass come riferimento per determinare dove avviare le istanze.

Nell'esempio seguente, creiamo un NodePool riferimento alla nostra «zona locale». NodeClass Aggiungiamo anche una macchia ai nodi per garantire che solo i pod con una tolleranza corrispondente possano essere programmati su questi nodi della Zona Locale. Ciò è particolarmente importante per i nodi della zona locale, che in genere hanno costi più elevati e devono essere utilizzati solo da carichi di lavoro che beneficiano specificamente della latenza ridotta.

Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

```
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"]
```

La contaminazione tra key `aws.amazon.com/local-zone` ed effect `NoSchedule` assicura che i pod senza una tolleranza corrispondente non vengano programmati su questi nodi. Ciò impedisce l'esecuzione accidentale di carichi di lavoro regolari nella zona locale, il che potrebbe comportare costi imprevisti.

Fase 4: Implementazione dei carichi di lavoro con tolleranza e affinità tra nodi

Per un controllo ottimale sul posizionamento dei carichi di lavoro sui nodi della zona locale, utilizzate entrambi taints/tolerations e l'affinità dei nodi insieme. Questo approccio combinato offre i seguenti vantaggi:

1. Controllo dei costi: The taint garantisce che solo i pod con tolleranze esplicite possano utilizzare risorse potenzialmente costose della Zona Locale.
2. Posizionamento garantito: l'affinità dei nodi garantisce che le applicazioni sensibili alla latenza vengano eseguite esclusivamente nella zona locale, non sui normali nodi del cluster.

Ecco un esempio di distribuzione configurata per l'esecuzione specifica sui nodi della zona locale:

```
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"
```

Questa distribuzione ha due configurazioni di pianificazione chiave:

1. La tolleranza consente di programmare i pod sui nodi con la contaminazione. `aws.amazon.com/local-zone`
2. Il requisito di affinità dei nodi garantisce che questi pod funzionino solo sui nodi con l'etichetta. `node-type: local-zone`

Insieme, garantiscono che l'applicazione sensibile alla latenza venga eseguita solo sui nodi della zona locale e che le applicazioni normali non consumino le risorse della zona locale a meno che non siano configurate esplicitamente per farlo.

Passaggio 5: verifica con la console AWS

Dopo aver configurato le NodeClass distribuzioni e le distribuzioni, è necessario verificare che i nodi vengano forniti nella zona locale come previsto e che i carichi di lavoro vengano eseguiti su di essi. NodePool È possibile utilizzare la console di AWS gestione per verificare che le EC2 istanze vengano avviate nella sottorete della zona locale corretta.

Inoltre, puoi controllare l'elenco dei nodi Kubernetes utilizzando `kubectl get nodes -o wide` per confermare che i nodi si uniscano al cluster con le etichette e le sfumature corrette:

```
kubectl get nodes -o wide
kubectl describe node <node-name> | grep -A 5 Taints
```

Puoi anche verificare che i pod del carico di lavoro siano pianificati sui nodi della zona locale:

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

Questo approccio garantisce che su questi nodi vengano pianificati solo i carichi di lavoro che tollerano specificamente la contaminazione della zona locale, aiutandoti a controllare i costi e a utilizzare nel modo più efficiente le risorse della zona locale.

Configurare le impostazioni di sicurezza avanzate per i nodi

Questo argomento descrive come configurare le impostazioni di sicurezza avanzate per i nodi Amazon EKS Auto Mode utilizzando le `advancedSecurity` specifiche nella tua classe di nodi.

Prerequisiti

Prima di iniziare, assicurati di disporre dei seguenti elementi:

- Un cluster della modalità automatica di Amazon EKS. Per ulteriori informazioni, consulta the section called "Creazione di un cluster".
- `kubectl` installato e configurato. Per ulteriori informazioni, consulta Configurazione.
- Comprensione della configurazione della classe di nodi. Per ulteriori informazioni, consulta the section called "Creazione di una classe di nodi".

Configura le impostazioni di sicurezza avanzate

Per configurare le impostazioni di sicurezza avanzate per i tuoi nodi, imposta i `advancedSecurity` campi nella specifica della tua classe di nodi:

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: security-hardened
spec:
  role: MyNodeRole

  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"

  securityGroupSelectorTerms:
```

```
    - tags:
        Name: "eks-cluster-sg"

  advancedSecurity:
    # Enable FIPS-compliant AMIs (US regions only)
    fips: true

    # Configure kernel lockdown mode
    kernelLockdown: "integrity"
```

Applica questa configurazione:

```
kubectl apply -f nodeclass.yaml
```

Fai riferimento a questa classe di nodi nella configurazione del tuo pool di nodi. Per ulteriori informazioni, consulta the section called "Creazione di pool di nodi".

Descrizioni dei campi

- `fips`(booleano, opzionale): se impostato su`true`, effettua il provisioning dei nodi utilizzando moduli crittografici AMIs convalidati FIPS 140-2. Questa impostazione seleziona la conformità a FIPS; i clienti sono responsabili della gestione dei propri requisiti di conformità AMIs. Per ulteriori informazioni, consulta Conformità FIPS.AWS Default: `false`.
- `kernelLockdown`(stringa, opzionale): Controlla la modalità del modulo di sicurezza Kernel Lockdown. Valori accettati:
  - `integrity`: blocca i metodi per sovrascrivere la memoria del kernel o modificare il codice del kernel. Impedisce il caricamento dei moduli del kernel non firmati.
  - `none`: disattiva la protezione dal blocco del kernel.

    Per ulteriori informazioni, consultate la documentazione relativa al blocco del kernel Linux.

Considerazioni

- Gli standard FIPS AMIs sono disponibili nelle regioni Stati Uniti orientali/occidentali, (AWS Stati Uniti) e Canada (AWS GovCloud Centro/Ovest). AWS Per ulteriori informazioni,AWS consulta Conformità FIPS.
- Durante l'utilizzo`kernelLockdown: "integrity"`, assicurati che i tuoi carichi di lavoro non richiedano il caricamento di moduli del kernel non firmati o la modifica della memoria del kernel.

Risorse correlate

- the section called "Creazione di una classe di nodi"- Guida completa alla configurazione della classe di nodi
- the section called "Creazione di pool di nodi"- Configurazione del pool di nodi

Scopri come funziona EKS Auto Mode

Utilizza questo capitolo per scoprire come funzionano i componenti dei cluster di Amazon EKS Auto Mode.

Argomenti

- Informazioni sulle istanze gestite della modalità automatica di Amazon EKS
- Informazioni su identità e accesso in modalità automatica di EKS
- Informazioni sulla rete VPC e sul bilanciamento del carico in modalità automatica di EKS

Informazioni sulle istanze gestite della modalità automatica di Amazon EKS

Questo argomento spiega come Amazon EKS Auto Mode gestisce EC2 le istanze Amazon nel tuo cluster EKS. Quando abiliti la modalità automatica EKS, le risorse di calcolo del cluster vengono automaticamente fornite e gestite da EKS, cambiando il modo in cui interagisci con le EC2 istanze che fungono da nodi nel cluster.

Comprendere come la modalità automatica di Amazon EKS gestisce le istanze è essenziale per pianificare la strategia di implementazione dei carichi di lavoro e le procedure operative. A differenza delle EC2 istanze tradizionali o dei gruppi di nodi gestiti, queste istanze seguono un modello di ciclo di vita diverso in cui EKS si assume la responsabilità di molti aspetti operativi, limitando al contempo determinati tipi di accesso e personalizzazione.

La modalità automatica di Amazon EKS automatizza le attività di routine per la creazione di nuove EC2 istanze e le collega come nodi al cluster EKS. La modalità automatica EKS rileva quando un carico di lavoro non può adattarsi ai nodi esistenti e crea una nuova istanza. EC2

La modalità automatica di Amazon EKS è responsabile della creazione, dell'eliminazione e dell'applicazione di patch EC2 alle istanze. Sei responsabile dei container e dei pod implementati sull'istanza.

EC2 Le istanze create da EKS Auto Mode sono diverse dalle altre EC2 istanze, sono istanze gestite. Queste istanze gestite sono di proprietà di EKS e sono più limitate. Non puoi accedere o installare direttamente il software sulle istanze gestite dalla modalità automatica EKS.

AWS suggerisce di eseguire EKS Auto Mode o Karpenter autogestito. Puoi installarli sia durante una migrazione che in una configurazione avanzata. Se hai già installato entrambi, configura i pool di nodi in modo tale che i carichi di lavoro siano associati a Karpenter o modalità automatica EKS.

Per ulteriori informazioni, consulta le istanze EC2 gestite da Amazon nella guida per l' EC2 utente di Amazon.

Tabella di confronto

Istanza standard EC2	Istanza gestita dalla modalità automatica EKS
Sei responsabile dell'applicazione di patch e dell'aggiornamento dell'istanza.	AWS corregge e aggiorna automaticamente l'istanza.
EKS non è responsabile del software sull'istanza.	EKS è responsabile di determinati software sull'istanza, come `kubelet`, il runtime del container e il sistema operativo.
È possibile eliminare l' EC2 istanza utilizzando l' EC2 API.	EKS determina il numero di istanze implementate nell'account. Se elimini un carico di lavoro, EKS ridurrà il numero di istanze nell'account.
È possibile utilizzare SSH per accedere all' EC2 istanza.	Puoi implementare pod e container nell'istanza gestita.
Puoi determinare il sistema operativo e l'immagine (AMI).	AWS determina il sistema operativo e l'immagine.
Puoi implementare carichi di lavoro che si basano sulle funzionalità di Windows o Ubuntu.	Puoi distribuire container basati su Linux, ma senza dipendenze specifiche del sistema operativo.
Puoi determinare il tipo e la famiglia di istanze da avviare.	AWS determina il tipo e la famiglia di istanze da avviare. Puoi utilizzare un pool di nodi

Istanza standard EC2	Istanza gestita dalla modalità automatica EKS
	per limitare i tipi di istanze tra cui la modalità automatica EKS effettua la selezione.

Le seguenti funzionalità sono valide sia per le istanze gestite che per le EC2 istanze standard:

- È possibile visualizzare l'istanza nella AWS console.
- Puoi utilizzare lo storage delle istanze come storage temporaneo per i carichi di lavoro.

Supporto AMI

Con EKS Auto Mode, AWS determina l'immagine (AMI) utilizzata per i nodi di calcolo. AWS monitora il lancio delle nuove versioni AMI EKS Auto Mode. Se riscontri problemi di carico di lavoro relativi a una versione AMI, crea una richiesta di supporto. Per ulteriori informazioni, consulta Creazione di casi di supporto e gestione dei casi nella AWS Support User Guide.

In genere, EKS rilascia una nuova AMI ogni settimana contenente CVE e correzioni di sicurezza.

Riferimento all'istanza supportata dalla modalità automatica EKS

La modalità automatica EKS crea solo istanze dei tipi supportati e che soddisfano un requisito di dimensione minima.

modalità automatica di EKS supporta i tipi di istanza indicati di seguito:

Family	Tipi di istanza
Compute Optimized (C)	c8i, c8i-flex, c8gd, c8gn, c8g, c7a, c7gn, c7gd, c7i, c7i-flex, c6a, c6g, c6i, c6gn, c6id, c6in, c6gd, c5, c5a, c5d, c5d, c5d annuncio, c5n, c4
General Purpose (M)	m8i, m8i-flex, m8a, m8gn, 8mb, m8gd, m7i, m7a, m7g, m7gd, m7i-flex, m6a, m6m, m6m, m6idn, m6id, m6gd, m5a, m5a, m5, m5n, m5dn, m5d, m5zn, m4
Memory Optimized (R)	r8i, r8i-flex, r8gn, r8gb, r8gd, r7a, r7iz, r7gd, r7i, r7g, r6a, r6i, r6id, r6in, r6idn, r6g, r6gd, r5, r5n, r5a, r5a, r5d, r5a, r5a, r5d, r5a, r5d, r5a, r5a, r5d, r5a, r5d, r5a, r5a, r5d, r5a, r5d, r5a, r5a, r5d, r5a, r5d, r5a, r5d, r5a, r5a, r5b, 5ad, 5d, 4r

Family	Tipi di istanza
Burstable (T)	t4g, t3, t3a, t2
High Memory (Z/X)	z1d, x8g, x2gd
Storage Optimized (I/D)	i8ge, i7i, i8g, i7ie, i4g, i4i, i3, i3en, is4gen, d3, d3en, im4gn
P/G/Inf/T rnCalcolo accelerato ()	p5, p4d, p4de, p3, p3dn, gr6, g6, g6e, g5g, g5, g4dn, inf2, inf1, trn1, trn1n
High Performance Computing (X2)	x2iezn, x2iedn, x2idn

Inoltre, EKS Auto Mode creerà solo EC2 istanze che soddisfano i seguenti requisiti:

- Più di 1 CPU
- Le dimensioni delle istanze non sono nano, micro o piccole

Per ulteriori informazioni, consulta le convenzioni di denominazione dei tipi di EC2 istanze di Amazon.

Servizio di metadati dell'istanza

- La modalità automatica EKS viene applicata IMDSv2 con un limite di hop pari a 1 per impostazione predefinita, aderendo alle AWS migliori pratiche di sicurezza.
- Questa configurazione predefinita non può essere modificata in modalità automatica.
- Per i componenti aggiuntivi che in genere richiedono l'accesso a IMDS, fornite i parametri (come la AWS regione) durante l'installazione per evitare le ricerche IMDS. Per ulteriori informazioni, consulta the section called "Campi che puoi personalizzare".
- Se un Pod richiede assolutamente l'accesso IMDS quando viene eseguito in modalità automatica, il Pod deve essere configurato per funzionare con `hostNetwork: true`. Ciò consente al Pod di accedere direttamente al servizio di metadati dell'istanza.

- Considerare le implicazioni sulla sicurezza quando si concede ai Pod l'accesso ai metadati dell'istanza.

Per ulteriori informazioni su Amazon EC2 Instance Metadata Service (IMDS), consulta le opzioni di Configurazione delle opzioni del servizio di metadati dell'istanza nella Amazon EC2 User Guide.

Considerazioni

- Se lo storage temporaneo configurato in NodeClass è più piccolo dello storage NVMe locale per l'istanza, EKS Auto Mode elimina la necessità di una configurazione manuale eseguendo automaticamente le seguenti azioni:
  - Utilizza un volume inferiore di dati Amazon EBS (20 GiB) per ridurre i costi.
  - Formatta e configura l'archiviazione NVMe locale per l'uso temporaneo dei dati. Ciò include la configurazione di un array RAID 0 se sono presenti più unità. NVMe
- Quando `ephemeralStorage.size` è uguale o superiore alla NVMe capacità locale, si verificano le seguenti azioni:
  - La modalità automatica salta il volume EBS piccolo.
  - Le NVMe unità sono esposte direttamente al carico di lavoro.
- La modalità automatica di Amazon EKS non supporta le seguenti azioni del servizio AWS Fault Injection:
  - `ec2:RebootInstances`
  - `ec2:SendSpotInstanceInterruptions`
  - `ec2:StartInstances`
  - `ec2:StopInstances`
  - `ec2:TerminateInstances`
  - `ec2:PauseVolumeIO`
- La modalità automatica di Amazon EKS supporta le azioni EKS Pod del servizio AWS Fault Injection. Per ulteriori informazioni, consulta Managing AWS Fault Injection Service e Usa le azioni FIS aws:eks:pod nella Guida per l' AWS utente di Resilience Hub.
- Non devi installare il `Neuron Device Plugin` sui nodi della modalità automatica EKS.

  Se hai altri tipi di nodi nel cluster, devi configurare il plug-in Neuron Device in modo tale che non venga eseguito su nodi in modalità automatica. Per ulteriori informazioni, consulta the section called "Controllare l'implementazione".

Informazioni su identità e accesso in modalità automatica di EKS

Questo argomento descrive i ruoli e le autorizzazioni di Identity and Access Management (IAM) necessari per utilizzare modalità automatica di EKS. La modalità automatica di EKS utilizza due ruoli IAM primari: un ruolo IAM del cluster e uno del nodo. Questi ruoli lavorano in combinazione con EKS Pod Identity e le voci di accesso EKS per fornire una gestione completa degli accessi per i cluster EKS.

Quando configuri la modalità automatica EKS, dovrai configurare questi ruoli IAM con autorizzazioni specifiche che consentano ai AWS servizi di interagire con le risorse del cluster. Ciò include le autorizzazioni per la gestione delle risorse di elaborazione, dei volumi di archiviazione, dei bilanciatori del carico e dei componenti di rete. La comprensione di queste configurazioni dei ruoli è essenziale per il corretto funzionamento e la sicurezza del cluster.

In EKS Auto Mode, i ruoli AWS IAM vengono mappati automaticamente alle autorizzazioni Kubernetes tramite le voci di accesso EKS, eliminando la necessità di configurazioni manuali o associazioni personalizzate. `aws-auth` ConfigMaps Quando crei un nuovo cluster in modalità auto, EKS crea automaticamente le autorizzazioni Kubernetes corrispondenti utilizzando le voci Access, assicurando che i AWS servizi e i componenti del cluster abbiano i livelli di accesso appropriati sia all'interno del sistema di autorizzazione che in Kubernetes. AWS Questa integrazione automatizzata riduce la complessità della configurazione e aiuta a prevenire i problemi relativi alle autorizzazioni che si verificano comunemente durante la gestione dei cluster EKS.

Ruolo IAM del cluster

Il ruolo Cluster IAM è un ruolo AWS Identity and Access Management (IAM) utilizzato da Amazon EKS per gestire le autorizzazioni per i cluster Kubernetes. Questo ruolo concede ad Amazon EKS le autorizzazioni necessarie per interagire con altri AWS servizi per conto del cluster e viene configurato automaticamente con le autorizzazioni Kubernetes utilizzando le voci di accesso EKS.

- È necessario collegare AWS le politiche IAM a questo ruolo.
- La modalità automatica di EKS assegna automaticamente le autorizzazioni Kubernetes a questo ruolo utilizzando le voci di accesso EKS.
- Con EKS Auto Mode, AWS suggerisce di creare un singolo ruolo Cluster IAM per AWS account.
- AWS suggerisce di assegnare un nome a questo ruolo`AmazonEKSAutoClusterRole`.
- Questo ruolo richiede le autorizzazioni per più AWS servizi per gestire le risorse, inclusi volumi EBS, Elastic Load Balancer e istanze. EC2

- La configurazione suggerita per questo ruolo include più politiche IAM AWS gestite, correlate alle diverse funzionalità di EKS Auto Mode.
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicy`
  - `AmazonEKSLoadBalancingPolicy`
  - `AmazonEKSNetworkingPolicy`
  - `AmazonEKSClusterPolicy`

Per ulteriori informazioni sul ruolo Cluster IAM e sulle politiche IAM AWS gestite, consulta:

- the section called " AWS politiche gestite"
- the section called "Ruolo IAM del cluster"

Per ulteriori informazioni sull'accesso a Kubernetes, consulta:

- the section called "Rivedere le policy di accesso"

Ruolo IAM del nodo

Il ruolo Node IAM è un ruolo AWS Identity and Access Management (IAM) utilizzato da Amazon EKS per gestire le autorizzazioni per i nodi di lavoro nei cluster Kubernetes. Questo ruolo concede EC2 alle istanze che funzionano come nodi Kubernetes le autorizzazioni necessarie per interagire con AWS servizi e risorse e viene configurato automaticamente con le autorizzazioni RBAC di Kubernetes utilizzando le voci di accesso EKS.

- È necessario collegare le politiche IAM a questo ruolo. AWS
- La modalità automatica di EKS assegna automaticamente le autorizzazioni Kubernetes RBAC a questo ruolo utilizzando le voci di accesso EKS.
- AWS suggerisce di assegnare un nome a questo ruolo`AmazonEKSAutoNodeRole`.
- Con EKS Auto Mode, AWS suggerisce di creare un singolo ruolo IAM Node per AWS account.
- Questo ruolo dispone di autorizzazioni limitate. Le autorizzazioni chiave includono l'assunzione di un ruolo Pod Identity e l'estrazione di immagini da ECR.
- AWS suggerisce le seguenti politiche IAM AWS gestite:
  - `AmazonEKSWorkerNodeMinimalPolicy`

- `AmazonEC2ContainerRegistryPullOnly`

Per ulteriori informazioni sul ruolo IAM del cluster e sulle politiche IAM AWS gestite, consulta:

- the section called " AWS politiche gestite"
- the section called "Ruolo IAM del nodo"

Per ulteriori informazioni sull'accesso a Kubernetes, consulta:

- the section called "Rivedere le policy di accesso"

Ruolo collegato al servizio

Amazon EKS utilizza un ruolo collegato ai servizi (SLR) per determinate operazioni. Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. I ruoli collegati ai servizi sono predefiniti da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

AWS crea e configura automaticamente la reflex. Puoi eliminare un SLR solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché non puoi rimuovere involontariamente delle autorizzazioni di accesso alle risorse.

La policy SLR concede ad Amazon EKS le autorizzazioni per osservare ed eliminare i componenti principali dell'infrastruttura: EC2 risorse (istanze, interfacce di rete, gruppi di sicurezza), risorse ELB (sistemi di bilanciamento del carico, gruppi target), CloudWatch funzionalità (registrazione e metriche) e ruoli IAM con prefisso «eks». Consente inoltre il networking privato degli endpoint tramite l'associazione di VPC/hosted zone e include autorizzazioni per il monitoraggio e la pulizia delle risorse con tag EKS. EventBridge

Per ulteriori informazioni, consulta:

- the section called "AWS politica gestita: Amazon EKSService RolePolicy"
- the section called "Autorizzazioni del ruolo collegato ai servizi per Amazon EKS"

Tag personalizzati per AWS le risorse EKS Auto

Per impostazione predefinita, le politiche gestite relative alla modalità automatica EKS non consentono l'applicazione di tag definiti dall'utente alle AWS risorse fornite da Auto Mode. Se si

desidera applicare tag definiti dall'utente alle AWS risorse, è necessario assegnare autorizzazioni aggiuntive al ruolo Cluster IAM con autorizzazioni sufficienti per creare e modificare tag sulle risorse. AWS Di seguito è riportato un esempio di policy che consentirà l'accesso ai tag senza restrizioni:

Visualizzare un esempio di policy sui tag personalizzata

```
{
    "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}"
                }
            }
        }
    ]
}
```

Riferimento alla policy di accesso

Per ulteriori informazioni sulle autorizzazioni di Kubernetes utilizzate dalla modalità automatica di EKS, consulta the section called "Rivedere le policy di accesso".

Informazioni sulla rete VPC e sul bilanciamento del carico in modalità automatica di EKS

Questo argomento spiega come configurare le funzionalità di rete e bilanciamento del carico del cloud privato virtuale (VPC) in modalità automatica di EKS. Sebbene modalità automatica di EKS gestisca automaticamente la maggior parte dei componenti di rete, puoi comunque personalizzare alcuni aspetti della configurazione di rete del cluster tramite risorse `NodeClass` e annotazioni del bilanciatore del carico.

Quando utilizzi EKS Auto Mode, AWS gestisce la configurazione VPC Container Network Interface (CNI) e il provisioning del load balancer per il tuo cluster. Puoi influenzare i comportamenti di rete

definendo oggetti `NodeClass` e applicando annotazioni specifiche alle risorse Service e Ingress, mantenendo al contempo il modello operativo automatizzato fornito dalla modalità automatica di EKS.

Funzionalità di rete

modalità automatica di EKS ha una nuova funzionalità di rete che gestisce la rete di nodi e pod. Puoi configurarlo creando un oggetto `NodeClass` di Kubernetes.

Le opzioni di configurazione per il precedente AWS VPC CNI non si applicheranno alla modalità automatica EKS.

Configura la rete con una **`NodeClass`**

La risorsa `NodeClass` in modalità automatica di EKS ti consente di personalizzare alcuni aspetti della funzionalità di rete. Tramite `NodeClass`, puoi specificare le selezioni dei gruppi di sicurezza, controllare il posizionamento dei nodi nelle sottoreti VPC, impostare le policy SNAT, configurare le policy di rete e abilitare la registrazione degli eventi di rete. Questo approccio mantiene il modello operativo automatizzato della modalità automatica di EKS fornendo al contempo flessibilità per la personalizzazione della rete.

Puoi usare una `NodeClass` per:

- Selezionare un gruppo di sicurezza per i nodi
- Controllare come vengono posizionati i nodi nelle sottoreti VPC
- Impostare la policy SNAT del nodo su `random` o `disabled`
- Abilitare le policy di rete Kubernetes, tra cui:
  - Impostare la policy di rete su Rifiuto per default o Autorizzazione per default
  - Abilitare la registrazione degli eventi di rete su un file.
- Isolare il traffico dei pod dal traffico del nodo collegando i pod a diverse sottoreti.

Scopri come creare un Amazon EKS NodeClass.

Considerazioni

La modalità automatica di EKS supporta:

- Policy di rete EKS.
- Le opzioni `HostPort` e `HostNetwork` per i pod di Kubernetes.
- Nodi e pod in sottoreti pubbliche o private.

- Memorizzazione nella cache di query DNS sul nodo.

La modalità automatica di EKS non supporta:

- Gruppi di sicurezza per pod (SGPP).
- Rete personalizzata nella `ENIConfig`. Puoi inserire i pod in più sottoreti o isolarli esclusivamente dal traffico dei nodi con the section called "Selezione della sottorete per i pod".
- Configurazioni warm IP, warm prefix e warm ENI.
- Configurazione minima delle destinazioni IP.
- Altre configurazioni supportate dal AWS VPC CNI open source.
- Configurazioni delle policy di rete come la personalizzazione del timer conntrack (l'impostazione predefinita è 300 s).
- Esportazione dei registri degli eventi di rete in. CloudWatch

Gestione delle risorse di rete

EKS Auto Mode gestisce prefissi, indirizzi IP e gestione dell'interfaccia di rete monitorando NodeClass le risorse per le configurazioni di rete. Il servizio esegue automaticamente diverse operazioni sulle chiavi:

Delega del prefisso

EKS Auto Mode utilizza per impostazione predefinita la delega dei prefissi (/28 prefissi) per il networking dei pod e mantiene un pool caldo predefinito di risorse IP scalabile in base al numero di pod pianificati. Quando viene rilevata la frammentazione della sottorete dei pod, Auto Mode fornisce indirizzi IP secondari (/32). Grazie a questo algoritmo di rete pod predefinito, Auto Mode calcola il numero massimo di pod per nodo in base al numero ENIs e al tipo di istanza IPs supportati (presupponendo il caso peggiore di frammentazione). Per ulteriori informazioni sul numero massimo di indirizzi IP ENIs e IPs per istanza, consultate la sezione Numero massimo di indirizzi IP per interfaccia di rete nella Guida per l'utente. EC2 Le famiglie di istanze di nuova generazione (Nitro v6 e successive) sono generalmente aumentate IPs per tipo di istanza, ENIs e la modalità automatica regola di conseguenza il calcolo del numero massimo di pod.

Per IPv6 i cluster, viene utilizzata solo la delega dei prefissi e la modalità automatica utilizza sempre un limite massimo di pod di 110 pod per nodo.

Gestione del tempo di raffreddamento

Il servizio implementa un pool di cooldown per prefissi o indirizzi secondari IPv4 che non sono più in uso. Dopo la scadenza del tempo di raffreddamento, queste risorse vengono rilasciate nuovamente al VPC. Tuttavia, se i pod riutilizzano queste risorse durante il tempo di raffreddamento, vengono ripristinate dal pool di raffreddamento.

IPv6 Support

Per IPv6 i cluster, EKS Auto Mode fornisce un `/80` IPv6 prefisso per nodo sull'interfaccia di rete principale.

Il servizio garantisce inoltre una corretta gestione e rimozione di oggetti inutili di tutte le interfacce di rete.

Sistema di bilanciamento del carico

È possibile configurare AWS Elastic Load Balancer forniti da EKS Auto Mode utilizzando annotazioni sulle risorse Service e Ingress.

Per ulteriori informazioni, consulta the section called "Creare una classe in ingresso" o the section called "Creare servizio".

Considerazioni sul bilanciamento del carico con la modalità automatica di EKS

- La modalità di targeting predefinita è la modalità IP, non la modalità istanza.
- La modalità automatica di EKS supporta solo la modalità del gruppo di sicurezza per Network Load Balancer.
- AWS non supporta la migrazione dei load balancer dal controller di bilanciamento del carico autogestito alla gestione tramite EKS Auto AWS Mode.
- Il campo `networking.ingress.ipBlock` nella specifica `TargetGroupBinding` non è supportato.
- Se i nodi worker utilizzano gruppi di sicurezza personalizzati (non schemi di denominazione `eks-cluster-sg-`), il ruolo del cluster richiede autorizzazioni IAM aggiuntive. La policy predefinita gestita da EKS consente solo a EKS di modificare i gruppi di sicurezza denominati `eks-cluster-sg-`. Senza l'autorizzazione a modificare i gruppi di sicurezza personalizzati, EKS non può aggiungere le regole di ingresso richieste che consentono al ALB/NLB traffico di raggiungere i pod.

Considerazioni su CoredNS

EKS Auto Mode non utilizza la tradizionale implementazione CoredNS per fornire la risoluzione DNS all'interno del cluster. Invece, i nodi Auto Mode utilizzano CoredNS in esecuzione come servizio di sistema direttamente su ciascun nodo. Se si esegue la transizione di un cluster tradizionale alla modalità automatica, è possibile rimuovere la distribuzione CoredNS dal cluster una volta che i carichi di lavoro sono stati spostati nei nodi in modalità automatica.

> **Important**
>
> Se si prevede di mantenere un cluster con nodi in modalità automatica e non in modalità automatica, è necessario mantenere la distribuzione CoredNS. I nodi non in modalità automatica si affidano ai tradizionali pod CoreDNS per la risoluzione DNS, in quanto non possono accedere al servizio DNS a livello di nodo fornito da Auto Mode.

Osservabilità per la modalità automatica EKS

Usa questo capitolo per conoscere le opzioni di osservabilità per i cluster Amazon EKS Auto Mode.

Argomenti

- Accedi ai registri dei componenti AWS gestiti per EKS Auto

Accedi ai registri dei componenti AWS gestiti per EKS Auto

È possibile accedere ai registri dei componenti AWS gestiti da EKS Auto Mode per ottenere una maggiore osservabilità delle operazioni del cluster. EKS Auto Mode supporta i log per le seguenti fonti:

- Scalabilità automatica del calcolo - Karpenter
- Archiviazione a blocchi - EBS CSI
- Bilanciamento del carico - AWS Load Balancer Controller
- Pod networking - Gestione degli indirizzi IP CNI VPC

I registri possono essere consegnati a una destinazione di consegna di tua scelta.

Quando create un cluster EKS Auto, avete la possibilità di abilitare la registrazione del piano di controllo (server API, Audit, Authenticator, Controller manager, Scheduler). I registri dei componenti gestiti automaticamente da EKS (come Compute, Block storage, Load balancing e IPAM) richiedono una configurazione separata tramite la consegna dei log.

Impostazione della consegna dei log

Per configurare la consegna dei log dei componenti AWS gestiti per il tuo cluster EKS Auto Mode, utilizza l'API Amazon CloudWatch Logs. Per istruzioni dettagliate sulla configurazione, consulta Enabling logging from AWS services nella Amazon CloudWatch Logs User Guide. Ogni funzionalità della modalità automatica può essere configurata come fonte di distribuzione individuale di CloudWatch Vend Logs, che consente di selezionare i log a cui si desidera avere accesso.

EKS Auto Mode supporta i seguenti tipi di log:

- AUTO_MODE_COMPUTE_LOGS
- AUTO_MODE_BLOCK_STORAGE_LOGS
- AUTO_MODE_LOAD_BALANCING_LOGS
- AUTO_MODE_IPAM_LOGS

Usare Amazon CloudWatch APIs

La configurazione della registrazione richiede tre passaggi:

1. Crea una fonte di distribuzione per la funzionalità utilizzando l'API CloudWatch PutDeliverySource
2. Crea una destinazione di consegna utilizzando PutDeliveryDestination
3. Crea una consegna per connettere l'origine e la destinazione utilizzando CreateDelivery

È possibile configurare i dettagli della destinazione per i log di Auto Mode utilizzando l' deliveryDestinationConfiguration oggetto nell' CloudWatch PutDeliveryDestination API. Richiede l'ARN di un gruppo di CloudWatch log, di un bucket S3 o del flusso di distribuzione di Kinesis Data Firehose.

È possibile configurare una singola funzionalità di modalità automatica (origine di consegna) per inviare i log a più destinazioni creando più consegne. Puoi anche creare più consegne per configurare più fonti di consegna per inviare i registri alla stessa destinazione di consegna.

autorizzazioni IAM

A seconda della destinazione selezionata, potrebbe essere necessario configurare le policy o i ruoli IAM per il gruppo di CloudWatch log, il bucket S3 e Kinesis Data Firehose per garantire una corretta consegna dei log. Inoltre, se stai inviando log tra AWS account, dovrai utilizzare l' PutDeliveryDestinationPolicy API per configurare una policy IAM che consenta la consegna alla destinazione. Per ulteriori informazioni, consulta la documentazione sulle autorizzazioni di CloudWatch Vend Logs.

Visualizzazione dei registri

Una volta configurata la consegna dei log, i log verranno consegnati alla destinazione specificata. Il metodo di accesso ai log dipende dal tipo di destinazione scelto:

- CloudWatch Registri: visualizza i log nella console Logs, usa CloudWatch i comandi CLI o esegui una AWS query con Logs Insights CloudWatch
- Amazon S3: accedi ai log come oggetti nel tuo bucket S3 tramite la console S3, l'interfaccia a riga di comando o strumenti di analisi come Amazon AWS Athena
- Amazon Data Firehose: i log vengono trasmessi in streaming al target Firehose configurato (ad esempio S3, Service, Redshift, ecc.) OpenSearch

Prezzi

CloudWatch Le tariffe di Vending Logs si applicano per la consegna e lo stoccaggio dei registri in base alla destinazione di consegna prescelta. CloudWatch Vending Logs consente una consegna affidabile e sicura dei log con AWS autenticazione e autorizzazione integrate a un prezzo ridotto rispetto ai registri standard. CloudWatch Per maggiori dettagli, consulta la sezione Vending Logs della pagina dei CloudWatch prezzi.

Risorse correlate

- Registrazione del piano di controllo Amazon EKS
- PutDeliverySource API nella guida di riferimento all'API CloudWatch Logs
- PutDeliveryDestination API nel riferimento all'API CloudWatch Logs
- CreateDelivery API nel riferimento all'API CloudWatch Logs

Risoluzione dei problemi della modalità automatica di EKS

Con EKS Auto Mode, AWS ti assumi maggiori responsabilità per le istanze EC2 presenti nel tuo account. AWS EKS si assume la responsabilità del runtime dei container sui nodi, del sistema operativo sui nodi e di determinati controller. Ciò include un controller per l'archiviazione a blocchi, uno per il bilanciamento del carico e uno di elaborazione.

È necessario utilizzare AWS e APIs Kubernetes per risolvere i problemi dei nodi. Puoi:

- Utilizzare una risorsa `NodeDiagnostic` di Kubernetes per recuperare i log dei nodi utilizzando the section called "Agente di monitoraggio del nodo". Per le altri fasi, consulta the section called "Ottenere i log dei nodi".
- Usa il `get-console-output` comando CLI AWS EC2 per recuperare l'output della console dai nodi. Per le altri fasi, consulta the section called "Ottieni l'output della console da un'istanza gestita da EC2 utilizzando la AWS CLI EC2".
- Usa i container di debug di Kubernetes per recuperare i log dei nodi. Per le altri fasi, consulta the section called "Ottieni i log dei nodi utilizzando i container di debug e `kubectl` CLI".

> **Note**
>
> modalità automatica di EKS utilizza istanze gestite EC2. Non puoi accedere direttamente alle istanze gestite EC2, anche tramite SSH.

Potresti riscontrare i seguenti problemi che hanno soluzioni specifiche per i componenti della modalità automatica di EKS:

- Pod bloccati nello stato `Pending`, che non vengono pianificati sui nodi Auto Mode. Per le soluzioni, consulta the section called "Risolvere i problemi relativi alla mancata pianificazione dei Pod sul nodo Auto Mode".
- Istanze gestite da EC2 che non entrano a far parte del cluster come nodi di Kubernetes. Per le soluzioni, consulta the section called "Risolvere i problemi del nodo che non si unisce al cluster".
- Errori e problemi con i `NodePools`, `PersistentVolumes` e `Services` che utilizzano i controller inclusi in modalità automatica di EKS. Per le soluzioni, consulta the section called "Risolvere i problemi relativi ai controller inclusi in Auto Mode".

- La sicurezza avanzata dei Pod impedisce la condivisione di volumi tra Pod. Per le soluzioni, consulta the section called "Condivisione di volumi tra pod".

Puoi utilizzare i seguenti metodi per risolvere i problemi relativi ai componenti della modalità automatica di EKS:

- the section called "Ottieni l'output della console da un'istanza gestita da EC2 utilizzando la AWS CLI EC2"
- the section called "Ottieni i log dei nodi utilizzando i container di debug e `kubectl` CLI"
- the section called "Visualizza le risorse associate alla modalità automatica EKS nella console AWS"
- the section called "Visualizza gli errori IAM nel tuo AWS account"
- the section called "Rilevare i problemi di connettività dei nodi con il `VPC Reachability Analyzer`"

Agente di monitoraggio del nodo

modalità automatica di EKS include l'agente di monitoraggio dei nodi di Amazon EKS. Puoi utilizzare questo agente per visualizzare le informazioni di risoluzione dei problemi e di debug relative ai nodi. L'agente di monitoraggio dei nodi pubblica gli `events` di Kubernetes e le `conditions` del nodo. Per ulteriori informazioni, consulta the section called "Integrità del nodo".

Ottieni l'output della console da un'istanza gestita da EC2 utilizzando la AWS CLI EC2

Questa procedura aiuta a risolvere problemi in fase di avvio o a livello di kernel.

Innanzitutto, devi determinare l'ID dell'istanza EC2 dell'istanza associata al carico di lavoro. In secondo luogo, utilizzate la AWS CLI per recuperare l'output della console.

1. Conferma di avere `kubectl` installato e connesso al cluster
2. (Facoltativo) Utilizza il nome di un'implementazione di Kubernetes per elencare i pod associati.

```
kubectl get pods -l app=<deployment-name>
```

3. Utilizza il nome del pod di Kubernetes per determinare l'ID dell'istanza EC2 del nodo associato.

```
kubectl get pod <pod-name> -o wide
```

4. Utilizza l'ID dell'istanza EC2 per recuperare l'output della console.

```
aws ec2 get-console-output --instance-id <instance id> --latest --output text
```

Ottieni i log dei nodi utilizzando i container di debug e **`kubectl`** CLI

Il modo consigliato per recuperare i log da un nodo della modalità automatica di EKS prevede l'utilizzo della risorsa `NodeDiagnostic`. Per la procedura, consulta the section called "Ottenere i log dei nodi".

Tuttavia, puoi eseguire lo streaming dei log in tempo reale da un'istanza utilizzando il comando `kubectl debug node`. Questo comando avvia un nuovo Pod sul nodo di cui desideri eseguire il debug, che puoi quindi utilizzare in modo interattivo.

1. Avvia un container di debug. Il comando seguente utilizza `i-0123456789012345`6 per l'ID di istanza del nodo, `-it` alloca a `tty` e allega `stdin` per l'utilizzo interattivo, e utilizza il profilo `sysadmin` del file kubeconfig.

```
kubectl debug node/i-0123456789012345`6 -it --profile=sysadmin --image=public.ecr.aws/
amazonlinux/amazonlinux:2023
```

   Di seguito viene riportato un output di esempio.

```
Creating debugging pod node-debugger-i-0123456789012345`6-nxb9c with container
 debugger on node i-0123456789012345`6.
If you don't see a command prompt, try pressing enter.
bash-5.2#
```

2. Dalla shell, puoi ora installare `util-linux-core` che fornisce il comando `nsenter`. Utilizza `nsenter` per inserire il namespace di montaggio di PID 1 (`init`) sull'host ed eseguire il comando `journalctl` per eseguire lo streaming dei log da `kubelet`:

```
yum install -y util-linux-core
nsenter -t 1 -m journalctl -f -u kubelet
```

Per motivi di sicurezza, l'immagine del container di Amazon Linux non installa molti file binari per impostazione predefinita. Puoi utilizzare il comando `yum whatprovides` per identificare il pacchetto da installare al fine di fornire un determinato file binario.

```
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
```

Visualizza le risorse associate alla modalità automatica EKS nella console AWS

È possibile utilizzare la AWS console per visualizzare lo stato delle risorse associate al cluster EKS Auto Mode.

- Volumi EBS
  - Visualizza i volumi della modalità automatica di EKS cercando la chiave del tag `eks:eks-cluster-name`
- Sistemi di bilanciamento del carico
  - Visualizza i bilanciatori del carico della modalità automatica di EKS cercando la chiave del tag `eks:eks-cluster-name`
- Istanze EC2
  - Visualizza le istanze della modalità automatica di EKS cercando la chiave del tag `eks:eks-cluster-name`

Visualizza gli errori IAM nel tuo AWS account

1. Vai alla CloudTrail console
2. Seleziona "Cronologia degli eventi" dal riquadro di navigazione a sinistra
3. Applica i filtri dei codici di errore:
   - AccessDenied
   - UnauthorizedOperation
   - InvalidClientTokenId

Cerca gli errori relativi al cluster EKS. Utilizza i messaggi di errore per aggiornare le voci di accesso EKS, il ruolo IAM del cluster o il ruolo IAM del nodo. Potresti dover allegare una nuova policy a questi ruoli con autorizzazioni per la modalità automatica di EKS.

Risolvere i problemi relativi alla mancata pianificazione dei Pod sul nodo Auto Mode

Se i pod rimangono nello stato `Pending` e non vengono pianificati su un nodo Auto Mode, verifica se il pod o il manifesto di implementazione dispone di un `nodeSelector`. Se è presente un `nodeSelector`, assicurati che utilizzi `eks.amazonaws.com/compute-type: auto` per essere pianificato sui nodi creati dalla modalità automatica di EKS. Per ulteriori informazioni sulle etichette dei nodi utilizzate dalla modalità automatica di EKS, consulta the section called "Controllare l'implementazione".

Risolvere i problemi del nodo che non si unisce al cluster

La modalità automatica di EKS configura automaticamente le nuove istanze EC2 con le informazioni corrette per unirsi al cluster, inclusi l'endpoint del cluster e l'autorità di certificazione (CA) del cluster. Tuttavia, queste istanze possono ancora non riuscire a unirsi al cluster EKS sotto forma di nodo. Esegui i seguenti comandi per identificare le istanze che non si sono unite al cluster:

1. Esegui `kubectl get nodeclaim` per verificare le `NodeClaims` che sono `Ready = False`.

```
kubectl get nodeclaim
```

2. Esegui `kubectl describe nodeclaim <node_claim>` e cerca su Stato per trovare eventuali problemi che impediscono al nodo di unirsi al cluster.

```
kubectl describe nodeclaim <node_claim>
```

Messaggi di errore comuni:

`Error getting launch template configs`

Potresti ricevere questo errore se stai impostando tag personalizzati nella `NodeClass` con le autorizzazioni del ruolo IAM del cluster predefinite. Per informazioni, consulta the section called "Identità e accesso".

`Error creating fleet`

Potrebbe esserci qualche problema di autorizzazione con la chiamata `RunInstances` dall'API EC2. Verifica AWS CloudTrail la presenza di errori e verifica the section called "Ruolo IAM del cluster Auto Mode" le autorizzazioni IAM richieste.

Rilevare i problemi di connettività dei nodi con il **`VPC Reachability Analyzer`**

> **Note**
>
> Viene addebitato un costo per ogni analisi eseguita da VPC Reachability Analyzer. Per i dettagli sui prezzi, consulta Prezzi di Amazon VPC.

Un motivo per cui un'istanza non si è unita al cluster è un problema di connettività di rete che impedisce di raggiungere il server API. Per diagnosticare questo problema, puoi utilizzare VPC Reachability Analyzer per eseguire un'analisi della connettività tra un nodo che non riesce a unirsi al cluster e al server API. Avrai bisogno di due informazioni:

- ID di istanza di un nodo che non è in grado di unirsi al cluster
- Indirizzo IP dell'endpoint del server dell'API Kubernetes

Per ottenere l'ID dell'istanza, dovrai creare un carico di lavoro sul cluster per far sì che modalità automatica di EKS avvii un'istanza EC2. Questo crea anche un oggetto `NodeClaim` nel cluster che avrà l'ID dell'istanza. Esegui `kubectl get nodeclaim -o yaml` per stampare tutte le `NodeClaims` del cluster. Ciascuna `NodeClaim` contiene l'ID dell'istanza come campo e di nuovo nel providerID:

```
kubectl get nodeclaim -o yaml
```

Di seguito viene riportato un output di esempio.

```
    nodeName: i-01234567890123456
    providerID: aws:///us-west-2a/i-01234567890123456
```

Puoi determinare l'endpoint del server dell'API Kubernetes eseguendo `kubectl get endpoint kubernetes -o yaml`. Gli indirizzi si trovano nel campo degli indirizzi:

```
kubectl get endpoints kubernetes -o yaml
```

Di seguito viene riportato un output di esempio.

```
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
```

Con queste due informazioni, puoi eseguire l'analisi s. Per prima cosa, accedi a VPC Reachability Analyzer in Console di gestione AWS.

1. Fai clic su "Crea e analizza il percorso"
2. Fornisci un nome per l'analisi (ad esempio "Errore nell'unione del nodo")
3. Per "Tipo di origine", seleziona "Istanze"
4. Inserisci l'ID dell'istanza del nodo soggetto a errore come "Origine"
5. Per la "Destinazione del percorso", seleziona "Indirizzo IP"
6. Inserisci uno degli indirizzi IP per il server API come "Indirizzo di destinazione"

7. Espandi la "Sezione di configurazione aggiuntiva dell'intestazione dei pacchetti"
8. Inserisci una "Porta di destinazione" di 443
9. Seleziona "Protocollo" come TCP se non è già selezionato
10. Fai clic su "Crea e analizza il percorso"
11. L'esecuzione dell'analisi potrebbe richiedere alcuni minuti. Se i risultati dell'analisi mostrano una mancata raggiungibilità, indicheranno dove si trovava l'errore nel percorso di rete in modo da poter risolvere il problema.

Condivisione di volumi tra pod

I nodi EKS Auto Mode sono SELinux configurati con una modalità di applicazione che fornisce un maggiore isolamento tra i Pod in esecuzione sullo stesso nodo. Quando SELinux è abilitata, alla maggior parte dei pod non privilegiati verrà applicata automaticamente la propria etichetta di sicurezza multicategoria (MCS). Questa etichetta MCS è unica per ogni Pod ed è progettata per garantire che un processo in un Pod non possa manipolare uno in un altro Pod o sull'host. Anche se un Pod etichettato viene eseguito come root e ha accesso al filesystem host, non sarà in grado di manipolare i file, effettuare chiamate di sistema sensibili sull'host, accedere al runtime del container oppure ottenere il materiale delle chiavi segrete di kubelet.

Per questo motivo, potresti riscontrare problemi durante il tentativo di condividere dati tra Pod. Ad esempio, una `PersistentVolumeClaim` con una modalità di accesso di `ReadWriteOnce` non consentirà comunque a più Pod di accedere contemporaneamente al volume.

Per abilitare questa condivisione tra i Pod, puoi usare i Pod `seLinuxOptions` per configurare la stessa etichetta MCS su quei Pod. In questo esempio, assegniamo le tre categorie `c123,c456,c789` al Pod. Ciò non entrerà in conflitto con le categorie assegnate automaticamente ai Pod sul nodo, poiché a esse verranno assegnate solo due categorie.

```
securityContext:
  seLinuxOptions:
    level: "s0:c123,c456,c789"
```

Visualizzare gli eventi di Karpenter nei log del piano di controllo

Per i cluster EKS con i log del piano di controllo abilitati, puoi ottenere informazioni sulle azioni e sul processo decisionale di Karpenter interrogando i log. Ciò può essere particolarmente utile per

risolvere i problemi della modalità automatica di EKS relativi al provisioning, al dimensionamento e alla terminazione dei nodi. Per visualizzare gli eventi relativi a Karpenter, utilizza la seguente query Logs Insights: CloudWatch

```
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
```

Questa query filtra gli eventi specifici relativi a Karpenter nei log di auditi kube-apiserver. Gli eventi includono vari stati di interruzione, errori di pianificazione, problemi di capacità e problemi relativi ai nodi. Analizzando questi log, puoi comprendere meglio:

- Perché Karpenter intraprende determinate azioni.
- Qualsiasi problema che impedisca il corretto provisioning, il dimensionamento o la terminazione dei nodi.
- Potenziali problemi di capacità o compatibilità con i tipi di istanze.
- Eventi del ciclo di vita dei nodi come interruzioni, espulsioni o terminazioni.

Per utilizzare questa query:

1. Vai CloudWatch alla console
2. Seleziona “Logs Insights” dal riquadro di navigazione a sinistra

3. Scegli il gruppo di log per i log del piano di controllo del cluster EKS
4. Incolla la query nell'editor di query
5. Regola l'intervallo di tempo in base alle esigenze
6. Eseguire la query

I risultati mostreranno una cronologia degli eventi correlati a Karpenter, aiutandoti a risolvere i problemi e a comprendere il comportamento della modalità automatica di EKS nel cluster. Per esaminare le azioni di Karpenter su un nodo specifico, puoi aggiungere il seguente filtro di riga che specifica l'ID dell'istanza alla suddetta query:

```
|filter @message like /[.replaceable]`i-12345678910123456`/
```

> **Note**
>
> Per utilizzare questa query, è necessario abilitare la registrazione del piano di controllo sul cluster EKS. Se non hai ancora eseguito questa operazione, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".

Risolvere i problemi relativi ai controller inclusi in Auto Mode

Se hai un problema con un controller, dovresti cercare:

- Se le risorse associate a quel controller sono valide e formattate correttamente.
- Se le risorse AWS IAM e Kubernetes RBAC sono configurate correttamente per il tuo cluster. Per ulteriori informazioni, consulta the section called "Identità e accesso".

Risorse correlate

Utilizza questi articoli di re:POST per procedure avanzate di AWS risoluzione dei problemi:

- Come risolvere i problemi di scalabilità comuni in EKS Auto-Mode?
- Come posso risolvere i problemi di provisioning di nodepool e nodeclass personalizzati nella modalità automatica di Amazon EKS?
- Come posso risolvere i pool di nodi integrati in modalità automatica di EKS con stato sconosciuto?

Rivedere le note di rilascio della modalità automatica di EKS

Questa pagina documenta gli aggiornamenti alla modalità automatica di Amazon EKS. Puoi controllare periodicamente questa pagina per annunci su funzionalità, correzioni di bug, problemi noti e funzionalità obsolete.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, puoi iscriverti al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/automode/auto-change.adoc.atom
```

2 febbraio 2026

Funzionalità: è stato aggiunto il supporto per disabilitare il traffico V4Egress dai IPv6 pod nei cluster EKS Auto Mode. IPv6 Per ulteriori informazioni, consulta the section called “Disabilita l' IPv4 uscita dai pod nei cluster. IPv6 IPv6”.

19 dicembre 2025

Funzionalità: è stato aggiunto il supporto per la modalità IP secondaria che fornisce indirizzi IP secondari anziché il prefisso ai nodi Auto. La modalità mantiene un IP secondario come minimo IPTarget e consente di risparmiare risorse IP per i clienti che non hanno bisogno di aggiungere altri prefissi IPs o secondari. Per ulteriori informazioni, consultare the section called “Specifiche della classe di nodi” e the section called “Modalità IP secondaria per i pod”.

19 novembre 2025

Funzionalità: abilitato il pull and unpack parallelo Seekable OCI (SOCI) per le istanze della famiglia G, P e Trn con archiviazione locale. NVMe Il pull and unpack parallelo SOCI viene sempre utilizzato per queste famiglie di istanze con EKS Auto Mode e non sono necessarie modifiche alla configurazione per abilitarlo. Per ulteriori informazioni su SOCI, consulta il blog di lancio.

19 novembre 2025

Funzionalità: è stato aggiunto il supporto per pool di nodi a capacità statica che mantengono un numero fisso di nodi. Per ulteriori informazioni, consulta the section called “Pool di nodi a capacità statica”.

23 ottobre 2025

Funzionalità: gli utenti con cluster nelle regioni degli Stati Uniti possono ora richiedere di utilizzare la compatibilità FIPS AMIs `spec.advancedSecurity.fips` specificando nella loro definizione. NodeClass

1° ottobre 2025

Funzionalità: EKS Auto Mode ora supporta l'implementazione di nodi su AWS Local Zones. Per ulteriori informazioni, consulta the section called "Usa Local Zones".

30 settembre 2025

Funzionalità: è stato aggiunto il supporto per InstanceProfile NodeClass `spec.instanceProfile` al quale si esclude a vicenda dal campo. `spec.role`

29 settembre 2025

Il DRA non è attualmente supportato da EKS Auto Mode.

10 settembre 2025

Compito: gli eventi generati dal controller Auto Mode Compute ora utilizzeranno il nome `eks-auto-mode/compute` anziché `karpenter`.

24 agosto 2025

Correzione di bug: se VPCs si utilizzava un'opzione DHCP impostata con un nome di dominio personalizzato contenente lettere maiuscole, i nodi non riuscivano a unirsi al cluster a causa della generazione di un nome host non valido. Questo problema è stato risolto e i nomi di dominio con lettere maiuscole ora funzionano correttamente.

15 agosto 2025

Correzione di bug: il Pod Identity Agent ora ascolterà solo l'indirizzo IPv4 Link Local in un cluster IPv4 EKS per evitare problemi in cui il Pod non riesce a raggiungere l'indirizzo. IPv6

6 agosto 2025

Caratteristica: è stata aggiunta una nuova configurazione NodeClass `spec.advancedNetworking.associatePublicIPAddress` che può essere utilizzata per impedire l'assegnazione di indirizzi IP pubblici ai nodi EKS Auto Mode

30 giugno 2025

Funzionalità: la modalità automatica NodeClass ora utilizza la chiave KMS personalizzata configurata per crittografare il volume root di sola lettura dell'istanza, oltre al volume di dati. read/write In precedenza, la chiave KMS personalizzata veniva utilizzata solo per crittografare il volume di dati.

20 giugno 2025

Funzionalità: Supporto per il controllo dell'implementazione dei carichi di lavoro in EC2 On-Demand Capacity Reservations (). ODCRs Ciò aggiunge la chiave opzionale `capacityReservationSelectorTerms` a NodeClass, che consente di controllare in modo esplicito l'utilizzo dei carichi di lavoro. ODCRs Per ulteriori informazioni, consulta the section called "Controllare le prenotazioni della capacità".

13 giugno 2025

Funzionalità: supporto per sottoreti dei pod separate nella `NodeClass`. Vengono aggiunte le chiavi facoltative `` `podSubnetSelectorTerms `` e `podSecurityGroupSelectorTerms` per impostare le sottoreti e i gruppi di sicurezza per i pod. Per ulteriori informazioni, consulta the section called "Selezione della sottorete per i pod".

30 aprile 2025

Funzionalità: supporto per proxy di rete inoltrati nella `NodeClass`. Viene aggiunta la chiave facoltativa `advancedNetworking` per impostare il proxy HTTPS. Per ulteriori informazioni, consulta the section called "Specifiche della classe di nodi".

18 aprile 2025

Funzionalità: supporto per la risoluzione di domini .local (in genere riservati per Multicast DNS) tramite DNS unicast.

11 aprile 2025

Funzionalità: aggiunta di `certificateBundles` e `ephemeralStorage.kmsKeyID` a `NodeClass` Per ulteriori informazioni, consulta the section called "Specifiche della classe di nodi".

Funzionalità: velocità di estrazione delle immagini migliorata, in particolare per i tipi di istanze con archiviazione locale delle istanze che possono sfruttare la decompressione più rapida delle immagini.

Correzione di bug: è stata risolta una condizione di gara che FailedCreatePodSandBox causava un errore durante la chiamata: dial tcp 127.0.0. 1:50051: connect: connection not, a volte si verificava un errore durante la programmazione dei Pod su un nodo immediatamente all'avvio.

4 aprile 2025

Funzionalità: aumentare `registryPullQPS` da 5 a 25 e `registryBurst` da 10 a 50 per ridurre la limitazione dell'estrazione delle immagini applicata dal client (`Failed to pull image xyz: pull QPS exceeded`)

31 marzo 2025

Correzione di bug: risoluzione di un problema per cui se un Core DNS Pod è in esecuzione su un nodo in Auto Mode, le query DNS dai Pod sul nodo raggiungevano quel Core DNS Pod anziché il server DNS locale del nodo. Le query DNS dai pod su un nodo in Auto Mode andranno sempre al DNS locale del nodo.

21 marzo 2025

Correzione di bug: i nodi in Auto Mode ora risolvono `kube-dns.kube-system.svc.cluster.local` correttamente quando non vi è un servizio `kube-dns` installato nel cluster. GitHub Risolve il problema #2546.

14 marzo 2025

Funzionalità: uscita `IPv4` abilitata nei cluster `IPv6`. Il traffico `IPv4` in uscita dai cluster `IPv6` in Auto Mode verrà ora tradotto automaticamente all'indirizzo `v4` del nodo ENI primario.

Ciclo di vita e configurazione del cluster di Amazon EKS

Questa sezione fornisce indicazioni approfondite sulla gestione dell'intero ciclo di vita dei cluster Kubernetes e sui diversi modi per configurarli. Tratta la creazione di cluster, il monitoraggio dell'integrità dei cluster, l'aggiornamento delle versioni di Kubernetes e l'eliminazione dei cluster. Include anche argomenti avanzati su come configurare l'accesso agli endpoint, il supporto di enable/disable Windows, la configurazione di cluster privati, l'implementazione della scalabilità automatica e come migliorare la resilienza con lo spostamento zonale per il reindirizzamento del traffico nelle configurazioni Multi-AZ.

Argomenti

- Creare un cluster della modalità automatica di Amazon EKS
- Crea un cluster Amazon EKS.
- Piano di controllo predisposto per Amazon EKS
- Prepararsi agli aggiornamenti delle versioni di Kubernetes e risolvere i problemi di configurazione errata con gli approfondimenti sui cluster
- Aggiornamento del cluster esistente alla nuova versione di Kubernetes
- Eliminazione di un cluster
- Endpoint del server API del cluster
- Implementazione dei nodi Windows su cluster EKS
- Disabilitare il supporto Windows
- Implementazione di cluster privati con accesso limitato a Internet
- Dimensionamento dell'elaborazione dei cluster con Karpenter e Cluster Autoscaler
- Ulteriori informazioni relative allo spostamento zonale in controller di ripristino delle applicazioni Amazon (ARC)
- Abilitazione dello spostamento zonale EKS per evitare zone di disponibilità compromesse

Creare un cluster della modalità automatica di Amazon EKS

Questo argomento offre istruzioni dettagliate per creare un cluster della modalità automatica di Amazon EKS usando opzioni di configurazione avanzate. Vengono trattati i prerequisiti, le opzioni di rete e le configurazioni aggiuntive. Il processo comprende l'impostazione dei ruoli IAM, la

configurazione delle impostazioni del cluster, la specificazione dei parametri di rete e la selezione dei componenti aggiuntivi. Gli utenti possono creare cluster utilizzando la Console di gestione AWS o la AWS CLI, step-by-step con indicazioni fornite per entrambi i metodi.

Gli utenti che desiderano una procedura di configurazione meno complessa possono fare riferimento alle seguenti istruzioni semplificate per la creazione di cluster:

- the section called "CLI di eksctl"
- the section called " AWS CLI"
- the section called "Console di gestione"

Questa guida alla configurazione avanzata è destinata agli utenti che necessitano di un controllo più granulare sulla configurazione del cluster della modalità automatica di EKS e che hanno familiarità con i concetti e i requisiti di Amazon EKS. Prima di iniziare la configurazione avanzata, assicurati di aver soddisfatto tutti i prerequisiti e di avere una conoscenza approfondita dei requisiti di rete e IAM per i cluster della modalità automatica di EKS.

La modalità automatica di EKS richiede autorizzazioni IAM aggiuntive. Per ulteriori informazioni, consulta:

- the section called "Ruoli IAM per cluster della modalità automatica di EKS"
- the section called "Identità e accesso"

> **Note**
>
> Se vuoi creare un cluster senza modalità automatica di EKS, consulta the section called "Creazione di un cluster ".
> In questo argomento viene trattata la configurazione avanzata. Se vuoi iniziare a usare modalità automatica di EKS, consulta the section called "Creazione di un cluster".

Prerequisiti

- Un VPC esistente e sottoreti che soddisfano i requisiti di Amazon EKS. Prima di implementare un cluster da utilizzare in produzione, ti consigliamo di approfondire le nozioni relative ai requisiti del VPC e delle sottoreti. Se non disponi di un VPC e di sottoreti, puoi crearli utilizzando un modello fornito da Amazon EKS. AWS CloudFormation

- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version`. Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando.
- Un IAM principale con le autorizzazioni per creare e modificare risorse EKS e IAM.

Crea cluster - AWS console

1. Aprire la Console Amazon EKS.
2. Seleziona Aggiungi cluster e quindi Crea.
3. In Opzioni di configurazione, seleziona Configurazione personalizzata.
   - In questo argomento viene trattata la configurazione personalizzata. Per ulteriori informazioni sulla configurazione rapida, consulta the section called "Console di gestione".
4. Conferma che l'opzione Usa modalità automatica di EKS sia abilitata.
   - In questo argomento viene trattata la creazione di cluster con la modalità automatica di EKS. Per ulteriori informazioni sulla creazione di cluster senza modalità automatica di EKS, consulta the section called "Creazione di un cluster ".
5. Nella pagina Configura cluster compila i campi seguenti:
   - Nome: un nome per il cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole), trattini e trattini bassi. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
   - Ruolo IAM del cluster: scegli il ruolo IAM del cluster Amazon EKS che hai creato per consentire al piano di controllo Kubernetes di gestire AWS le risorse per tuo conto. Se non hai mai creato in precedenza un ruolo IAM del cluster per la modalità automatica di EKS, seleziona il pulsante Crea ruolo consigliato per creare il ruolo con le autorizzazioni richieste nella console IAM.
   - Versione Kubernetes – la versione di Kubernetes da utilizzare per il cluster. È preferibile selezionare la versione più recente, a meno che non occorra una versione precedente.

- Policy di aggiornamento: la policy della versione di Kubernetes che vuoi impostare per il tuo cluster. Se vuoi che il cluster venga eseguito solo su una versione di supporto standard, puoi scegliere Standard. Se vuoi che il tuo cluster acceda al supporto esteso alla fine del supporto standard per una versione, puoi scegliere Esteso. Se selezioni una versione di Kubernetes che attualmente è in supporto esteso, non puoi selezionare il supporto standard come opzione.

6. Nella sezione Auto Mode Compute della pagina di configurazione del cluster, compila i seguenti campi:
   - Pool di nodi: stabilisci se vuoi utilizzare i pool di nodi incorporati. Per ulteriori informazioni, consulta the section called "Riesamina i pool di nodi integrati".
   - Ruolo IAM del nodo: se abiliti un pool di nodi integrato, devi selezionare un ruolo IAM del nodo. La modalità automatica di EKS assegnerà tale ruolo ai nuovi nodi. Non puoi modificare questo valore dopo la creazione del cluster. Se non hai mai creato in precedenza un ruolo IAM del nodo per la modalità automatica di EKS, seleziona il pulsante Crea ruolo consigliato per creare il ruolo con le autorizzazioni richieste. Per ulteriori informazioni su questo ruolo, consulta the section called "Identità e accesso".
7. Nella sezione Accesso al cluster della pagina Configura cluster, inserisci i seguenti campi:
   - Accesso amministratore cluster Bootstrap: il creatore del cluster diventa automaticamente amministratore Kubernetes. Se vuoi disabilitare questa opzione, seleziona Impedisci l'accesso all'amministratore del cluster.
   - Modalità di autenticazione cluster: modalità automatica di EKS richiede voci di accesso EKS, la modalità di autenticazione API EKS. Facoltativamente, puoi abilitare la modalità di **`ConfigMap`** autenticazione selezionando EKS API e. ConfigMap
8. Inserisci i campi rimanenti nella pagina di configurazione del cluster:
   - Crittografia dei segreti (Facoltativo): scegli di abilitare la crittografia dei segreti per i segreti Kubernetes tramite una chiave KMS. Puoi abilitare questa funzionalità anche dopo la creazione del cluster. Assicurati, tuttavia, di acquisire familiarità con le informazioni contenute nella sezione the section called "Abilitare la crittografia dei segreti".
   - Spostamento zonale ARC: modalità automatica di EKS non supporta lo spostamento zonale Arc.
   - Tag: (facoltativo) aggiunge eventuali tag al cluster. Per ulteriori informazioni, consulta the section called "Assegnazione di tag alle risorse ".

     Quando hai finito con questa pagina, seleziona Avanti.
9. Nella pagina Specifica reti, seleziona i valori dei campi riportati di seguito:

- VPC: crea il cluster in un VPC esistente conforme ai requisiti su VPC di Amazon EKS. Prima di scegliere un VPC, ti consigliamo di acquisire familiarità con i requisiti e le considerazioni nella sezione the section called "Requisiti del VPC e delle sottoreti". Dopo la creazione del cluster, non potrai più modificare il VPC da utilizzare. Se non VPCs ne è elencata nessuna, devi prima crearne una. Per ulteriori informazioni, consulta the section called "Crea un VPC".
- Sottoreti: per impostazione predefinita, tutte le sottoreti disponibili nel VPC specificato nel campo precedente sono preselezionate. È necessario selezionarne almeno due.

  Le sottoreti scelte devono soddisfare i requisiti delle sottoreti di Amazon EKS. Prima di selezionare le sottoreti, consigliamo di acquisire familiarità con tutti i requisiti e le considerazioni su cluster VPC e sottoreti di Amazon EKS.

  Gruppi di sicurezza: (facoltativo) specifica uno o più gruppi di sicurezza da associare alle interfacce di rete create da Amazon EKS.

  Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS.
- Scegli la famiglia di indirizzi IP del cluster: puoi scegliere IPv4tra IPv6.

  Kubernetes assegna indirizzi `IPv4` ai pod e ai servizi, per impostazione predefinita. Prima di utilizzare la famiglia `IPv6`, assicurati di conoscere tutte le considerazioni e i requisiti indicati negli argomenti Requisiti e considerazioni relativi al VPC, the section called "Considerazioni e requisiti relativi alle sottoreti", the section called "Requisiti relativi al gruppo di sicurezza" e the section called "IPv6". Se scegli la famiglia `IPv6`, non puoi specificare un intervallo di indirizzi a cui Kubernetes può assegnare indirizzi di servizio `IPv6` come puoi fare per la famiglia `IPv4`. Kubernetes assegna gli indirizzi del servizio dall'intervallo di indirizzi locale univoco (`fc00::/7`).
- (Facoltativo) Scegli Configure Kubernetes Service IP address range (Configura l'intervallo di indirizzi IP del servizio Kubernetes) e specifica un intervallo **`IPv4`** del servizio.

  L'indicazione di un intervallo personalizzato consente di evitare conflitti tra servizi Kubernetes e altre reti con peering o connesse al VPC. Immetti un intervallo in una notazione CIDR. Ad esempio: `10.2.0.0/16`.

  Il blocco CIDR deve soddisfare i seguenti requisiti:

- Essere compreso in uno degli intervalli seguenti: `10.0.0.0/8` `172.16.0.0/12` o `192.168.0.0/16`.
- Avere una dimensione minima di `/24` e una dimensione massima di `/12`.
- Non sovrapporsi all'intervallo del VPC per le risorse Amazon EKS.

Puoi specificare questa opzione solo con la famiglia di indirizzi `IPv4` e solo durante la creazione del cluster. Se non specifichi un'opzione, Kubernetes assegna gli indirizzi IP del servizio dai blocchi CIDR `10.100.0.0/16` o `172.20.0.0/16`.

- Per Cluster endpoint access (Accesso all'endpoint del cluster), seleziona un'opzione. Puoi modificare questa opzione dopo aver creato il cluster. Prima di selezionare un'opzione non predefinita, assicurati di familiarizzare con le opzioni e le relative implicazioni. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

  Quando hai finito con questa pagina, seleziona Avanti.

10. (Facoltativo) Nella pagina Configura osservabilità, scegli le opzioni Parametri e Registrazione del piano di controllo da attivare. Per impostazione predefinita, i tipi di log sono disattivati.
    - Per ulteriori informazioni sull'opzione relativa alle metriche di Prometheus, consulta the section called "Passaggio 1: attivazione delle metriche Prometheus".
    - Per ulteriori informazioni sulle opzioni Registrazione del piano di controllo, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".
    - Quando hai finito con questa pagina, seleziona Avanti.
11. Nella pagina Select add-ons (Seleziona componenti aggiuntivi), scegli i componenti aggiuntivi da aggiungere al tuo cluster. Puoi scegliere tutti i componenti aggiuntivi Amazon EKS e i componenti aggiuntivi AWS Marketplace di cui hai bisogno. Se i componenti aggiuntivi del AWS Marketplace che desideri installare non sono elencati, puoi fare clic sulla numerazione delle pagine per visualizzare altri risultati di pagina o cercare i componenti aggiuntivi di AWS Marketplace disponibili inserendo il testo nella casella di ricerca. Puoi anche filtrare per categoria, fornitore o modello di prezzi e quindi scegliere i componenti aggiuntivi dai risultati della ricerca. Quando crei un cluster, puoi visualizzare, selezionare e installare tutti i componenti aggiuntivi che supportano EKS Pod Identity, come descritto in dettaglio in the section called "Pod Identity".
    - La modalità automatica di EKS automatizza la funzionalità di alcuni componenti aggiuntivi. Se prevedi di implementare i gruppi di nodi gestiti da EKS nel tuo cluster della modalità automatica di EKS, seleziona Altri componenti aggiuntivi Amazon EKS ed esamina le opzioni. Potrebbe essere necessaria l'installazione di componenti aggiuntivi come CoredNS e kube-proxy. EKS

installerà i componenti aggiuntivi presenti in questa sezione solo su nodi e gruppi di nodi autogestiti.

- Quando hai finito con questa pagina, seleziona Avanti.

12 Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona la versione da installare. Dopo la creazione del cluster, puoi eseguire in qualsiasi momento l'aggiornamento a una versione successiva.

Per i componenti aggiuntivi che supportano EKS Pod Identities, puoi utilizzare la console per generare automaticamente il ruolo con il nome, la politica AWS gestita e la politica di attendibilità precompilati specificamente per il componente aggiuntivo. È possibile riutilizzare i ruoli esistenti o crearne di nuovi per i componenti aggiuntivi supportati Per i passaggi necessari per usare la console per creare ruoli per i componenti aggiuntivi che supportano EKS Pod Identity, consulta the section called "Crea componente aggiuntivo (console)AWS". Se un componente aggiuntivo non supporta EKS Pod Identity, viene mostrato un messaggio con le istruzioni per usare la procedura guidata per creare i ruoli IAM per gli account di servizio (IRSA) dopo la creazione del cluster.

Puoi aggiornare la configurazione di ogni componente aggiuntivo dopo la creazione del cluster. Per ulteriori informazioni sulla configurazione dei componenti aggiuntivi, consulta the section called "Aggiornamento di un componente aggiuntivo". Quando hai finito con questa pagina, seleziona Avanti.

13 Nella pagina Rivedi e crea, controlla le informazioni che hai inserito o selezionato nelle pagine precedenti. Se devi apportare modifiche, seleziona Edit (Modifica). Al termine della configurazione, seleziona Create (Crea). Durante il provisioning del cluster, nel campo Stato viene visualizzato il messaggio CREAZIONE.

> **Note**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

Il provisioning del cluster richiede diversi minuti.

Crea cluster - AWS CLI

Le seguenti istruzioni sulle CLI riguardano la creazione di risorse IAM e la creazione del cluster.

Creazione di un ruolo IAM per un cluster della modalità automatica di EKS

Passaggio 1: creazione della policy di attendibilità

Crea una policy di attendibilità che consenta al servizio Amazon EKS di assumere tale ruolo. Salva la policy come `trust-policy.json`:

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

Passaggio 2: creazione del ruolo IAM

Usa la policy di attendibilità per creare il ruolo IAM del cluster:

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

Passaggio 3: annotazione del ruolo ARN

Recupera e salva l'ARN del nuovo ruolo per usarlo nelle fasi successive:

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

Passaggio 4: collegamento delle policy obbligatorie

Allega le seguenti politiche AWS gestite al ruolo IAM del cluster per concedere le autorizzazioni necessarie:

EKSClusterPolitica di Amazon:

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

EKSComputePolitica di Amazon:

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

Amazon EKSBlock StoragePolicy:

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

Amazon EKSLoad BalancingPolicy:

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

EKSNetworkingPolitica di Amazon:

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

Creazione di un ruolo IAM per un nodo modalità automatica di EKS

Passaggio 1: creazione della policy di attendibilità

Crea una policy di attendibilità che consenta al servizio Amazon EKS di assumere tale ruolo. Salva la policy come `node-trust-policy.json`:

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

Passaggio 2: creare il ruolo IAM del nodo

Utilizza il node-trust-policyfile.json del passaggio precedente per definire quali entità possono assumere il ruolo. Per creare il ruolo IAM del nodo, esegui il seguente comando:

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

Passaggio 3: annotazione del ruolo ARN

Una volta creato il ruolo, recupera e salva l'ARN del ruolo IAM del nodo. Sarà necessario usare questo ARN nei passaggi successivi. Usa il seguente comando per ottenere l'ARN:

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

Passaggio 4: collegamento delle policy obbligatorie

Allega le seguenti policy AWS gestite al ruolo Node IAM per fornire le autorizzazioni necessarie:

Amazon EKSWorker NodeMinimalPolicy:

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

Amazon EC2 ContainerRegistryPullOnly:

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

```
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Crea un cluster

1. Crea un cluster con il comando seguente. Prima di eseguire il comando, apporta le modifiche seguenti:
   - Sostituiscilo *`region-code`* con la AWS regione in cui desideri creare il cluster.
   - Sostituisci *`my-cluster`* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole), trattini e trattini bassi. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
   - Sostituisci *`1.30`* con qualsiasi versione supportata da Amazon EKS.
   - *`111122223333`*Sostituiscilo con l'ID del tuo account
   - Se hai creato ruoli IAM con nomi diversi per i ruoli Cluster e Node, sostituisci ARNs.
   - Sostituisci i valori di `subnetIds` con quelli in tuo possesso. Puoi anche aggiungerne altri IDs. È necessario specificare almeno due sottoreti IDs.

     Le sottoreti scelte devono soddisfare i requisiti delle sottoreti di Amazon EKS. Prima di selezionare le sottoreti, consigliamo di acquisire familiarità con tutti i requisiti e le considerazioni su cluster VPC e sottoreti di Amazon EKS.
   - Se non vuoi specificare un ID del gruppo di sicurezza, rimuovi `,securityGroupIds=sg-<ExampleID1>` dal comando. Se desideri specificare uno o più gruppi di sicurezza IDs, sostituisci i valori di `securityGroupIds` con i tuoi. Puoi anche aggiungerne altri IDs.

     Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da 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**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

Di seguito sono riportate le impostazioni facoltative da aggiungere al comando precedente, quando necessario. Puoi abilitare queste opzioni esclusivamente durante la creazione del cluster, non in seguito.

- Se vuoi specificare il blocco di instradamento interdominio senza classi (CIDR) `IPv4` da cui Kubernetes assegna gli indirizzi IP del servizio, aggiungi `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>` al comando seguente.

  L'indicazione di un intervallo personalizzato consente di evitare conflitti tra servizi Kubernetes e altre reti con peering o connesse al VPC. Immetti un intervallo in una notazione CIDR. Ad esempio: `10.2.0.0/16`.

  Il blocco CIDR deve soddisfare i seguenti requisiti:

  - Essere compreso in uno degli intervalli seguenti: `10.0.0.0/8` `172.16.0.0/12` o `192.168.0.0/16`.
  - Avere una dimensione minima di `/24` e una dimensione massima di `/12`.
  - Non sovrapporsi all'intervallo del VPC per le risorse Amazon EKS.

Puoi specificare questa opzione solo con la famiglia di indirizzi `IPv4` e solo durante la creazione del cluster. Se non specifichi un'opzione, Kubernetes assegna gli indirizzi IP del servizio dai blocchi CIDR `10.100.0.0/16` o `172.20.0.0/16`.

- Se stai creando un cluster e vuoi che il cluster assegni gli indirizzi `IPv6` ai pod e ai servizi anziché a indirizzi `IPv4`, aggiungi `--kubernetes-network-config ipFamily=ipv6` al seguente comando.

  Kubernetes assegna indirizzi `IPv4` ai pod e ai servizi, per impostazione predefinita. Prima di utilizzare la famiglia `IPv6`, assicurati di conoscere tutte le considerazioni e i requisiti indicati negli argomenti Requisiti e considerazioni relativi al VPC, the section called "Considerazioni e requisiti relativi alle sottoreti", the section called "Requisiti relativi al gruppo di sicurezza" e the section called "IPv6". Se scegli la famiglia `IPv6`, non puoi specificare un intervallo di indirizzi a cui Kubernetes può assegnare indirizzi di servizio `IPv6` come puoi fare per la famiglia `IPv4`. Kubernetes assegna gli indirizzi del servizio dall'intervallo di indirizzi locale univoco (`fc00::/7`).

2. Il provisioning del cluster richiede diversi minuti. È possibile eseguire query sullo stato del cluster con il comando seguente.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

Fasi successive

- the section called "Accedere al cluster con kubectl"
- the section called "Voci di accesso"
- Abilitare la crittografia dei segreti per il cluster.
- Configurare la registrazione per il cluster.
- Aggiungi nodi al cluster.

Crea un cluster Amazon EKS.

> **Note**
>
> Questo argomento tratta la creazione di cluster Amazon EKS senza EKS Auto Mode.

Per istruzioni dettagliate su come creare un cluster della modalità automatica di EKS, consulta the section called "Creare un cluster automatico".
Per iniziare a usare modalità automatica di EKS, consulta the section called "Crea cluster (EKS Auto Mode)".

Questo argomento offre una panoramica delle opzioni disponibili e descrivere gli aspetti da considerare quando crei un cluster Amazon EKS. Se devi creare un cluster con la tua infrastruttura on-premises come elaborazione per i nodi, consulta the section called "Crea cluster". Se è la prima volta che crei un cluster Amazon EKS, consigliamo di seguire le nostre guide in Nozioni di base. Le guide consentono di creare un cluster predefinito in modo semplice, senza approfondire tutte le opzioni disponibili.

Prerequisiti

- Un VPC esistente e sottoreti che soddisfano i requisiti di Amazon EKS. Prima di implementare un cluster da utilizzare in produzione, ti consigliamo di approfondire le nozioni relative ai requisiti del VPC e delle sottoreti. Se non disponi di un VPC e di sottoreti, puoi crearli utilizzando un modello fornito da Amazon EKS. AWS CloudFormation
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Un principale IAM con autorizzazioni per `create` e `describe` un cluster Amazon EKS. Per ulteriori informazioni, consultare the section called "Creazione di un cluster Kubernetes locale su un Outpost" e the section called "Elencare o descrivere tutti i cluster".

Passaggio 1: crea un ruolo IAM del cluster

1. Se disponi già di un ruolo IAM del cluster o intendi creare il cluster con `eksctl`, ignora questo passaggio. Per impostazione predefinita, `eksctl` crea un ruolo per te.
2. Per creare un file JSON della policy di attendibilità IAM, esegui il comando seguente.

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

3. Crea il ruolo IAM del cluster Amazon EKS. Se necessario, inserisci il prefisso *`eks-cluster-role-trust-policy.json`* nel percorso del computer in cui hai eseguito la scrittura del file nella Passaggio precedente. Il comando associa il criterio di attendibilità creato nella Passaggio precedente al ruolo. Per creare un ruolo IAM, l'azione `iam:CreateRole` (autorizzazione) deve essere assegnata al principale IAM che sta creando il ruolo.

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

4. Puoi assegnare la policy gestita da Amazon EKS o creare una policy personalizzata. Per conoscere le autorizzazioni minime che devi utilizzare nella policy personalizzata, consulta the section called "Ruolo IAM del cluster".

   Allega la policy gestita di Amazon EKS denominata Amazon EKSCluster Policy al ruolo. Per allegare una policy IAM a un principale IAM, è necessario assegnare al principale che sta allegando la policy una delle azioni IAM (autorizzazioni) seguenti: `iam:AttachUserPolicy` o `iam:AttachRolePolicy`.

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

Ruolo legato al servizio

Amazon EKS crea automaticamente un ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKS`.

Ciò avviene in aggiunta al ruolo IAM del cluster. Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. Il ruolo consente ad Amazon EKS di gestire i cluster nell'account. Per ulteriori informazioni, consulta the section called "Ruolo del cluster".

L'identità IAM usata per creare il cluster EKS deve disporre dell'autorizzazione per creare il ruolo collegato al servizio. Ciò include l'autorizzazione `iam:CreateServiceLinkedRole`.

Se il ruolo collegato al servizio non esiste già e il ruolo IAM attuale non dispone delle autorizzazioni sufficienti per crearlo, l'operazione di creazione del cluster non andrà a buon fine.

Fase 2: Creare un cluster

Puoi creare un cluster utilizzando:

- `eksctl`
- il Console di gestione AWS
- la AWS CLI

Crea cluster - eksctl

1. È necessaria la versione `0.215.0` o una versione successiva dello strumento da riga di `eksctl` comando installata sul dispositivo o AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Crea un `IPv4` cluster Amazon EKS con la versione Kubernetes predefinita di Amazon EKS nella tua regione predefinita. AWS Prima di eseguire il comando, apporta le sostituzioni seguenti:
3. Sostituiscilo *`region-code`* con la AWS regione in cui desideri creare il cluster.
4. Sostituisci *`my-cluster`* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
5. Sostituisci *`1.33`* con qualsiasi versione supportata da Amazon EKS.

6. Modifica i valori di `vpc-private-subnets` in base alle tue esigenze. Puoi anche aggiungerne altri IDs. È necessario specificare almeno due sottoreti IDs. tuttavia, se preferisci specificare le sottoreti pubbliche, puoi modificare `--vpc-private-subnets` in `--vpc-public-subnets`. Alle sottoreti pubbliche è associata una tabella di routing con un routing a un gateway Internet, al contrario delle sottoreti private. Per questo motivo, ti consigliamo di utilizzare le sottoreti private quando possibile.

   Le sottoreti scelte devono soddisfare i requisiti delle sottoreti di Amazon EKS. Prima di selezionare le sottoreti, consigliamo di acquisire familiarità con tutti i requisiti e le considerazioni su cluster VPC e sottoreti di Amazon EKS.

7. Esegui il comando seguente:

   ```
   eksctl create cluster --name my-cluster --region region-code --version 1.33 --vpc-private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
   ```

   Il provisioning del cluster richiede diversi minuti. Durante la creazione del cluster, vengono visualizzate diverse righe di output. L'ultima riga di output è simile alla seguente riga di esempio.

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

8. Continuare con the section called "Passaggio 3: aggiornamento di kubeconfig"

Impostazioni facoltative

Per visualizzare la maggior parte delle opzioni che è possibile specificare durante la creazione di un cluster con `eksctl`, utilizza il comando `eksctl create cluster --help`. Per visualizzare tutte le opzioni disponibili, puoi utilizzare un file `config`. Per ulteriori informazioni, consulta Uso dei file config e lo Schema dei file config nella documentazione di `eksctl`. Gli esempi di file di configurazione sono disponibili su GitHub.

Di seguito sono riportate le impostazioni facoltative da aggiungere al comando precedente, quando necessario. Puoi abilitare queste opzioni esclusivamente durante la creazione del cluster, non in seguito. Per specificare queste opzioni, non usare il comando precedente. Crea invece il cluster con un file di configurazione eksctl e specifica tali impostazioni.

- Se vuoi specificare uno o più gruppi di sicurezza assegnati da Amazon EKS alle interfacce di rete create, scegli l'opzione securityGroup.

Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS.

- Se desideri specificare da quale blocco CIDR (`IPv4`Classless Inter-domain Routing) Kubernetes assegna gli indirizzi IP del servizio, specifica l'opzione CIDR del servizio. IPv4

  L'indicazione di un intervallo personalizzato consente di evitare conflitti tra servizi Kubernetes e altre reti con peering o connesse al VPC. Immetti un intervallo in una notazione CIDR. Ad esempio: `10.2.0.0/16`.

  Il blocco CIDR deve soddisfare i seguenti requisiti:

  - Essere compreso in uno degli intervalli seguenti: `10.0.0.0/8` `172.16.0.0/12` o `192.168.0.0/16`.
  - Avere una dimensione minima di `/24` e una dimensione massima di `/12`.
  - Non sovrapporsi all'intervallo del VPC per le risorse Amazon EKS.

    Puoi specificare questa opzione solo con la famiglia di indirizzi `IPv4` e solo durante la creazione del cluster. Se non specifichi un'opzione, Kubernetes assegna gli indirizzi IP del servizio dai blocchi CIDR `10.100.0.0/16` o `172.20.0.0/16`.
- Se stai creando un cluster e vuoi che il cluster assegni gli indirizzi `IPv6` ai pods e ai servizi anziché a indirizzi `IPv4`, scegli l'opzione ipFamily.

  Kubernetes assegna indirizzi `IPv4` ai pod e ai servizi, per impostazione predefinita. Prima di utilizzare la famiglia `IPv6`, assicurati di conoscere tutte le considerazioni e i requisiti indicati negli argomenti the section called "Considerazioni e requisiti relativi al VPC", the section called "Considerazioni e requisiti relativi alle sottoreti", the section called "Requisiti relativi al gruppo di sicurezza" e the section called "IPv6". Se scegli la famiglia `IPv6`, non puoi specificare un intervallo di indirizzi a cui Kubernetes può assegnare indirizzi di servizio `IPv6` come puoi fare per la famiglia `IPv4`. Kubernetes assegna gli indirizzi del servizio dall'intervallo di indirizzi locale univoco (`fc00::/7`).

Crea cluster - console AWS

1. Aprire la Console Amazon EKS.
2. Seleziona Aggiungi cluster e quindi Crea.
3. In Opzioni di configurazione, seleziona Configurazione personalizzata
   - Per informazioni sulla creazione rapida di cluster con la modalità automatica di EKS, consulta the section called "Console di gestione".
4. In modalità automatica di EKS, disattiva l'opzione Usa modalità automatica di EKS.
   - Per informazioni sulla creazione di cluster della modalità automatica di EKS con configurazione personalizzata, consulta the section called "Creare un cluster automatico".
5. Nella pagina Configura cluster compila i campi seguenti:
   - Nome: un nome per il cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole), trattini e trattini bassi. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
   - Ruolo IAM del cluster: scegli il ruolo IAM del cluster Amazon EKS che hai creato per consentire al piano di controllo Kubernetes di gestire AWS le risorse per tuo conto.
   - Versione Kubernetes – la versione di Kubernetes da utilizzare per il cluster. È preferibile selezionare la versione più recente, a meno che non occorra una versione precedente.
   - Tipo di supporto: la policy della versione di Kubernetes che vuoi impostare per il tuo cluster. Se vuoi che il cluster venga eseguito solo su una versione di supporto standard, puoi scegliere Supporto standard. Se vuoi che il tuo cluster acceda al supporto esteso alla fine del supporto standard per una versione, puoi scegliere Supporto esteso. Se selezioni una versione di Kubernetes che attualmente è in supporto esteso, non puoi selezionare il supporto standard come opzione.
   - Crittografia dei segreti (Facoltativo): scegli di abilitare la crittografia dei segreti per i segreti Kubernetes tramite una chiave KMS. Puoi abilitare questa funzionalità anche dopo la creazione del cluster. Assicurati, tuttavia, di acquisire familiarità con le informazioni contenute nella sezione the section called "Abilitare la crittografia dei segreti".
   - Tag: (facoltativo) aggiunge eventuali tag al cluster. Per ulteriori informazioni, consulta the section called "Assegnazione di tag alle risorse ".
   - Spostamento zonale ARC - (facoltativo) Puoi usare il controller Route53 Application Recovery per mitigare le zone di disponibilità compromesse. Per ulteriori informazioni, consulta the section called "Ulteriori informazioni relative allo spostamento zonale".

6. Nella sezione Accesso al cluster della pagina Configura cluster, inserisci i seguenti campi:
   - Accesso amministratore cluster Bootstrap: il creatore del cluster diventa automaticamente amministratore Kubernetes. Se vuoi disabilitare questa opzione, seleziona Impedisci l'accesso all'amministratore del cluster.
   - Modalità di autenticazione del cluster: determina come concedere agli utenti e ai ruoli IAM l'accesso a Kubernetes. APIs Per ulteriori informazioni, consulta the section called "Set Cluster Authentication Mode".

   Quando hai finito con questa pagina, seleziona Avanti.
7. Nella pagina Specifica reti, seleziona i valori dei campi riportati di seguito:
   - VPC: crea il cluster in un VPC esistente conforme ai requisiti su VPC di Amazon EKS. Prima di scegliere un VPC, ti consigliamo di acquisire familiarità con i requisiti e le considerazioni nella sezione the section called "Requisiti del VPC e delle sottoreti". Dopo la creazione del cluster, non potrai più modificare il VPC da utilizzare. Se non VPCs ne è elencata nessuna, devi prima crearne una. Per ulteriori informazioni, consulta the section called "Crea un VPC".
   - Sottoreti: per impostazione predefinita, tutte le sottoreti disponibili nel VPC specificato nel campo precedente sono preselezionate. È necessario selezionarne almeno due.

     Le sottoreti scelte devono soddisfare i requisiti delle sottoreti di Amazon EKS. Prima di selezionare le sottoreti, consigliamo di acquisire familiarità con tutti i requisiti e le considerazioni su cluster VPC e sottoreti di Amazon EKS.

     Gruppi di sicurezza: (facoltativo) specifica uno o più gruppi di sicurezza da associare alle interfacce di rete create da Amazon EKS.

     Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS.
   - Scegli la famiglia di indirizzi IP del cluster: puoi scegliere IPv4tra IPv6.

     Kubernetes assegna indirizzi `IPv4` ai pod e ai servizi, per impostazione predefinita. Prima di utilizzare la famiglia `IPv6`, assicurati di conoscere tutte le considerazioni e i requisiti indicati negli argomenti Requisiti e considerazioni relativi al VPC, the section called "Considerazioni e requisiti relativi alle sottoreti", the section called "Requisiti relativi al gruppo di sicurezza" e the

section called "IPv6". Se scegli la famiglia `IPv6`, non puoi specificare un intervallo di indirizzi a cui Kubernetes può assegnare indirizzi di servizio `IPv6` come puoi fare per la famiglia `IPv4`. Kubernetes assegna gli indirizzi del servizio dall'intervallo di indirizzi locale univoco (`fc00::/7`).

- (Facoltativo) Scegli Configure Kubernetes Service IP address range (Configura l'intervallo di indirizzi IP del servizio Kubernetes) e specifica un intervallo **`IPv4`** del servizio.

  L'indicazione di un intervallo personalizzato consente di evitare conflitti tra servizi Kubernetes e altre reti con peering o connesse al VPC. Immetti un intervallo in una notazione CIDR. Ad esempio: `10.2.0.0/16`.

  Il blocco CIDR deve soddisfare i seguenti requisiti:

  - Essere compreso in uno degli intervalli seguenti: `10.0.0.0/8` `172.16.0.0/12` o `192.168.0.0/16`.
  - Avere una dimensione minima di `/24` e una dimensione massima di `/12`.
  - Non sovrapporsi all'intervallo del VPC per le risorse Amazon EKS.

  Puoi specificare questa opzione solo con la famiglia di indirizzi `IPv4` e solo durante la creazione del cluster. Se non specifichi un'opzione, Kubernetes assegna gli indirizzi IP del servizio dai blocchi CIDR `10.100.0.0/16` o `172.20.0.0/16`.

- Per Cluster endpoint access (Accesso all'endpoint del cluster), seleziona un'opzione. Puoi modificare questa opzione dopo aver creato il cluster. Prima di selezionare un'opzione non predefinita, assicurati di familiarizzare con le opzioni e le relative implicazioni. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

  Quando hai finito con questa pagina, seleziona Avanti.

8. (Facoltativo) Nella pagina Configura osservabilità, scegli le opzioni Parametri e Registrazione del piano di controllo da attivare. Per impostazione predefinita, i tipi di log sono disattivati.
   - Per ulteriori informazioni sull'opzione relativa alle metriche di Prometheus, consulta the section called "Passaggio 1: attivazione delle metriche Prometheus".
   - Per ulteriori informazioni sulle opzioni Registrazione del piano di controllo, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".

   Quando hai finito con questa pagina, seleziona Avanti.
9. Nella pagina Select add-ons (Seleziona componenti aggiuntivi), scegli i componenti aggiuntivi da aggiungere al tuo cluster. Certi componenti aggiuntivi sono preselezionati. Puoi scegliere tutti i componenti aggiuntivi Amazon EKS e i componenti aggiuntivi AWS Marketplace di cui

hai bisogno. Se i componenti aggiuntivi del AWS Marketplace che desideri installare non sono elencati, puoi fare clic sulla numerazione delle pagine per visualizzare altri risultati di pagina o cercare i componenti aggiuntivi di AWS Marketplace disponibili inserendo il testo nella casella di ricerca. Puoi anche filtrare per categoria, fornitore o modello di prezzi e quindi scegliere i componenti aggiuntivi dai risultati della ricerca. Quando crei un cluster, puoi visualizzare, selezionare e installare tutti i componenti aggiuntivi che supportano EKS Pod Identity, come descritto in dettaglio in the section called "Pod Identity".

Quando hai finito con questa pagina, seleziona Avanti.

Alcuni componenti aggiuntivi, come Amazon VPC CNI, CoreDNS e kube-proxy, sono installati per impostazione predefinita. Disabilitare uno dei componenti aggiuntivi predefiniti potrebbe influire sulla tua capacità di eseguire le applicazioni Kubernetes.

10. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona la versione da installare. Dopo la creazione del cluster, puoi eseguire in qualsiasi momento l'aggiornamento a una versione successiva.

    Per i componenti aggiuntivi che supportano EKS Pod Identities, puoi utilizzare la console per generare automaticamente il ruolo con il nome, la politica AWS gestita e la politica di attendibilità precompilati specificamente per il componente aggiuntivo. È possibile riutilizzare i ruoli esistenti o crearne di nuovi per i componenti aggiuntivi supportati Per i passaggi necessari per usare la console per creare ruoli per i componenti aggiuntivi che supportano EKS Pod Identity, consulta the section called "Crea componente aggiuntivo (console)AWS". Se un componente aggiuntivo non supporta EKS Pod Identity, viene mostrato un messaggio con le istruzioni per usare la procedura guidata per creare i ruoli IAM per gli account di servizio (IRSA) dopo la creazione del cluster.

    Puoi aggiornare la configurazione di ogni componente aggiuntivo dopo la creazione del cluster. Per ulteriori informazioni sulla configurazione dei componenti aggiuntivi, consulta the section called "Aggiornamento di un componente aggiuntivo". Quando hai finito con questa pagina, seleziona Avanti.

11. Nella pagina Rivedi e crea, controlla le informazioni che hai inserito o selezionato nelle pagine precedenti. Se devi apportare modifiche, seleziona Edit (Modifica). Al termine della configurazione, seleziona Create (Crea). Durante il provisioning del cluster, nel campo Stato viene visualizzato il messaggio CREAZIONE.

> **Note**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

Il provisioning del cluster richiede diversi minuti.

12. Continua con the section called "Passaggio 3: aggiornamento di kubeconfig"

Crea cluster - AWS CLI

1. Crea un cluster con il comando seguente. Prima di eseguire il comando, apporta le modifiche seguenti:
   - Sostituisci *region-code* con la AWS regione in cui desideri creare il cluster.
   - Sostituisci *my-cluster* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole), trattini e trattini bassi. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
   - Sostituisci *1.33* con qualsiasi versione supportata da Amazon EKS.
   - Sostituisci *111122223333* con il tuo ID account e *myAmazonEKSClusterRole* con il nome del ruolo IAM del cluster.
   - Sostituisci i valori di `subnetIds` con quelli in tuo possesso. Puoi anche aggiungerne altri IDs. È necessario specificare almeno due sottoreti IDs.

     Le sottoreti scelte devono soddisfare i requisiti delle sottoreti di Amazon EKS. Prima di selezionare le sottoreti, consigliamo di acquisire familiarità con tutti i requisiti e le considerazioni su cluster VPC e sottoreti di Amazon EKS.
   - Se non vuoi specificare un ID del gruppo di sicurezza, rimuovi `,securityGroupIds=sg-<ExampleID1>` dal comando. Se desideri specificare uno o più gruppi di sicurezza IDs, sostituisci i valori di `securityGroupIds` con i tuoi. Puoi anche aggiungerne altri IDs.

Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da 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**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

Di seguito sono riportate le impostazioni facoltative da aggiungere al comando precedente, quando necessario. Puoi abilitare queste opzioni esclusivamente durante la creazione del cluster, non in seguito.

- Durante la creazione del cluster, EKS installa più componenti aggiuntivi di rete per impostazione predefinita. Sono inclusi Amazon VPC CNI, CoreDNS e kube-proxy.

  Se vuoi disabilitare l'installazione di questi componenti aggiuntivi di rete predefiniti, usa il parametro seguente. Questo può essere usato per alternative CNIs, come Cilium. Per ulteriori informazioni, consulta la Documentazione di riferimento delle API di EKS.

  ```
  aws eks create-cluster --bootstrapSelfManagedAddons false
  ```

- Se vuoi specificare il blocco di instradamento interdominio senza classi (CIDR) `IPv4` da cui Kubernetes assegna gli indirizzi IP del servizio, aggiungi `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>` al comando seguente.

L'indicazione di un intervallo personalizzato consente di evitare conflitti tra servizi Kubernetes e altre reti con peering o connesse al VPC. Immetti un intervallo in una notazione CIDR. Ad esempio: `10.2.0.0/16`.

Il blocco CIDR deve soddisfare i seguenti requisiti:

- Essere compreso in uno degli intervalli seguenti: `10.0.0.0/8` `172.16.0.0/12` o `192.168.0.0/16`.
- Avere una dimensione minima di `/24` e una dimensione massima di `/12`.
- Non sovrapporsi all'intervallo del VPC per le risorse Amazon EKS.

Puoi specificare questa opzione solo con la famiglia di indirizzi `IPv4` e solo durante la creazione del cluster. Se non specifichi un'opzione, Kubernetes assegna gli indirizzi IP del servizio dai blocchi CIDR `10.100.0.0/16` o `172.20.0.0/16`.

- Se stai creando un cluster e vuoi che il cluster assegni gli indirizzi `IPv6` ai pod e ai servizi anziché a indirizzi `IPv4`, aggiungi `--kubernetes-network-config ipFamily=ipv6` al seguente comando.

  Kubernetes assegna indirizzi `IPv4` ai pod e ai servizi, per impostazione predefinita. Prima di utilizzare la famiglia `IPv6`, assicurati di conoscere tutte le considerazioni e i requisiti indicati negli argomenti Requisiti e considerazioni relativi al VPC, the section called "Considerazioni e requisiti relativi alle sottoreti", the section called "Requisiti relativi al gruppo di sicurezza" e the section called "IPv6". Se scegli la famiglia `IPv6`, non puoi specificare un intervallo di indirizzi a cui Kubernetes può assegnare indirizzi di servizio `IPv6` come puoi fare per la famiglia `IPv4`. Kubernetes assegna gli indirizzi del servizio dall'intervallo di indirizzi locale univoco (`fc00::/7`).

2. Il provisioning del cluster richiede diversi minuti. È possibile eseguire query sullo stato del cluster con il comando seguente.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

   Non passare alla fase successiva finché l'output restituito non è `ACTIVE`.

3. Continua con the section called "Passaggio 3: aggiornamento di kubeconfig"

Passaggio 3: aggiornamento di kubeconfig

1. Se hai creato il cluster utilizzando `eksctl`, puoi ignorare questo passaggio, poiché è già stato completato da `eksctl`. Abilita `kubectl` per consentire la comunicazione con il cluster aggiungendo un nuovo contesto al file `config` `kubectl`. Per ulteriori informazioni su come creare e aggiornare il file, consulta the section called "Accedere al cluster con kubectl".

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

   Di seguito viene riportato un output di esempio:

   ```
   Added new context arn:aws: eks:region-code:111122223333:cluster/my-cluster to /home/
   username/.kube/config
   ```

2. Conferma la comunicazione con il cluster eseguendo il comando seguente.

   ```
   kubectl get svc
   ```

   Di seguito viene riportato un output di esempio.

   ```
   NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
   ```

Passaggio 4: configurazione del cluster

1. (Consigliato) Per utilizzare alcuni componenti aggiuntivi di Amazon EKS o per consentire a singoli carichi di lavoro Kubernetes di disporre di autorizzazioni IAM (AWS Identity and Access Management) specifiche, crea un provider IAM OpenID Connect (OIDC) per il tuo cluster. È necessario creare un provider OIDC IAM per il cluster una sola volta. Per ulteriori informazioni sui componenti aggiuntivi di Amazon EKS, consulta the section called "Componenti aggiuntivi Amazon EKS". Per ulteriori informazioni sull'assegnazione di autorizzazioni IAM specifiche ai carichi di lavoro, consulta the section called "Credenziali con IRSA".
2. (Consigliato) Configura il cluster per il plug-in Amazon VPC CNI per Kubernetes prima di distribuire i nodi Amazon nel cluster. EC2 Per impostazione predefinita, il plugin è stato installato con il cluster. Quando aggiungi EC2 nodi Amazon al tuo cluster, il plug-in viene distribuito automaticamente su ogni EC2 nodo Amazon che aggiungi. Il componente aggiuntivo richiede di

collegare una delle policy IAM seguenti a un ruolo IAM. Se il cluster usa la famiglia `IPv4`, usa la policy IAM gestita AmazonEKS_CNI_Policy. Se il tuo cluster usa la famiglia `IPv6`, utilizza una policy IAM creata da te.

Il ruolo IAM a cui si collega la policy può essere il ruolo IAM del nodo o un ruolo dedicato utilizzato solo per il plugin. Consigliamo di allegare la policy a quest'ultimo ruolo. Per ulteriori informazioni sulla creazione del ruolo, consulta the section called "Configurare per IRSA" o il ruolo IAM del nodo Amazon EKS.

3. Se hai distribuito il cluster utilizzando Console di gestione AWS, puoi saltare questo passaggio. Per impostazione predefinita, Console di gestione AWS implementa il plug-in Amazon VPC CNI per i componenti aggiuntivi Kubernetes, CoredNS e Amazon EKS. `kube-proxy`

   Se distribuisci il cluster utilizzando una delle due opzioni `eksctl` o la AWS CLI, vengono implementati il plug-in Amazon VPC CNI per Kubernetes, CoredNS e componenti aggiuntivi autogestiti. `kube-proxy` Puoi eseguire la migrazione dei componenti aggiuntivi CNI di Amazon VPC per Kubernetes, CoreDNS e dei componenti autogestiti `kube-proxy` implementati con il cluster verso i componenti aggiuntivi di Amazon EKS. Per ulteriori informazioni, consulta the section called "Componenti aggiuntivi Amazon EKS".
4. (Facoltativo) Se non l'hai già fatto, puoi abilitare le metriche Prometheus per il cluster. Per ulteriori informazioni, consulta Creare uno scraper nella Guida per l'utente di Amazon Managed Service for Prometheus.
5. Se hai intenzione di implementare nel tuo cluster carichi di lavoro che utilizzano volumi Amazon EBS, devi installare il Amazon EBS CSI nel cluster prima di implementare i carichi di lavoro.

Fasi successive

- L'utente o il principale IAM che ha creato il cluster è l'unico principale in grado di accedere al cluster. Concedi autorizzazioni ad altri principali IAM in modo che possano accedere al tuo cluster.
- Se il principale IAM che ha creato il cluster dispone solo delle autorizzazioni IAM minime a cui si fa riferimento nei prerequisiti, potresti voler aggiungere ulteriori autorizzazioni Amazon EKS per quel principale IAM. Per ulteriori informazioni su come concedere le autorizzazioni Amazon EKS ali principali IAM, consulta the section called "Riferimento IAM".
- Se desideri che il principale IAM che ha creato il cluster o qualsiasi altro principale visualizzi le risorse Kubernetes nella console Amazon EKS, concedi le autorizzazioni necessarie alle entità.
- Se desideri che i nodi e i principali IAM accedano al cluster dall'interno della VPC, attiva l'endpoint privato per il cluster. L'endpoint pubblico è abilitato per impostazione predefinita. Puoi disabilitare

l'endpoint pubblico dopo aver abilitato l'endpoint privato, se lo desideri. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

- Abilitare la crittografia dei segreti per il cluster.
- Configurare la registrazione per il cluster.
- Aggiungi nodi al cluster.

Piano di controllo predisposto per Amazon EKS

Panoramica di

Amazon EKS Provisioned Control Plane è una funzionalità che consente agli amministratori dei cluster di scegliere tra una serie di livelli di scalabilità e di designare il livello prescelto per prestazioni molto elevate e prevedibili dal piano di controllo del cluster. Ciò consente agli amministratori del cluster di garantire che il piano di controllo sia sempre dotato della capacità specificata.

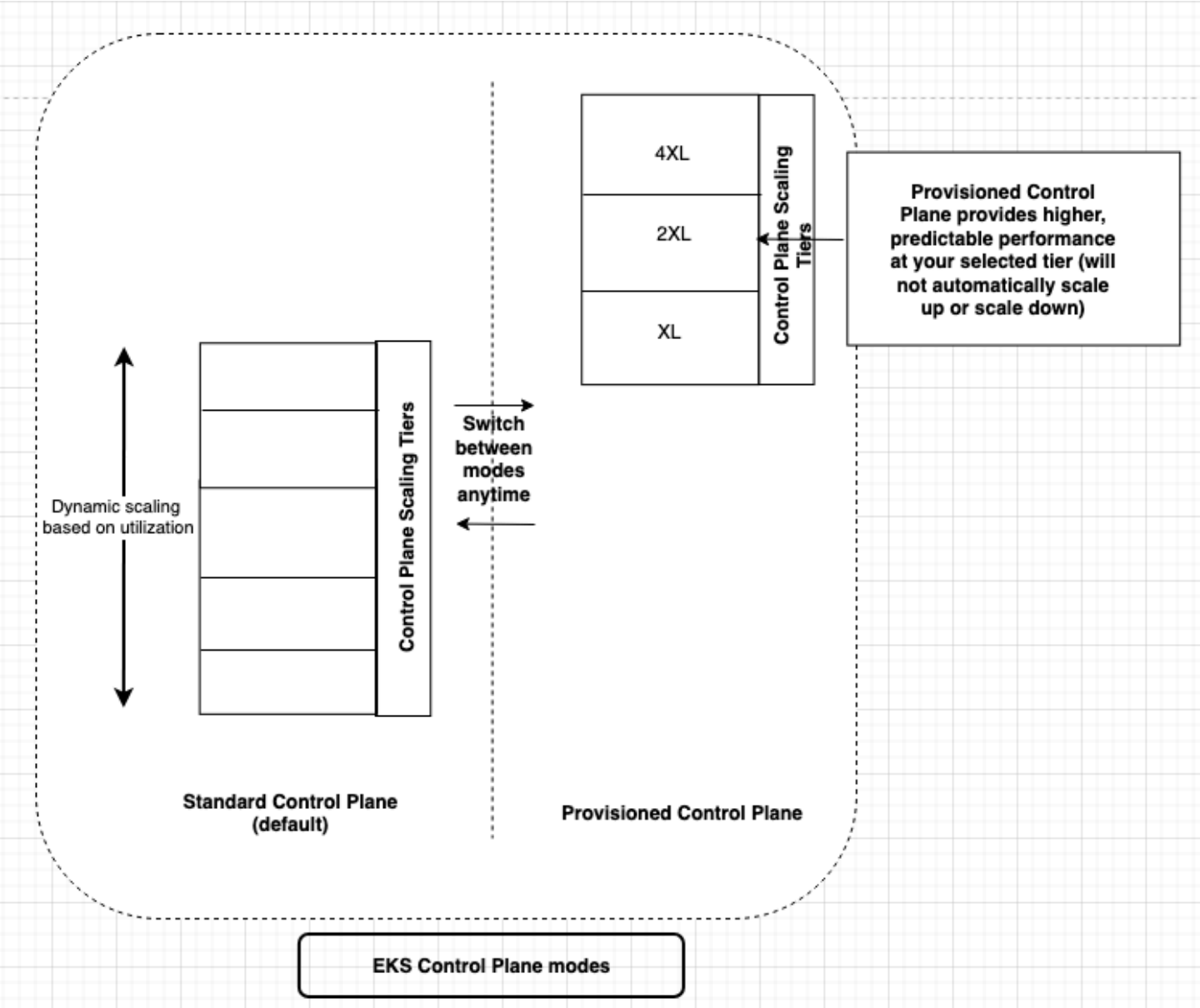
Amazon EKS offre due modalità operative per il piano di controllo del cluster. Per impostazione predefinita, i cluster Amazon EKS utilizzano la modalità Standard, in cui il piano di controllo si ridimensiona automaticamente verso l'alto e verso il basso in base alle richieste del carico di lavoro. La modalità Standard alloca dinamicamente una capacità del piano di controllo sufficiente a soddisfare le esigenze di carico di lavoro ed è la soluzione consigliata per la maggior parte dei casi d'uso. Tuttavia, per carichi di lavoro specializzati che non possono tollerare alcuna variabilità delle prestazioni dovuta alla scalabilità del piano di controllo o che richiedono quantità molto elevate di capacità del piano di controllo, è possibile utilizzare facoltativamente la modalità Provisioned. La modalità Provisioned consente di preallocare la capacità del piano di controllo che è sempre pronta a gestire requisiti di carico di lavoro impegnativi.

> **Note**
>
> La modalità Provisioned è una modalità operativa del piano di controllo aggiuntiva rispetto alla modalità Standard predefinita. L'introduzione della modalità Provisioned non modifica il comportamento della modalità Standard.

Con EKS Provisioned Control Plane, gli amministratori del cluster possono predisporre in anticipo la capacità desiderata del piano di controllo, fornendo prestazioni prevedibili e elevate dal piano di controllo del cluster, sempre disponibile. EKS Provisioned Control Plane consente inoltre agli

amministratori del cluster di fornire la stessa capacità del piano di controllo in tutti gli ambienti, dallo staging ai siti di produzione e di disaster recovery. Questo è importante per garantire che le prestazioni del piano di controllo ottenute in tutti gli ambienti siano coerenti e prevedibili. Infine, EKS Provisioned Control Plane consente di accedere a livelli molto elevati di prestazioni del piano di controllo, consentendo l'esecuzione di carichi di lavoro di intelligenza artificiale estremamente scalabili, elaborazione ad alte prestazioni e carichi di lavoro di elaborazione dati su larga scala su Kubernetes.

Per impostazione predefinita, tutti i cluster Amazon EKS nuovi e esistenti funzionano in modalità Standard. Per i cluster che richiedono prestazioni elevate e prevedibili dal piano di controllo, puoi scegliere di utilizzare la funzione EKS Provisioned Control Plane. Ti verrà fatturata la tariffa oraria per il particolare livello di scalabilità del Control Plane, oltre alle tariffe orarie EKS per il supporto standard o esteso. Per ulteriori informazioni sui prezzi, consulta i prezzi di Amazon EKS.



Casi d'uso

EKS Provisioned Control Plane è progettato per affrontare scenari specifici in cui prestazioni elevate e prevedibili del piano di controllo sono fondamentali per le operazioni. La comprensione di questi casi d'uso può aiutarvi a determinare se EKS Provisioned Control Plane è la soluzione giusta per i vostri carichi di lavoro.

Carichi di lavoro critici per le prestazioni: per i carichi di lavoro che richiedono una latenza minima e le massime prestazioni dal piano di controllo Kubernetes, EKS Provisioned Control Plane offre una capacità che elimina la variabilità delle prestazioni con la scalabilità del piano di controllo.

Carichi di lavoro estremamente scalabili: se esegui carichi di lavoro altamente scalabili come l'addestramento e l'inferenza basati sull'intelligenza artificiale, l'elaborazione ad alte prestazioni o l'elaborazione di dati su larga scala che richiedono un gran numero di nodi in esecuzione nel cluster, Provisioned Control Plane fornisce la capacità del piano di controllo necessaria per supportare questi carichi di lavoro impegnativi.

Eventi previsti ad alta richiesta: quando si prevede un improvviso aumento delle richieste di control plane a causa di un evento imminente, come vendite o promozioni di e-commerce, lanci di prodotti, periodi di shopping natalizio o importanti eventi sportivi o di intrattenimento, Provisioned Control Plane consente di scalare in anticipo la capacità del piano di controllo. Questo approccio proattivo assicura che il piano di controllo sia pronto a gestire l'aumento del carico senza attendere che la scalabilità automatica risponda alla domanda.

Coerenza dell'ambiente: Provisioned Control Plane consente di abbinare la capacità e le prestazioni del piano di controllo negli ambienti di staging e produzione, aiutando a identificare potenziali problemi prima dell'implementazione in produzione. Mantenendo lo stesso livello del piano di controllo in tutti gli ambienti, è possibile garantire che i risultati dei test riflettano accuratamente il comportamento di produzione, riducendo il rischio di sorprese legate alle prestazioni durante l'implementazione.

Disaster recovery e business continuity: per gli scenari di disaster recovery, Provisioned Control Plane consente di effettuare il provisioning di ambienti di failover con lo stesso livello di capacità dell'ambiente primario. Ciò garantisce interruzioni minime e un ripristino rapido durante gli eventi di failover, poiché il cluster di disaster recovery avrà caratteristiche prestazionali del piano di controllo identiche a quelle del cluster di produzione dal momento in cui viene attivato.

Control Plane Scaling Tiers

EKS Provisioned Control Plane offre livelli di scalabilità denominati utilizzando le taglie delle t-shirt (XL, 2XL, 4XL). Ogni livello definisce la propria capacità attraverso tre attributi chiave di Kubernetes che determinano le caratteristiche prestazionali del piano di controllo del cluster. La comprensione di questi attributi consente di selezionare il livello appropriato per i requisiti del carico di lavoro.

La concorrenza delle richieste API misura il numero di richieste che il server API del piano di controllo Kubernetes può elaborare contemporaneamente, il che è fondamentale per carichi di lavoro con throughput elevato.

La frequenza di pianificazione dei pod indica la velocità con cui lo scheduler Kubernetes predefinito può pianificare i pod sui nodi, misurata in pod al secondo.

La dimensione del database del cluster indica lo spazio di archiviazione allocato a etcd, il database che contiene lo stato/i metadati del cluster.

Quando si effettua il provisioning del piano di controllo del cluster su un determinato livello di scalabilità utilizzando Provisioned Control Plane, EKS garantisce che il piano di controllo del cluster mantenga i limiti corrispondenti a tale livello. I limiti dei livelli di scalabilità del piano di controllo variano in base alla versione di Kubernetes, come illustrato nelle tabelle seguenti.

EKS v1.28 e v1.29

Livello di scalabilità del piano di controllo fornito	Concorrenza delle richieste API (postazioni)	Velocità di pianificazione dei pod (pods/sec)	Dimensioni del database del cluster (GB)
XL	1700	100	16
2XL	3400	100	16
4XL	6800	100	16

EKS v1.30 e versioni successive

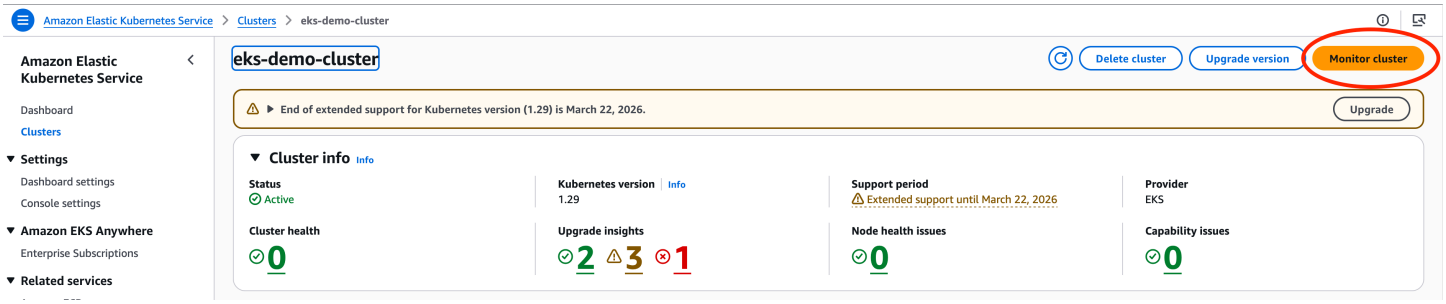
Provisioned Control Plane Scaling Tier	Concorrenza delle richieste API (postazioni)	Velocità di pianificazione dei pod (pods/sec)	Dimensioni del database del cluster (GB)
XL	1700	167	16
2XL	3400	283	16
4 XL	6800	400	16

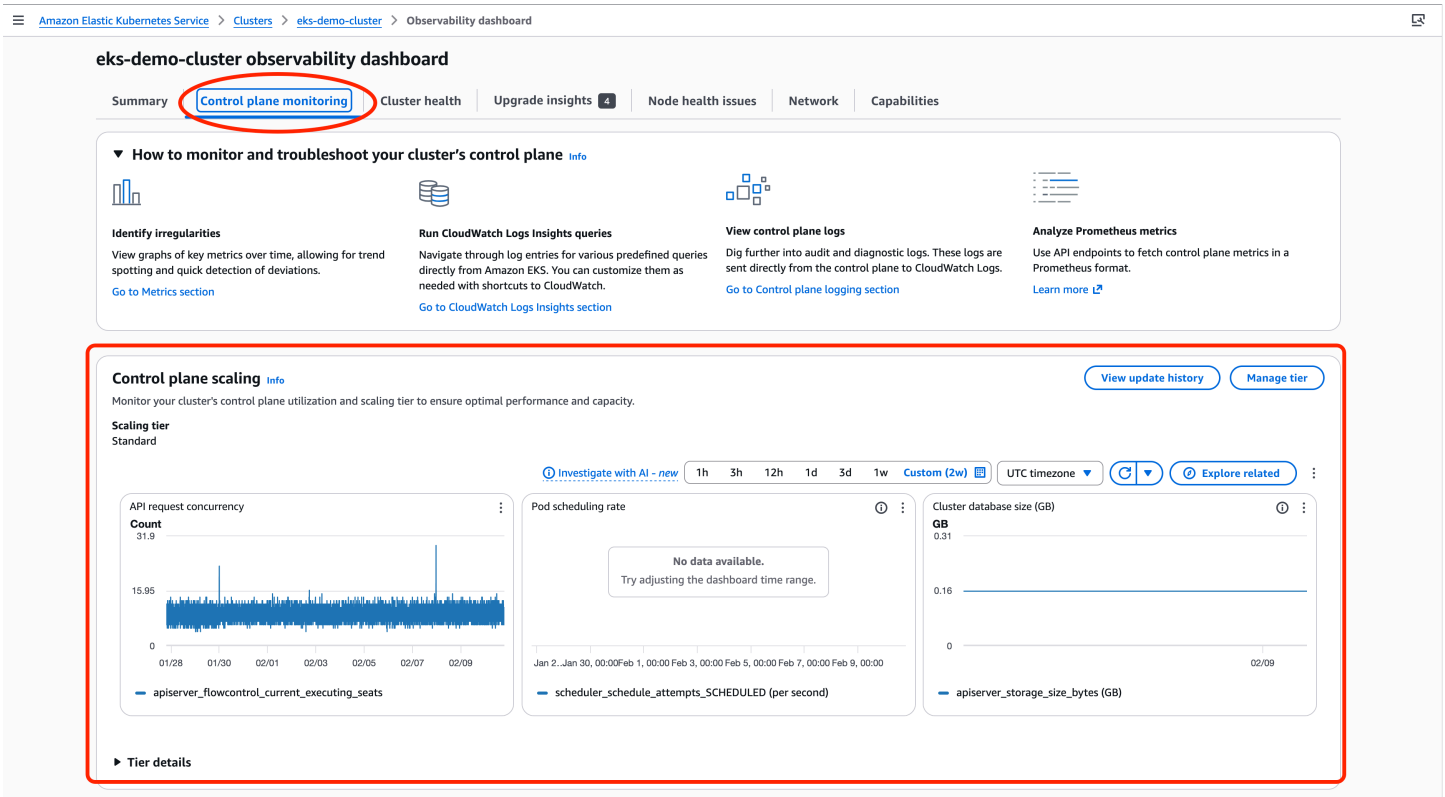
Monitoraggio del piano di controllo, scalabilità dell'utilizzo dei livelli

Amazon EKS fornisce diverse metriche per aiutarti a monitorare l'utilizzo dei livelli del piano di controllo. Queste metriche sono pubblicate come CloudWatch metriche Amazon e sono accessibili tramite la CloudWatch console EKS. Inoltre, queste metriche possono essere recuperate dall'endpoint Prometheus del cluster EKS (vedi qui).

	Metrica Prometheus	CloudWatch Parametro
Concorrenza delle richieste API	apiserver_flowcontrol_current_executing_seats	apiserver_flowcontrol_current_executing_seats
Frequenza di pianificazione del pod	scheduler_schedule_attempts_total	scheduler_schedule_attempts_total, scheduler_schedule_attempts_scheduled, scheduler_schedule_attempts_unschedulable
Dimensioni del database del cluster	apiserver_storage_size_bytes	apiserver_storage_size_bytes

Puoi visualizzare l'utilizzo del piano di controllo nella console Amazon EKS. Dalla pagina di panoramica del cluster, scegli Monitor cluster per accedere alla dashboard di osservabilità, quindi seleziona la scheda Control plane monitoring per visualizzare l'utilizzo del piano di controllo nella sezione Control plane scaling.





Comprensione della capacità di Tier rispetto alle prestazioni effettive

Quando selezioni un livello di scalabilità Provisioned Control Plane, gli attributi del livello rappresentano le configurazioni sottostanti che Amazon EKS applica al tuo piano di controllo. Tuttavia, le prestazioni effettive ottenute dipendono dai modelli di carico di lavoro specifici, dalle configurazioni e dall'aderenza alle best practice di Kubernetes. Ad esempio, mentre un livello 4XL configura API Priority and Fairness (APF) con 6.800 postazioni di richieste simultanee, la velocità effettiva delle richieste ottenuta dal piano di controllo dipende dai tipi di operazioni eseguite. Ad esempio, Kubernetes penalizza le richieste di elenco più di quelle get, e quindi il numero effettivo di richieste di elenco elaborate contemporaneamente dal piano di controllo è inferiore alle richieste get (vedi qui). Allo stesso modo, sebbene lo scheduler predefinito QPS sia impostato su 400 per un livello 4XL, la velocità di pianificazione effettiva dei pod dipende da fattori come la disponibilità dei nodi e lo stato di salute per la pianificazione. Per ottenere prestazioni ottimali, assicurati che le tue applicazioni seguano le best practice di Kubernetes (vedi qui) e siano configurate correttamente per le caratteristiche del tuo carico di lavoro.

Considerazioni

- Capacità del piano di controllo standard: la modalità del piano di controllo standard di EKS offre il miglior rapporto prezzo/prestazioni ed è l'opzione consigliata per la maggior parte dei casi

d'uso. Tuttavia, per carichi di lavoro specializzati che non possono tollerare alcuna variabilità delle prestazioni dovuta alla scalabilità del piano di controllo o che richiedono quantità molto elevate di capacità del piano di controllo, è possibile prendere in considerazione l'utilizzo della modalità Provisioned.

- È richiesto il consenso esplicito: i cluster esistenti non passeranno automaticamente dal piano di controllo Standard a un livello EKS Provisioned Control Plane più costoso. È necessario attivare esplicitamente uno dei nuovi livelli di scalabilità EKS Provisioned Control Plane.
- Restrizione all'uscita: la modalità piano di controllo standard supporta fino a 8 GB di dimensioni del database cluster (etcd). Se la dimensione del database del cluster supera gli 8 GB durante l'utilizzo della modalità Provisioned, non è possibile tornare alla modalità Standard finché non si riducono le dimensioni del database al di sotto di 8 GB. Ad esempio, se si utilizzano 14 GB di storage del database in modalità Provisioned, è necessario innanzitutto ridurre l'utilizzo del database a meno di 8 GB prima di tornare alla modalità Standard.
- Nessuna scalabilità automatica dei livelli: EKS Provisioned Control Plane non esegue automaticamente la scalabilità tra i livelli. Una volta selezionato un livello di scalabilità, il piano di controllo del cluster rimane fissato a quel livello, garantendo prestazioni coerenti e prevedibili. Tuttavia, avete la flessibilità necessaria per implementare la vostra soluzione di scalabilità automatica monitorando le metriche di utilizzo dei livelli e utilizzando l'EKS Provisioned Control Plane APIs per scalare verso l'alto o verso il basso quando queste metriche superano le soglie definite, offrendovi il pieno controllo sulla strategia di scalabilità e sull'ottimizzazione dei costi.
- Visualizzazione del livello corrente: puoi utilizzare la console Amazon EKS, l'interfaccia a riga di comando di Amazon Web Services o l'API per visualizzare l'attuale livello di scalabilità del piano di controllo. Nella CLI, puoi eseguire il `describe-cluster` comando: `aws eks describe-cluster --name cluster-name`
- Tempo di transizione dei livelli: puoi utilizzare la console Amazon EKS, Amazon EKS APIs o la CLI per uscire o passare da un livello di scalabilità all'altro. Amazon EKS ha introdotto un nuovo tipo di aggiornamento del cluster chiamato`ScalingTierConfigUpdate`, che puoi ispezionare per monitorare l'avanzamento della transizione. Dopo aver eseguito un comando di cambio di livello, puoi elencare gli aggiornamenti sul cluster per visualizzare un nuovo aggiornamento di tipo `ScalingTierConfigUpdate` con stato`Updating`. Lo stato cambia `Successful` al termine dell'aggiornamento o in `Failed` caso di errore. Il campo di errore nell'aggiornamento indica il motivo dell'errore. Non ci sono restrizioni sulla frequenza con cui è possibile passare da un livello all'altro. Il completamento della modifica del livello del piano di controllo richiede diversi minuti.
- Selezione del livello ottimale: per determinare il livello di scalabilità Provisioned Control Plane ottimale per il cluster, è possibile eseguire test di carico effettuando il provisioning del cluster sul

livello più alto (4XL). Quindi esegui un test di carico per simulare la domanda di picco sul piano di controllo del cluster. Osservate le metriche di utilizzo del livello di control plane al picco di carico e utilizzate queste osservazioni come fattore guida per selezionare il livello appropriato per la modalità Provisioned.

- Prezzi Provisioned Control Plane: ti verrà addebitata la tariffa oraria per il livello di scalabilità Provisioned Control Plane in cui si trova il cluster. Questo si aggiunge alle tariffe orarie di supporto standard o esteso. Per ulteriori informazioni, consulta la pagina dei prezzi di Amazon EKS.
- Livello di scalabilità più ampio: se intendi eseguire il cluster su un livello di scalabilità superiore a 4XL, contatta il team dell'account di Amazon Web Services per ulteriori informazioni sui prezzi.
- Supporto per versioni e regioni di Kubernetes: EKS Provisioned Control Plane è supportato in tutte le regioni commerciali di Amazon Web Services GovCloud e in Cina. Provisioned Control Plane funziona su EKS v1.28 e versioni successive.

Guida introduttiva all'Amazon EKS Provisioned Control Plane

Questa guida illustra i passaggi per configurare e utilizzare EKS Provisioned Control Plane utilizzando AWS Console di gestione AWS CLI e.

Prerequisiti

Prima di iniziare, assicurati di disporre dei seguenti elementi:

- AWS CLI: uno strumento da riga di comando per lavorare con AWS servizi, incluso Amazon EKS. Per ulteriori informazioni, consulta la Guida per l'utente all'installazione nell'interfaccia a riga di AWS comando. Dopo aver installato la AWS CLI, ti consigliamo di configurarla anche. Per ulteriori informazioni, consulta Configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. Nota che la AWS CLI v2 è necessaria per utilizzare l'opzione update-kubeconfig mostrata in questa pagina.
- Autorizzazioni IAM richieste: il responsabile della sicurezza IAM che stai utilizzando deve disporre delle autorizzazioni per lavorare con i ruoli IAM di Amazon EKS, i ruoli collegati ai servizi AWS CloudFormation, un VPC e le risorse correlate. Per ulteriori informazioni, consulta Azioni e utilizzo dei ruoli collegati ai servizi nella Guida per l'utente IAM. È necessario che tutti i passaggi di questa guida siano completati dallo stesso utente. Esegui il comando seguente per controllare l'utente corrente:

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

> **Note**
>
> Ti consigliamo di completare la procedura descritta in questo argomento in una shell Bash. In alternativa, puoi apportare alcune modifiche alla tua shell per alcuni comandi di script, come i caratteri di continuazione della riga, e per il modo in cui le variabili vengono impostate e utilizzate. Inoltre, le regole di escape e di utilizzo delle virgolette per la shell (interprete di comandi) potrebbero essere diverse. Per ulteriori informazioni, consulta Uso delle virgolette con le stringhe nella AWS CLI nella Guida per l'utente dell'interfaccia a riga di AWS comando.

Piano di controllo predisposto EKS - AWS CLI

Crea cluster con EKS Provisioned Control Plane Scaling Tier

```
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
```

Risposta:

```
{
    "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"
        }
    }
}
```

Visualizza il livello Control Plane Scaling del cluster

```
aws eks describe-cluster --name prod-cluster
```

Risposta:

```
{
    "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"
        }
    }
}
```

Aggiorna il cluster per utilizzare EKS Provisioned Control Plane

```
aws eks update-cluster-config --name prod-cluster \
--control-plane-scaling-config tier=tier-2xl
```

Risposta:

```
{
    "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": []
    }
}
```

Visualizza l'aggiornamento di Control Plane Scaling

```
aws eks list-updates --name example
```

Risposta:

```
{
    "updateIds": [
        "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd1"
    ]
}
```

Uscire da Provisioned Control Plane a Standard Control Plane

```
aws eks update-cluster-config --name prod-cluster \
--control-plane-scaling-config tier=standard
```

Risposta:

```
{
    "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": []
    }
}
```

Piano di controllo predisposto EKS - Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli Crea cluster.
3. In Opzioni di configurazione, seleziona Configurazione personalizzata.
4. Scorri verso il basso fino al livello di scalabilità del piano di controllo. Seleziona Usa un livello di scalabilità per abilitare Provisioned Control Plane.
5. Seleziona il livello di scalabilità del piano di controllo che desideri fornire per il cluster tra varie opzioni di scaling tier come XL, 2XL e 4XL.
6. Seleziona altre opzioni di configurazione del cluster in base alle esigenze. Nel passaggio finale seleziona Crea cluster. Nota che il completamento della creazione del cluster potrebbe richiedere alcuni minuti.

Prepararsi agli aggiornamenti delle versioni di Kubernetes e risolvere i problemi di configurazione errata con gli approfondimenti sui cluster

Gli approfondimenti sui cluster di Amazon EKS rilevano i problemi e forniscono consigli per risolverli e gestire il cluster. Ogni cluster Amazon EKS è soggetto a controlli automatici e ricorrenti basati su un elenco di approfondimenti curato da Amazon EKS. Questi controlli sugli approfondimenti sono completamente gestiti da Amazon EKS e offrono consigli su come affrontare eventuali esiti.

Tipi di approfondimenti del cluster

- Approfondimenti sulla configurazione: identificano le configurazioni errate nella configurazione di nodi ibridi EKS che potrebbero compromettere la funzionalità del cluster o dei carichi di lavoro.
- Approfondimenti sugli aggiornamenti: identificano i problemi che potrebbero influire sulla capacità di aggiornamento a nuove versioni di Kubernetes.

Considerazioni

- Frequenza: Amazon EKS aggiorna le informazioni sul cluster ogni 24 ore oppure è possibile aggiornarle manualmente per visualizzare lo stato più recente. Ad esempio, puoi aggiornare manualmente le informazioni sul cluster dopo aver risolto un problema per vedere se il problema è stato risolto.
- Autorizzazioni: Amazon EKS crea automaticamente una voce di accesso al cluster per le informazioni sui cluster in ogni cluster EKS. Questa voce concede l'autorizzazione a EKS per visualizzare le informazioni relative al cluster. Amazon EKS utilizza queste informazioni per generare approfondimenti. Per ulteriori informazioni, consulta the section called "Amazon EKSCluster InsightsPolicy".

Casi d'uso

Gli approfondimenti sui cluster in Amazon EKS forniscono controlli automatizzati per aiutare a mantenere l'integrità, l'affidabilità e la configurazione ottimale dei cluster Kubernetes. Di seguito sono riportati i principali casi d'uso degli approfondimenti sui cluster, tra cui la preparazione all'aggiornamento e la risoluzione dei problemi di configurazione.

Approfondimenti sugli aggiornamenti

Gli approfondimenti sugli aggiornamenti sono un tipo specifico di controlli approfonditi all'interno degli approfondimenti del cluster. Questi controlli restituiscono approfondimenti riguardanti la preparazione all'aggiornamento della versione di Kubernetes. Amazon EKS esegue controlli approfonditi sugli aggiornamenti su ogni cluster EKS.

> **Important**
>
> Amazon EKS ha temporaneamente eseguito il rollback di una funzionalità che richiedeva l'utilizzo di un flag `--force` per aggiornare il cluster in caso di determinati problemi di

approfondimenti sul cluster. Per ulteriori informazioni, consulta Temporary rollback of enforcing upgrade insights on update cluster version su GitHub.
Per ulteriori informazioni sull'aggiornamento del cluster, consulta the section called "Fase 3: aggiornamento del piano di controllo del cluster".

Prima di aggiornare la versione di Kubernetes del cluster, puoi utilizzare la scheda Approfondimenti sugli aggiornamenti della dashboard di osservabilità nella console Amazon EKS. Se il cluster ha identificato problemi, consultali e apporta le correzioni appropriate. I problemi includono collegamenti alla documentazione di Amazon EKS e Kubernetes. Dopo aver risolto il problema, aggiornare gli aggiornamenti sui cluster su richiesta per recuperare gli approfondimenti più recenti. Se tutti i problemi sono stati risolti, aggiorna il cluster.

Amazon EKS restituisce approfondimenti riguardanti la preparazione all'aggiornamento della versione di Kubernetes. Gli approfondimenti sugli aggiornamenti identificano possibili problemi che possono influire sugli aggiornamenti del cluster di Kubernetes. Questo riduce l'impegno richiesto agli amministratori nella preparazione degli aggiornamenti e aumenta l'affidabilità delle applicazioni sulle versioni più recenti di Kubernetes. I cluster vengono scansionati automaticamente da Amazon EKS in base a un elenco di possibili aggiornamenti di versione di Kubernetes che influiscono sui problemi. Amazon EKS aggiorna frequentemente l'elenco dei controlli sugli approfondimenti in base alle revisioni delle modifiche apportate in ogni versione di Kubernetes.

Gli approfondimenti sugli aggiornamenti offerti da Amazon EKS velocizzano il processo di test e verifica per le nuove versioni. Inoltre, consentono agli amministratori dei cluster e agli sviluppatori di applicazioni di avvalersi delle funzionalità di Kubernetes più recenti, evidenziando eventuali problemi e offrendo consigli per porvi rimedio.

Approfondimenti sulla configurazione

Gli approfondimenti sui cluster di EKS analizzano automaticamente i cluster di Amazon EKS con nodi ibridi per identificare problemi di configurazione che compromettono la comunicazione tra piano di controllo e webhook di Kubernetes, comandi kubectl come exec e logs, e altro ancora. Gli approfondimenti sulla configurazione evidenziano i problemi e forniscono consigli per la correzione, accelerando i tempi per una configurazione di nodi ibridi perfettamente funzionante.

Inizia a usare

Per visualizzare l'elenco dei controlli sugli approfondimenti eseguiti e gli eventuali problemi rilevanti identificati da Amazon EKS, puoi usare Console di gestione AWS, AWS CLI, AWS SDK

e l'operazione dell'API `ListInsights` di Amazon EKS. Per iniziare, consulta the section called "Visualizzazione degli approfondimenti sui cluster".

Visualizzazione degli approfondimenti sui cluster

Amazon EKS fornisce due tipi di approfondimenti: Approfondimenti sulla configurazione e Approfondimenti sugli aggiornamenti. Gli approfondimenti sulla configurazione identificano le configurazioni errate nella configurazione di nodi ibridi EKS che potrebbero compromettere la funzionalità del cluster o dei carichi di lavoro. Gli approfondimenti sugli aggiornamenti identificano i problemi che potrebbero influire sulla capacità di aggiornamento a nuove versioni di Kubernetes.

Per visualizzare l'elenco dei controlli approfonditi eseguiti e gli eventuali problemi rilevanti identificati da Amazon EKS, puoi chiamare il look in the Console di gestione AWS, la AWS CLI e il funzionamento AWS SDKs dell'`ListInsights`API Amazon EKS.

Visualizzare approfondimenti sulla configurazione (console)

1. Aprire la Console Amazon EKS.
2. Dall'elenco dei cluster, scegli il nome del cluster Amazon EKS per il quale desideri visualizzare gli approfondimenti.
3. Scegli Monitora cluster.
4. Scegli la scheda Integrità del cluster.
5. Nella tabella Approfondimenti sulla configurazione, vedrai le seguenti colonne:
   - Nome: il controllo eseguito da Amazon EKS sul cluster.
   - Stato di approfondimento: un approfondimento con stato `Error` indica una configurazione errata che potrebbe influire sulla funzionalità del cluster. Un approfondimento con uno stato `Warning` indica che la configurazione non corrisponde all'approccio documentato, ma che la funzionalità del cluster potrebbe funzionare se configurata intenzionalmente. Un approfondimento con lo stato `Passing` indica che Amazon EKS non ha riscontrato alcun problema associato a questo controllo sull'approfondimento nel cluster.
   - Versione: la versione applicabile.
   - Ora dell'ultimo aggiornamento: l'ora in cui lo stato dell'approfondimento è stato aggiornato l'ultima volta per questo cluster.
   - Descrizione: informazioni tratte dal controllo sull'approfondimento, che include l'avviso e le azioni consigliate per la correzione.

Visualizzare gli approfondimenti sugli aggiornamenti (console)

1. Aprire la Console Amazon EKS.
2. Dall'elenco dei cluster, scegli il nome del cluster Amazon EKS per il quale desideri visualizzare gli approfondimenti.
3. Scegli Monitora cluster.
4. Scegli la scheda Approfondimenti sugli aggiornamenti.
5. Per visualizzare i dati più recenti, scegli il pulsante Aggiorna approfondimenti e attendi il completamento dell'operazione di aggiornamento.
6. Nella tabella Approfondimenti sugli aggiornamenti, vedrai le seguenti colonne:
   - Nome: il controllo eseguito da Amazon EKS sul cluster.
   - Stato dell'approfondimento: un approfondimento che presenta lo stato "Errore" in genere indica che la versione di Kubernetes interessata è N+1 rispetto alla versione corrente del cluster, mentre lo stato "Avviso" indica che l'approfondimento si applica a una versione di Kubernetes futura N+2 o superiore. Un approfondimento con lo stato "Ammesso" indica che Amazon EKS non ha riscontrato alcun problema associato a questo controllo sull'approfondimento nel cluster. Lo stato dell'approfondimento "Sconosciuto" significa che Amazon EKS non è in grado di determinare se il cluster è interessato da questo controllo sull'approfondimento.
   - Versione: la versione di Kubernetes che l'approfondimento ha verificato per individuare possibili problemi.
   - Ora dell'ultimo aggiornamento: l'ora in cui lo stato dell'approfondimento è stato aggiornato l'ultima volta per questo cluster.
   - Ora dell'ultima transizione: l'ora in cui lo stato di questo approfondimento è cambiato l'ultima volta.
   - Descrizione: informazioni tratte dal controllo sull'approfondimento, che include l'avviso e le azioni consigliate per la correzione.

Visualizza informazioni dettagliate sul cluster (AWS CLI)

1. Utilizza gli approfondimenti per un cluster specificato per visualizzare i dati più recenti. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:
   - Sostituiscilo *region-code* con il codice della tua AWS regione.
   - Sostituisci *my-cluster* con il nome del cluster.

```
aws eks start-insights-refresh --region region-code --cluster-name my-cluster
```

2. Esegui il comando seguente per tenere traccia dello stato di un aggiornamento di approfondimenti. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks describe-insights-refresh --cluster-name my-cluster
```

Di seguito viene riportato un output di esempio.

```
{
    "message": "Insights refresh is in progress",
    "status": "IN_PROGRESS",
    "startedAt": "2025-07-30T13:36:09-07:00"
}
```

3. Elenca gli approfondimenti per un cluster specificato. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:
   - Sostituiscilo *region-code* con il codice AWS della tua regione.
   - Sostituisci *my-cluster* con il nome del cluster.

```
aws eks list-insights --region region-code --cluster-name my-cluster
```

Di seguito viene riportato un output di esempio.

```
{
"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. Esegui il comando seguente per visualizzare le informazioni dettagliate relative a un approfondimento. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:
   - Sostituiscilo *region-code* con il codice AWS della tua regione.
   - *a1b2c3d4-5678-90ab-cdef-EXAMPLE22222*Sostituiscilo con un ID di approfondimento recuperato dall'elenco degli approfondimenti del cluster.
   - Sostituisci *my-cluster* con il nome del cluster.

   ```
   aws eks describe-insight --region region-code --id a1b2c3d4-5678-90ab-cdef-
   EXAMPLE22222 --cluster-name my-cluster
   ```

   Di seguito viene riportato un output di esempio.

   ```
   {
     "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": [] },
    },
}
```

Aggiornamento del cluster esistente alla nuova versione di Kubernetes

Tip

Registrati ai prossimi workshop Amazon EKS.

Quando è disponibile una nuova versione di Kubernetes in Amazon EKS, è possibile aggiornare il cluster Amazon EKS alla versione più recente.

Important

Una volta aggiornato un cluster, non è possibile effettuare il downgrade a una versione precedente. Prima di eseguire l'aggiornamento a una nuova versione di Kubernetes,

consigliamo di esaminare le informazioni contenute in Comprensione del ciclo di vita della versione di Kubernetes su EKS e nei passaggi di aggiornamento in questo argomento.

Le nuove versioni di Kubernetes hanno introdotto modifiche significative. Pertanto ti consigliamo di verificare il comportamento delle applicazioni rispetto alla nuova versione di Kubernetes prima di eseguire l'aggiornamento sui cluster di produzione. È possibile eseguire questa operazione mediante la creazione di un flusso di lavoro di integrazione continua per testare il comportamento totale dell'applicazione prima di passare a una nuova versione di Kubernetes.

Il processo di aggiornamento consiste nell'avvio, da parte di Amazon EKS, di nuovi nodi del server API con la versione aggiornata di Kubernetes per sostituire quelli esistenti. Amazon EKS esegue i controlli dell'integrità dell'infrastruttura standard e dello stato di preparazione per il traffico di rete su questi nuovi nodi per verificare che funzionino come previsto. Tuttavia, una volta avviato l'aggiornamento del cluster, non è possibile metterlo in pausa o interromperlo. Se uno di questi controlli non va a buon fine, Amazon EKS ripristina l'implementazione dell'infrastruttura e il cluster rimane nella versione precedente di Kubernetes. Le applicazioni in esecuzione non sono interessate e il cluster non è mai lasciato in uno stato non deterministico o irrecuperabile. Amazon EKS esegue regolarmente il backup di tutti i cluster gestiti e, se necessario, dispone di meccanismi per il recupero dei cluster. Stiamo valutando e migliorando costantemente i nostri processi di gestione dell'infrastruttura di Kubernetes.

Per aggiornare il cluster, Amazon EKS richiede fino a cinque indirizzi IP disponibili dalle sottoreti specificate al momento della creazione del cluster. Amazon EKS crea nuove interfacce di rete elastiche (interfacce di rete) per il cluster in una delle sottoreti specificate, che può essere diversa rispetto a quella in cui si trovano le interfacce di rete esistenti. Assicurati quindi che le regole del gruppo di sicurezza consentano la comunicazione necessaria con il cluster per una delle sottoreti specificate al momento della creazione del cluster. Se nessuna delle sottoreti che hai specificato durante la creazione del cluster non esiste, se non hai abbastanza indirizzi IP disponibili o se non disponi di regole del gruppo di sicurezza per la comunicazione con il cluster, l'aggiornamento potrebbe non riuscire.

Per garantire che l'endpoint del server API per il tuo cluster sia sempre accessibile, Amazon EKS fornisce un'elevata disponibilità del piano di controllo Kubernetes ed esegue aggiornamenti continui delle istanze del server API durante le operazioni di aggiornamento. Per tenere conto della modifica degli indirizzi IP delle istanze del server API che supportano l'endpoint del server API Kubernetes, devi assicurarti che i client del tuo server API gestiscano le riconnessioni in modo efficace. Le

versioni recenti di `kubectl` e le librerie di client Kubernetes sono ufficialmente supportati, esegui questo processo di riconnessione in modo trasparente.

> **Note**
>
> Per ulteriori informazioni sui componenti di un aggiornamento del cluster, consulta Best practice per gli aggiornamenti dei cluster nella Guida alle best practice EKS. Questa risorsa consente di pianificare un aggiornamento e comprendere la strategia di aggiornamento di un cluster.

Considerazioni per la modalità automatica di Amazon EKS

- La funzionalità di elaborazione della modalità automatica di Amazon EKS controlla la versione di Kubernetes dei nodi. Dopo aver aggiornato il piano di controllo, modalità automatica EKS inizierà ad aggiornare in modo incrementale i nodi gestiti. Modalità automatica EKS rispetta i budget relativi alle interruzioni dei pod.
- Non è necessario aggiornare manualmente le funzionalità di modalità automatica di Amazon EKS, incluse le funzionalità di scalabilità automatica di calcolo, archiviazione a blocchi e bilanciamento del carico.

Riepilogo

Il riepilogo approfondito del processo di aggiornamento del cluster Amazon EKS è il seguente:

1. Assicurati che il cluster sia in uno stato che supporti un aggiornamento. Ciò include il controllo di Kubernetes APIs utilizzato dalle risorse distribuite nel cluster, per garantire che il cluster sia privo di problemi di salute. È necessario utilizzare gli approfondimenti sull'aggiornamento di Amazon EKS per valutare la preparazione all'aggiornamento del cluster.
2. Aggiorna il piano di controllo alla versione secondaria successiva (ad esempio, da 1.32 a 1.33).
3. Aggiorna i nodi nel piano dati in modo che corrispondano a quelli del piano di controllo.
4. Aggiorna tutte le applicazioni aggiuntive eseguite sul cluster (ad esempio, `cluster-autoscaler`).
5. Esegui l'aggiornamento dei componenti aggiuntivi forniti da Amazon EKS, come quelli inclusi per impostazione predefinita:
   - Versione consigliata di Amazon VPC CNI

- Versione consigliata di CoredNS
- Versione suggerita `kube-proxy`

6. Aggiorna tutti i client che comunicano con il cluster (ad esempio, `kubectl`).

Passaggio 1: preparazione per l'aggiornamento

Confrontare la versione Kubernetes del piano di controllo cluster con la versione Kubernetes dei nodi.

- Ottieni la versione di Kubernetes del piano di controllo del cluster.

```
kubectl version
```

- Ottieni la versione di Kubernetes dei tuoi nodi. Questo comando restituisce tutti i nodi Amazon EC2, Fargate e ibridi autogestiti e gestiti. Ciascun pod Fargate è presente nell'elenco assieme al rispettivo nodo.

```
kubectl get nodes
```

Prima di aggiornare il piano di controllo a una nuova versione di Kubernetes, assicurarsi che la versione secondaria di Kubernetes dei nodi gestiti e dei nodi Fargate nel cluster sia la stessa versione del piano di controllo. Ad esempio, se il piano di controllo è in versione in esecuzione `1.29` e uno dei nodi è in esecuzione`1.28`, è necessario aggiornare i nodi alla versione `1.29` prima di aggiornare il piano di controllo alla versione 1.30. Prima di aggiornare il piano di controllo, si consiglia inoltre di aggiornare i nodi autogestiti e i nodi ibridi alla stessa versione del piano di controllo. Per ulteriori informazioni, consultare the section called "Aggiornamento", the section called "Metodi di aggiornamento" e the section called "Aggiornamento dei nodi ibridi". Se hai nodi Fargate con una versione secondaria inferiore rispetto alla versione del piano di controllo, devi prima eliminare il pod rappresentato dal nodo. a aggiornare il piano di controllo (control-plane). Tutti i pod rimanenti saranno aggiornati alla nuova versione dopo averli nuovamente implementati.

Passaggio 2: esaminare le considerazioni sull'aggiornamento

Gli approfondimento sull'aggiornamento di Amazon EKS esegue automaticamente la scansione dei cluster e li mette a confronto con un elenco di potenziali problemi di aggiornamento della versione di Kubernetes, come l'utilizzo di API Kubernetes obsoleti. Amazon EKS aggiorna periodicamente l'elenco dei controlli approfonditi da eseguire in base alle valutazioni delle modifiche nel progetto

Kubernetes. Amazon EKS aggiorna anche l'elenco dei controlli approfonditi man mano che sono introdotte modifiche nel servizio Amazon EKS insieme a nuove versioni. Per ulteriori informazioni, consulta the section called "Approfondimenti sui cluster".

Esamina la Guida alla migrazione delle API obsolete nei documenti di Kubernetes.

Esaminare gli approfondimenti sull'aggiornamento

Utilizza approfondimenti sull'aggiornamento di Amazon EKS per identificare i problemi. Per ulteriori informazioni, consulta the section called "Visualizzare gli approfondimenti sugli aggiornamenti (console)".

Considerazioni dettagliate

- Poiché Amazon EKS viene eseguito in un piano di controllo ad alta disponibilità, è possibile aggiornare solo una versione secondaria alla volta. Per ulteriori informazioni su questo requisito, consulta Kubernetes Version and Version Skew Support Policy (Policy per la versione Kubernetes e il supporto Skew della versione). Supponiamo che la versione corrente del cluster sia `1.28` e che desideri aggiornarlo alla versione `1.30`. Devi prima aggiornare la versione `1.28` del cluster alla versione `1.29`, quindi aggiornare la versione `1.29` del cluster alla versione `1.30`.
- Esamina la discrepanza di versione tra Kubernetes `kube-apiserver` e `kubelet` sui nodi.
  - A partire dalla versione `1.28` di Kubernetes, `kubelet` può essere fino a tre versioni minori precedenti rispetto a `kube-apiserver`. Consulta la sezione Kubernetes upstream version skew policy.
  - Se `kubelet` sui nodi gestiti e Fargate è nella versione di Kubernetes `1.25` o più recente, è possibile aggiornare il cluster fino a tre versioni successive senza aggiornare la versione di `kubelet`. Ad esempio, se `kubelet` è nella versione `1.25`, è possibile aggiornare la versione del cluster Amazon EKS dalla `1.25` alla `1.26`, `1.27` e `1.28` mentre `kubelet` rimane sulla versione `1.25`.
- Come best practice prima di iniziare un aggiornamento, assicurati che `kubelet` sui nodi sia alla stessa versione di Kubernetes del tuo piano di controllo.
- Se il cluster è configurato con una versione del plugin CNI di Amazon VPC per Kubernetes precedente a `1.8.0`, consigliamo di aggiornare il plugin all'ultima versione prima di aggiornare il cluster. Per aggiornare il plug-in, consulta the section called "CNI di Amazon VPC".
- Puoi eseguire un backup del tuo cluster Amazon EKS, per consentirti di ripristinare lo stato del cluster Amazon EKS e lo storage persistente in caso di errori durante il processo di aggiornamento. Per informazioni, consultare the section called " AWS Backup".

Fase 3: aggiornamento del piano di controllo del cluster

> **Important**
>
> Amazon EKS ha temporaneamente eseguito il rollback di una funzionalità che richiedeva l'utilizzo di un flag `--force` per aggiornare il cluster in caso di determinati problemi di approfondimenti sul cluster. Per ulteriori informazioni, consulta la sezione Ripristino temporaneo dell'applicazione degli approfondimenti sull'aggiornamento della versione del cluster su. GitHub
> Amazon EKS aggiorna un approfondimento sul cluster 24 ore dopo "l'ultimo aggiornamento". È possibile confrontare l'ora in cui hai risolto un problema con "l'ora dell'ultimo aggiornamento" dell'approfondimento sul cluster.
> Inoltre, l'aggiornamento dello stato degli approfondimenti può richiedere fino a 30 giorni dopo la risoluzione dell'utilizzo di API obsolete. Gli approfondimenti sull'aggiornamento cercano sempre l'utilizzo di API obsolete in una finestra di 30 giorni consecutivi.

È possibile inviare la richiesta di aggiornamento della versione del tuo piano di controllo EKS utilizzando:

- eksctl
- la console AWS
- la AWS CLI

Aggiornare il cluster: eksctl

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione e sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

Aggiorna la versione di Kubernetes del tuo piano di controllo di Amazon EKS. Sostituisci `<cluster-name>` con il nome del cluster. Sostituisci `<version-number>` con il numero di versione supportato

da Amazon EKS a cui aggiornare il tuo cluster. Per l'elenco completo delle versioni supportate, consulta Versioni supportate da Amazon EKS.

```
eksctl upgrade cluster --name <cluster-name> --version <version-number> --approve
```

Il processo di aggiornamento può richiedere alcuni minuti per il completamento.

Continua su the section called "Fase 4: aggiornamento dei componenti del cluster".

Aggiorna cluster - console AWS

1. Aprire la Console Amazon EKS.
2. Scegli Aggiorna ora per un cluster che desideri aggiornare.
3. Seleziona la versione a cui aggiornare il cluster e scegli Aggiorna.
4. Il processo di aggiornamento può richiedere alcuni minuti per il completamento. Continua su the section called "Fase 4: aggiornamento dei componenti del cluster".

Aggiorna cluster - AWS CLI

1. Verifica che la AWS CLI sia installata e che tu abbia effettuato l'accesso. Per ulteriori informazioni, consulta Installazione o aggiornamento alla versione più recente della AWS CLI.
2. Aggiorna il tuo cluster Amazon EKS con il seguente AWS comando CLI. Sostituisci `<cluster-name>` e `<region-code>` del cluster che desideri aggiornare. Sostituisci `<version-number>` con il numero di versione supportato da Amazon EKS a cui aggiornare il tuo cluster. Per l'elenco completo delle versioni supportate, consulta Versioni supportate da Amazon EKS.

   ```
   aws eks update-cluster-version --name <cluster-name> \
     --kubernetes-version <verion-number> --region <region-code>
   ```

   Di seguito viene riportato un output di esempio.

   ```
   {
       "update": {
           "id": "<update-id>",
           "status": "InProgress",
           "type": "VersionUpdate",
           "params": [
               {
   ```

```
                "type": "Version",
                "value": "<version-number>"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
```

3. Il processo di aggiornamento può richiedere alcuni minuti per il completamento. É possibile monitorare lo stato di aggiornamento del cluster attraverso il seguente comando. Oltre a utilizzare gli stessi `<cluster-name>` e `<region-code>`, utilizza `<update-id>` restituito dal comando precedente.

   ```
   aws eks describe-update --name <cluster-name> \
      --region <region-code> --update-id <update-id>
   ```

   Quando viene visualizzato lo stato `Successful`, l'aggiornamento è completo.

4. Continua su the section called "Fase 4: aggiornamento dei componenti del cluster".

Fase 4: aggiornamento dei componenti del cluster

1. Dopo che l'aggiornamento del cluster è completo, aggiornare i nodi di lavoro alla stessa versione secondaria di Kubernetes del cluster aggiornato. Per ulteriori informazioni, consultare the section called "Metodi di aggiornamento", the section called "Aggiornamento" e the section called "Aggiornamento dei nodi ibridi". Tutti i nuovi pod avviati in Fargate hanno una versione di `kubelet` che corrisponde alla versione del cluster. I pod Fargate esistenti non sono modificati.
2. (Facoltativo) Se è stato implementato Kubernetes Cluster Autoscaler nel cluster prima di aggiornarlo, aggiornare Cluster Autoscaler alla versione più recente che corrisponde alla versione principale e secondaria di Kubernetes per cui è stato eseguito l'aggiornamento.
   a. Apri la pagina delle versioni di Cluster Autoscaler in un browser Web e trova la versione più recente di Cluster Autoscaler corrispondente alla versione principale e secondaria di Kubernetes del cluster. Ad esempio, se la versione Kubernetes del cluster è `1.30`, cerca la versione più recente di Cluster Autoscaler che inizia con `1.30`. Registra il numero di versione semantica (`1.30.n`, ad esempio) per tale versione da utilizzare nella fase successiva.

b. Impostare il tag image di Cluster Autoscaler sulla versione registrata nella fase precedente mediante il comando seguente. Se necessario, sostituire `X.XX.X` con il valore in proprio possesso.

```
kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-
autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:vX.XX.X
```

3. (Solo cluster con nodi GPU) Se il tuo cluster ha gruppi di nodi con supporto GPU (ad esempio,`p3.2xlarge`), devi aggiornare il plug-in del dispositivo NVIDIA per Kubernetes sul tuo cluster. DaemonSet Sostituiscilo `<vX.X.X>` con la versione di s-device-pluginNVIDIA/K8 desiderata prima di eseguire il comando seguente.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/<vX.X.X>/
deployments/static/nvidia-device-plugin.yml
```

4. Aggiorna il plugin CNI di Amazon VPC per Kubernetes, CoredNS e componenti aggiuntivi `kube-proxy`. È preferibile aggiornare i componenti aggiuntivi alle versioni minime elencate nei Token dell'account di servizio.
   - Se stai utilizzando i componenti aggiuntivi di Amazon EKS, nella console Amazon EKS seleziona Clusters (Cluster), quindi seleziona il nome del cluster aggiornato nel riquadro di navigazione a sinistra. Le notifiche vengono visualizzate nella console Ti viene segnalato che è disponibile una nuova versione per ogni componente aggiuntivo per cui è disponibile un aggiornamento. Per aggiornare un componente aggiuntivo, seleziona la scheda Add-ons (Componenti aggiuntivi). In una delle caselle relative al componente aggiuntivo che dispone di un aggiornamento disponibile, selezionare Aggiorna ora, selezionare una versione disponibile e quindi selezionare Aggiorna.
   - In alternativa, puoi utilizzare la AWS CLI `eksctl` o aggiornare i componenti aggiuntivi. Per ulteriori informazioni, consulta the section called "Aggiornamento di un componente aggiuntivo".
5. Se necessario, aggiorna la tua versione di `kubectl`. Utilizza la versione secondaria `kubectl` immediatamente precedente a quella del piano di controllo del cluster Amazon EKS.

Effettuare il downgrade della versione di Kubernetes di un cluster Amazon EKS

Non è possibile effettuare il downgrade di Kubernetes di un cluster Amazon EKS. Invece, crea un nuovo cluster su una versione precedente di Amazon EKS e migra i carichi di lavoro.

Eliminazione di un cluster

Terminato l'utilizzo del cluster Amazon EKS, è necessario eliminare le risorse a esso associate per non dover sostenere costi superflui.

È possibile eliminare un cluster con `eksctl` Console di gestione AWS, o la AWS CLI.

Considerazioni

- Se viene visualizzato un errore in seguito alla rimozione del creatore del cluster, consultare questo articolo per la risoluzione.
- Le risorse del Servizio gestito da Amazon per Prometheus non rientrano nel ciclo di vita del cluster e devono essere gestite indipendentemente dal cluster. Quando si elimina il cluster, assicurarsi di eliminare anche tutti gli scraper applicabili per bloccare i relativi costi. Per ulteriori informazioni, consultare Find and delete scrapers nella Guida per l'utente del Servizio gestito da Amazon per Prometheus.
- Per rimuovere un cluster connesso, consultare the section called "Annullamento della registrazione di un cluster"
- Prima di poter eliminare un cluster, assicurati che la protezione da eliminazione sia disabilitata per il cluster.

Considerazioni per la modalità automatica EKS

- Eventuali nodi della modalità automatica EKS saranno eliminati, comprese le istanze gestite EC2.
- Tutti i bilanciatori del carico saranno eliminati.

Per ulteriori informazioni, consulta the section called "Disabilita modalità automatica di EKS".

Prerequisiti

Di seguito sono riportati i passaggi da eseguire prima di poter eliminare un cluster. Questi passaggi si applicano indipendentemente dal metodo utilizzato per eliminare il cluster.

1. Elenca tutti i servizi in esecuzione nel cluster.

```
kubectl get svc --all-namespaces
```

2. Elimina i servizi che hanno un valore `EXTERNAL-IP` associato. Questi servizi sono anticipati da un Elastic Load Balancing, (load balancer) ed è necessario eliminarli in Kubernetes per consentire al sistema e alle risorse associate di essere rilasciate correttamente. Sostituisci *`service-name`* con il nome di ogni servizio elencato come descritto.

```
kubectl delete svc service-name
```

3. Elimina anche tutte le risorse in ingresso. Se non elimini le risorse in ingresso, l'application load balancer rimane attivo anche se hai eliminato il cluster. *`ingress-name`*Sostituiscilo con il nome delle tue risorse in ingresso.

```
kubectl get ingress --all-namespaces
```

```
kubectl delete ing ingress-name
```

Eliminare un cluster (eksctl)

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

1. Segui i passaggi preliminari. Dopo averlo fatto, elimina il cluster e i nodi associati con il seguente comando, sostituendolo *`prod`* con il nome del cluster.

```
eksctl delete cluster --name prod
```

Output:

```
[#]  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
```

Elimina cluster (AWS console)

1. Segui i passaggi preliminari. Dopo averlo fatto, elimina tutti i gruppi di nodi e i profili Fargate.
   a. Aprire la Console Amazon EKS.
   b. Nel pannello di navigazione a sinistra, scegli Clusters (Cluster) Amazon EKS, quindi nell'elenco a schede dei cluster scegli il nome del cluster da eliminare.
   c. Seleziona la scheda Compute (Calcolo), quindi scegli un gruppo di nodi da eliminare. Scegli Delete (Elimina), immetti il nome del gruppo di nodi, quindi seleziona Delete (Elimina). Eliminare tutti i gruppi di nodi del cluster.

      > **Note**
      >
      > L'elenco presenta solo gruppi di nodi gestiti.

   d. Scegli un profilo Fargate da eliminare, seleziona Delete (Elimina), immetti il nome del profilo e infine scegli Delete (Elimina). Eliminare tutti i profili di Fargate nel cluster.
2. Elimina tutti gli stack di nodi AWS CloudFormation autogestiti.
   a. Apri la AWS CloudFormation console.
   b. Scegli lo stack del nodo da eliminare, quindi scegli Elimina.
   c. Nella finestra di dialogo di conferma Delete stack (Elimina stack) scegliere Delete stack.(Elimina stack). Eliminare tutte le pile di nodi autogestiti nel cluster.
3. Eliminare il cluster.
   a. Aprire la Console Amazon EKS.
   b. Seleziona il cluster da eliminare e scegli Delete (Elimina).
   c. Nella schermata di conferma dell'eliminazione del cluster, scegliere Elimina.
4. (Facoltativo) Eliminare lo stack VPC. AWS CloudFormation
   a. Apri la AWS CloudFormation console.
   b. Selezionare lo stack del VPC da eliminare, quindi scegliere Delete (Elimina).
   c. Nella finestra di dialogo di conferma Delete stack (Elimina stack) scegliere Delete stack.(Elimina stack).

Elimina cluster (AWS CLI)

1. Segui i passaggi preliminari. Dopo averlo fatto, elimina tutti i gruppi di nodi e i profili Fargate.
   a. Elencare i gruppi di nodi nel cluster con il comando seguente.

   ```
   aws eks list-nodegroups --cluster-name my-cluster
   ```

   > **Note**
   >
   > L'elenco presenta solo gruppi di nodi gestiti.

   b. Eliminare ogni gruppo di nodi con il comando seguente. Eliminare tutti i gruppi di nodi del cluster.

   ```
   aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
   ```

   c. Elenca i proqfili Fargate nel cluster con il comando seguente.

   ```
   aws eks list-fargate-profiles --cluster-name my-cluster
   ```

   d. Eliminare ogni profilo di Fargate con il comando seguente. Eliminare tutti i profili di Fargate nel cluster.

   ```
   aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --
   cluster-name my-cluster
   ```

2. Elimina tutti gli stack di nodi AWS CloudFormation autogestiti.
   a. Elenca gli AWS CloudFormation stack disponibili con il seguente comando. Trovare il nome del modello del nodo nell'output risultante.

   ```
   aws cloudformation list-stacks --query "StackSummaries[].StackName"
   ```

   b. Elimina lo stack di ogni nodo con il seguente comando, sostituendo *node-stack* con il nome del tuo stack. Eliminare tutte le pile di nodi autogestiti nel cluster.

   ```
   aws cloudformation delete-stack --stack-name node-stack
   ```

3. Elimina il cluster con il seguente comando, sostituendo *my-cluster* con il nome del tuo cluster.

```
aws eks delete-cluster --name my-cluster
```

4. (Facoltativo) Eliminare lo stack VPC. AWS CloudFormation

   a. Elenca gli AWS CloudFormation stack disponibili con il seguente comando. Trovare il nome del modello di VPC nell'output risultante.

   ```
   aws cloudformation list-stacks --query "StackSummaries[].StackName"
   ```

   b. Elimina lo stack VPC con il seguente comando, sostituendo *my-vpc-stack* con il nome dello stack VPC.

   ```
   aws cloudformation delete-stack --stack-name my-vpc-stack
   ```

Proteggi i cluster EKS dall'eliminazione accidentale

L'eliminazione accidentale di un cluster EKS può compromettere le operazioni del cluster Kubernetes.

Ora è possibile proteggere i cluster EKS dall'eliminazione accidentale. Se la protezione dall'eliminazione è abilitata su un cluster, prima di poterlo eliminare è necessario disabilitare tale protezione.

Lo scopo della protezione dall'eliminazione è prevenire gli incidenti. È necessario limitare attentamente gli utenti autorizzati a eliminare i cluster.

Se si cerca di eliminare un cluster attivo con la protezione dall'eliminazione attivata, si riceverà un'eccezione `InvalidRequestException`.

> **Important**
>
> Se abiliti la protezione da eliminazione su un cluster, devi disporre sia delle UpdateClusterConfig autorizzazioni che di DeleteCluster IAM per rimuovere prima la protezione da eliminazione e infine eliminare il cluster.

> **Note**
>
> Se lo stato del cluster è Creazione, Non riuscito o Eliminazione, è possibile eliminarlo anche se è attivata la protezione dall'eliminazione.

Per abilitare la protezione dall'eliminazione per un cluster esistente

È possibile eseguire questo comando solo su un cluster con stato attivo.

```
aws eks update-cluster-config --name <cluster-name> --region <aws-region> --deletion-protection
```

Per disabilitare la protezione dall'eliminazione per un cluster esistente

```
aws eks update-cluster-config --name <cluster-name> --region <aws-region> --no-deletion-protection
```

Endpoint del server API del cluster

Questo argomento consente di abilitare l'accesso privato per l'endpoint del server API Kubernetes del cluster Amazon EKS e limitare, o disabilitare completamente, l'accesso pubblico in modo che non sia accessibile da Internet.

Quando si crea un nuovo cluster, Amazon EKS crea un endpoint per il server API Kubernetes gestito utilizzato per comunicare con il cluster (usando strumenti di gestione Kubernetes, ad esempio `kubectl`). Per impostazione predefinita, questo endpoint del server API è pubblico su Internet e l'accesso al server API è protetto utilizzando una combinazione di AWS Identity and Access Management (IAM) e RBAC (Role Based Access Control) nativo di Kubernetes. Questo endpoint è noto come l'endpoint pubblico del cluster. Esiste anche un endpoint privato del cluster. Per maggiori informazioni sull'endpoint privato del cluster, consulta la sezione the section called "Endpoint privato del cluster" seguente.

formato dell'endpoint del cluster **`IPv6`**

EKS crea un endpoint dual-stack unico nel seguente formato per i nuovi cluster `IPv6` creati dopo ottobre 2024. Un IPv6 cluster è un cluster selezionato `IPv6` nell'impostazione della famiglia IP () del cluster. `ipFamily`

Example

AWS

public/private Endpoint del cluster EKS: `eks-cluster.region.api.aws`

AWS GovCloud (US)

public/private Endpoint del cluster EKS: `eks-cluster.region.api.aws`

Amazon Web Services in China

public/private Endpoint del cluster EKS: `eks-cluster.region.api.amazonwebservices.com.cn`

> **Note**
>
> L'endpoint del cluster dual-stack è stato introdotto nell'ottobre 2024. Per ulteriori informazioni sui cluster `IPv6`, consulta the section called "IPv6". I cluster creati prima di ottobre 2024 usano invece il seguente formato di endpoint.

formato dell'endpoint del cluster **`IPv4`**

EKS crea un endpoint unico nel seguente formato per ogni cluster che seleziona `IPv4` nell'impostazione della famiglia IP (ipFamily) del cluster:

Example

AWS

Endpoint del cluster public/private EKS `eks-cluster.region.eks.amazonaws.com`

AWS GovCloud (US)

Endpoint del cluster public/private EKS `eks-cluster.region.eks.amazonaws.com`

Amazon Web Services in China

Endpoint del cluster public/private EKS `eks-cluster.region.amazonwebservices.com.cn`

> **Note**
>
> Prima di ottobre 2024, anche i cluster `IPv6` usavano questo formato di endpoint. Per questi cluster, sia l'endpoint pubblico che l'endpoint privato dispongono solo di indirizzi `IPv4` risolti da questo endpoint.

Endpoint privato del cluster

Puoi abilitare l'accesso privato al server API Kubernetes in modo che tutte le comunicazioni tra i nodi di lavoro e il server API rimangano all'interno del VPC. È possibile limitare gli indirizzi IP che possono accedere al server API da Internet o disabilitare completamente l'accesso a Internet al server API.

> **Note**
>
> Poiché questo endpoint è destinato al server API Kubernetes e non a un AWS PrivateLink endpoint tradizionale per la comunicazione con un' AWS API, non viene visualizzato come endpoint nella console Amazon VPC.

Quando si abilita l'accesso privato all'endpoint per il cluster, Amazon EKS crea una zona ospitata privata Route 53 per conto dell'utente e la associa al VPC del cluster. Questa zona ospitata privata è gestita da Amazon EKS e non viene visualizzata nelle risorse Route 53 dell'account. Affinché la zona ospitata privata instradi correttamente il traffico verso il tuo server API, il VPC deve avere `enableDnsHostnames` e `enableDnsSupport` impostati su `true` e le opzioni DHCP impostate per il VPC devono includere `AmazonProvidedDNS` nell'elenco dei server dei nomi di dominio. Per ulteriori informazioni, consultare Visualizzazione e aggiornamento del supporto DNS per il VPC nella Guida per l'utente di Amazon VPC.

È possibile definire i requisiti di accesso all'endpoint del server API quando si crea un nuovo cluster e aggiornare l'accesso endpoint del server API per un cluster in qualsiasi momento.

Modifica dell'accesso all'endpoint del cluster

Utilizza le procedure in questa sezione per modificare l'accesso all'endpoint per un cluster esistente. La tabella seguente mostra le combinazioni di accesso all'endpoint del server API supportate e il comportamento associato.

Accesso pubblico all'endpoint	Accesso privato all'endpoint	Comportamento
Abilitato	Disabilitato	• Questo è il comportamento di default per nuovi cluster Amazon EKS. • Le richieste API Kubernetes provenienti dal VPC del cluster (ad esempio, comunicazione tra nodo e piano di controllo) lasciano il VPC ma non la rete Amazon. • Il server API del cluster è accessibile da Internet. Facoltativamente, puoi utilizzare l'elenco degli accessi pubblici CIDRs per controllare l'accesso all'endpoint pubblico tramite un elenco di blocchi CIDR. Se limiti l'accesso a blocchi CIDR specifici, ti consigliamo di abilitare anche l'endpoint privato o di assicurarti che i blocchi CIDR specificati includano gli indirizzi da cui i nodi e i Fargate Pod (se li usi) accedono all'endpoint pubblico.
Abilitato	Abilitato	• Le richieste API Kubernetes all'interno del VPC del cluster (ad esempio, comunicazione tra nodo di lavoro e piano di controllo) utilizzano l'endpoint VPC privato. Utilizzate le regole del gruppo di sicurezza del cluster per controllare l'accesso all'endpoint privato. • Il server API del cluster è accessibile da Internet. È possibile, facoltativamente, utilizzare l'elenco degli accessi pubblici CIDRs per controllare l'accesso all'endpoint pubblico tramite un elenco di blocchi CIDR. • Se utilizzi nodi ibridi con il tuo cluster Amazon EKS, non è consigliabile abilitare l'accesso pubblico e privato agli endpoint. Dato che i nodi ibridi sono in esecuzion e all'esterno del VPC, risolveranno l'endpoint del cluster negli indirizzi IP pubblici. Si consiglia di utilizzar e l'accesso pubblico o privato agli endpoint per i cluster con nodi ibridi.

Accesso pubblico all'endpoint	Accesso privato all'endpoint	Comportamento

Disabilitato	Abilitato	• Tutto il traffico verso il server API del cluster deve provenire dal VPC del cluster o da una rete connessa.
		• Non esiste alcun accesso pubblico al server API da Internet. Tutti i comandi `kubectl` devono provenire dall'interno del VPC o da una rete connessa. Per le opzioni di connettività, consultare the section called "Accesso a un server API solo privato".
		• Si utilizzano le regole del gruppo di sicurezza del cluster per controllare l'accesso all'endpoint privato. Tieni presente che l'accesso pubblico CIDRs non influisce sull'endpoint privato.
		• L'endpoint del server API del cluster viene risolto dai server DNS pubblici a un indirizzo IP privato dal VPC. In passato, l'endpoint poteva essere risolto solo dall'interno del VPC.
		
		Se l'endpoint non viene risolto in un indirizzo IP privato all'interno del VPC per un cluster esistente, è possibile:
		  • Abilitare l'accesso pubblico e quindi disabilitarlo di nuovo. È necessario farlo solo una volta per un cluster e l'endpoint si risolverà in un indirizzo IP privato da quel punto in avanti.
		  • Aggiornare il cluster.

Controlli di accesso agli endpoint

Tieni presente che ciascuno dei seguenti metodi per controllare l'accesso agli endpoint influisce solo sul rispettivo endpoint.

Gruppo di sicurezza del cluster

Il gruppo di sicurezza del cluster controlla due tipi di connessioni: connessioni all'API Kubelet e all'endpoint privato. Le connessioni all'`kubelet`API vengono utilizzate nei comandi`kubectl attach`,, `kubectl cp` `kubectl exec`, `kubectl logs`e. `kubectl port-forward` Il gruppo di sicurezza del cluster non influisce sull'endpoint pubblico.

Accesso pubblico CIDRs

L'accesso pubblico CIDRs controlla l'accesso all'endpoint pubblico tramite un elenco di blocchi CIDR. Tieni presente che l'accesso pubblico CIDRs non influisce sull'endpoint privato. L'accesso pubblico CIDRs si comporta in modo diverso nei `IPv6` cluster e nei `IPv4` cluster a seconda della data di creazione, descritta di seguito:

Blocchi CIDR nell'endpoint pubblico (cluster **`IPv6`**)

Puoi aggiungere blocchi `IPv6` e `IPv4` CIDR all'endpoint pubblico di un cluster `IPv6`, in quanto l'endpoint pubblico è dual-stack. Questo vale solo per i nuovi cluster con `ipFamily` impostato su `IPv6` creati a ottobre 2024 o successivamente. Puoi identificare questi cluster in base al nuovo nome di dominio dell'endpoint `api.aws`.

Blocchi CIDR nell'endpoint pubblico (cluster **`IPv4`**)

Puoi aggiungere blocchi CIDR `IPv4` all'endpoint pubblico di un cluster `IPv4`. Non puoi aggiungere blocchi CIDR `IPv6` all'endpoint pubblico di un cluster `IPv4`. Se provi a farlo, EKS restituirà il seguente messaggio di errore: `The following CIDRs are invalid in publicAccessCidrs`

Blocchi CIDR nell'endpoint pubblico (cluster **`IPv6`** creato prima di ottobre 2024)

Puoi aggiungere blocchi CIDR `IPv4` all'endpoint pubblico dei cluster `IPv6` precedenti creati prima di ottobre 2024. Puoi identificare questi cluster in base all'endpoint `eks.amazonaws.com`. Non puoi aggiungere blocchi CIDR `IPv6` all'endpoint pubblico dei cluster `IPv6` precedenti creati prima di ottobre 2024. Se provi a farlo, EKS restituirà il seguente messaggio di errore: `The following CIDRs are invalid in publicAccessCidrs`

Accesso a un server API solo privato

Se è stato disabilitato l'accesso pubblico all'endpoint del server API Kubernetes del cluster, è possibile accedere al server di API solo dal VPC o da una rete connessa. Di seguito sono elencati alcuni possibili modi per accedere all'endpoint del server API Kubernetes:

Rete connessa

È possibile connettere la rete al VPC con un gateway di transito AWS o un'altra opzione di connettività e quindi utilizzare un computer nella rete connessa. Il gruppo di sicurezza del piano di controllo di Amazon EKS deve contenere le regole per consentire il traffico in ingresso sulla porta 443 dalla rete connessa.

Host Amazon EC2 Bastion

Puoi avviare un' EC2 istanza Amazon in una sottorete pubblica nel VPC del cluster e quindi accedere tramite SSH a quell'istanza per eseguire i comandi. `kubectl` Per ulteriori informazioni, consultare Bastion host Linux in AWS. Il gruppo di sicurezza del piano di controllo di Amazon EKS deve contenere le regole per consentire il traffico in ingresso sulla porta 443 dal bastion host. Per ulteriori informazioni, consulta the section called “Requisiti relativi al gruppo di sicurezza”.

Quando esegui la configurazione `kubectl` per il tuo host bastion, assicurati di utilizzare AWS le credenziali già mappate alla configurazione RBAC del cluster oppure aggiungi il principale IAM che verrà utilizzato dal tuo bastion alla configurazione RBAC prima di rimuovere l'accesso pubblico agli endpoint. Per ulteriori informazioni, consultare the section called “Accesso API Kubernetes” e the section called “Accesso negato o non autorizzato (`kubectl`)”.

AWS Cloud9 IDE

AWS Cloud9 è un ambiente di sviluppo integrato (IDE) basato su cloud che consente di scrivere, eseguire ed eseguire il debug del codice con un semplice browser. Puoi creare un IDE AWS Cloud9 nel VPC del cluster e utilizzare l'IDE per comunicare con il cluster. Per ulteriori informazioni, consulta Creazione di un ambiente in AWS Cloud9. È necessario assicurarsi che il gruppo di sicurezza del piano di controllo Amazon EKS contenga regole per consentire il traffico in ingresso sulla porta 443 dal gruppo di sicurezza IDE. Per ulteriori informazioni, consulta the section called “Requisiti relativi al gruppo di sicurezza”.

Quando esegui la configurazione `kubectl` per il tuo AWS IDE Cloud9, assicurati di AWS utilizzare credenziali già mappate alla configurazione RBAC del cluster o aggiungi il principale IAM che l'IDE utilizzerà alla configurazione RBAC prima di rimuovere l'accesso pubblico agli endpoint. Per ulteriori informazioni, consultare the section called “Accesso API Kubernetes” e the section called “Accesso negato o non autorizzato (`kubectl`)”.

Modifica questa pagina su GitHub

Configurare l'accesso di rete all'endpoint del server API del cluster

È possibile modificare l'accesso agli endpoint del server API del cluster utilizzando Console di gestione AWS o la AWS CLI nelle sezioni seguenti.

Configura l'accesso agli endpoint - console AWS

1. Aprire la Console Amazon EKS.
2. Scegliere il nome del cluster per visualizzare le informazioni sul cluster.
3. Scegli la scheda Networking e scegli Gestisci accesso all'endpoint.
4. Per Accesso privato, scegli se attivare o disattivare l'acceso privato per l'endpoint del server API Kubernetes del cluster. Se abiliti l'accesso privato, le richieste API Kubernetes che provengono dal VPC del cluster utilizzano l'endpoint VPC privato. Per disabilitare l'accesso pubblico è necessario abilitare l'accesso privato.
5. Per Accesso pubblico, scegli se attivare o disattivare l'acceso pubblico per l'endpoint del server API Kubernetes del cluster. Se disabiliti l'accesso pubblico, il server API Kubernetes del cluster può ricevere solo richieste dal VPC del cluster.
6. (Facoltativo) Se abilitati l'Accesso pubblico, puoi specificare quali indirizzi Internet possono comunicare con l'endpoint pubblico. Seleziona Impostazioni avanzate. Immettere un blocco CIDR, ad esempio *203.0.113.5/32*. L'intervallo non può includere indirizzi riservati. Puoi immettere intervalli aggiuntivi selezionando Aggiungi origine. Puoi specificare un numero massimo di intervalli CIDR. Per ulteriori informazioni, consulta the section called "Quote di servizio". Se non si specificano gli intervalli, l'endpoint del server API pubblico riceve richieste da tutti gli indirizzi IP sia per `IPv4` (`0.0.0.0/0`) che in aggiunta `IPv6` (`::/0`) per il cluster `IPv6` a doppio stack. Se limitati l'accesso all'endpoint pubblico utilizzando gli intervalli CIDR, ti consigliamo di abilitare anche l'accesso agli endpoint privati in modo che i nodi e Fargate Pods (se utilizzati) possano comunicare con il cluster. Senza l'endpoint privato abilitato, le origini CIDR dell'endpoint di accesso pubblico devono includere le origini di uscita dal VPC. Ad esempio, se si dispone di un nodo in una sottorete privata che comunica su Internet tramite un gateway NAT, sarà necessario aggiungere l'indirizzo IP in uscita del gateway NAT come parte di un blocco CIDR nella whitelist nell'endpoint pubblico.
7. Scegliere Update (Aggiorna) per terminare.

Configurazione dell'accesso agli endpoint - AWS CLI

Completa i seguenti passaggi utilizzando la versione AWS CLI `1.27.160` o successiva. È possibile verificare la versione corrente con `aws --version`. Per installare o aggiornare la AWS CLI, vedi Installazione della AWS CLI.

1. Aggiorna l'accesso agli endpoint del server API del cluster con il seguente comando AWS CLI. Sostituire il nome del cluster e i valori di accesso dell'endpoint desiderati. Se si imposta `endpointPublicAccess=true`, è possibile (facoltativamente) immettere un singolo blocco CIDR o un elenco separato da virgole di blocchi CIDR per `publicAccessCidrs`. I blocchi non possono includere indirizzi riservati. Se si specificano blocchi CIDR, l'endpoint del server API pubblico riceverà solo le richieste dai blocchi elencati. Puoi specificare un numero massimo di intervalli CIDR. Per ulteriori informazioni, consulta the section called "Quote di servizio". Se limiti l'accesso all'endpoint pubblico utilizzando i blocchi CIDR, è consigliabile abilitare anche l'accesso agli endpoint privati in modo che i nodi e i Pods Fargate (se utilizzati) possano comunicare con il cluster. Senza l'endpoint privato abilitato, le origini CIDR dell'endpoint di accesso pubblico devono includere le origini di uscita dal VPC. Ad esempio, se si dispone di un nodo in una sottorete privata che comunica su Internet tramite un gateway NAT, sarà necessario aggiungere l'indirizzo IP in uscita del gateway NAT come parte di un blocco CIDR nella whitelist nell'endpoint pubblico. Se non specifichi intervalli CIDR, l'endpoint del server API pubblico riceve richieste da tutti (0.0.0.0/0) gli indirizzi IP e in aggiunta `IPv6` (`::/0`) per il cluster `IPv6` a doppio stack.

> **Note**
>
> Il comando seguente consente l'accesso privato e l'accesso pubblico da un singolo indirizzo IP per l'endpoint del server API. Sostituire *`203.0.113.5/32`* con un singolo blocco CIDR o un elenco separato da virgole di blocchi CIDR a cui si desidera limitare l'accesso alla rete.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --resources-vpc-config
 endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

Di seguito viene riportato un output di esempio:

```
{
    "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. Monitorare lo stato di aggiornamento dell'accesso all'endpoint con il comando seguente, utilizzando il nome del cluster, e aggiornare l'ID restituito dal comando precedente. L'aggiornamento è completo quando lo stato è illustrato come `Successful`.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000
```

Di seguito viene riportato un output di esempio.

```
{
    "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": []
    }
}
```

Modifica questa pagina su GitHub

Implementazione dei nodi Windows su cluster EKS

Informazioni su come abilitare e gestire il supporto Windows per i cluster Amazon EKS per l'esecuzione di container Windows insieme a container Linux.

Considerazioni

Prima di implementare i nodi di Windows, tenere conto delle seguenti considerazioni.

- La modalità automatica EKS non supporta i nodi Windows
- Puoi utilizzare la rete host sui nodi Windows utilizzando i Pod `HostProcess` . Per ulteriori informazioni, consulta Creare un Windows HostProcessPod nella documentazione di Kubernetes.
- I cluster Amazon EKS devono contenere uno o più nodi Linux o Fargate per eseguire i pod del sistema core che funzionano solo su Linux, come CoreDNS.
- I log degli eventi `kubelet` e `kube-proxy` sono reindirizzati al log degli eventi `EKS Windows` e sono impostati su un limite di 200 MB.
- Non è possibile utilizzare Assegna gruppi di sicurezza a singoli pod con i pod in esecuzione su nodi Windows.

- Non è possibile utilizzare reti personalizzate con nodi Windows.
- Non è possibile utilizzare `IPv6` con nodi Windows.
- I nodi di Windows supportano un'interfaccia di rete elastica per nodo. Per impostazione predefinita, il numero di pod che è possibile eseguire per ogni nodo Windows è uguale al numero di indirizzi IP (meno uno) disponibili per interfaccia di rete elastica relativa al tipo di istanza del nodo. Per ulteriori informazioni, consulta Indirizzi IP per interfaccia di rete per tipo di istanza nella Amazon EC2 User Guide.
- In un cluster Amazon EKS, un singolo servizio con un bilanciatore del carico può supportare fino a 1024 pod back-end. Ogni pod ha il proprio indirizzo IP univoco. Il limite precedente di 64 pod non è più valido, dopo un aggiornamento di Windows Server a partire da Build del sistema operativo 17763.2746.
- I container Windows non sono supportati per i pod Amazon EKS su Fargate.
- Non è possibile utilizzare Amazon EKS Hybrid Nodes con Windows come sistema operativo per l'host.
- Non è possibile recuperare i log dal pod `vpc-resource-controller`. In precedenza era possibile quando si implementava il controller sul piano dati.
- Esiste un periodo di raffreddamento prima che un indirizzo `IPv4` venga assegnato a un nuovo pod. In questo modo, si impedisce che il traffico vada verso un pod precedente con lo stesso indirizzo `IPv4` a causa della mancata validità delle regole `kube-proxy`.
- L'origine del controller è gestita su GitHub. Per contribuire o segnalare problemi al controller, visita il progetto su GitHub.
- Quando specifichi un ID AMI personalizzato per i gruppi di nodi gestiti da Windows, aggiungilo `eks:kube-proxy-windows` alla mappa di configurazione di AWS IAM Authenticator. Per ulteriori informazioni, consulta the section called "Limiti e condizioni quando si specifica un ID AMI".
- Se la conservazione IPv4 degli indirizzi disponibili è fondamentale per la sottorete, consultate la EKS Best Practices Guide - Windows Networking IP Address Management per una guida.
- Considerazioni relative a voci di accesso EKS
  - Le voci di accesso da utilizzare con i nodi Windows richiedono il tipo di `EC2_WINDOWS`. Per ulteriori informazioni, consulta the section called "Creare voci di accesso".

    Per creare una voce di accesso per un nodo Windows:

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
 iam::111122223333:role/<role-name> --type EC2_Windows
```

Prerequisiti

- Un cluster esistente.
- Il tuo cluster deve avere almeno un nodo Linux (ne consigliamo almeno due) o un pod Fargate per eseguire CoreDNS. Se si abilita il supporto Windows legacy, è necessario utilizzare un nodo Linux (non è possibile utilizzare un pod Fargate) per eseguire CoreDNS.
- Un ruolo IAM del cluster Amazon EKS esistente.

Abilitazione del supporto Windows

1. Se non disponi di nodi Amazon Linux nel cluster e utilizzi gruppi di sicurezza per i pod, vai al passaggio successivo. Altrimenti, conferma che la policy gestita di `AmazonEKSVPCResourceController` è collegata al tuo ruolo del cluster. Sostituisci *`eksClusterRole`* con il nome del ruolo del cluster.

   ```
   aws iam list-attached-role-policies --role-name eksClusterRole
   ```

   Di seguito viene riportato un output di esempio:

   ```
   {
       "AttachedPolicies": [
           {
               "PolicyName": "AmazonEKSClusterPolicy",
               "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSClusterPolicy"
           },
           {
               "PolicyName": "AmazonEKSVPCResourceController",
               "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSVPCResourceController"
           }
       ]
   }
   ```

   Se la policy è collegata, come nell'output precedente, salta il passaggio successivo.

2. Collega la policy gestita di Amazon EKSVPCResource Controller al tuo ruolo IAM del cluster Amazon EKS. Sostituisci *`eksClusterRole`* con il nome del ruolo del cluster.

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

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

3. Aggiorna il VPC CNI ConfigMap per abilitare Windows IPAM. Nota, se il VPC CNI è installato sul tuo cluster utilizzando un grafico Helm o come componente aggiuntivo Amazon EKS, potresti non essere in grado di modificare direttamente il. ConfigMap Per informazioni sulla configurazione dei componenti aggiuntivi Amazon EKS, consulta the section called "Campi che puoi personalizzare".

   a. Crea un file denominato *vpc-resource-controller-configmap.yaml* con i seguenti contenuti.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: amazon-vpc-cni
     namespace: kube-system
   data:
     enable-windows-ipam: "true"
   ```

   b. Applica la `ConfigMap` al cluster.

   ```
   kubectl apply -f vpc-resource-controller-configmap.yaml
   ```

4. Se il cluster ha la modalità di autenticazione impostata per abilitare configmap `aws-auth`:

   - Verifica che `aws-auth` `ConfigMap` contenga una mappatura per il ruolo di istanza del nodo Windows per includere il gruppo di autorizzazioni `eks:kube-proxy-windows` RBAC. Puoi eseguire la verifica eseguendo il comando seguente.

     ```
     kubectl get configmap aws-auth -n kube-system -o yaml
     ```

     Di seguito viene riportato un output di esempio.

     ```
     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}}
[...]
```

Dovresti vedere `eks:kube-proxy-windows` elencato nei gruppi. Se il gruppo non è specificato, devi aggiornare `ConfigMap` o crearla per includere il gruppo richiesto. Per ulteriori informazioni su `aws-auth` `ConfigMap`, consulta the section called "Applica la `aws-auth` `ConfigMap` al cluster".

5. Se il cluster ha la modalità di autenticazione impostata per disabilitare configmap `aws-auth`, è possibile utilizzare le voci di accesso EKS. La creazione di un nuovo ruolo del nodo da utilizzare con le istanze Windows ed EKS creerà automaticamente una voce di accesso di tipo `EC2_WINDOWS`.

Implementazione di pods Windows

Quando si implementano i pod nel cluster, è necessario specificare il sistema operativo utilizzato se si esegue una combinazione di tipi di nodi.

Per i pod Linux, utilizzare il seguente testo del selettore dei nodi nei manifesti.

```
nodeSelector:
        kubernetes.io/os: linux
        kubernetes.io/arch: amd64
```

Per i pod Windows, utilizzare il seguente testo del selettore dei nodi nei manifesti.

```
nodeSelector:
        kubernetes.io/os: windows
        kubernetes.io/arch: amd64
```

È possibile implementare un'applicazione di esempio per vedere i selettori dei nodi in uso.

Supporto di maggiore densità di pod sui nodi di Windows

Su Amazon EKS a ciascun pod è assegnato un indirizzo `IPv4` dal tuo VPC. Per questo motivo, il numero di pod che è possibile distribuire su un nodo è vincolato agli indirizzi IP disponibili, anche

se ci sono risorse sufficienti per eseguire più pod sul nodo. Poiché un nodo di Windows supporta una sola interfaccia di rete elastica, per impostazione predefinita, il numero massimo di indirizzi IP disponibili su un nodo di Windows è pari a:

```
Number of private IPv4 addresses for each interface on the node - 1
```

Un indirizzo IP è utilizzato come indirizzo IP principale dell'interfaccia di rete, quindi non può essere assegnato ai pod.

È possibile abilitare una maggiore densità di pod sui nodi Windows abilitando la delega del prefisso IP. Questa funzionalità consente di assegnare un prefisso/28 `IPv4` all'interfaccia di rete principale, anziché assegnare indirizzi `IPv4` secondari. L'assegnazione di un prefisso IP aumenta il numero massimo di indirizzi `IPv4` disponibili sul nodo:

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) *
 16
```

Con questo numero di indirizzi IP disponibili di gran lunga maggiore, gli indirizzi IP disponibili non dovrebbero limitare la capacità di scalare il numero di pod sui tuoi nodi. Per ulteriori informazioni, consulta the section called "Aumentare gli indirizzi IP".

Disabilitare il supporto Windows

1. Se il cluster contiene nodi Amazon Linux e si utilizzano gruppi di sicurezza per i pod, saltare questo passaggio.

   Rimuovi la policy IAM gestita `AmazonVPCResourceController` dal ruolo del cluster. Sostituire *`eksClusterRole`* con il nome del ruolo del cluster.

   ```
   aws iam detach-role-policy \
       --role-name eksClusterRole \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
   ```

2. Disabilita Windows IPAM nella ConfigMap per `amazon-vpc-cni`.

   ```
   kubectl patch configmap/amazon-vpc-cni \
                       -n kube-system \
                       --type merge \
   ```

```
-p '{"data":{"enable-windows-ipam":"false"}}'
```

Implementazione di cluster privati con accesso limitato a Internet

Questo argomento descrive come distribuire un cluster Amazon EKS distribuito sul AWS cloud, ma che non dispone di accesso a Internet in uscita. Se hai un cluster locale su AWS Outposts, consultathe section called "Nodi", invece di questo argomento.

Se non si ha dimestichezza con la rete Amazon EKS, consulta Demistificare le reti cluster per i nodi nodo worker Amazon EKS. Se il cluster non dispone di accesso a Internet in uscita, deve soddisfare i seguenti requisiti:

Requisiti dell'architettura del cluster

- Il tuo cluster deve estrarre immagini da un registro di container che si trova nel tuo VPC. Puoi creare un Amazon Elastic Container Registry nel VPC e copiare al suo interno le immagini dei container per i nodi da cui estrarre. Per ulteriori informazioni, consulta the section called "Copia di un'immagine in un repository".
- Il tuo cluster deve avere l'accesso privato all'endpoint abilitato. Ciò è obbligatorio per la registrazione dei nodi con l'endpoint del cluster. L'accesso pubblico all'endpoint è facoltativo. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

Requisiti dei nodi

- Prima di essere lanciati, i nodi Linux e Windows autogestiti devono includere i seguenti argomenti di bootstrap. Questi argomenti ignorano l'introspezione Amazon EKS e non richiedono l'accesso all'API Amazon EKS dal VPC.
  a. Determina l'endpoint del cluster con il comando seguente. Sostituisci *my-cluster* con il nome del cluster.

  ```
  aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
  ```

  Di seguito viene riportato un output di esempio.

  ```
  https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
  ```

b. Determina il valore dell'autorità di certificazione del cluster con il seguente comando. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --
output text
```

L'output è una stringa molto lunga.

c. Sostituisci i valori di `apiServerEndpoint` e `certificateAuthority` nell' NodeConfig oggetto con i valori restituiti nell'output dei comandi precedenti. Per ulteriori informazioni sulla specificazione degli argomenti di bootstrap all'avvio di nodi Amazon Linux 2023 autogestiti, consulta e. the section called "Amazon Linux" the section called "Windows"

- Per i nodi 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=
    ...
```

Per ulteriori argomenti, consulta lo script bootstrap su. GitHub

- Per i nodi Windows:

> **Note**
>
> Se utilizzi il servizio personalizzato CIDR, devi specificarlo utilizzando il parametro `-ServiceCIDR`. In caso contrario, la risoluzione DNS per i pod nel cluster avrà esito negativo.

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

Per ulteriori argomenti, consulta.the section called "Parametri di configurazione dello script di bootstrap".

- `aws-auth ConfigMap` del tuo cluster deve essere creata all'interno del tuo VPC. Per ulteriori informazioni sulla creazione e l'aggiunta di voci a `aws-auth ConfigMap`, digita `eksctl create iamidentitymapping --help` nel terminale. Se `ConfigMap` non esiste sul server, `eksctl` la creerà quando usi il comando per aggiungere una mappatura dell'identità.

Requisiti del pod

- Pod Identity: i pod configurati con EKS Pod Identity acquisiscono le credenziali dall'API EKS Auth. Se non è presente un accesso a Internet in uscita, è necessario creare e utilizzare un endpoint VPC per l'API EKS Auth: `com.amazonaws.region-code.eks-auth`. Per ulteriori informazioni sugli endpoint VCP EKS ed EKS Auth, consulta the section called " AWS PrivateLink".
- IRSA - I pod configurati con ruoli IAM per gli account di servizio acquisiscono le credenziali da una chiamata API AWS Security Token Service (AWS STS). Se non è disponibile un accesso a Internet in uscita, è necessario creare e utilizzare un endpoint VPC AWS STS nel VPC. La maggior parte AWS `v1` SDKs utilizza l'endpoint AWS STS globale di default (`sts.amazonaws.com`), che non utilizza l'endpoint VPC AWS STS. Per utilizzare l'endpoint VPC AWS STS, potrebbe essere necessario configurare l'SDK per utilizzare l'endpoint AWS STS regionale (). `sts.region-code.amazonaws.com` Per ulteriori informazioni, consulta the section called "Endpoint STS".
- Le sottoreti VPC del cluster devono disporre di un endpoint di interfaccia VPC per tutti i AWS servizi a cui i Pod devono accedere. Per ulteriori informazioni, consulta Accedere a un AWS servizio utilizzando un endpoint VPC di interfaccia. Nella tabella seguente sono elencati alcuni servizi ed endpoint di uso comune. Per un elenco completo degli endpoint, consulta Servizi AWS che si integrano con AWS PrivateLink nella Guida di AWS PrivateLink .

Ti consigliamo di abilitare i nomi DNS privati per gli endpoint VPC, in modo che i carichi di lavoro possano continuare a utilizzare gli endpoint del AWS servizio pubblico senza problemi.

Servizio	Endpoint
Amazon EC2	com.amazonaws. *region-code* .ec2
Amazon Elastic Container Registry (per estrarre immagini di container)	com.amazonaws.it. *region-code* .ecr.api, com.amazonaws. *region-code* .ecr.dkr e com.amazonaws. *region-code* .s3
Application Load Balancer e Network Load Balancer di Amazon	com.amazonaws. *region-code* . bilanciam ento elastico del carico
(Opzionale) AWS X-Ray (necessario per il tracciamento inviato a X-Ray) AWS	com.amazonaws. *region-code* .raggi x
(Facoltativo) Amazon SSM (necessario per l'agente SSM per le attività di gestione dei nodi. Alternativa a SSH)	com.amazonaws.it. *region-code* .ssm
Amazon CloudWatch Logs (richiesto per i log di nodi e pod inviati ad Amazon CloudWatch Logs)	com.amazonaws. *region-code* .registri
AWS Security Token Service (richiesto quando si utilizzano i ruoli IAM per gli account di servizio)	com.amazonaws. *region-code* .sts
Amazon EKS Auth (necessario quando si utilizzano le associazioni Pod Identity)	com.amazonaws.it. *region-code* .eks-auth
Amazon EKS	com.amazonaws. *region-code* .eks

- Qualsiasi nodo autogestito deve essere implementato in sottoreti con gli endpoint di interfaccia VPC richiesti. Se crei un gruppo di nodi gestiti, il gruppo di sicurezza dell'endpoint di interfaccia VPC deve consentire il CIDR per le sottoreti oppure dovrai aggiungere il gruppo di sicurezza del nodo creato al gruppo di sicurezza dell'endpoint di interfaccia VPC.

- Storage EFS: se i tuoi pod utilizzano volumi Amazon EFS, prima di distribuire Store, un file system elastico con Amazon EFS, devi modificare il file kustomization.yaml del driver per impostare le immagini del contenitore in modo che utilizzino la stessa regione del cluster Amazon EKS. AWS
- Route53 non supporta. AWS PrivateLink Non è possibile gestire i record DNS di Route53 da un cluster Amazon EKS privato. Ciò ha un impatto su external-dnsdi Kubernetes.
- Se utilizzi l'AMI ottimizzata EKS, devi abilitare l'endpoint `ec2` nella tabella precedente. In alternativa, esiste la possibilità di impostare manualmente il nome DNS del nodo. L'AMI ottimizzata utilizza l'AMI EC2 APIs per impostare automaticamente il nome DNS del nodo.
- Puoi utilizzare AWS Load Balancer Controller per distribuire AWS Application Load Balancer (ALB) e Network Load Balancer nel tuo cluster privato. Quando lo implementi, devi usare i flag della riga di comando per impostare`enable-shield`, `enable-waf` e `enable-wafv2` su fasle. Il rilevamento dei certificati con i nomi host degli oggetti in entrata non è supportato. Questo perché il controller deve raggiungere AWS Certificate Manager, che non dispone di un endpoint di interfaccia VPC.

  Il controller supporta bilanciatori di carico di rete destinazioni IP, necessari per l'utilizzo con Fargate. Per ulteriori informazioni, consultare the section called "Bilanciamento del carico di applicazione" e the section called "Creazione di un Network Load Balancer".
- Cluster Autoscaler è supportato. Quando si implementano i pod di Cluster Autoscaler, assicurarsi che la riga di comando includa `--aws-use-static-instance-list=true`. Per ulteriori informazioni, consulta Use Static Instance List on. GitHub Il VPC del nodo di lavoro deve includere anche l'endpoint VPC STS e l'endpoint AWS VPC con scalabilità automatica.
- Alcuni prodotti software container utilizzano chiamate API che accedono al AWS Marketplace Metering Service per monitorare l'utilizzo. I cluster privati non consentono queste chiamate, pertanto questi tipi di container non possono essere utilizzati nei cluster privati.

Dimensionamento dell'elaborazione dei cluster con Karpenter e Cluster Autoscaler

Il dimensionamento automatico è una funzione che dimensiona automaticamente in verticale e in orizzontale le risorse per soddisfare le esigenze in continua evoluzione. Questa è una delle principali funzioni Kubernetes, la cui esecuzione manuale richiederebbe altrimenti diverse risorse umane.

EKS Auto Mode

EKS Auto Mode di Amazon dimensiona automaticamente le risorse di calcolo del cluster. Se un pod non può adattarsi ai nodi esistenti, EKS Auto Mode ne crea uno nuovo. EKS Auto Mode consolida anche i carichi di lavoro ed elimina i nodi. EKS Auto Mode si basa su Karpenter.

Per ulteriori informazioni, consultare:

- Modalità automatica EKS
- the section called "Creazione di pool di nodi"
- the section called "Implementazione di un carico di lavoro di decompressione"

Ulteriori soluzioni

Amazon EKS supporta due ulteriori prodotti con dimensionamento automatico:

Karpenter

Karpenter è un Cluster Autoscaler Kubernetes flessibile e ad alte prestazioni che aiuta a migliorare la disponibilità delle applicazioni e l'efficienza del cluster. Karpenter avvia risorse di calcolo di dimensioni corrette, ad esempio le istanze di Amazon EC2, in risposta alla modifica del carico dell'applicazione in meno di un minuto. Integrando Kubernetes con AWS, Karpenter è in grado di effettuare il provisioning di risorse di calcolo just-in-time che soddisfano con precisione i requisiti del carico di lavoro. Karpenter effettua automaticamente il provisioning di nuove risorse di calcolo in base ai requisiti specifici dei carichi di lavoro del cluster. Questi includono requisiti di calcolo, archiviazione, accelerazione e pianificazione. Amazon EKS supporta i cluster che utilizzano Karpenter, anche se Karpenter funziona con qualsiasi cluster di Kubernetes conforme. Per ulteriori informazioni, consulta la documentazione di Karpenter.

> **Important**
>
> Karpenter è un software open source che i clienti AWS sono tenuti a installare, configurare e gestire nei propri cluster Kubernetes. AWS fornisce supporto tecnico quando Karpenter viene eseguito senza modifiche usando una versione compatibile nei cluster Amazon EKS. È fondamentale che i clienti mantengano la disponibilità e la sicurezza del controller Karpenter, nonché adeguate procedure di test quando lo aggiornano o aggiornano il cluster Kubernetes in cui è in esecuzione, proprio come qualsiasi altro software gestito dal cliente. Non esiste un accordo sul livello di servizio (SLA) AWS per

Karpenter e i clienti sono responsabili di garantire che le istanze EC2 avviate da Karpenter soddisfino i loro requisiti aziendali.

Cluster Autoscaler

Il Cluster Autoscaler Kubernetes regola automaticamente il numero di nodi nel cluster quando i pod non riescono o vengono riprogrammati su altri nodi. Il Cluster Autoscaler utilizza i gruppi con scalabilità automatica. Per ulteriori informazioni consultare Cluster Autoscaler su AWS.

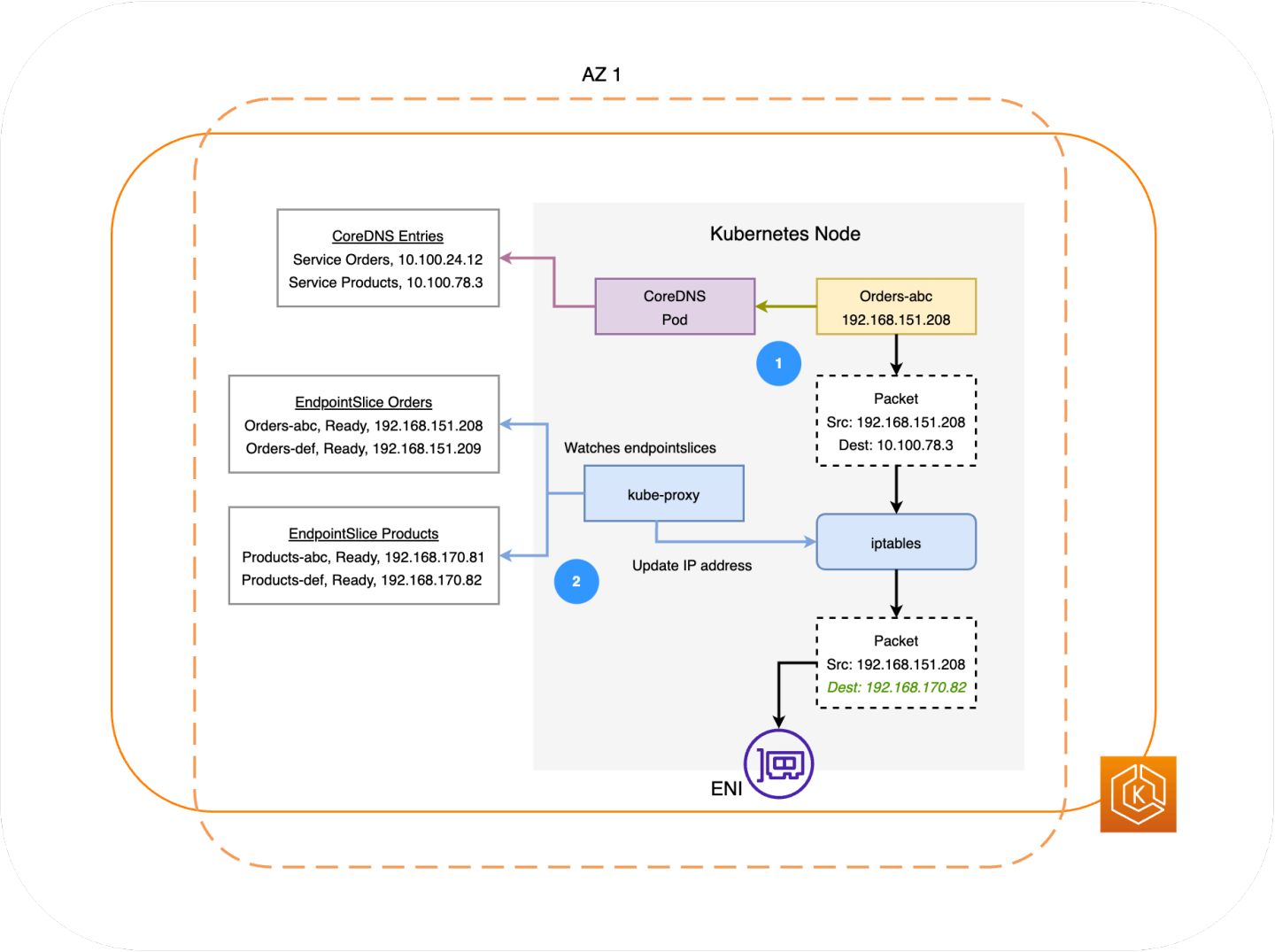
Ulteriori informazioni relative allo spostamento zonale in controller di ripristino delle applicazioni Amazon (ARC)

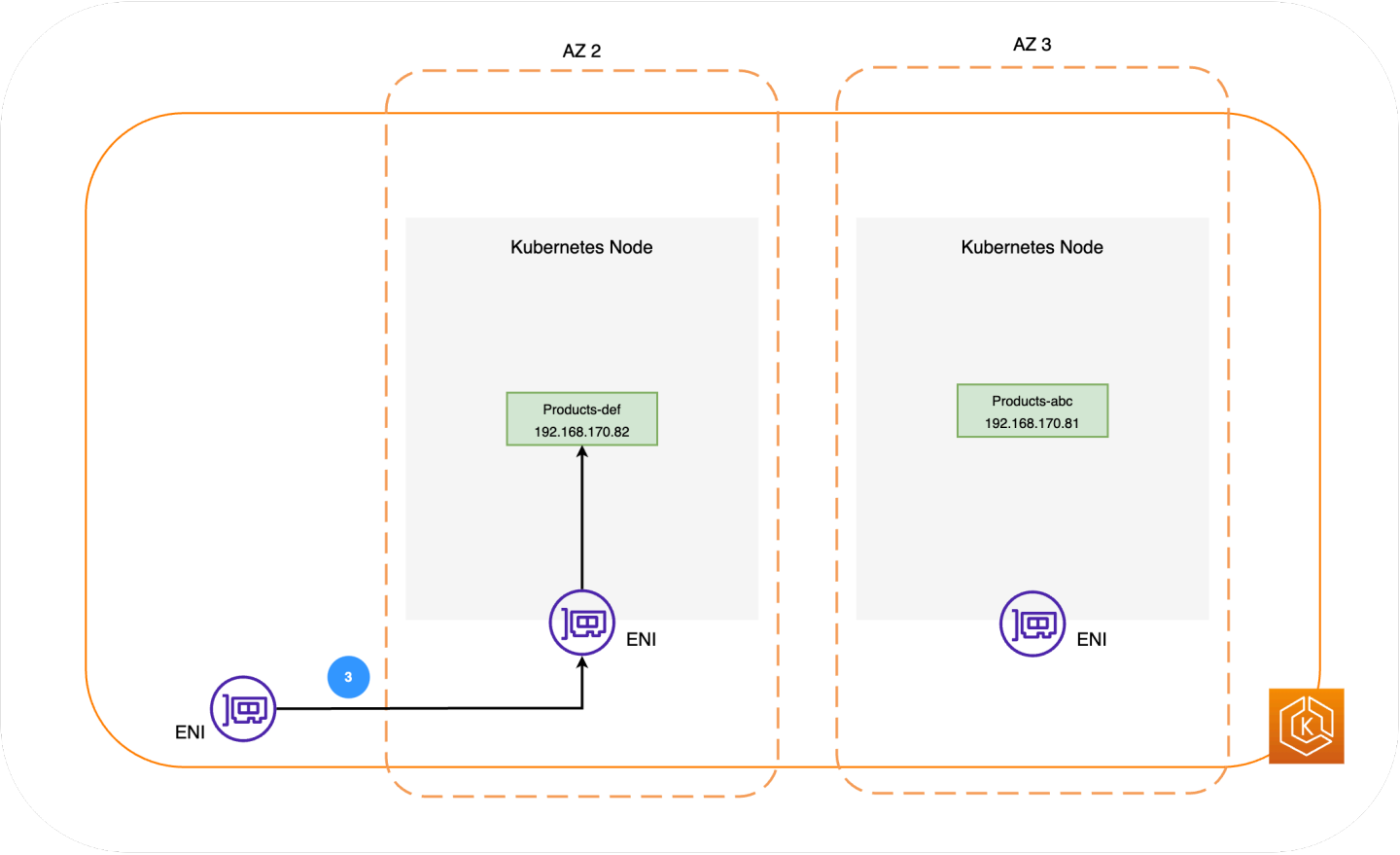
Kubernetes dispone di funzionalità native che consentono di rendere le applicazioni più resistenti a eventi come il deterioramento dello stato di integrità o il deterioramento di una zona di disponibilità (AZ). Quando esegui i carichi di lavoro in un cluster Amazon EKS, è possibile migliorare ulteriormente la tolleranza ai guasti e il ripristino delle applicazioni dell'ambiente applicativo utilizzando lo spostamento zonale o lo spostamento zonale automatico del controller di ripristino delle applicazioni Amazon (ARC). Lo spostamento zonale ARC è progettato per essere una misura temporanea che consente di spostare il traffico di una risorsa lontano da una AZ interessata fino alla scadenza dello spostamento zonale o all'annullamento dello stesso. È possibile estendere lo spostamento zonale, se necessario.

Puoi avviare uno spostamento di zona per un cluster EKS oppure puoi consentire AWS di spostare il traffico per te abilitando lo spostamento automatico di zona. Questa modifica aggiorna il flusso del traffico di east-to-west rete nel cluster in modo da prendere in considerazione solo gli endpoint di rete per i Pod in esecuzione sui nodi di lavoro in condizioni di integrità. AZs Inoltre, qualsiasi ALB o NLB che gestisca il traffico in ingresso per le applicazioni del cluster EKS indirizzerà automaticamente il traffico verso gli obiettivi integri. AZs Per i clienti che cercano gli obiettivi di disponibilità più elevati, nel caso in cui una zona AZ sia compromessa, può essere importante consentire di indirizzare tutto il traffico lontano dalla AZ compromessa fino a quando non si ripristina. Per questo, puoi anche abilitare ALB o NLB con spostamento zonale ARC.

Comprensione del flusso di traffico di rete est-ovest tra i pod

Il diagramma seguente illustra due esempi di carichi di lavoro, Ordini e Prodotti. Lo scopo di questo esempio è mostrare come comunicano carichi di lavoro e Pod diversi. AZs


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
1
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
2
Update IP address
Packet
Src: 192.168.151.208
Dest: 192.168.170.82
ENI



1. Affinché gli Ordini comunichino con i Prodotti, gli Ordini devono prima risolvere il nome DNS del servizio di destinazione. Ordini comunica con CoredNS per recuperare l'indirizzo IP virtuale (IP cluster) per quel servizio. Dopo che Ordini ha risolto il nome del servizio Prodotti, invia il traffico a quell'indirizzo IP di destinazione.
2. Il kube-proxy viene eseguito su ogni nodo del cluster e controlla continuamente i servizi. EndpointSlices Quando viene creato un servizio, un servizio EndpointSlice viene creato e gestito in background dal controller. EndpointSlice Ciascuno EndpointSlice ha un elenco o una tabella di endpoint che contiene un sottoinsieme di indirizzi Pod, insieme ai nodi su cui sono in esecuzione. Il kube-proxy imposta le regole di routing per ciascuno di questi endpoint di pod utilizzando `iptables` sui nodi. Il kube-proxy è anche responsabile di una forma base di bilanciamento del carico, che consiste nel reindirizzare il traffico destinato all'indirizzo IP del cluster di un servizio per inviarlo invece direttamente all'indirizzo IP di un pod. Il kube-proxy esegue questa operazione riscrivendo l'indirizzo IP di destinazione sulla connessione in uscita.
3. I pacchetti di rete vengono quindi inviati al Products Pod in AZ 2 utilizzando i ENIs rispettivi nodi, come mostrato nel diagramma precedente.

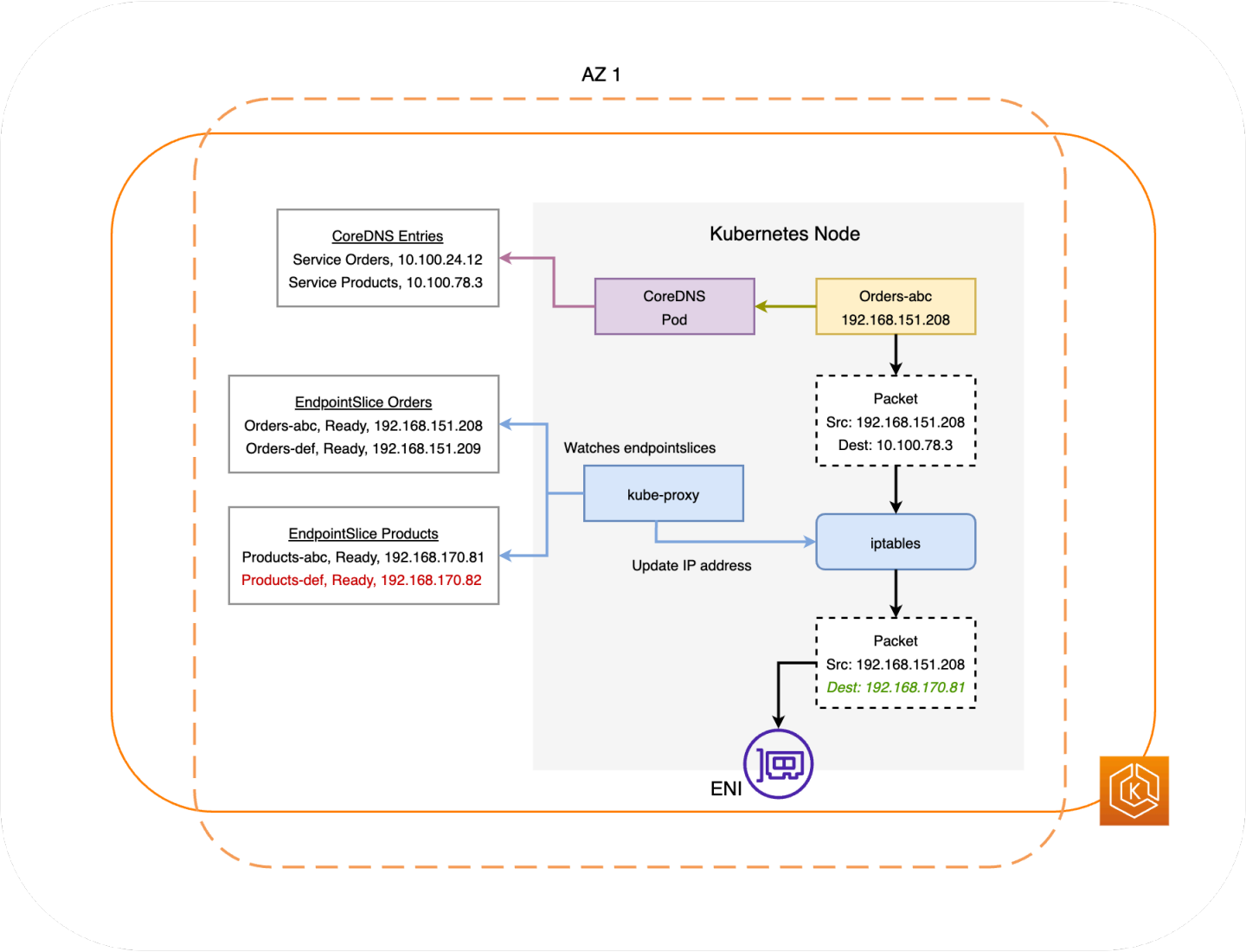
Comprensione dello spostamento zonale ARC in Amazon EKS

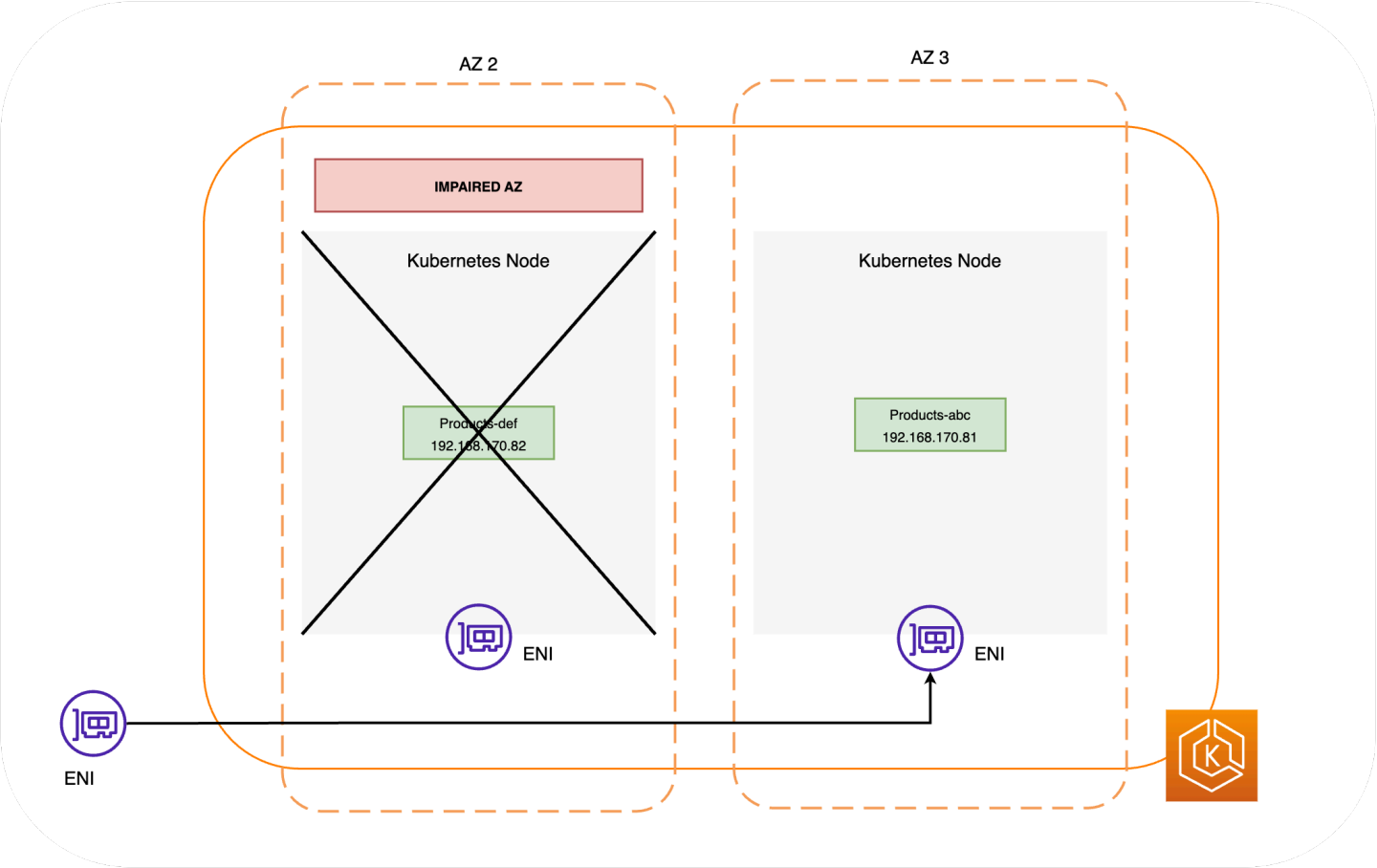
Se il tuo ambiente presenta un deterioramento di AZ, è possibile avviare uno spostamento zonale per l'ambiente del cluster EKS. In alternativa, puoi consentire di gestire AWS al posto tuo il traffico variabile con lo spostamento automatico zonale. Con lo spostamento automatico zonale, AWS monitora lo stato generale della zona a zona e risponde a un potenziale deterioramento della zona Z spostando automaticamente il traffico dall'area AZ compromessa nell'ambiente del cluster.

Dopo che il cluster Amazon EKS ha abilitato lo spostamento zonale con ARC, puoi avviare uno spostamento zonale o abilitare lo spostamento automatico di zona utilizzando la console ARC, la AWS CLI o lo spostamento zonale e lo spostamento automatico zonale. APIs Durante uno spostamento zonale EKS, è eseguito automaticamente quanto segue:

- Tutti i nodi della zona AZ interessata sono isolati. Ciò impedisce al sistema di pianificazione Kubernetes di pianificare nuovi pod su nodi in una zona AZ non integra.
- Se utilizzi i gruppi di nodi gestiti, il ribilanciamento delle zone di disponibilità viene sospeso e il gruppo Auto Scaling viene aggiornato per garantire che i nuovi nodi del piano dati EKS vengano lanciati solo in condizioni ottimali. AZs
- I nodi all'interno di AZ non integra non sono terminati e i pod non sono espulsi dai nodi. In questo modo, quando uno spostamento zonale scade o è annullato, il traffico può essere restituito in tutta sicurezza ad AZ per sfruttarne la piena capacità.
- Il EndpointSlice controller trova tutti gli endpoint Pod nella zona AZ compromessa e li rimuove dai relativi. EndpointSlices Ciò garantisce che solo gli endpoint Pod integri AZs siano destinati a ricevere il traffico di rete. Quando uno spostamento di zona viene annullato o scade, il EndpointSlice controller lo aggiorna EndpointSlices per includere gli endpoint nella AZ ripristinata.

I seguenti diagrammi forniscono una panoramica di alto livello su come uno spostamento zonale EKS garantisce che solo gli endpoint dei pod integri siano presi di mira nell'ambiente del cluster.


AZ 1
Kubernetes Node
CoreDNS Entries
Service Orders, 10.100.24.12
Service Products, 10.100.78.3
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
EndpointSlice Products
Products-abc, Ready, 192.168.170.81
Products-def, Ready, 192.168.170.82
Update IP address
iptables
Packet
Src: 192.168.151.208
Dest: 192.168.170.81
ENI



Requisiti per lo spostamento zonale EKS

Affinché lo spostamento zonale funzioni correttamente con EKS, è necessario configurare in anticipo l'ambiente del cluster in modo che sia resiliente a una compromissione di AZ. Di seguito è riportato un elenco di opzioni di configurazione che aiutano a garantire la resilienza.

- Effettua il provisioning dei nodi di lavoro del cluster su più nodi AZs
- Consulta a una capacità di elaborazione sufficiente per consentire la rimozione di una singola AZ
- Pre-scala i tuoi pod, incluso CoredNS, in ogni AZ
- Distribuisci più repliche Pod su tutti AZs, per assicurarti che anche quando ti allontani da una singola AZ, avrai comunque una capacità sufficiente
- Effettua la co-locazione di pod interdipendenti o correlati nella stessa AZ
- Verifica che il tuo ambiente cluster funzioni come previsto senza una AZ avviando manualmente uno spostamento zonale dalla zona AZ. In alternativa, è possibile abilitare lo spostamento zonale automatico e fare affidamento sulle esecuzioni pratiche di spostamento automatico. I test con

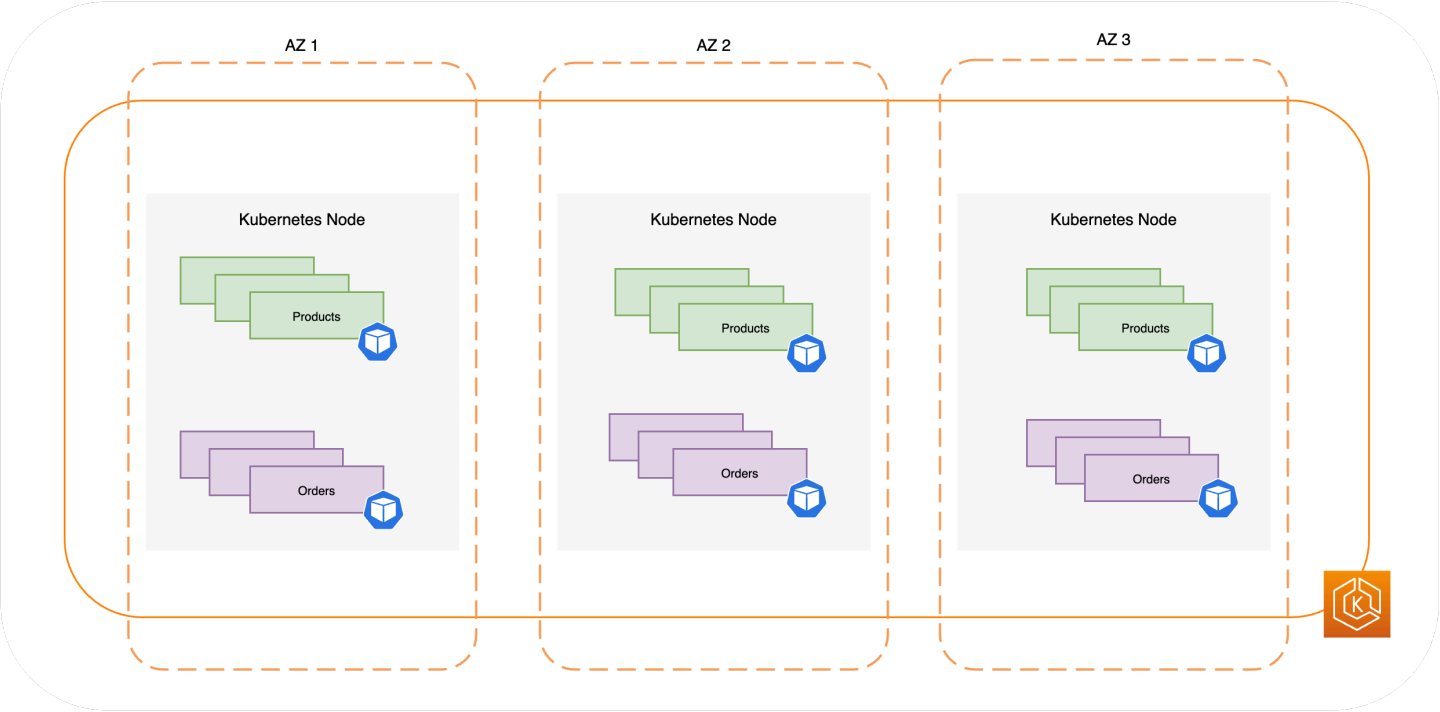
spostamenti zonali manuali o pratici non sono necessari affinché lo spostamento zonale funzioni in EKS, ma sono fortemente consigliati.

Esegui il provisioning dei nodi workerEKS in più zone di disponibilità

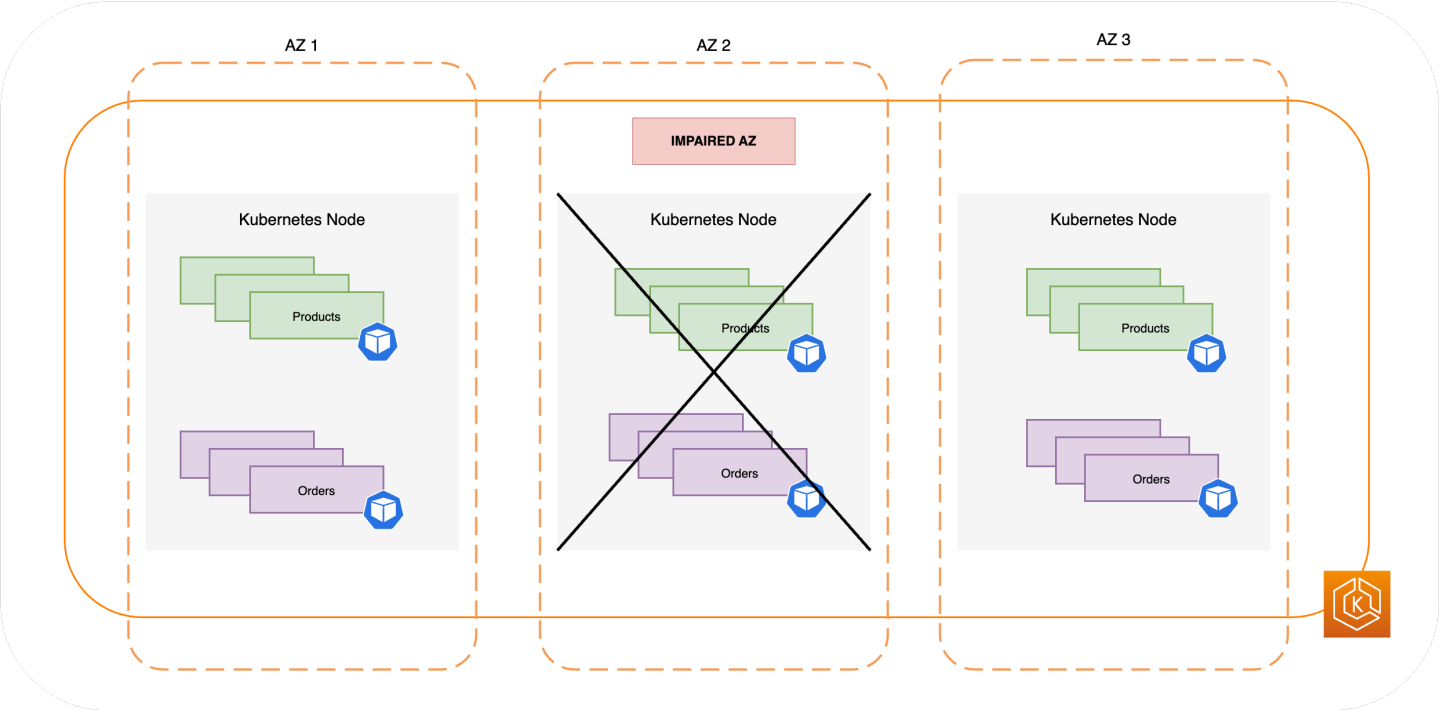
AWS Le regioni hanno più sedi separate con data center fisici, note come zone di disponibilità (AZs). AZs sono progettate per essere isolate fisicamente l'una dall'altra per evitare impatti simultanei che potrebbero interessare un'intera regione. Quando si effettua il provisioning di un cluster EKS, si consiglia di distribuire i nodi di lavoro su più nodi AZs in una regione. Ciò contribuisce a rendere l'ambiente del cluster più resiliente ai danni di una singola AZ e consente di mantenere un'elevata disponibilità per le applicazioni eseguite nell'altra. AZs Quando iniziate un cambiamento di zona rispetto alla zona di zona interessata, la rete interna al cluster dell'ambiente EKS si aggiorna automaticamente per utilizzarla solo in condizioni ottimali AZs, per contribuire a mantenere un'elevata disponibilità del cluster.

La garanzia di disporre di una configurazione multi-AZ per l'ambiente EKS migliora l'affidabilità complessiva del sistema. Tuttavia, gli ambienti multi-AZ influenzano il modo in cui i dati delle applicazioni sono trasferiti ed elaborati, il che a sua volta ha un impatto sui costi di rete dell'ambiente. In particolare, il traffico in uscita frequente tra le zone (traffico distribuito tra le zone AZs) può avere un impatto importante sui costi relativi alla rete. È possibile applicare diverse strategie per controllare la quantità di traffico tra le zone tra i pod nel cluster EKS e ridurre i costi associati. Per ulteriori informazioni su come ottimizzare i costi di rete durante l'esecuzione di ambienti EKS ad alta disponibilità, consulta queste best practice.

Il diagramma seguente illustra un ambiente EKS ad alta disponibilità con tre ambienti integri. AZs



Il diagramma seguente illustra come un ambiente EKS con tre elementi AZs sia resiliente a una compromissione AZ e rimanga altamente disponibile perché ne rimangono due integri. AZs



Consulta a una capacità di elaborazione sufficiente per resistere alla rimozione di una singola zona di disponibilità

Per ottimizzare l'utilizzo delle risorse e i costi per l'infrastruttura di elaborazione nel piano dati EKS, è consigliabile allineare la capacità di elaborazione ai requisiti del carico di lavoro. Tuttavia, se tutti i nodi worker sono a piena capacità, è necessario aggiungere nuovi nodi worker al piano dati EKS prima di poter pianificare nuovi pod. Quando si eseguono carichi di lavoro critici, in genere è buona norma impiegare una capacità ridondante online per gestire scenari quali aumenti improvvisi del carico e problemi di integrità dei nodi. Se prevedi di utilizzare lo spostamento zonale, intendi rimuovere un'intera AZ di capacità in caso di compromissione. Ciò significa che è necessario regolare la capacità di elaborazione ridondante in modo che sia sufficiente a gestire il carico anche con uno dei sistemi offline. AZs

Quando si ridimensionano le risorse di elaborazione, il processo di aggiunta di nuovi nodi al piano dati EKS richiede del tempo. Ciò può avere implicazioni sulle prestazioni e sulla disponibilità in tempo reale delle applicazioni, soprattutto in caso di una compromissione zonale. Il tuo ambiente EKS dovrebbe essere in grado di assorbire il peso della perdita di una AZ senza compromettere l'esperienza degli utenti finali o dei clienti. Ciò significa ridurre al minimo o eliminare il ritardo tra il momento in cui è necessario un nuovo pod e quello in cui è effettivamente pianificato su un nodo worker.

Inoltre, in caso di problemi a livello di zona, dovreste mirare a mitigare il rischio di incorrere in un limite di capacità di elaborazione che impedirebbe l'aggiunta di nuovi nodi necessari al piano dati EKS quando sono integri. AZs

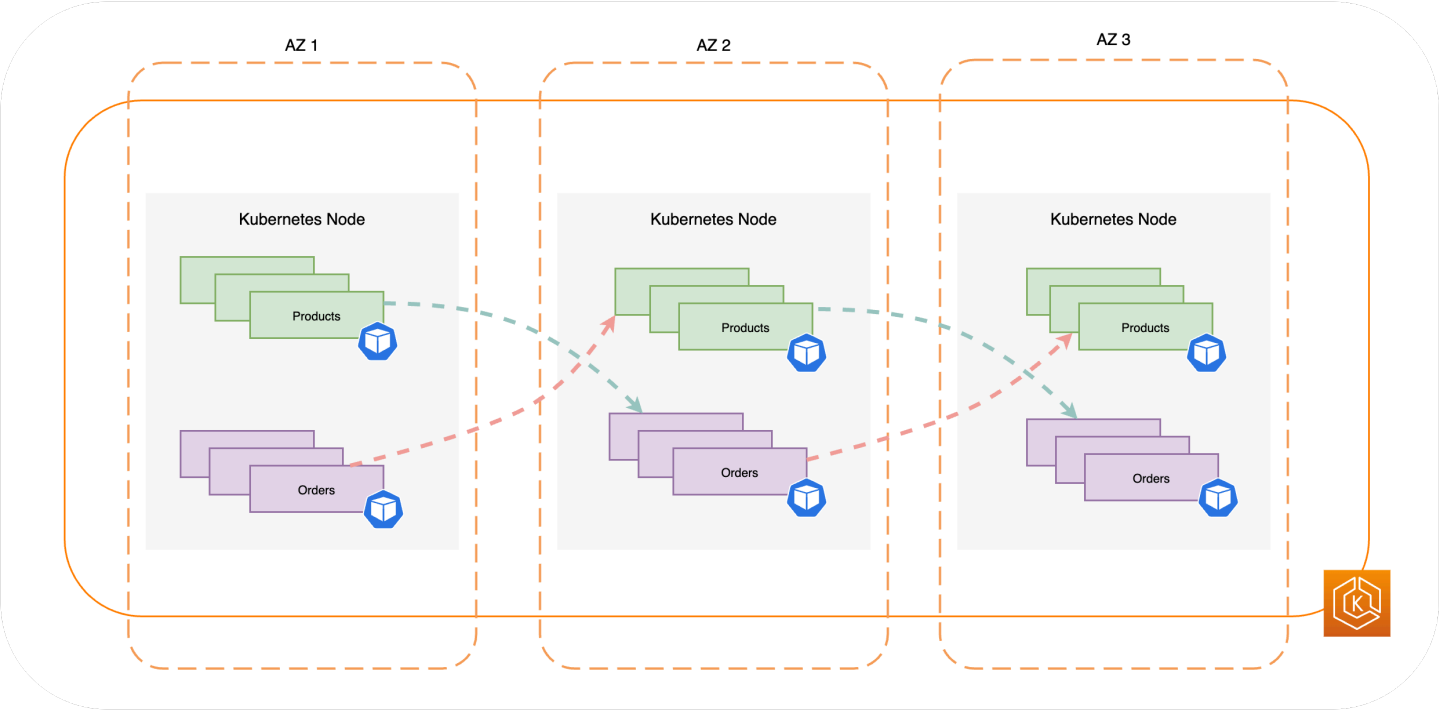
Per ridurre il rischio di questi potenziali impatti negativi, consigliamo di fornire una capacità di elaborazione eccessiva in alcuni nodi di lavoro di ciascuno di essi. AZs In questo modo, Kubernetes Scheduler dispone di una capacità preesistente per il posizionamento di nuovi Pod, il che è particolarmente importante quando si perde uno di essi nell'ambiente. AZs

Esegui e distribuisci più repliche di pod tra le zone di disponibilità

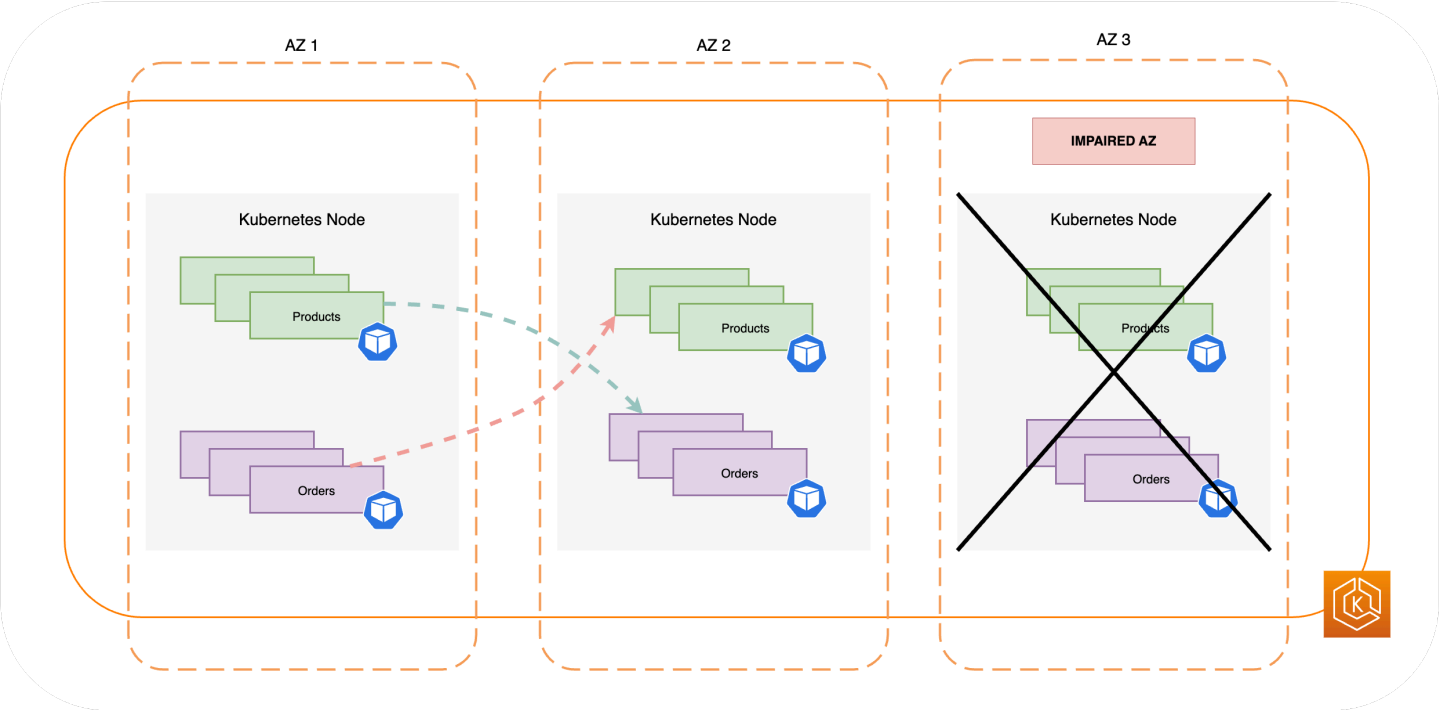
Kubernetes ti consente di pre-scalare i carichi di lavoro eseguendo più istanze (repliche di pod) di una singola applicazione. L'esecuzione di più repliche di pod per un'applicazione elimina i singoli punti di guasto e aumenta le prestazioni complessive riducendo il carico di sovraccarico su una singola replica. Tuttavia, per garantire un'elevata disponibilità e una migliore tolleranza ai guasti per le applicazioni, si consiglia di eseguire più repliche dell'applicazione e di distribuirle su diversi domini di guasto, noti anche come domini di topologia. I domini di guasto in questo scenario sono

le zone di disponibilità. Utilizzando i vincoli di diffusione della topologia, è possibile configurare le applicazioni in modo che abbiano una stabilità statica pre-esistente. Quindi, in caso di problemi di AZ, il tuo ambiente avrà un numero sufficiente di repliche in buone condizioni per gestire immediatamente eventuali picchi o picchi AZs di traffico.

Il diagramma seguente illustra un ambiente EKS con flusso di traffico quando tutto è east-to-west in buone condizioni. AZs



Il diagramma seguente illustra un ambiente EKS con un flusso di east-to-west traffico in cui una singola AZ ha avuto un errore ed è stato avviato un cambiamento di zona.



Il seguente frammento di codice è un esempio di come configurare il carico di lavoro con più repliche in 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
```

Soprattutto, è necessario eseguire più repliche del software del server DNS (CoreDNS/Kube-DNS) e applicare vincoli di diffusione della topologia simili, se non sono configurati per impostazione predefinita. In questo modo è possibile garantire che, in caso di un solo problema della zona di disponibilità, siano presenti abbastanza pod DNS integri per continuare AZs a gestire le richieste di rilevamento del servizio per gli altri Pod comunicanti del cluster. Il componente aggiuntivo CoreDNS EKS dispone di impostazioni predefinite per i CoredNS Pods che assicurano che, se sono presenti AZs più nodi disponibili, siano distribuiti nelle zone di disponibilità del cluster. Se si desidera, è possibile sostituire queste impostazioni predefinite con le proprie configurazioni personalizzate.

Quando installi CoreDNS con Helm, è possibile aggiornare `replicaCount` nel file values.yaml per assicurarti di avere un numero sufficiente di repliche in ciascuna AZ. Inoltre, per garantire che queste repliche siano distribuite tra i diversi ambienti del cluster, assicuratevi di aggiornare la proprietà AZs nello stesso file. `topologySpreadConstraints` `values.yaml` Il seguente frammento di codice illustra come configurare CoreDNS per eseguire tale operazione.

values.yaml CoredNS Helm

```
replicaCount: 6
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

In caso di compromissione di AZ, è possibile assorbire l'aumento del carico sui pod CoreDNS, utilizzando un sistema di scalabilità automatica per CoreDNS. Il numero di istanze DNS necessarie dipende dal numero di carichi di lavoro in esecuzione nel cluster. CoreDNS è legato alla CPU, il che gli consente di scalare in base alla CPU utilizzando l'Horizontal Pod Autoscaler (HPA). Di seguito è riportato un esempio che è possibile modificare in base alle esigenze.

```
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
```

In alternativa, EKS può gestire la scalabilità automatica dell'implementazione CoreDNS nella versione del componente aggiuntivo EKS di CoreDNS. Questo autoscaler di CoreDNS monitora continuamente lo stato del cluster, incluso il numero di nodi e core CPU. Sulla base di tali informazioni, il controller regola dinamicamente il numero di repliche dell'implementazione CoreDNS in un cluster EKS.

Per abilitare la configurazione di scalabilità automatica nel componente aggiuntivo CoredNS EKS, utilizzare la seguente impostazione di configurazione:

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

Puoi anche usare NodeLocal DNS o l'autoscaler proporzionale del cluster per scalare CoredNS. Per ulteriori informazioni, consulta Scalare CoredNS orizzontalmente.

Co-locare i pod interdipendenti nella stessa zona di disponibilità

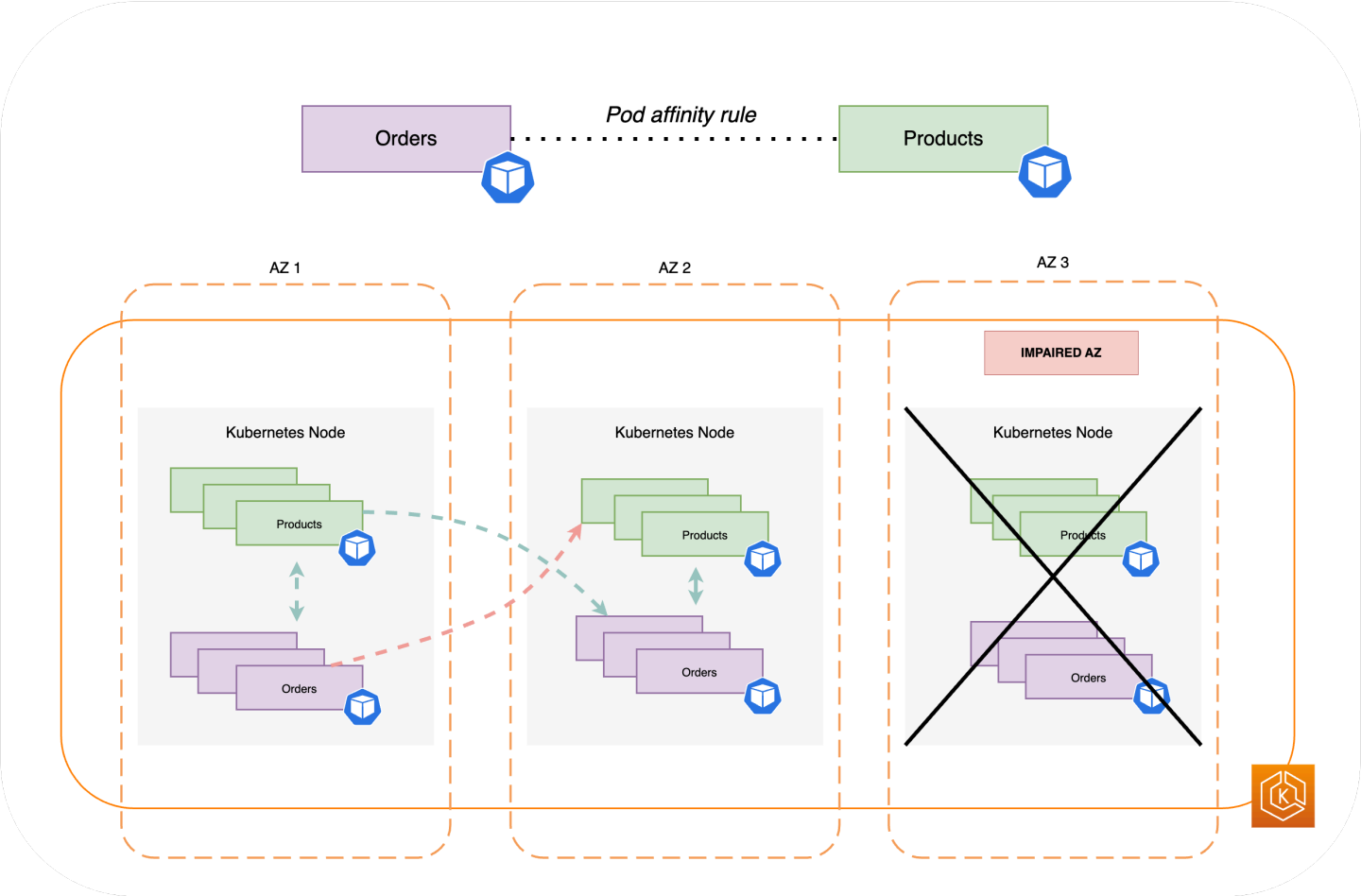
In genere, le applicazioni hanno carichi di lavoro distinti che devono comunicare tra loro per completare correttamente un processo. end-to-end Se queste applicazioni distinte sono distribuite su più aree AZs e non sono collocate nella stessa zona di zona, una singola area di disponibilità può influire sul processo. end-to-end Ad esempio, se l'applicazione A ha più repliche in AZ 1 e AZ 2, ma l'applicazione B ha tutte le repliche in AZ 3, la perdita di AZ 3 influirà sui end-to-end processi tra i due carichi di lavoro, l'applicazione A e l'applicazione B. Se combini i vincoli di diffusione della topologia con l'affinità dei pod, puoi migliorare la resilienza dell'applicazione distribuendo i Pod su tutti. AZs Inoltre, ciò configura una relazione tra determinati pod per garantire che siano co-locati.

Con le regole di affinità dei pod, è possibile definire le relazioni tra i carichi di lavoro per influenzare il comportamento del sistema di pianificazione Kubernetes in modo che collochi i pod sullo stesso nodo worker o nella stessa AZ. Inoltre, è possibile configurare la rigidità dei vincoli di pianificazione.

```
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"
```

Il diagramma seguente mostra diversi pod che sono stati co-locati sullo stesso nodo utilizzando le regole di affinità dei pod.



Verifica che il tuo ambiente cluster sia in grado di gestire la perdita di una AZ

Dopo aver completato i requisiti descritti nelle sezioni precedenti, il passaggio successivo consiste nel verificare di disporre di una capacità di elaborazione e carico di lavoro sufficiente per gestire la perdita di una AZ. Per eseguire tale operazione è possibile avviare manualmente uno spostamento zonale in EKS. In alternativa, è possibile abilitare lo spostamento zonale automatico e configurare le sessioni pratiche, che consentono anche di verificare che le applicazioni funzionino come previsto con una AZ in meno nell'ambiente cluster.

Domande frequenti

Qual è il vantaggio di utilizzare questa funzionalità?

Utilizzando lo spostamento zonale ARC o lo spostamento zonale automatico nel cluster EKS, è possibile mantenere meglio la disponibilità delle applicazioni Kubernetes automatizzando il processo di ripristino rapido che consiste nello spostamento del traffico di rete interno al cluster da una zona AZ

compromessa. Con ARC, è possibile evitare passaggi lunghi e complicati, che possono portare a un periodo di recupero prolungato in caso di eventi AZ compromessi.

Come funziona questa funzionalità con altri servizi? AWS

EKS si integra con ARC, che fornisce l'interfaccia principale in cui eseguire le operazioni di ripristino. AWS Per garantire che il traffico all'interno del cluster sia indirizzato in modo appropriato lontano da una zona di disponibilità compromessa, EKS apporta modifiche all'elenco degli endpoint di rete per i pod in esecuzione sul piano dati Kubernetes. Se utilizzi Elastic Load Balancing per instradare il traffico esterno verso il cluster, è possibile registrare i bilanciatori del carico con ARC e avviare uno spostamento zonale su di essi per impedire che il traffico fluisca verso la zona AZ deteriorata. Zonal shift funziona anche con i gruppi di Amazon EC2 Auto Scaling creati da gruppi di nodi gestiti da EKS. Per evitare che una AZ compromessa sia utilizzata per nuovi pod Kubernetes o avvii di nodi, EKS rimuove la AZ compromessa dai gruppi Auto Scaling.

In che modo questa funzionalità è diversa dalle protezioni Kubernetes predefinite?

Tale funzionalità funziona in combinazione con diverse protezioni integrate di Kubernetes che aiutano la resilienza delle applicazioni dei clienti. È possibile configurare le sonde readiness e liveness di pod che decidono quando un Pod deve ricevere traffico. Quando queste sonde si guastano, Kubernetes rimuove questi pod come obiettivi di un servizio e il traffico non è più inviato al pod. Sebbene ciò sia utile, non è semplice per i clienti configurare tali controlli dell'integrità in modo da garantire che abbiano esito negativo in caso di deterioramento di una AZ. La funzione di spostamento zonale ARC fornisce una rete di sicurezza aggiuntiva che ti aiuta a isolare completamente una AZ deteriorata quando le protezioni native di Kubernetes non erano sufficienti. Lo spostamento zonale offre anche un modo semplice per testare la prontezza operativa e la resilienza dell'architettura.

Posso AWS avviare un cambiamento di zona per mio conto?

Sì, se desideri un modo completamente automatizzato di utilizzare lo spostamento zonale ARC, puoi abilitare lo spostamento zonale automatico ARC. Con l'autoshift zonale, puoi contare sul AWS monitoraggio dello stato del tuo cluster EKS e sull' AZs avvio automatico di uno spostamento di zona quando viene rilevata una compromissione della zona A Z.

Cosa succede se utilizzo questa funzionalità e i miei nodi worker e i miei carichi di lavoro non sono predimensionati?

Se non si esegue una pre-scalabilità e si affida al provisioning di nodi o pod aggiuntivi durante uno spostamento zonale, si rischia un ripristino ritardato. Il processo di aggiunta di nuovi nodi al piano

dati di Kubernetes richiede del tempo, il che può influire sulle prestazioni e sulla disponibilità in tempo reale delle applicazioni, soprattutto in caso di compromissione zonale. Inoltre, in caso di compromissione zonale, è possibile che si verifichi un potenziale limite di capacità di elaborazione che potrebbe impedire l'aggiunta dei nuovi nodi necessari a quelli integri. AZs

Se i carichi di lavoro non sono prescalati e distribuiti su tutto il cluster, un AZs problema zonale potrebbe influire sulla disponibilità di un'applicazione che viene eseguita solo sui nodi di lavoro nelle zone di lavoro interessate. Per mitigare il rischio di un'interruzione completa della disponibilità dell'applicazione, EKS dispone di un sistema a prova di guasto per l'invio del traffico agli endpoint di pod in una zona compromessa se il carico di lavoro ha tutti i suoi endpoint in una AZ non integra. Tuttavia, ti consigliamo vivamente di pre-scalare e distribuire le applicazioni su tutte AZs per mantenere la disponibilità in caso di problemi zonali.

Come funziona se sto eseguendo un'applicazione stateful?

Se stai eseguendo un'applicazione stateful, devi valutarne la tolleranza ai guasti, in base al tuo caso d'uso e all'architettura. Se disponi di un' active/standby architettura o di un pattern, potrebbero esserci casi in cui l'attivo si trova in una zona AZ compromessa. A livello di applicazione, se lo standby non è attivato, è possibile che si verifichino problemi con l'applicazione. Potresti inoltre riscontrare problemi quando i nuovi Kubernetes Pod vengono lanciati in modalità integra AZs, poiché non saranno in grado di collegarsi ai volumi persistenti limitati alla zona AZ compromessa.

Questa funzionalità funziona con Karpenter?

Il supporto di Karpenter non è attualmente disponibile con uno spostamento zonale ARC e uno spostamento zonale automatico in EKS. Se una AZ è compromessa, puoi modificare la NodePool configurazione di Karpenter pertinente rimuovendo quella non integra in modo che i nuovi nodi di lavoro vengano avviati solo nell'altra. AZs

Questa funzionalità funziona con EKS Fargate?

Questa funzionalità non funziona con EKS Fargate. Per impostazione predefinita, quando EKS Fargate riconosce un evento sanitario zonale, i Pods preferiranno essere eseguiti nell'altra. AZs

Il piano di controllo Kubernetes gestito da EKS ne risentirà?

No, per impostazione predefinita Amazon EKS esegue e ridimensiona il piano di controllo Kubernetes su più piani AZs per garantire un'elevata disponibilità. Lo spostamento zonale ARC e lo spostamento zonale automatico agiscono solo sul piano dati Kubernetes.

Ci sono dei costi associati a questa nuova funzionalità?

È possibile utilizzare uno spostamento zonale ARC e uno spostamento zonale automatico nel cluster EKS senza costi aggiuntivi. Tuttavia, continuerai a pagare per le istanze assegnate e consigliamo vivamente di pre-scalare il tuo piano dati Kubernetes prima di utilizzare questa funzionalità. È necessario considerare un equilibrio tra costi e disponibilità delle applicazioni.

Risorse aggiuntive

- Come funziona uno spostamento zonale
- Best practice per spostamenti zonali in ARC
- Risorse e scenari supportati per lo spostamento zonale e lo spostamento zonale automatico
- Funzionamento di carichi di lavoro resilienti su Amazon EKS
- Elimina il ritardo di scalabilità dei nodi Kubernetes con la priorità dei pod e l'eccesso di provisioning
- Scala i pod CoreDNS per un traffico DNS elevato

Abilitazione dello spostamento zonale EKS per evitare zone di disponibilità compromesse

Amazon Application Recovery Controller (ARC) ti aiuta a gestire e coordinare il ripristino per le tue applicazioni nelle zone di disponibilità (AZs) e funziona con molti servizi, tra cui Amazon EKS. Con il supporto EKS per lo spostamento zonale ARC, è possibile spostare il traffico di rete interno al cluster da una AZ compromessa. Puoi anche AWS autorizzare il monitoraggio dello stato della tua rete AZs e spostare temporaneamente il traffico di rete da una zona non sicura per tuo conto.

Come utilizzare uno spostamento zonale EKS:

1. Abilita il tuo cluster EKS con Amazon Application Recovery Controller (ARC). Questa operazione viene eseguita a livello di cluster utilizzando la console Amazon EKS, la AWS CLI o CloudFormation eksctl.
2. Una volta abilitato, puoi gestire i turni zonali o gli spostamenti automatici zonali utilizzando la console ARC, la AWS CLI o lo spostamento zonale e lo spostamento automatico zonale. APIs

Tieni presente che dopo aver registrato un cluster EKS con ARC, devi comunque configurare ARC. Per esempio, è possibile utilizzare la console ARC per configurare lo spostamento zonale automatico.

Per informazioni più dettagliate su come funziona lo spostamento zonale EKS e su come progettare i carichi di lavoro per gestire le zone di disponibilità compromesse, consulta the section called "Ulteriori informazioni relative allo spostamento zonale".

Considerazioni

- La modalità automatica EKS non supporta Amazon Application Recovery Controller, lo spostamento zonale e lo spostamento zonale automatico.
- Consigliamo di attendere almeno 60 secondi tra le operazioni di spostamento zonale per garantire la corretta elaborazione di ciascuna richiesta.

  Quando si tenta di eseguire spostamenti zonali in rapida successione (entro 60 secondi l'uno dall'altro), il servizio Amazon EKS potrebbe non elaborare correttamente tutte le richieste di spostamento. Ciò è dovuto all'attuale meccanismo di polling che aggiorna lo stato zonale del cluster. Se è necessario eseguire più spostamenti zonali, assicuratevi che vi sia un periodo di tempo adeguato tra le operazioni affinché il sistema elabori ogni modifica.

What is Amazon Application Recovery Controller?

Amazon Application Recovery Controller (ARC) ti aiuta a prepararti e a portare a termine ripristini più rapidi per le applicazioni in esecuzione su AWS. Lo spostamento zonale consente di riprendersi rapidamente dai problemi relativi alla zona di disponibilità (AZ), spostando temporaneamente il traffico di una risorsa supportata da una zona di disponibilità a un livello ottimale nella regione. AZs AWS

Ulteriori informazioni su Amazon Application Recovery Controller (ARC)

Che cos'è uno spostamento zonale?

Lo spostamento zonale è una funzionalità di ARC che consente di spostare il traffico per una risorsa come un cluster EKS o un Elastic Load Balancer lontano da una zona di disponibilità in AWS una regione per mitigare rapidamente un problema e ripristinare rapidamente l'applicazione. È possibile scegliere di spostare il traffico, per esempio, perché una cattiva implementazione causa problemi di latenza o perché la zona di disponibilità è compromessa. Uno spostamento zonale non richiede passaggi di configurazione avanzati.

Ulteriori informazioni sullo spostamento zonale ARC

Cos'è lo spostamento automatico zonale automatico?

L'autoshift zonale è una funzionalità di ARC che puoi abilitare per autorizzare lo spostamento del traffico da una zona di zona AWS alle risorse supportate, per tuo conto, a un traffico funzionante nella regione. AZs AWS AWS avvia uno spostamento automatico quando la telemetria interna indica che in una zona AZ in una regione si è verificata una compromissione che potrebbe avere un impatto sui clienti. La telemetria interna incorpora metriche provenienti da più fonti, tra cui la rete e i servizi AWS Amazon ed Elastic Load EC2 Balancing.

AWS interrompe gli spostamenti automatici quando gli indicatori indicano che non esiste più un problema o un problema potenziale.

Ulteriori informazioni sullo spostamento zonale automatico ARC

Cosa fa EKS durante uno spostamento automatico?

EKS aggiorna le configurazioni di rete per evitare di indirizzare il traffico verso persone con problemi. AZs Inoltre, se utilizzi Managed Node Groups, EKS lancerà nuovi nodi nell'area integra solo AZs durante un cambio di zona. Quando lo spostamento scade o viene annullato, le configurazioni di rete saranno ripristinate per includere l'AZ che in precedenza era stato rilevato come non integra.

Ulteriori informazioni sullo spostamento zonale EKS

Registra il cluster EKS con Amazon Application Recovery Controller (ARC) (AWS console)

1. Trova il nome e la regione del cluster EKS che desideri registrare con ARC.
2. Vai alla Console EKS in quella regione e seleziona il tuo cluster.
3. Nella pagina delle Informazioni sul cluster, seleziona la scheda Panoramica.
4. Sotto l'intestazione Spostamento zonale, seleziona il pulsante Gestisci.
5. Seleziona abilita o disabilita per lo spostamento zonale EKS.

Ora il tuo cluster EKS è registrato con ARC.

Se desideri AWS rilevare ed evitare zone di disponibilità ridotta, devi configurare l'autoshift zonale ARC. Ad esempio, è possibile eseguire questa operazione nella console ARC.

Fasi successive

- Scopri come abilitare lo spostamento zonale automatico.
- Scopri come avviare manualmente uno spostamento zonale.

Informazioni su come funziona il controllo degli accessi in Amazon EKS

Scopri come gestire l'accesso al tuo cluster Amazon EKS. L'utilizzo di Amazon EKS richiede la conoscenza di come Kubernetes e AWS Identity and Access Management (AWS IAM) gestiscono il controllo degli accessi.

Questa sezione include:

the section called "Accesso API Kubernetes": scopri come permettere alle applicazioni o agli utenti di autenticarsi nell'API Kubernetes. Puoi utilizzare access entry, aws-auth ConfigMap o un provider OIDC esterno.

the section called "Accesso alle risorse di cluster"— Scopri come configurare la comunicazione con Console di gestione AWS il tuo cluster Amazon EKS. Utilizza la console per visualizzare le risorse Kubernetes nel cluster, come namespace, nodi e pod.

the section called "Modifica le risorse del cluster"— Scopri le autorizzazioni necessarie per modificare le risorse Kubernetes.

the section called "Accedere al cluster con kubectl" : scopri come configurare kubectl per comunicare con il tuo cluster Amazon EKS. Usa la AWS CLI per creare un file kubeconfig.

the section called "Accesso ai carichi di lavoro a AWS "— Scopri come associare un account di servizio Kubernetes a IAM Roles. AWS Puoi usare Pod Identity o i ruoli IAM per gli account di servizio (IRSA).

Attività comuni

- Concedi l'accesso all'API Kubernetes agli sviluppatori. Visualizza le risorse Kubernetes in. Console di gestione AWS
  - Soluzione: utilizza le voci di accesso per associare le autorizzazioni RBAC di Kubernetes agli utenti o ai ruoli IAM. AWS
- Configura kubectl per comunicare con un cluster Amazon EKS utilizzando le credenziali. AWS
  - Soluzione: utilizzate la AWS CLI per creare un file kubeconfig.
- Usa un provider di identità esterno, come Ping Identity, per autenticare gli utenti nell'API Kubernetes.

  - Soluzione: collega un provider OIDC esterno.
- Concedi ai carichi di lavoro sul tuo cluster Kubernetes la possibilità di chiamare. AWS APIs
  - Soluzione: utilizza Pod Identity per associare un ruolo AWS IAM a un account di servizio Kubernetes.

Contesto

- Scopri come funzionano gli account di servizio Kubernetes.
- Esamina il modello di controllo degli accessi basato sui ruoli (Role Based Access Control, RBAC) di Kubernetes
- Per ulteriori informazioni sulla gestione dell'accesso alle AWS risorse, consulta la AWS IAM User Guide. In alternativa, segui un corso introduttivo gratuito sull'uso AWS di IAM.

Considerazioni relative alla modalità automatica di EKS

La modalità automatica di EKS si integra con le voci di accesso EKS Pod Identity ed EKS EKS.

- La modalità automatica di EKS usa le voci di accesso per concedere le autorizzazioni Kubernetes del piano di controllo di EKS. Ad esempio, le policy di accesso consentono alla modalità automatica di EKS di leggere informazioni sugli endpoint e sui servizi di rete.
  - Non puoi disabilitare le voci di accesso su un cluster della modalità automatica di EKS.
  - Facoltativamente, puoi abilitare `aws-auth ConfigMap`.
  - Le voci di accesso per la modalità automatica di EKS sono configurate automaticamente. Puoi visualizzare queste voci di accesso, ma non puoi modificarle.
  - Se utilizzi un NodeClass ruolo Node IAM personalizzato, devi creare una voce di accesso per il ruolo utilizzando la policy di EKSAuto NodePolicy accesso Amazon.
- Se desideri concedere ai carichi di lavoro le autorizzazioni per AWS i servizi, usa EKS Pod Identity.
  - Non è necessario installare l'agente Pod Identity sui cluster della modalità automatica di EKS.

Concedi agli utenti e ai ruoli IAM l'accesso a Kubernetes APIs

Il tuo cluster dispone di un endpoint dell'API Kubernetes. Kubectl utilizza questa API. Puoi autenticarti su questa API usando due tipi di identità:

- Un principale (ruolo o utente) di AWS Identity and Access Management (IAM): questo tipo richiede l'autenticazione su IAM. Gli utenti possono accedere AWS come utenti IAM o con un'identità federata utilizzando le credenziali fornite tramite una fonte di identità. Gli utenti possono accedere solo con un'identità federata se l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando gli utenti accedono AWS utilizzando la federazione, assumono indirettamente un ruolo. Quando gli utenti utilizzano questo tipo di identità:
  - Puoi assegnare loro autorizzazioni Kubernetes in modo che possano lavorare con gli oggetti Kubernetes sul tuo cluster. Per ulteriori informazioni su come assegnare le autorizzazioni ai principali IAM in modo che possano accedere agli oggetti Kubernetes sul cluster, consulta la sezione the section called "Voci di accesso".
  - Puoi assegnare loro le autorizzazioni IAM in modo che possano lavorare con il tuo cluster Amazon EKS e le sue risorse utilizzando l'API Amazon EKS, la AWS CLI,, AWS CloudFormation o. Console di gestione AWS`eksctl` Per ulteriori informazioni, consulta Actions defined by Amazon Elastic Kubernetes Service nella Documentazione di riferimento per l'autorizzazione ai servizi.
  - I nodi si aggiungono al cluster assumendo un ruolo IAM. La capacità di accedere al cluster utilizzando i principali (IAM) è abilitata dall'autenticatore AWS IAM per Kubernetes, che viene eseguito sul piano di controllo di Amazon EKS.
- Un utente del tuo provider OpenID Connect (OIDC): questo tipo richiede l'autenticazione al tuo provider OIDC. Per ulteriori informazioni sulla configurazione del provider OIDC con il cluster Amazon EKS, consulta the section called "Collegare il provider OIDC". Quando gli utenti utilizzano questo tipo di identità:
  - Puoi assegnare loro autorizzazioni Kubernetes in modo che possano lavorare con gli oggetti Kubernetes sul tuo cluster.
  - Non puoi assegnare loro autorizzazioni IAM in modo che possano lavorare con il tuo cluster Amazon EKS e le sue risorse utilizzando l'API Amazon EKS, la AWS CLI,, AWS CloudFormation o. Console di gestione AWS`eksctl`

Puoi utilizzare entrambi i tipi di identità con il tuo cluster. Il metodo di autenticazione IAM non può essere disabilitato. Il metodo di autenticazione OIDC è facoltativo.

Associa le identità IAM alle autorizzazioni Kubernetes

L'Autenticatore AWS IAM per Kubernetes è installato sul piano di controllo del tuo cluster. Abilita i principali AWS Identity and Access Management (ruoli e utenti) a cui consenti di accedere alle risorse

Kubernetes sul tuo cluster. Puoi consentire ai principali IAM di accedere agli oggetti Kubernetes sul tuo cluster utilizzando uno dei metodi seguenti:

- Creazione di voci di accesso: se la versione del cluster corrisponde o è successiva alla versione della piattaforma elencata nella sezione Prerequisiti per la versione del cluster, ti consigliamo di utilizzare questa opzione.

  Utilizza le voci di accesso per gestire le autorizzazioni Kubernetes dei principali IAM dall'esterno del cluster. Puoi aggiungere e gestire l'accesso al cluster utilizzando l'API EKS, l'interfaccia a riga di AWS comando e AWS SDKs. AWS CloudFormation Console di gestione AWS Ciò significa che puoi gestire gli utenti con gli stessi strumenti con cui hai creato il cluster.

  Per iniziare, segui Cambia la modalità di autenticazione per utilizzare le voci di accesso, quindi Migrazione delle voci aws-auth esistenti alle ConfigMap voci di accesso.
- Aggiunta di voci a **`aws-auth` `ConfigMap`**: se la versione della piattaforma del cluster è precedente alla versione elencata nella sezione Prerequisiti, è necessario utilizzare questa opzione. Se la versione della piattaforma del cluster corrisponde o è successiva a quella elencata nella sezione Prerequisiti per la versione di Kubernetes del cluster e hai aggiunto delle voci a `ConfigMap`, ti consigliamo di eseguire la migrazione di tali voci per potervi accedere. Tuttavia, non puoi migrare le voci aggiunte da Amazon EKS a `ConfigMap`, come le voci per i ruoli IAM utilizzati con gruppi di nodi gestiti o profili Fargate. Per ulteriori informazioni, consulta the section called "Accesso API Kubernetes".
  - Se è necessario utilizzare l'opzione `aws-auth ConfigMap`, è possibile aggiungere voci a `ConfigMap` utilizzando il comando `eksctl create iamidentitymapping`. Per ulteriori informazioni, consulta Manage IAM users and roles nella documentazione di `eksctl`.

Set Cluster Authentication Mode

Ogni cluster dispone di una modalità di autenticazione. La modalità di autenticazione determina quali metodi è possibile utilizzare per consentire ai principali IAM di accedere agli oggetti Kubernetes sul cluster. Sono disponibili tre modalità di autenticazione.

> **Important**
>
> Una volta abilitato, il metodo di immissione dell'accesso non può essere disabilitato.

Se il metodo `ConfigMap` non è abilitato durante la creazione del cluster, non può essere abilitato in seguito. Tutti i cluster creati prima dell'introduzione delle voci di accesso hanno il metodo `ConfigMap` abilitato.
Se si utilizzano nodi ibridi con il cluster, è necessario utilizzare le modalità di autenticazione del cluster `API` o `API_AND_CONFIG_MAP`.

Il **`aws-auth` `ConfigMap`** all'interno del cluster

Questa è la modalità di autenticazione originale per i cluster Amazon EKS. Il principale IAM che ha creato il cluster è l'utente iniziale che può accedere al cluster utilizzando `kubectl`. L'utente iniziale deve aggiungere altri utenti all'elenco in `aws-auth` `ConfigMap` e assegnare autorizzazioni che influiscono sugli altri utenti all'interno del cluster. Questi altri utenti non possono gestire o rimuovere l'utente iniziale, poiché non c'è una voce da gestire in `ConfigMap`.

Sia le voci di accesso che quelle di accesso **`ConfigMap`**

Con questa modalità di autenticazione, è possibile utilizzare entrambi i metodi per aggiungere i principali IAM al cluster. Si noti che ogni metodo memorizza voci separate; ad esempio, se si aggiunge una voce di accesso dalla AWS CLI, non `aws-auth` `ConfigMap` viene aggiornata.

Accedi solo alle voci

Con questa modalità di autenticazione, puoi utilizzare l'API EKS, l'interfaccia Console di gestione AWS a riga di AWS comando e gestire l'accesso al cluster per i principali IAM. AWS SDKs AWS CloudFormation

Ogni voce di accesso include un tipo, che consente di utilizzare la combinazione di un ambito di accesso per limitare il principale a un namespace specifico e una policy di accesso per definire policy di autorizzazione riutilizzabili preconfigurate. In alternativa, è possibile utilizzare il tipo STANDARD e i gruppi Kubernetes RBAC per assegnare autorizzazioni personalizzate.

Modalità di autenticazione	Metodi
Solo `ConfigMap` (`CONFIG_MAP`)	`aws-auth` `ConfigMap`
API EKS e `ConfigMap` (`API_AND_CONFIG_MAP`)	accedere alle voci nell'API EKS, AWS nell'interfaccia a riga di comando AWS SDKs AWS

Modalità di autenticazione	Metodi
	CloudFormation, e e Console di gestione AWS `aws-auth ConfigMap`
Solo API EKS (`API`)	accedere alle voci nell'API EKS, AWS nell'interfaccia a riga di comando AWS SDKs, AWS CloudFormation, e Console di gestione AWS

> **Note**
>
> La modalità automatica di Amazon EKS richiede le voci di accesso.

Concedere agli utenti IAM l'accesso a Kubernetes con le voci di accesso EKS

Questa sezione è progettata per mostrarti come gestire l'accesso principale IAM ai cluster di Kubernetes in Amazon Elastic Kubernetes Service (EKS) utilizzando voci di accesso e policy. Sono disponibili dettagli sulla modifica delle modalità di autenticazione, la migrazione da voci `aws-auth` ConfigMap precedenti, la creazione, l'aggiornamento e l'eliminazione delle voci di accesso, l'associazione delle policy alle voci, la revisione delle autorizzazioni delle policy predefinite e i prerequisiti e le considerazioni chiave per la gestione sicura degli accessi.

Panoramica

Le voci di accesso EKS sono il modo migliore per concedere agli utenti l'accesso all'API Kubernetes. Ad esempio, puoi utilizzare le voci di accesso per concedere agli sviluppatori l'accesso all'uso di kubectl. Fondamentalmente, una voce di accesso EKS associa un set di autorizzazioni di Kubernetes a un'identità IAM, ad esempio un ruolo IAM. Ad esempio, uno sviluppatore può assumere un ruolo IAM e utilizzarlo per autenticarsi in un cluster EKS.

Funzionalità

- Autenticazione e autorizzazione centralizzate: controlla l'accesso ai cluster di Kubernetes direttamente tramite le API di Amazon EKS, eliminando la necessità di passare fra AWS e le API Kubernetes per le autorizzazioni utente.

- Gestione granulare delle autorizzazioni: utilizza le voci e le policy di accesso per definire autorizzazioni granulari per i principali IAM AWS, inclusa la modifica o la revoca dell'accesso da amministratore del cluster dal creatore.
- Integrazione con strumenti IaC: supporta l'infrastruttura sotto forma di strumenti di codice come AWS CloudFormation, Terraform e AWS CDK per definire le configurazioni di accesso durante la creazione del cluster.
- Ripristino della configurazione errata: consente di ripristinare l'accesso al cluster tramite l'API Amazon EKS senza accesso diretto all'API Kubernetes.
- Sovraccarico ridotto e sicurezza avanzata: centralizza le operazioni per ridurre il sovraccarico sfruttando al contempo funzionalità IAM AWS come la registrazione degli audit CloudTrail e l'autenticazione a più fattori.

Come allegare le autorizzazioni

Puoi allegare le autorizzazioni Kubernetes per accedere alle voci in due modi:

- Utilizzare una policy di accesso. Le policy di accesso sono modelli di autorizzazioni di Kubernetes predefiniti gestiti da AWS. Per ulteriori informazioni, consulta the section called "Rivedere le policy di accesso".
- Fare riferimento a un gruppo di Kubernetes. Se associ un'identità IAM a un gruppo di Kubernetes, puoi creare risorse di Kubernetes che concedono le autorizzazioni al gruppo. Per ulteriori informazioni, consulta Using RBAC Authorization nella documentazione di Kubernetes.

Considerazioni

Quando si abilitano le voci di accesso EKS sui cluster esistenti, tenere presente quanto segue:

- Comportamento legacy dei cluster: per i cluster creati prima dell'introduzione delle voci di accesso (quelli con versioni iniziali della piattaforma precedenti a quelle specificate in Platform version requirements), EKS crea automaticamente una voce di accesso che riflette le autorizzazioni preesistenti. Questa voce include l'identità IAM che ha originariamente creato il cluster e le autorizzazioni amministrative concesse a tale identità durante la creazione del cluster.
- Gestione di **`aws-auth`** ConfigMap legacy: se il cluster si basa su `aws-auth` ConfigMap legacy per la gestione degli accessi, dopo aver abilitato le voci di accesso, viene creata automaticamente solo la voce di accesso per il creatore del cluster originale. I ruoli o le autorizzazioni aggiuntivi aggiunti a ConfigMap (ad esempio, ruoli IAM personalizzati per sviluppatori o servizi) non vengono

migrati automaticamente. Per risolvere questo problema, creare manualmente le voci di accesso corrispondenti.

Inizia a usare

1. Determina l'identità IAM e la policy di accesso che desideri utilizzare.
   - the section called "Rivedere le policy di accesso"
2. Abilita le voci di accesso EKS sul cluster. Conferma di avere una versione della piattaforma supportata.
   - the section called "Modalità di autenticazione"
3. Creare una voce di accesso che associ un'identità IAM all'autorizzazione di Kubernetes.
   - the section called "Creare voci di accesso"
4. Effettuare l'autenticazione nel cluster utilizzando l'identità IAM.
   - the section called "Configurazione di AWS CLI"
   - the section called "Configura kubectl ed eksctl"

Associare le policy di accesso alle voci di accesso

È possibile assegnare una o più policy di accesso alle voci di accesso di tipo `STANDARD`. Amazon EKS concede automaticamente agli altri tipi di voci di accesso le autorizzazioni necessarie per funzionare correttamente nel cluster. Le policy di accesso di Amazon EKS includono le autorizzazioni di Kubernetes, non le autorizzazioni IAM. Prima di associare una policy di accesso a una voce di accesso, assicurati di conoscere le autorizzazioni di Kubernetes incluse in ogni policy di accesso. Per ulteriori informazioni, consulta the section called "Rivedere le policy di accesso". Se nessuna delle policy di accesso soddisfa i requisiti, non associarne alcuna a una voce di accesso. Specificare invece uno o più nomi dei gruppi per la voce di accesso, e creare e gestire oggetti di controllo degli accessi basato sui ruoli di Kubernetes. Per ulteriori informazioni, consulta the section called "Creare voci di accesso".

- Una voce di accesso esistente. Per crearne uno, consulta the section called "Creare voci di accesso".
- Un ruolo o un utente AWS Identity and Access Management con le seguenti autorizzazioni: `ListAccessEntries`, `DescribeAccessEntry`, `UpdateAccessEntry`, `ListAccessPolicies`, `AssociateAccessPolicy` e `DisassociateAccessPolicy`. Per

ulteriori informazioni, consulta Actions defined by Amazon Elastic Kubernetes Service in Service Authorization Reference.

Prima di associare policy di accesso a voci di accesso, considera i seguenti requisiti:

- È possibile associare più policy di accesso a ciascuna voce di accesso, ma è possibile associare ogni policy a una voce di accesso solo una volta. Se associ più policy di accesso, il principale IAM della voce di accesso dispone di tutte le autorizzazioni incluse in tutte le policy di accesso associate.
- Puoi applicare una policy di accesso a tutte le risorse di un cluster o specificando il nome di uno o più namespace di Kubernetes. Puoi utilizzare caratteri jolly per il nome di un namespace. Ad esempio, se desideri applicare una policy di accesso a tutti i namespace che iniziano con `dev-`, puoi specificare `dev-*` come nome del namespace. Assicurati che i namespace siano presenti sul cluster e che quanto digitato corrisponda esattamente al nome reale del namespace sul cluster. Amazon EKS, infatti, non conferma l'ortografia o l'esistenza dei namespace sul cluster.
- È possibile modificare l'ambito di accesso per una policy di accesso dopo averla associata a una voce di accesso. Se hai limitato l'ambito della policy di accesso ai namespace di Kubernetes, puoi aggiungere e rimuovere i namespace per l'associazione, secondo necessità.
- Se associ una policy di accesso a una voce di accesso in cui sono specificati anche i nomi dei gruppi, il principale IAM dispone di tutte le autorizzazioni in tutte le policy di accesso associate. Inoltre, dispone di tutte le autorizzazioni in qualsiasi oggetto `Role` o `ClusterRole` di Kubernetes specificato in qualsiasi oggetto `Role` e `RoleBinding` di Kubernetes che specifica i nomi dei gruppi.
- Se esegui il comando `kubectl auth can-i --list`, non vedrai alcun permesso di Kubernetes assegnato dalle policy di accesso associate a una voce di accesso per il principale IAM che stai utilizzando quando viene eseguito il comando. Il comando mostra le autorizzazioni di Kubernetes solo se le hai concesse a oggetti `Role` o `ClusterRole` di Kubernetes associati ai nomi dei gruppi o al nome utente specificato per una voce di accesso.
- Se impersoni un utente o un gruppo di Kubernetes durante l'interazione con oggetti di Kubernetes sul cluster, ad esempio utilizzando il comando `kubectl` con `--as` *`username`* o `--as-group` *`group-name`* , stai forzando l'uso dell'autorizzazione RBAC di Kubernetes. Di conseguenza, il principale IAM non dispone di autorizzazioni assegnate da alcuna policy di accesso associata alla voce di accesso. Le uniche autorizzazioni di Kubernetes di cui dispone l'utente o il gruppo che il principale IAM sta impersonando sono quelle di Kubernetes che hai concesso negli oggetti `Role` o `ClusterRole` di Kubernetes associati ai nomi dei gruppi o al nome utente. Per garantire che il

principale IAM abbia le autorizzazioni nelle policy di accesso associate, evitare di impersonare un utente o un gruppo di Kubernetes. Il principale IAM avrà comunque anche tutte le autorizzazioni concesse negli oggetti `Role` o `ClusterRole` di Kubernetes associati ai nomi dei gruppi o al nome utente specificato per la voce di accesso. Per ulteriori informazioni, consulta User impersonation nella documentazione di Kubernetes.

Puoi associare una policy di accesso a una voce di accesso utilizzando Console di gestione AWS o AWS CLI.

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster con una voce di accesso da associare a una policy di accesso.
3. Scegliere la scheda Accesso.
4. Se il tipo di voce di accesso è Standard, puoi associare o dissociare le policy di accesso di Amazon EKS. Se il tipo di voce di accesso è diverso da Standard, questa opzione non è disponibile.
5. Scegliere Associa policy di accesso.
6. Per Nome della policy, seleziona la policy con le autorizzazioni che desideri assegnare al principale IAM. Per visualizzare i permessi inclusi in ciascuna policy, consulta the section called "Rivedere le policy di accesso".
7. Per Ambito di accesso, scegli un ambito di accesso. Se scegli Cluster, le autorizzazioni nella policy di accesso vengono concesse al principale IAM per le risorse in tutti i namespace di Kubernetes. Se scegli Namespace di Kubernetes, puoi scegliere Aggiungi nuovo namespace. Nel campo Namespace che appare, puoi inserire il nome di un namespace di Kubernetes sul cluster. Se desideri che il principale IAM disponga delle autorizzazioni su più namespace, puoi inserirne più di uno.
8. Scegliere Aggiungi policy di accesso.

CLI AWS

1. La versione `2.12.3` o successive oppure `1.27.160` o successive dell'interfaccia a riga di comando AWS (AWS CLI) installata e configurata sul dispositivo o in AWS CloudShell. Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I programmi di gestione dei pacchetti, come `yum`, `apt-get` o Homebrew per macOS, spesso sono aggiornati a versioni precedenti di AWS CLI. Per installare la versione più recente, consulta

Installing e Quick configuration with aws configure nella Guida per l'utente dell'interfaccia a riga di comando AWS. La versione della AWS CLI installata in AWS CloudShell potrebbe anche essere precedente di diverse versioni rispetto alla più recente. Per aggiornarla, consulta Installing AWS CLI to your home directory nella Guida per l'utente di AWS CloudShell.

2. Visualizza le policy di accesso disponibili.

```
aws eks list-access-policies --output table
```

Di seguito viene riportato un output di esempio.

```
-----------------------------------------------------------------------------------------------
|                                          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          ||
|+-----------------------------------------------------------------
+------------------------------+|
```

Per visualizzare i permessi inclusi in ciascuna policy, consulta the section called "Rivedere le policy di accesso".

3. Visualizza le voci di accesso esistenti. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks list-access-entries --cluster-name my-cluster
```

Di seguito viene riportato un output di esempio.

```
{
    "accessEntries": [
        "arn:aws:iam::111122223333:role/my-role",
        "arn:aws:iam::111122223333:user/my-user"
    ]
}
```

4. Associa una policy di accesso a una voce di accesso. L'esempio seguente associa la policy di accesso `AmazonEKSViewPolicy` a una voce di accesso. Ogni volta che il ruolo IAM *`my-role`* tenta di accedere agli oggetti di Kubernetes sul cluster, Amazon EKS autorizzerà il ruolo a utilizzare le autorizzazioni della policy per accedere agli oggetti di Kubernetes solo nei namespace *`my-namespace1`* e *`my-namespace2`* di Kubernetes. Sostituisci *`my-cluster`* con il nome del cluster, *`111122223333`* con l'ID dell'account AWS e *`my-role`* con il nome del ruolo IAM che desideri che Amazon EKS autorizzi ad accedere agli oggetti del cluster di 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 desideri che il principale IAM disponga delle autorizzazioni a livello di cluster, sostituisci `type=namespace,namespaces=`*`my-namespace1`*`,`*`my-namespace2`* con `type=cluster`. Se desideri associare più policy di accesso alla voce di accesso, esegui il comando più volte, ogni volta con una policy di accesso unica. Ogni policy di accesso associata ha il proprio ambito.

> **Note**
>
> Se in seguito desideri modificare l'ambito di una policy di accesso associata, esegui nuovamente il comando precedente con il nuovo ambito. Ad esempio, se desideri rimuovere *`my-namespace2`*, eseguiresti nuovamente il comando utilizzando solo `type=namespace,namespaces=`*`my-namespace1`* . Se desidera modificare l'ambito da `namespace` a `cluster`, esegui nuovamente il comando utilizzando `type=cluster` e rimuovendo `type=namespace,namespaces=`*`my-namespace1`*`,`*`my-namespace2`* .

5. Determina quali policy di accesso sono associate a una voce di accesso.

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

Di seguito viene riportato un output di esempio.

```
{
    "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"
        }
    ]
}
```

Nell'esempio precedente, il principale IAM per questa voce di accesso ha permessi di visualizzazione in tutti i namespace del cluster e permessi di amministratore per due namespace di Kubernetes.

6. Dissociazione di una policy di accesso da una voce di accesso. In questo esempio, la policy `AmazonEKSAdminPolicy` è dissociata da una voce di accesso. Tuttavia, il principale IAM

mantiene le autorizzazioni nella policy di accesso di `AmazonEKSViewPolicy` per gli oggetti nei namespace *`my-namespace1`* e *`my-namespace2`*, poiché tale policy di accesso non è dissociata dalla voce di accesso.

```
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
```

Per elencare le policy di accesso disponibili, consultare the section called "Rivedere le policy di accesso".

Migrazione delle voci **`aws-auth ConfigMap`** esistenti alle voci di accesso

Se hai aggiunto voci a `aws-auth ConfigMap` sul tuo cluster, ti consigliamo di creare voci di accesso per quelle esistenti in `aws-auth ConfigMap`. Dopo aver creato le voci di accesso, puoi rimuovere le voci da `ConfigMap`. Non puoi associare le policy di accesso alle voci presenti in `aws-auth ConfigMap`. Se desideri associare policy di accesso ai tuoi principali IAM, crea voci di accesso.

> **Important**
>
> - Quando un cluster è in modalità di autenticazione `API_AND_CONFIGMAP` ed è presente una mappatura per lo stesso ruolo IAM sia in `aws-auth ConfigMap` che nelle voci di accesso, il ruolo utilizzerà la mappatura della voce di accesso per l'autenticazione. Le voci di accesso hanno la precedenza sulle voci `ConfigMap` dello stesso principale IAM.
> - Prima di rimuovere le voci `aws-auth ConfigMap` esistenti create da Amazon EKS per un gruppo di nodi gestiti o un profilo Fargate nel tuo cluster, ricontrolla se nel tuo cluster Amazon EKS esistono le voci di accesso corrette per quelle risorse specifiche. Se rimuovi le voci che Amazon EKS ha creato in `ConfigMap` senza avere le voci di accesso equivalenti, il cluster non funzionerà correttamente.

Prerequisiti

- Familiarità con le voci di accesso e le policy di accesso. Per ulteriori informazioni, consulta the section called "Voci di accesso" e the section called "Associare le policy di accesso".
- Un cluster esistente con una versione della piattaforma corrispondente o successiva alle versioni elencate nei Prerequisiti dell'argomento the section called "Voci di accesso".

- La versione `0.215.0` o quelle successive dello strumento a riga di comando `eksctl` deve essere installata sul dispositivo o nella AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
- Le autorizzazioni di Kubernetes per modificare `aws-auth ConfigMap` nel namespace `kube-system`.
- Un ruolo o un utente di AWS Identity and Access Management con le seguenti autorizzazioni: `CreateAccessEntry` `eListAccessEntries`. Per ulteriori informazioni, consulta Actions defined by Amazon Elastic Kubernetes Service nella Documentazione di riferimento per l'autorizzazione ai servizi.

`eksctl`

1. Verifica le voci esistenti in `aws-auth ConfigMap`. Sostituisci *`my-cluster`* con il nome del cluster.

```
eksctl get iamidentitymapping --cluster my-cluster
```

Di seguito viene riportato un output di esempio.

```
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 "Creare voci di accesso" per tutte le voci `ConfigMap` che hai creato e che sono state restituite nell'output precedente. Quando crei le voci di accesso, assicurati di specificare gli stessi valori per `ARN`, `USERNAME`, `GROUPS` e `ACCOUNT` restituiti nell'output. Nell'output di esempio,

dovresti creare voci di accesso per tutte le voci tranne le ultime due, poiché tali voci sono state create da Amazon EKS per un profilo Fargate e un gruppo di nodi gestito.

3. Elimina le voci da `ConfigMap` per tutte le voci di accesso che hai creato. Se non elimini la voce da `ConfigMap`, le impostazioni per la voce di accesso per l'ARN principale IAM hanno la precedenza sulla voce `ConfigMap`. Sostituiscilo *`111122223333`* con l'ID del tuo AWS account e *`EKS-my-cluster-my-namespace-Viewers`* con il nome del ruolo nella voce del tuo`ConfigMap`. Se la voce che stai rimuovendo riguarda un utente IAM, anziché un ruolo IAM, `role` sostituiscila con `user` e *`EKS-my-cluster-my-namespace-Viewers`* con il nome utente.

```
eksctl delete iamidentitymapping --arn arn:aws: iam::111122223333:role/EKS-my-
cluster-my-namespace-Viewers --cluster my-cluster
```

Rivedere le autorizzazioni della policy di accesso

Le policy di accesso includono `rules` che contengono `verbs` di Kubernetes (autorizzazioni) e `resources`. Le policy di accesso non includono le autorizzazioni o le risorse IAM. Analogamente agli oggetti `Role` e `ClusterRole` di Kubernetes, le policy di accesso includono solo `allow` `rules`. Non puoi modificare il contenuto di una policy di accesso. Non puoi creare policy di accesso personalizzate. Se le autorizzazioni nelle policy di accesso non soddisfano le tue esigenze, crea oggetti RBAC di Kubernetes e specificare nomi dei gruppi per le voci di accesso. Per ulteriori informazioni, consulta the section called "Creare voci di accesso". Le autorizzazioni contenute nelle policy di accesso sono simili a quelle nei ruoli del cluster rivolti agli utenti di Kubernetes. Per ulteriori informazioni, consulta User-facing roles nella documentazione di Kubernetes.

Elencare tutte le policy

Utilizza una delle politiche di accesso elencate in questa pagina o recupera un elenco di tutte le politiche di accesso disponibili utilizzando la AWS CLI:

```
aws eks list-access-policies
```

L'output previsto dovrebbe essere simile al seguente (abbreviato per brevità):

```
{
    "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
    ]
}
```

EKSAdminPolitica di Amazon

Questa policy di accesso include autorizzazioni che concedono a un principale IAM la maggior parte delle autorizzazioni per le risorse. Quando è associato a una voce di accesso, il relativo ambito di accesso è in genere uno o più namespace di Kubernetes. Se invece desideri che un principale IAM disponga dell'accesso di amministratore a tutte le risorse del cluster, allora associa la policy di accesso the section called "Amazon EKSCluster AdminPolicy" alla voce di accesso.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminPolicy`

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
`apps`	`daemonsets` , `deployments` , `deployments/rollback` , `deployments/scale` , `replicasets` , `replicasets/`	`create`, `delete`, `deletecollection` , `patch`, `update`

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
	scale, statefulsets , statefulsets/scale	
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
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

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
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
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

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizz azioni)
rbac.authorization .k8s.io	rolebindings , roles	create, delete, deletecol lection , get, list, patch, update, watch
	configmaps , endpoints , persistentvolumecl aims , persisten tvolumeclaims/stat us , pods, replicati oncontrollers , replicationcontrol lers/scale , serviceac counts , services, services/status	get,list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	configmaps , events, persistentvolumecl aims , replicati oncontrollers , replicationcontrol lers/scale , secrets, serviceaccounts , services, services/ proxy	create, delete, deletecol lection , patch, update
	pods, pods/attach , pods/exec , pods/port forward , pods/proxy	create, delete, deletecol lection , patch, update

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
	`serviceaccounts`	`impersonate`
	`bindings`, `events`, `limitranges` , `namespaces/status` , `pods/log`, `pods/status` , `replicationcontrollers/status` , `resourcequotas` , `resourcequotas/status`	`get`, `list`, `watch`
	`namespaces`	`get`,`list`, `watch`

Amazon EKSCluster AdminPolicy

Questa policy di accesso include le autorizzazioni che concedono a un principale IAM l'accesso di amministratore per un cluster. Quando è associato a una voce di accesso, il relativo ambito di accesso è in genere il cluster, anziché un namespace di Kubernetes. Se desideri che un principale IAM abbia un ambito amministrativo più limitato, valuta invece la possibilità di associare la policy di accesso the section called "EKSAdminPolitica di Amazon" alla voce di accesso.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy`

Gruppi di API Kubernetes	Kubernetes NonResource URLs	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
*		*	*
	*		*

Amazon EKSAdmin ViewPolicy

Questa politica di accesso include autorizzazioni che garantiscono un accesso principale IAM a list/view tutte le risorse di un cluster. Questo include Kubernetes Secrets.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy`

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizz azioni)
*	*	get, list, watch

EKSEditPolitica di Amazon

Questa policy di accesso include autorizzazioni che consentono a un principale IAM di modificare la maggior parte delle risorse di Kubernetes.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSEditPolicy`

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizz azioni)
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

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
autoscaling	horizontalpodautoscalers, horizontalpodautoscalers/status	get, list, watch
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

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizz azioni)
extensions	daemonsets , daemonset s/status , deploymen ts , deployments/scale , deployments/status , ingresses , ingresses /status , networkpo licies , replicasets , replicasets/scale , replicasets/status , replicationcontrol lers/scale	get, list, watch
networking.k8s.io	ingresses , networkpo licies	create, delete, deletecol lection , patch, update
networking.k8s.io	ingresses , ingresses /status , networkpo licies	get, list, watch
policy	poddisruptionbudgets	create, delete, deletecol lection , patch, update
policy	poddisruptionbudgets , poddisruptionbudge ts/status	get, list, watch
	namespaces	get, list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	serviceaccounts	impersonate

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizz azioni)
	pods, pods/attach , pods/exec , pods/port forward , pods/proxy	create, delete, deletecol lection , patch, update
	configmaps , events, persistentvolumecl aims , replicati oncontrollers , replicationcontrol lers/scale , secrets, serviceaccounts , services, services/ proxy	create, delete, deletecol lection , patch, update
	configmaps , endpoints , persistentvolumecl aims , persisten tvolumeclaims/stat us , pods, replicati oncontrollers , replicationcontrol lers/scale , serviceac counts , services, services/status	get, list, watch
	bindings, events, limitranges , namespace s/status , pods/log, pods/status , replicati oncontrollers/stat us , resourcequotas , resourcequotas/sta tus	get, list, watch

EKSViewPolitica di Amazon

Questa policy di accesso include autorizzazioni che consentono a un principale IAM di visualizzare la maggior parte delle risorse di Kubernetes.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy`

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
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 , networkpo	get, list, watch

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
	licies , replicasets , replicasets/scale , replicasets/status , 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

Amazon EKSSecret ReaderPolicy

Questa politica di accesso include autorizzazioni che consentono a un preside IAM di leggere Kubernetes Secrets.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSSecretReaderPolicy`

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
	`secrets`	`get`, `list`, `watch`

Amazon EKSAuto NodePolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy`

Questa policy include le seguenti autorizzazioni che consentono ai componenti di Amazon EKS di completare le attività seguenti:

- `kube-proxy`: monitorare gli endpoint e i servizi di rete e gestire gli eventi correlati. Ciò consente la funzionalità proxy di rete a livello di cluster.
- `ipamd`— Gestione delle risorse di rete AWS VPC e delle interfacce di rete container (CNI). Ciò consente al daemon di gestione degli indirizzi IP di gestire il networking dei pod.
- `coredns`: accedere a risorse di rilevamento servizi come endpoint e servizi. Ciò consente la risoluzione DNS all'interno del cluster.
- `ebs-csi-driver`: lavorare con risorse relative all'archiviazione per i volumi Amazon EBS. Ciò consente il provisioning e il collegamento dinamico di volumi persistenti.
- `neuron`— Monitora nodi e pod per i dispositivi Neuron. AWS Ciò consente la gestione degli acceleratori AWS Inferentia e Trainium.
- `node-monitoring-agent`: accedere alla diagnostica dei nodi e agli eventi. Ciò consente il monitoraggio dell'integrità del cluster e la raccolta della diagnostica.

Ogni componente utilizza un account di servizio dedicato ed è limitato alle sole autorizzazioni necessarie per la sua funzione specifica.

Se specifichi manualmente un ruolo Node IAM in a NodeClass, devi creare un Access Entry che associ il nuovo ruolo Node IAM a questa policy di accesso.

Amazon EKSBlock StoragePolicy

> **Note**
>
> Questa politica è destinata solo ai ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSBlockStoragePolicy`

Questa policy include le autorizzazioni che consentono ad Amazon EKS di gestire le risorse per l'elezione e il coordinamento dei leader per le operazioni di archiviazione:

- `coordination.k8s.io`: creare e gestire oggetti in leasing per l'elezione dei leader. Ciò consente ai componenti di archiviazione EKS di coordinare le proprie attività all'interno del cluster attraverso un meccanismo di elezione dei leader.

La policy si applica a specifiche risorse di leasing utilizzate dai componenti di archiviazione EKS per prevenire conflitti di accesso ad altre risorse di coordinamento del cluster.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di storage a blocchi.

Amazon EKSLoad BalancingPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSLoadBalancingPolicy`

Questa policy include le autorizzazioni che consentono ad Amazon EKS di gestire le risorse per le elezioni per i leader per il bilanciamento del carico:

- `coordination.k8s.io`: creare e gestire oggetti in leasing per l'elezione dei leader. Ciò consente ai componenti di bilanciamento del carico EKS di coordinare le attività su più repliche eleggendo un leader.

La policy riguarda specificamente il bilanciamento del carico delle risorse di leasing per garantire un coordinamento adeguato impedendo al contempo l'accesso ad altre risorse di leasing del cluster.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di rete.

EKSNetworkingPolitica di Amazon

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSNetworkingPolicy`

Questa policy include le autorizzazioni che consentono ad Amazon EKS di gestire le risorse per le elezioni dei leader per le reti:

- `coordination.k8s.io`: creare e gestire oggetti in leasing per l'elezione dei leader. Ciò consente ai componenti di rete EKS di coordinare le attività di allocazione degli indirizzi IP eleggendo un leader.

La policy riguarda specificamente le reti delle risorse di leasing per garantire un coordinamento adeguato impedendo al contempo l'accesso ad altre risorse di leasing del cluster.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di rete.

EKSComputePolitica di Amazon

> **Note**
>
> Questa politica è destinata solo ai ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSComputePolicy`

Questa policy include le autorizzazioni che consentono ad Amazon EKS di gestire le risorse per l'elezione dei leader per le operazioni di elaborazione:

- `coordination.k8s.io`: creare e gestire oggetti in leasing per l'elezione dei leader. Ciò consente ai componenti di elaborazione EKS di coordinare le attività di scalabilità dei nodi eleggendo un leader.

La policy riguarda specificamente la gestione delle risorse di elaborazione in leasing e consente al contempo l'accesso di base in lettura (`get`, `watch`) a tutte le risorse di leasing del cluster.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di rete.

Amazon EKSBlock StorageClusterPolicy

> **Note**
>
> Questa politica è destinata solo ai ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSBlockStorageClusterPolicy`

Questa policy concede le autorizzazioni necessarie per la funzionalità di storage a blocchi della modalità automatica di Amazon EKS. Consente una gestione efficiente delle risorse di archiviazione a blocchi all'interno dei cluster Amazon EKS. La policy include le seguenti autorizzazioni:

Gestione del driver CSI:

- Creare, leggere, aggiornare ed eliminare i driver CSI, in particolare per l'archiviazione a blocchi.

Gestione dei volumi:

- Elencare, guardare, creare, aggiornare, applicare patch ed eliminare i volumi persistenti.
- Elencare, guardare e aggiornare le dichiarazioni di volume persistenti.
- Applicare patch agli stati di dichiarazione di volume persistente.

Interazione tra nodi e pod:

- Leggere le informazioni su nodi e pod.
- Gestire gli eventi relativi alle operazioni di archiviazione.

Classi e attributi di archiviazione:

- Leggere le classi di archiviazione e i nodi CSI.
- Leggere le classi di attributi del volume.

Allegati dei volumi:

- Elencare, guardare e modificare gli allegati del volume e il relativo stato.

Operazioni di snapshot:

- Gestire snapshot di volume, contenuti di snapshot e classi di snapshot.
- Gestire le operazioni per gli snapshot dei gruppi di volumi e le risorse correlate.

Questa policy è progettata per supportare la gestione completa dell'archiviazione a blocchi all'interno dei cluster Amazon EKS in esecuzione in Auto Mode. Combina le autorizzazioni per varie operazioni, tra cui il provisioning, il collegamento, il ridimensionamento e la creazione di snapshot di volumi di archiviazione a blocchi.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di storage a blocchi.

Amazon EKSCompute ClusterPolicy

> **Note**
>
> Questa politica è destinata solo ai ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSComputeClusterPolicy`

Questa policy concede le autorizzazioni necessarie per la funzionalità di gestione dell'elaborazione della modalità automatica di Amazon EKS. Consente l'orchestrazione e la scalabilità efficienti delle risorse di elaborazione all'interno dei cluster Amazon EKS. La policy include le seguenti autorizzazioni:

Gestione dei nodi:

- Crea, leggi, aggiorna, elimina e gestisci lo stato di e. NodePools NodeClaims

- Gestione NodeClasses, inclusa la creazione, la modifica e l'eliminazione.

Pianificazione e gestione delle risorse:

- Accesso in lettura a pod, nodi, volumi persistenti, attestazioni di volume persistenti, controller di replica e namespace.
- Accesso in lettura alle classi di archiviazione, ai nodi CSI e agli allegati dei volumi.
- Elencare e controllare implementazioni, set di daemon, set di repliche e set stateful.
- Leggere i budget di interruzione dei pod.

Gestione degli eventi:

- Creare, leggere e gestire gli eventi del cluster.

Revoca del provisioning dei nodi ed espulsione dei pod:

- Aggiornare, applicare patch ed eliminare i nodi.
- Creare espulsioni dei pod ed eliminarli quando necessario.

Gestione di definizione di risorse personalizzate (CRD):

- Crea nuovo CRDs.
- Gestisci specifiche CRDs relative alla gestione dei nodi (NodeClasses NodePools, NodeClaims, e NodeDiagnostics).

Questa policy è progettata per supportare la gestione completa dell'elaborazione all'interno dei cluster Amazon EKS in esecuzione in Auto Mode. Combina le autorizzazioni per varie operazioni, tra cui il provisioning dei nodi, la pianificazione, la scalabilità e l'ottimizzazione delle risorse.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di gestione di elaborazione.

Amazon EKSLoad BalancingClusterPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSLoadBalancingClusterPolicy`

Questa policy concede le autorizzazioni necessarie per la funzionalità di bilanciamento del carico della modalità automatica di Amazon EKS. Consente una configurazione e una gestione efficiente delle risorse di bilanciamento del carico all'interno dei cluster Amazon EKS. La policy include le seguenti autorizzazioni:

Gestione di eventi e risorse:

- Creare e applicare patch agli eventi.
- Accesso in lettura a pod, nodi, endpoint e namespace.
- Aggiornare gli stati dei pod.

Gestione dei servizi e degli ingressi:

- Gestione completa dei servizi e dei relativi stati.
- Controllo completo sugli ingressi e sui relativi stati.
- Accesso in lettura alle sezioni degli endpoint e alle classi di ingresso.

Associazioni per gruppi di destinazione:

- Creare e modificare le associazioni per gruppi di destinazione e i relativi stati.
- Accesso in lettura ai parametri della classe di ingresso.

Gestione di definizione di risorse personalizzate (CRD):

- Crea e leggi tutto CRDs.
- Gestione specifica di targetgroupbindings.eks.amazonaws.com e ingressclassparams.eks.amazonaws.com. CRDs

Configurazione Webhook:

- Creare e leggere configurazioni webhook mutanti e convalidanti.
- eks-load-balancing-webhookGestisci la configurazione.

Questa policy è progettata per supportare la gestione completa del bilanciamento del carico all'interno dei cluster Amazon EKS in esecuzione in Auto Mode. Combina le autorizzazioni per

varie operazioni, tra cui esposizione del servizio, routing in ingresso e integrazione con i servizi di bilanciamento AWS del carico.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di bilanciamento del carico.

Amazon EKSNetworking ClusterPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSNetworkingClusterPolicy`

Amazon EKSNetworking ClusterPolicy

Questa policy concede le autorizzazioni necessarie per la funzionalità di rete della modalità automatica di Amazon EKS. Consente una configurazione e una gestione efficiente delle risorse di rete all'interno dei cluster Amazon EKS. La policy include le seguenti autorizzazioni:

Gestione di nodi e pod:

- Leggi gli accessi NodeClasses e i relativi stati.
- Leggi gli accessi NodeClaims e i relativi stati.
- Accesso in lettura ai pod.

Gestione dei nodi CNI:

- Autorizzazioni CNINodes e relativi stati, tra cui creazione, lettura, aggiornamento, eliminazione e patch.

Gestione di definizione di risorse personalizzate (CRD):

- Crea e leggi tutto. CRDs
- Gestione specifica (aggiornamento, applicazione di patch, eliminazione) della CRD cninodes.eks.amazonaws.com.

Gestione degli eventi:

- Creare e applicare patch agli eventi.

Questa policy è progettata per supportare la gestione completa della rete all'interno dei cluster Amazon EKS in esecuzione in Auto Mode. Combina le autorizzazioni per varie operazioni, tra cui la configurazione di rete dei nodi, la gestione della CNI (Container Network Interface) e la relativa gestione personalizzata delle risorse.

La policy consente ai componenti di rete di interagire con le risorse relative ai nodi, gestire le configurazioni dei nodi specifiche per CNI e gestire le risorse personalizzate fondamentali per le operazioni di rete nel cluster.

Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per il ruolo IAM del cluster quando Auto Mode è abilitata, garantendo che siano disponibili le autorizzazioni necessarie per il corretto funzionamento della capacità di rete.

EKSHybridPolitica di Amazon

> **Note**
>
> Questa politica è destinata solo ai ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

Questa policy di accesso include le autorizzazioni che concedono a EKS l'accesso di EKS ai nodi di un cluster. Quando è associato a una voce di accesso, il relativo ambito di accesso è in genere il cluster, anziché un namespace di Kubernetes. Questa policy viene utilizzata da Amazon EKS Hybrid Nodes.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSHybridPolicy`

Gruppi di API Kubernetes	Kubernetes NonResource URLs	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
*		nodes	list

Amazon EKSCluster InsightsPolicy

> **Note**
>
> Questa politica è destinata solo ai ruoli AWS collegati ai servizi e non può essere utilizzata con ruoli gestiti dal cliente.

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSClusterInsightsPolicy`

Questa policy concede le autorizzazioni di sola lettura alla funzionalità di approfondimenti del cluster di Amazon EKS. La policy include le seguenti autorizzazioni:

Accesso ai nodi: - Elencare e visualizzare i nodi del cluster - Leggere le informazioni sullo stato dei nodi

DaemonSet Accesso: - Accesso in lettura alla configurazione kube-proxy

Questa policy viene gestita automaticamente dal servizio EKS per gli approfondimenti del cluster. Per ulteriori informazioni, consulta the section called "Approfondimenti sui cluster".

AWSBackupFullAccessPolicyForBackup

ARN: `arn:aws: eks::aws:cluster-access-policy/AWSBackupFullAccessPolicyForBackup`

AWSBackupFullAccessPolicyForBackup

Questa politica concede le autorizzazioni necessarie a AWS Backup per gestire e creare backup del cluster EKS. Questa policy include le seguenti autorizzazioni:

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
*	*	`list`, `get`

AWSBackupFullAccessPolicyForRestore

ARN: `arn:aws: eks::aws:cluster-access-policy/AWSBackupFullAccessPolicyForRestore`

AWSBackupFullAccessPolicyForRestore

Questa politica concede le autorizzazioni necessarie a AWS Backup per gestire e ripristinare i backup del cluster EKS. Questa policy include le seguenti autorizzazioni:

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
*	*	`list`, `get`, `create`

Amazon EKSACKPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSACKPolicy`

Questa politica concede le autorizzazioni necessarie per consentire AWS ai Controllers for Kubernetes (ACK) di gestire le risorse di Kubernetes. AWS La policy include le seguenti autorizzazioni:

Gestione personalizzata delle risorse ACK:

- Accesso completo a tutte le risorse personalizzate del servizio ACK su oltre 50 AWS servizi tra cui S3, RDS, DynamoDB, Lambda, EC2 e altri.
- Crea, leggi, aggiorna ed elimina le definizioni delle risorse personalizzate ACK.

Accesso al namespace:

- Accesso in lettura ai namespace per l'organizzazione delle risorse.

Elezione del leader:

- Crea e leggi i contratti di coordinamento per l'elezione del leader.
- Aggiorna ed elimina i leasing specifici dei controller di servizio ACK.

Gestione degli eventi:

- Crea e correggi eventi per le operazioni ACK.

Questa policy è progettata per supportare la gestione completa AWS delle risorse tramite Kubernetes APIs. Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per la funzionalità (ruolo IAM) che fornisci quando viene creata la funzionalità ACK.

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizz azioni)
	`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.aws` , `codeartifact.services.k8s.aws` , `cognitoidentityprovider.services.k8s.aws` ,	*	*

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
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.aws , networkfirewall.services.k8s.aws , opensearchservice.services.k8s.aws , organizations.serv		

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
ices.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
coordination.k8s.io	leases(solo leasing di controller di servizio ACK specifici)	delete, update, patch
	events	create, patch

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
`apiextensions.k8s.io`	`customresourcedefinitions`	`*`

EKSArgoCDClusterPolitica di Amazon

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSArgoCDClusterPolicy`

Questa politica concede le autorizzazioni a livello di cluster necessarie per consentire alla funzionalità di Argo CD di scoprire risorse e gestire oggetti con ambito cluster. La policy include le seguenti autorizzazioni:

Gestione del namespace:

- Crea, leggi, aggiorna ed elimina i namespace per la gestione dei namespace delle applicazioni.

Gestione personalizzata delle definizioni delle risorse:

- Gestione di Argo CD specifici CRDs (applicazioni,, AppProjects). ApplicationSets

Scoperta delle API:

- Accesso in lettura agli endpoint dell'API Kubernetes per la scoperta delle risorse.

Questa policy è progettata per supportare le operazioni Argo CD a livello di cluster, tra cui la gestione dei namespace e l'installazione di CRD. Amazon EKS crea automaticamente una voce di accesso con questa politica di accesso per la funzionalità (ruolo IAM) che fornisci quando viene creata la funzionalità Argo CD.

Gruppi di API Kubernetes	Kubernetes NonResource URLs	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
		`namespaces`	`create`, `get`, `update`, `patch`, `delete`

Gruppi di API Kubernetes	Kubernetes NonResource URLs	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
`apiextensions.k8s.io`		`customresourcedefinitions`	`create`
`apiextensions.k8s.io`		`customresourcedefinitions` (Solo Argo CD) CRDs	`get`, `update`, `patch`, `delete`
	`/api`, `/api/*`, `/apis`, `/apis/*`		`get`

Amazon EKSArgo CDPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSArgoCDPolicy`

Questa politica concede le autorizzazioni a livello di namespace necessarie per consentire alla funzionalità di Argo CD di distribuire e gestire le applicazioni. La policy include le seguenti autorizzazioni:

Gestione segreta:

- Accesso completo ai segreti per le credenziali Git e i segreti del cluster.

ConfigMap Accesso:

- Accesso in lettura per ConfigMaps inviare avvisi se i clienti cercano di utilizzare un CD Argo non supportato. ConfigMaps

Gestione degli eventi:

- Leggi e crea eventi per il monitoraggio del ciclo di vita delle applicazioni.

Gestione delle risorse Argo CD:

- Accesso completo alle applicazioni ApplicationSets, e AppProjects.
- Gestisci i finalizzatori e lo stato delle risorse Argo CD.

Questa policy è progettata per supportare le operazioni di Argo CD a livello di namespace, tra cui la distribuzione e la gestione delle applicazioni. Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per la funzionalità (ruolo IAM) che fornisci al momento della creazione della funzionalità Argo CD, nell'ambito dello spazio dei nomi Argo CD.

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizzazioni)
	`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`	`*`

Amazon EKSKROPolicy

ARN: `arn:aws: eks::aws:cluster-access-policy/AmazonEKSKROPolicy`

Questa politica concede le autorizzazioni necessarie per consentire a kro (Kube Resource Orchestrator) di creare e gestire Kubernetes personalizzati. APIs La policy include le seguenti autorizzazioni:

Gestione delle risorse kro:

- Accesso completo a tutte le risorse kro, comprese le istanze ResourceGraphDefinitions di risorse personalizzate.

Gestione personalizzata della definizione delle risorse:

- Crea, leggi, aggiorna ed elimina CRDs per impostazioni personalizzate APIs definite da ResourceGraphDefinitions.

Elezione del leader:

- Crea e leggi i contratti di coordinamento per l'elezione del leader.
- Aggiorna ed elimina il contratto di locazione del controller kro.

Gestione degli eventi:

- Crea e correggi eventi per le operazioni kro.

Questa politica è progettata per supportare la composizione completa delle risorse e la gestione personalizzata delle API tramite kro. Amazon EKS crea automaticamente una voce di accesso con questa policy di accesso per la funzionalità (ruolo IAM) che fornisci quando viene creata la funzionalità kro.

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizz azioni)
`kro.run`	`*`	`*`
`apiextensions.k8s.io`	`customresourcedefi nitions`	`*`
`coordination.k8s.io`	`leases`	`create`, `get`, `list`, `watch`
`coordination.k8s.io`	`leases`(solo controller kro lease)	`delete`, `update`, `patch`

Gruppi di API Kubernetes	Risorse di Kubernetes	Verbi di Kubernetes (autorizz azioni)
	`events`	`create`, `patch`

Aggiornamenti della policy di accesso

Visualizza i dettagli sugli aggiornamenti alle policy di accesso, da quando sono stati introdotti. Per gli avvisi automatici sulle modifiche apportate a questa pagina, iscriviti al feed RSS in Cronologia dei documenti.

Modifica	Descrizione	Data
Aggiungi politiche per EKS Capabilities	Pubblica `AmazonEKS ACKPolicy` `AmazonEKS ArgoCDClusterPolic y` `AmazonEKSArgoCDPol icy` , e `AmazonEKS KROPolicy` per la gestione delle funzionalità EKS	22 novembre 2025
Add `AmazonEKSSecretRea derPolicy`	Aggiungi una nuova politica per l'accesso in sola lettura ai segreti	6 novembre 2025
Aggiungi una policy per gli approfondimenti sui cluster di EKS	Pubblica `AmazonEKS ClusterInsightsPol icy`	2 dicembre 2024
Aggiungi policy per Amazon EKS Hybrid	Pubblica `AmazonEKS HybridPolicy`	2 dicembre 2024
Aggiungi policy per la modalità automatica di Amazon EKS	Queste policy di accesso autorizzano Cluster IAM Role e Node IAM Role a chiamare APIs Kubernetes. AWS li utilizza per automatizzare le	2 dicembre 2024

Modifica	Descrizione	Data
	attività di routine per le risorse di archiviazione, elaborazione e rete.	
Add `AmazonEKSAdminViewPolicy`	Aggiungi una nuova policy per un accesso esteso alle visualizzazioni, incluse risorse come Secrets.	23 aprile 2024
Policy di accesso introdotte.	Amazon EKS ha introdotto policy di accesso.	29 maggio 2023

Cambia la modalità di autenticazione per utilizzare le voci di accesso

Per iniziare a utilizzare le voci di accesso, è necessario modificare la modalità di autenticazione del cluster nelle modalità `API_AND_CONFIG_MAP` o `API`. Questa operazione aggiunge l'API per le voci di accesso.

Console AWS

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster in cui desideri creare una voce di accesso.
3. Scegli la scheda Accesso.
4. Modalità di autenticazione mostra la modalità corrente di autenticazione del cluster. Se la modalità mostra API EKS, puoi già aggiungere voci di accesso e puoi saltare i passaggi rimanenti.
5. Scegli Gestisci accesso.
6. Per Modalità di autenticazione del cluster, seleziona una modalità con l'API EKS. Nota che non è possibile ripristinare la modalità di autenticazione in una modalità che rimuove l'API EKS e le voci di accesso.
7. Scegli Save changes (Salva modifiche). Amazon EKS inizia ad aggiornare il cluster, lo stato del cluster cambia in In aggiornamento e la modifica è registrata nella scheda Cronologia degli aggiornamenti.

8. Attendi che lo stato del cluster ritorni su Attivo. Quando il cluster è Attivo, è possibile seguire i passaggi indicati in the section called "Creare voci di accesso" per aggiungere l'accesso al cluster per i principali IAM.

CLI AWS

1. Installa AWS CLI come descritto in Installazione nella Guida per l'utente dell'interfaccia a riga di comando AWS.
2. Esegui il comando seguente. Sostituisci *my-cluster* con il nome del tuo cluster. Se desideri disabilitare il metodo `ConfigMap` in modo permanente, sostituisci `API_AND_CONFIG_MAP` con `API`.

   Amazon EKS inizia ad aggiornare il cluster, lo stato del cluster cambia in IN AGGIORNAMENTO e la modifica è registrata negli aggiornamenti dell'elenco eks aws .

   ```
   aws eks update-cluster-config --name my-cluster --access-config
    authenticationMode=API_AND_CONFIG_MAP
   ```
3. Attendi che lo stato del cluster ritorni su Attivo. Quando il cluster è Attivo, è possibile seguire i passaggi indicati in the section called "Creare voci di accesso" per aggiungere l'accesso al cluster per i principali IAM.

Versione della piattaforma richiesta

Per utilizzare le voci di accesso, il cluster deve avere una versione della piattaforma uguale o successiva a quella elencata nella tabella seguente o una versione Kubernetes successiva a quelle elencate nella tabella. Se la tua versione di Kubernetes non è elencata, tutte le versioni della piattaforma supportano le voci di accesso.

Versione di Kubernetes 1.15	Versione della piattaforma
Non elencata	Tutte supportate
`1.30`	`eks.2`
`1.29`	`eks.1`
`1.28`	`eks.6`

Per ulteriori informazioni, consulta Versioni della piattaforma .

Creare voci di accesso

Prima di creare voci di accesso, considera quanto segue:

- Una modalità di autenticazione impostata in modo corretto. Consultare the section called "Modalità di autenticazione".
- Una voce di accesso include il nome della risorsa Amazon (ARN) di un principale (e solo di uno) IAM esistente. Un principale IAM non può essere incluso in più di una voce di accesso. Considerazioni aggiuntive per l'ARN specificato:
  - Le best practice IAM consigliano di accedere al cluster utilizzando ruoli IAM con credenziali a breve termine, anziché utenti IAM con credenziali a lungo termine. Per maggiori informazioni, consulta Richiedere utenti umani per usare la federazione con un provider di identità per accedere a AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.
  - Se l'ARN è per un ruolo IAM, può includere un percorso. Gli ARN nelle voci `aws-auth ConfigMap`, non possono includere un percorso. Ad esempio, il tuo ARN può essere `arn:aws:iam::<111122223333>:role/<development/apps/my-role>` o `arn:aws:iam::<111122223333>:role/<my-role>`.
  - Se il tipo di voce di accesso è diverso da `STANDARD` (vedi la prossima considerazione sui tipi), l'ARN deve trovarsi nello stesso account AWS in cui si trova il cluster. Se il tipo è `STANDARD`, l'ARN può essere nell'account AWS uguale o diverso dall'account in cui si trova il cluster.
  - Non è possibile modificare il principale IAM dopo aver creato la voce di accesso.
  - Se elimini il principale IAM con questo ARN, la voce di accesso non viene eliminata automaticamente. Ti consigliamo di eliminare la voce di accesso con un ARN per un principale IAM che elimini. Se non cancelli la voce di accesso e in futuro ricrei il principale IAM, anche se ha lo stesso ARN, la voce di accesso non funzionerà. Questo perché, anche se l'ARN è lo stesso per il principale IAM ricreato, il `roleID` o `userID` (puoi vederlo con il comando CLI `aws sts get-caller-identity` AWS) è diverso per il principale IAM ricreato rispetto al principale IAM originale. Anche se non visualizzi il `roleID` o il `userID` del principale IAM per una voce di accesso, Amazon EKS lo memorizza insieme alla voce di accesso.
- Ogni voce di accesso ha un tipo. Il tipo di voce di accesso dipende dal tipo di risorsa a cui viene associata e non definisce le autorizzazioni. Se non specifichi un tipo, Amazon EKS lo imposta automaticamente su `STANDARD`
  - `EC2_LINUX`: per un ruolo IAM usato con nodi Linux o Bottlerocket autogestiti
  - `EC2_WINDOWS`: per un ruolo IAM usato con nodi autogestiti di Windows

  - `FARGATE_LINUX`: per un ruolo IAM usato con AWS Fargate (Fargate)
  - `HYBRID_LINUX`: per un ruolo IAM usato con nodi ibridi
  - `STANDARD`: tipo predefinito se non specificato
  - `EC2`: per le classi di nodi personalizzate EKS Auto Mode. Per ulteriori informazioni, consulta the section called "Creazione di una voce di accesso alla classe di nodi".
  - Non è possibile modificare il tipo dopo aver creato la voce di accesso.
- Non è necessario creare una voce di accesso per un ruolo IAM utilizzato per un gruppo di nodi gestiti o un profilo Fargate. EKS creerà voci di accesso (se abilitate) o aggiornerà la mappa di configurazione dell'autenticazione (se non sono disponibili le voci di accesso)
- Se il tipo di voce di accesso è `STANDARD`, puoi specificare un nome utente per la voce di accesso. Se non specifichi un valore per il nome utente, Amazon EKS assegnerà automaticamente uno dei valori seguenti, in funzione del tipo di voce di accesso e del principale IAM da te indicato, che si tratti di un ruolo IAM o di un utente IAM. A meno che tu non abbia un motivo particolare per specificare il tuo nome utente, consigliamo di non specificarne uno e di lasciare che Amazon EKS lo generi automaticamente. Se decidi di specificare il tuo nome utente:
  - Tieni presente che non può iniziare con `system:`, `eks:`, `aws:`, `amazon:` o `iam:`.
  - Se il nome utente è per un ruolo IAM, ti consigliamo di aggiungere `{{SessionName}}` o `{{SessionNameRaw}}` alla fine del nome utente. Se aggiungi `{{SessionName}}` o `{{SessionNameRaw}}` al tuo nome utente, il nome utente deve includere i due punti prima di {{sessionName}}. Quando si assume questo ruolo, il nome della sessione AWS STS specificata al momento dell'assunzione del ruolo viene passato automaticamente al cluster e verrà visualizzato nei log di CloudTrail. Ad esempio, non puoi avere un nome utente del tipo `john{{SessionName}}`. Il nome utente deve essere `:john{{SessionName}}` o `jo:hn{{SessionName}}`. Solo i due punti devono essere prima di `{{SessionName}}`. Il nome utente generato da Amazon EKS nella tabella seguente include un ARN. Poiché un ARN include i due punti, soddisfa questo requisito. I due punti non sono obbligatori se non includi `{{SessionName}}` nel nome utente. Nota che in `{{SessionName}}` il carattere speciale "@" viene sostituito con "-" nel nome della sessione. `{{SessionNameRaw}}` mantiene tutti i caratteri speciali nel nome della sessione.

Il tipo di principale IAM	Tipo	Valore del nome utente che Amazon EKS imposta automaticamente
Utente	`STANDARD`	L'ARN dell'utente. Esempio: `arn:aws:iam::<111122223333>:user/<my-user>`
Ruolo	`STANDARD`	L'ARN STS del ruolo quando viene assunto. Amazon EKS aggiunge `{{SessionName}}` al ruolo. Esempio: `arn:aws:sts::<111122223333>:assumed-role/<my-role>/{{SessionName}}` Se l'ARN del ruolo che hai specificato conteneva un percorso, Amazon EKS lo rimuove nel nome utente generato.
Ruolo	`EC2_LINUX` o `EC2_Windows`	`system:node:{{EC2PrivateDNSName}}`
Ruolo	`FARGATE_LINUX`	`system:node:{{SessionName}}`
Ruolo	`HYBRID_LINUX`	`system:node:{{SessionName}}`

Puoi modificare il nome utente dopo aver creato la voce di accesso.

- Se il tipo di una voce di accesso è `STANDARD` e desideri utilizzare l'autorizzazione RBAC di Kubernetes, puoi aggiungere uno o più nomi dei gruppi alla voce di accesso. Dopo aver creato una voce di accesso, puoi aggiungere e rimuovere i nomi dei gruppi. Affinché il principale IAM abbia accesso agli oggetti Kubernetes sul cluster, è necessario creare e gestire oggetti di autorizzazione basati sui ruoli (RBAC) Kubernetes. Creare oggetti Kubernetes `RoleBinding` o `ClusterRoleBinding` nel cluster che specificano il nome del gruppo come `subject` per `kind: Group`. Kubernetes autorizza l'accesso del principale IAM a qualsiasi oggetto del cluster specificato in un oggetto `Role` Kubernetes o `ClusterRole` specificato anche nel `roleRef` delle tue associazioni. Se specifichi nomi di gruppo, ti consigliamo di acquisire familiarità con gli oggetti di autorizzazione basata sui ruoli (RBAC) di Kubernetes. Per ulteriori informazioni, consulta Utilizzo dell'autorizzazione RBAC nella documentazione di Kubernetes.

  > **Important**
  >
  > Amazon EKS non conferma che gli oggetti RBAC Kubernetes presenti nel cluster includano uno dei nomi di gruppo specificati. Ad esempio, se si crea una voce di accesso per un gruppo che attualmente non esiste, EKS creerà il gruppo invece di restituire un errore.

  Puoi collegare policy di accesso Amazon EKS a una voce di accesso, come alternativa o complemento all'autorizzazione concessa da Kubernetes al principale IAM per accedere agli oggetti Kubernetes sul tuo cluster. Amazon EKS autorizza i principali IAM ad accedere agli oggetti Kubernetes sul tuo cluster con le autorizzazioni previste dalla policy di accesso. Puoi definire l'ambito delle autorizzazioni di una policy di accesso ai namespace di Kubernetes che specifichi. L'uso delle policy di accesso non richiede la gestione degli oggetti RBAC di Kubernetes. Per ulteriori informazioni, consulta the section called "Associare le policy di accesso".
- Se crei una voce di accesso con tipo `EC2_LINUX` o `EC2_Windows`, il principale IAM che crea la voce di accesso deve disporre dell'autorizzazione `iam:PassRole`. Per ulteriori informazioni, consulta Concessione di autorizzazioni utente per il passaggio di un ruolo a un servizio AWS nella Guida per l'utente di IAM.
- Analogamente al comportamento di IAM standard, la creazione e gli aggiornamenti delle voci di accesso alla fine sono coerenti e potrebbero essere necessari alcuni secondi prima che la chiamata API iniziale risulti completata con successo. È necessario progettare le applicazioni in modo da tenere in considerazione questi potenziali ritardi Si consiglia di non includere creazioni o aggiornamenti degli accessi nei percorsi critici e ad alta disponibilità del codice dell'applicazione. Al contrario, apportare modifiche in un'inizializzazione separata o in una routine di configurazione

che si esegue meno frequentemente. Inoltre, assicurarsi di verificare che le modifiche siano state propagate prima che i flussi di lavoro di produzione dipendano da esse.

- Le voci di accesso non supportano i ruoli collegati al servizio. Non puoi creare voci di accesso in cui l'ARN principale è un ruolo collegato al servizio. Puoi identificare i ruoli collegati al servizio in base al relativo ARN, che è nel formato `arn:aws:iam::*:role/aws-service-role/*`.

È possibile creare una voce di accesso utilizzando la Console di gestione AWS o la AWS CLI.

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster in cui desideri creare una voce di accesso.
3. Scegli la scheda Accesso.
4. Seleziona Crea voce di accesso.
5. Per Principale IAM, seleziona un ruolo o un utente IAM esistente. Le best practice IAM consigliano di accedere al cluster utilizzando ruoli IAM con credenziali a breve termine, anziché utenti IAM con credenziali a lungo termine. Per maggiori informazioni, consulta Richiedere utenti umani per usare la federazione con un provider di identità per accedere a AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.
6. Per Tipo, se la voce di accesso è per il ruolo del nodo utilizzato per i nodi Amazon EC2 autogestiti, seleziona EC2 Linux o EC2 Windows. Altrimenti, accetta l'impostazione predefinita (Standard).
7. Se il Tipo che hai scelto è Standard e desideri specificare un Nome utente, inserisci il nome utente.
8. Se il Tipo che hai scelto è Standard e desideri utilizzare l'autorizzazione RBAC di Kubernetes per il principale IAM, specifica uno o più nomi per i Gruppi. Se non specifichi nomi dei gruppi e desideri utilizzare l'autorizzazione Amazon EKS, puoi associare una policy di accesso in un passaggio successivo o dopo la creazione della voce di accesso.
9. (Facoltativo) Per Tag, assegna etichette alla voce di accesso. Ad esempio, per facilitare la ricerca di tutte le risorse con lo stesso tag.
10. Scegli Next (Successivo).
11. Nella pagina Aggiungi policy di accesso, se il tipo che hai scelto era Standard e desideri che Amazon EKS autorizzi il principale IAM a disporre delle autorizzazioni per gli oggetti Kubernetes sul tuo cluster, completa i seguenti passaggi. Altrimenti, scegli Next (Successivo).
    a. Per Nome della policy, scegli una policy di accesso. Non puoi visualizzare le autorizzazioni delle policy di accesso, tuttavia includono autorizzazioni simili a quelle degli oggetti `ClusterRole`

di Kubernetes rivolti all'utente. Per ulteriori informazioni, consulta User-facing roles nella documentazione di Kubernetes.

b. Selezionare una delle seguenti opzioni:

- Cluster: scegli questa opzione se desideri che Amazon EKS autorizzi il principale IAM a disporre delle autorizzazioni nella policy di accesso per tutti gli oggetti Kubernetes sul tuo cluster.
- Namespace Kubernetes: scegli questa opzione se desideri che Amazon EKS autorizzi il principale IAM a disporre delle autorizzazioni nella policy di accesso per tutti gli oggetti Kubernetes in uno specifico namespace Kubernetes sul tuo cluster. Per Namespace, inserisci il nome del namespace Kubernetes sul tuo cluster. Se desideri aggiungere altri spazi dei nomi, scegli Aggiungi nuovo spazio dei nomi e inserisci il nome dello spazio dei nomi.

c. Se desideri aggiungere ulteriori policy, scegli Aggiungi policy. Puoi definire l'ambito di ciascuna policy in modo diverso, ma puoi aggiungere ogni policy solo una volta.

d. Scegli Next (Successivo).

12. Verifica la configurazione per la tua voce di accesso. Se noti qualcosa di errato, scegli Precedente per tornare indietro e correggere l'errore. Se la configurazione è corretta, scegli Crea.

CLI AWS

1. Installare e configurare AWS CLI come descritto nella pagina Installing della Guida per l'utente dell'Interfaccia della linea di comando AWS.
2. Per creare una voce di accesso Per creare voci di accesso è possibile utilizzare uno qualsiasi degli esempi seguenti:

- Crea una voce di accesso per un gruppo di nodi Amazon EC2 Linux autogestito. Sostituisci *my-cluster* con il nome del cluster, *111122223333* con l'ID del tuo account AWS e *EKS-my-cluster-self-managed-ng-1* con il nome del ruolo IAM del nodo. Nel caso in cui il tuo gruppo di nodi operi su Windows, sostituisci *EC2_LINUX* con `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
```

Non puoi utilizzare l'opzione `--kubernetes-groups` quando specifichi un tipo diverso da `STANDARD`. Non puoi associare una policy di accesso a questa voce di accesso, perché il suo tipo è un valore diverso da `STANDARD`.

- Crea una voce di accesso che permetta a un ruolo IAM, non utilizzato per un gruppo di nodi autogestiti di Amazon EC2, di essere autorizzato da Kubernetes per accedere al tuo cluster. Sostituisci *my-cluster* con il nome del cluster, *111122223333* con l'ID del tuo account AWS e *my-role-name* con il nome del tuo ruolo IAM. Sostituisci *Visualizzatori* con il nome di un gruppo che hai specificato in un oggetto Kubernetes `RoleBinding` o `ClusterRoleBinding` sul tuo 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
```

- Crea una voce di accesso che consenta a un utente IAM di autenticarsi nel tuo cluster. Questo esempio viene fornito perché è possibile, anche se le best practice IAM consigliano di accedere al cluster utilizzando ruoli IAM con credenziali a breve termine, anziché utenti IAM con credenziali a lungo termine. Per maggiori informazioni, consulta Richiedere utenti umani per usare la federazione con un provider di identità per accedere a AWS utilizzando credenziali temporanee nella Guida per l'utente 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 desideri che questo utente abbia un accesso al tuo cluster maggiore rispetto alle autorizzazioni nei ruoli di rilevamento delle API Kubernetes, è necessario associare una policy di accesso alla voce di accesso, poiché l'opzione `--kubernetes-groups` non viene utilizzata. Per ulteriori informazioni, consulta the section called "Associare le policy di accesso" e API discovery roles nella documentazione di Kubernetes.

Aggiornamento delle voci di accesso

È possibile aggiornare una voce di accesso utilizzando Console di gestione AWS o AWS CLI.

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster in cui desideri creare una voce di accesso.
3. Scegli la scheda Accesso.
4. Scegli la voce di accesso che desideri aggiornare.

5. Scegli Modifica.
6. Per Nome utente, puoi modificare il valore esistente.
7. Per Gruppi, puoi rimuovere i nomi dei gruppi esistenti o aggiungere nuovi nomi dei gruppi. Se esistono i seguenti nomi di gruppi, non rimuoverli: system:nodes o system:bootstrappers. La rimozione di questi gruppi può causare un malfunzionamento del cluster. Se non specifichi nomi dei gruppi e desideri utilizzare l'autorizzazione Amazon EKS, associa una policy di accesso in un passaggio successivo.
8. Per Tag, puoi assegnare etichette alla voce di accesso. Ad esempio, per facilitare la ricerca di tutte le risorse con lo stesso tag. Puoi anche rimuovere i tag esistenti.
9. Scegli Save changes (Salva modifiche).
10. Se desideri associare una policy di accesso alla voce, consulta la sezione the section called "Associare le policy di accesso".

CLI AWS

1. Installa AWS CLI come descritto in Installazione nella Guida per l'utente dell'interfaccia a riga di comando AWS.
2. Per aggiornare una voce di accesso, sostituisci *my-cluster* con il nome del cluster, *111122223333* con l'ID del tuo account AWS e *EKS-my-cluster-my-namespace-Viewers* con il nome di un ruolo 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
```

   Non è possibile utilizzare l'opzione `--kubernetes-groups` se il tipo di voce di accesso è un valore diverso da `STANDARD`. Inoltre, non è possibile associare una policy di accesso a una voce di accesso con un tipo diverso da `STANDARD`.

Eliminare le voci di accesso

Se scopri di aver eliminato una voce di accesso per errore, puoi sempre ricrearla. Se la voce di accesso da eliminare è associata a policy di accesso, le associazioni vengono eliminate automaticamente. Non è necessario dissociare le policy di accesso da una voce di accesso prima di eliminarla.

È possibile eliminare una voce di accesso utilizzando la Console di gestione AWS o AWS CLI.

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster da cui desideri eliminare una voce di accesso.
3. Scegli la scheda Accesso.
4. Nell'elenco Voci di accesso, seleziona la voce di accesso da eliminare.
5. Scegliere Delete (Elimina).
6. Nella finestra di dialogo di conferma, seleziona Elimina.

CLI AWS

1. Installare AWS CLI come descritto nella pagina Installing della Guida per l'utente dell'Interfaccia della linea di comando AWS.
2. Per eliminare una voce di accesso, sostituire *my-cluster* con il nome del cluster, *111122223333* con l'ID dell'account AWS e *my-role* con il nome del ruolo IAM a cui si desidera revocare l'accesso al cluster.

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

Impostazione di un nome utente personalizzato per le voci di accesso EKS

Quando crei voci di accesso per Amazon EKS, è possibile utilizzare il nome utente generato automaticamente o specificare un nome utente personalizzato. Questa pagina spiega entrambe le opzioni e ti guida nell'impostazione di un nome utente personalizzato.

Panoramica di

Il nome utente in una voce di accesso è utilizzato per identificare il principale IAM nei log e negli audit trail di Kubernetes. Per impostazione predefinita, Amazon EKS genera un nome utente basato sull'ARN dell'identità IAM, ma è possibile specificare un nome utente personalizzato se necessario.

Generazione di un nome utente predefinito

Se non specifichi un valore per il nome utente, Amazon EKS genera automaticamente un nome utente basato sull'identità IAM:

- Per un utente IAM:
  - EKS imposta il nome utente Kubernetes sull'ARN dell'utente IAM
  - Esempio:

```
{arn-aws}iam::<111122223333>:user/<my-user>
```

- Per ruoli IAM:
  - EKS imposta il nome utente Kubernetes in base all'ARN del ruolo IAM
  - L'ARN STS del ruolo quando è assunto. Amazon EKS aggiunge `{{SessionName}}` al ruolo. Se l'ARN del ruolo che hai specificato conteneva un percorso, Amazon EKS lo rimuove nel nome utente generato.
  - Esempio:

```
{arn-aws}sts::<111122223333>:assumed-role/<my-role>/{{SessionName}}
```

A meno che tu non abbia un motivo particolare per specificare il tuo nome utente, consigliamo di non specificarne uno e di lasciare che Amazon EKS lo generi automaticamente.

Impostare un nome utente personalizzato

Quando si crea una voce di accesso, è possibile specificare un nome utente personalizzato utilizzando il parametro `--username`:

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD --username <custom-username>
```

Requisiti per nomi utente personalizzati

Se decidi specificare un nome utente personalizzato:

- Il nome utente non può iniziare con `system:`, `eks:`, `aws:`, `amazon:` o `iam:`.
- Se il nome utente è per un ruolo IAM, ti consigliamo di aggiungere `{{SessionName}}` o `{{SessionNameRaw}}` alla fine del nome utente.
  - Se aggiungi uno dei due `{{SessionName}}` o `{{SessionNameRaw}}` al tuo nome utente, il nome utente deve includere i due punti prima di {{SessionName}}.

Crea una voce di accesso per un utente o un ruolo IAM usando una policy di accesso e la AWS CLI

Crea voci di accesso Amazon EKS che usano le policy di accesso EKS gestite da AWS per dare alle identità IAM autorizzazioni standardizzate per accedere e gestire i cluster Kubernetes.

Panoramica

Le voci di accesso in Amazon EKS definiscono il modo in cui le identità IAM (utenti e ruoli) possono accedere ai cluster Kubernetes e interagire con essi. Creando voci di accesso con le policy di accesso EKS, puoi:

- Concedere a utenti o a ruoli IAM specifici l'autorizzazione ad accedere al tuo cluster EKS
- Controllare le autorizzazioni usando policy di accesso EKS gestite da AWS che forniscono set di autorizzazioni standardizzati e predefiniti
- Limitare le autorizzazioni a namespace specifici o a tutto il cluster
- Semplificare la gestione degli accessi senza modificare il ConfigMap `aws-auth` o creare risorse Kubernetes RBAC
- Usare l'approccio integrato con AWS al controllo degli accessi Kubernetes che copre i casi d'uso più comuni, mantenendo al contempo le migliori pratiche di sicurezza.

Questo approccio è consigliato per la maggior parte dei casi d'uso perché fornisce autorizzazioni standardizzate gestite da AWS senza richiedere la configurazione manuale di Kubernetes RBAC. Le policy di accesso EKS eliminano la necessità di configurare manualmente le risorse Kubernetes RBAC e forniscono set di autorizzazioni predefiniti che coprono casi d'uso comuni.

Prerequisiti

- La modalità di autenticazione del cluster deve essere configurata per abilitare le voci di accesso. Per ulteriori informazioni, consulta the section called "Modalità di autenticazione".
- Installare e configurare AWS CLI come descritto nella pagina Installing della Guida per l'utente dell'Interfaccia della linea di comando AWS.

Fase 1: definire la voce di accesso

1. Trovare l'ARN dell'identità IAM, come un utente o un ruolo, a cui concedere le autorizzazioni.
   - Ogni identità IAM può disporre di una sola voce di accesso EKS.

2. Stabilisci se vuoi che le autorizzazioni della policy di accesso Amazon EKS si applichino solo a uno specifico namespace Kubernetes o all'intero cluster.
   - Se vuoi limitare le autorizzazioni a un namespace specifico, prendi nota del nome del namespace.
3. Seleziona la policy di accesso EKS che desideri per l'identità IAM. Questa policy fornisce le autorizzazioni all'interno del cluster. Annota l'ARN della policy.
   - Per un elenco delle policy, consulta le policy di accesso disponibili.
4. Stabilire se il nome utente generato automaticamente è adeguato per la voce di accesso o se è necessario specificarne uno manualmente.
   - AWS genera questo valore automaticamente in base all'identità IAM. È possibile impostare un nome utente personalizzato. Tale nome è visibile nei log di Kubernetes.
   - Per ulteriori informazioni, consulta the section called "Impostazione del nome utente personalizzato".

Fase 2: creare una voce di accesso

Dopo aver pianificato la voce di accesso, usa la AWS CLI per crearla.

L'esempio seguente copre la maggior parte dei casi d'uso. Visualizzare il riferimento CLI per tutte le opzioni di configurazione.

Nella fase successiva allegherai la policy di accesso.

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD
```

Fase 3: associare la policy di accesso

Il comando varia a seconda che si voglia limitare la policy a un namespace Kubernetes specificato.

È necessario l'ARN della policy di accesso. Visualizza le policy di accesso disponibili.

Crea una policy senza l'ambito del namespace

```
aws eks associate-access-policy --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --policy-arn <access-policy-arn>
```

Crea con l'ambito del 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>
```

Passaggi successivi

- Crea un kubeconfig in modo da poter utilizzare kubectl con un'identità IAM

Creare una voce di accesso utilizzando i gruppi Kubernetes con AWS CLI

Creare voci di accesso Amazon EKS che utilizzano gruppi Kubernetes per l'autorizzazione e richiedono una configurazione di controllo degli accessi basato sul ruolo (RBAC) manuale.

> **Note**
>
> Per la maggior parte dei casi d'uso, consigliamo di utilizzare le policy di accesso EKS anziché l'approccio con gruppi Kubernetes descritto in questa pagina. Le policy di accesso EKS offrono un modo più semplice e integrato con AWS per gestire l'accesso senza richiedere la configurazione RBAC manuale. Utilizzare l'approccio con gruppi Kubernetes solo quando è necessario un controllo più granulare di quello offerto dalle policy di accesso EKS.

Panoramica

Le voci di accesso definiscono il modo in cui le identità IAM (utenti e ruoli) accedono ai cluster Kubernetes. L'approccio con gruppi Kubernetes concede agli utenti o ai ruoli IAM l'autorizzazione ad accedere al cluster EKS tramite gruppi Kubernetes RBAC standard. Questo metodo richiede la creazione e la gestione di risorse Kubernetes RBAC (Roles, RoleBindings, ClusterRoles e ClusterRoleBindings) ed è consigliato quando sono necessari set di autorizzazioni altamente personalizzati, requisiti di autorizzazione complessi o quando si desidera mantenere modelli di controllo degli accessi coerenti in ambienti Kubernetes ibridi.

Questo argomento non tratta la creazione di voci di accesso per le identità IAM utilizzate dalle istanze Amazon EC2 per unirsi ai cluster EKS.

Prerequisiti

- La modalità di autenticazione del cluster deve essere configurata per abilitare le voci di accesso. Per ulteriori informazioni, consulta the section called "Modalità di autenticazione".
- Installare e configurare AWS CLI come descritto nella pagina Installing della Guida per l'utente dell'Interfaccia della linea di comando AWS.
- Si consiglia di avere familiarità con il controllo RBAC Kubernetes. Per ulteriori informazioni, consulta Utilizzo dell'autorizzazione RBAC nella documentazione di Kubernetes.

Fase 1: definire la voce di accesso

1. Trovare l'ARN dell'identità IAM, come un utente o un ruolo, a cui concedere le autorizzazioni.
   - Ogni identità IAM può disporre di una sola voce di accesso EKS.
2. Determinare quali gruppi Kubernetes devono essere associati a tale identità IAM.
   - Sarà necessario creare o utilizzare le risorse Kubernetes `Role`/`ClusterRole` e `RoleBinding`/`ClusterRoleBinding` esistenti che fanno riferimento a tali gruppi.
3. Stabilire se il nome utente generato automaticamente è adeguato per la voce di accesso o se è necessario specificarne uno manualmente.
   - AWS genera questo valore automaticamente in base all'identità IAM. È possibile impostare un nome utente personalizzato. Tale nome è visibile nei log di Kubernetes.
   - Per ulteriori informazioni, consulta the section called "Impostazione del nome utente personalizzato".

Fase 2: creare una voce di accesso con i gruppi Kubernetes

Dopo aver definito la voce di accesso, utilizzare AWS CLI per crearla con i gruppi Kubernetes adeguati.

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD --kubernetes-groups <groups>
```

Sostituire:

- `<cluster-name>` con il nome del cluster EKS.
- `<iam-identity-arn>` con l'ARN del ruolo o dell'utente IAM.

- `<groups>` con un elenco separato da virgole di gruppi Kubernetes (ad esempio, "system:developers,system:readers").

Visualizzare il riferimento CLI per tutte le opzioni di configurazione.

Fase 3: configurazione del controllo RBAC Kubernetes

Affinché il principale IAM abbia accesso agli oggetti Kubernetes nel cluster, è necessario creare e gestire oggetti del controllo degli accessi basato sul ruolo (RBAC) Kubernetes:

1. Creare oggetti Kubernetes `Role` o `ClusterRole` che definiscono le autorizzazioni.
2. Creare oggetti Kubernetes `RoleBinding` o `ClusterRoleBinding` nel cluster che specificano il nome del gruppo come `subject` per `kind: Group`.

Per informazioni dettagliate sulla configurazione di gruppi e autorizzazioni in Kubernetes, consultare Using RBAC Authorization nella documentazione di Kubernetes.

Passaggi successivi

- Create a kubeconfig so you can use kubectl with an IAM identity

Concedi agli utenti IAM l'accesso a Kubernetes con un ConfigMap

> **Important**
>
> La funzionalità `aws-auth ConfigMap` è obsoleta. Per il metodo consigliato per gestire l'accesso a Kubernetes, consulta. APIs the section called "Voci di accesso"

L'accesso al cluster utilizzando i principali IAM è abilitato dall'autenticatore AWS IAM per Kubernetes, che viene eseguito sul piano di controllo di Amazon EKS. L'autenticatore riceve le informazioni di configurazione da `aws-auth ConfigMap`. Per tutte le impostazioni di `ConfigMap aws-auth`, consulta Formato di configurazione completo su GitHub.

Aggiunta di principali IAM al cluster Amazon EKS

Quando crei un cluster Amazon EKS, il principale IAM che crea il cluster riceve automaticamente le autorizzazioni `system:masters` nella configurazione del controllo degli accessi basato sul

ruolo (RBAC) nel piano di controllo di Amazon EKS. Questo principale non è visualizzato in una configurazione visibile qualsiasi, quindi assicurati di tenere traccia di quale principale ha originariamente creato il cluster. Per concedere a ulteriori principali IAM la capacità di interagire con il cluster, modifica `aws-auth ConfigMap` all'interno di Kubernetes e crea `rolebinding` o `clusterrolebinding` Kubernetes con il nome di `group` specificato in `aws-auth ConfigMap`.

> **Note**
>
> Per ulteriori informazioni sulla configurazione del controllo degli accessi basato sul ruolo (RBAC) di Kubernetes, consultare Using RBAC Authorization nella documentazione di Kubernetes.

1. Determina quali credenziali `kubectl` vengono utilizzate per accedere al cluster. Sul computer, è possibile vedere quali credenziali `kubectl` utilizza con il seguente comando. Sostituiscilo *`~/.kube/config`* con il percorso `kubeconfig` del file se non utilizzi il percorso predefinito.

   ```
   cat ~/.kube/config
   ```

   Di seguito viene riportato un output di esempio.

   ```
   [...]
   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
   [...]
   ```

   Nell'output dell'esempio precedente, le credenziali per un utente denominato *`admin`* sono configurate per un cluster denominato *`my-cluster`*. Se si tratta dell'utente che ha creato il cluster, ha già accesso al cluster. Se l'utente non ha creato il cluster, devi completare i passaggi rimanenti per abilitare l'accesso al cluster per gli altri principali IAM. Le best practice IAM consigliano di concedere le autorizzazioni ai ruoli anziché agli utenti. Per visualizzare gli altri principali che hanno attualmente accesso al cluster, esegui il comando seguente:

   ```
   kubectl describe -n kube-system configmap/aws-auth
   ```

Di seguito viene riportato un output di esempio.

```
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>
```

L'esempio precedente è una `aws-auth` `ConfigMap` predefinita. Solo il ruolo dell'istanza del nodo ha accesso al cluster.

2. Assicurati di disporre di `roles` e `rolebindings` o `clusterroles` e `clusterrolebindings` di Kubernetes da mappare ai principali IAM. Per ulteriori informazioni su queste risorse, consulta Utilizzo dell'autorizzazione RBAC nella documentazione di Kubernetes.
   a. Visualizza i `roles` o `clusterroles` Kubernetes esistenti. L'ambito di `Roles` è `namespace`, mentre l'ambito di `clusterroles` è il cluster.

      ```
      kubectl get roles -A
      ```

      ```
      kubectl get clusterroles
      ```

   b. Visualizza i dettagli dei valori `role` o `clusterrole` restituiti nell'output precedente e verifica di disporre delle autorizzazioni (`rules`) da concedere ai principali IAM del cluster.

Sostituisci *role-name* con un nome del `role` restituito nell'output del comando precedente. Sostituisci *kube-system* con lo spazio dei nomi del `role`.

```
kubectl describe role role-name -n kube-system
```

Sostituisci *cluster-role-name* con un nome del `clusterrole` restituito nell'output del comando precedente.

```
kubectl describe clusterrole cluster-role-name
```

c. Visualizza i `rolebindings` o `clusterrolebindings` Kubernetes esistenti. L'ambito di `Rolebindings` è `namespace`, mentre l'ambito di `clusterrolebindings` è il cluster.

```
kubectl get rolebindings -A
```

```
kubectl get clusterrolebindings
```

d. Visualizza i dettagli di `rolebinding` o `clusterrolebinding` e verifica che dispongano di un `role` or `clusterrole` della fase precedente elencato come un `roleRef` e un nome di gruppo elencato per `subjects`.

Sostituisci *role-binding-name* con un nome del `rolebinding` restituito nell'output del comando precedente. Sostituisci *kube-system* con lo `namespace` del `rolebinding`.

```
kubectl describe rolebinding role-binding-name -n kube-system
```

Di seguito viene riportato un output di esempio.

```
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
```

Sostituisci *cluster-role-binding-name* con un nome del `clusterrolebinding` restituito nell'output del comando precedente.

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

Di seguito viene riportato un output di esempio.

```
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. Modificare `aws-auth` `ConfigMap`. Per aggiornare `ConfigMap` puoi usare uno strumento adeguato, ad esempio `eksctl`, oppure puoi eseguire l'aggiornamento in modo manuale tramite modifica.

> **Important**
>
> Ti consigliamo di utilizzare `eksctl`, o uno strumento simile, per modificare `ConfigMap`. Per informazioni su altri strumenti che puoi utilizzare, consulta Utilizzare gli strumenti per apportare modifiche a aws- authConfigMap nelle guide alle best practice di Amazon EKS. Una formattazione impropria di `aws-auth` `ConfigMap` può causare la perdita dell'accesso al cluster.

- Visualizzare le fasi per modificare configmap con eksctl.
- Visualizzare le fasi per modificare configmap manualmente.

Modificare Configmap con Eksctl

1. È necessaria la versione `0.215.0` o una versione successiva dello strumento da riga di `eksctl` comando installata sul dispositivo o AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Visualizza le mappature correnti in `ConfigMap`. Sostituisci *`my-cluster`* con il nome del cluster. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

   Di seguito viene riportato un output di esempio.

```
ARN
            USERNAME                                GROUPS
 ACCOUNT
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

3. Aggiungi una mappatura per un ruolo. Sostituisci *`my-role`* con il nome del tuo ruolo. Sostituisci *`eks-console-dashboard-full-access-group`* con il nome del gruppo specificato nel tuo Kubernetes `RoleBinding` o nell'oggetto. `ClusterRoleBinding` Sostituisci *`111122223333`* con l'ID del tuo account. Puoi sostituire *`admin`* con un nome a tua scelta.

```
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**
   >
   > L'ARN del ruolo non può includere un percorso, ad esempio `role/my-team/developers/my-role`. Il formato dell'ARN deve essere `arn:aws: iam::111122223333:role/my-role` . In questo esempio, `my-team/developers/` deve essere rimosso.

   Di seguito viene riportato un output di esempio.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "{arn-aws}iam::111122223333:role/my-role" to
 auth ConfigMap
```

4. Aggiungi una mappatura per un utente. Le best practice IAM consigliano di concedere le autorizzazioni ai ruoli anziché agli utenti. Sostituisci *my-user* con il tuo nome utente. Sostituisci *eks-console-dashboard-restricted-access-group* con il nome del gruppo specificato nel tuo `RoleBinding` Kubernetes o nell'oggetto. `ClusterRoleBinding` Sostituisci *111122223333* con l'ID del tuo account. Puoi sostituire *my-user* con un nome a tua scelta.

```
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
```

   Di seguito viene riportato un output di esempio.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws: iam::111122223333:user/my-user" to
 auth ConfigMap
```

5. Visualizza nuovamente le mappature nella `ConfigMap`.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

   Di seguito viene riportato un output di esempio.

```
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
```

Modificare Configmap manualmente

1. Apri `ConfigMap` per la modifica.

```
kubectl edit -n kube-system configmap/aws-auth
```

> **Note**
>
> Se ricevi un errore che indica "`Error from server (NotFound): configmaps "aws-auth" not found`«, utilizza la procedura in Applica l'aws-auth al tuo cluster ConfigMap per applicare lo stock. `ConfigMap`

2. Aggiungi i tuoi principali IAM alla `ConfigMap`. Un gruppo IAM non è un principale IAM, quindi non può essere aggiunto alla `ConfigMap`.
   - Per aggiungere un ruolo IAM (ad esempio, per utenti federati): aggiungere i dettagli del ruolo alla sezione `mapRoles` della `ConfigMap`, in `data`. Aggiungi questa sezione se non esiste già nel file. Ogni voce supporta i seguenti parametri:
     - rolearn: l'ARN; del ruolo IAM da aggiungere. Questo valore non può includere un percorso. Ad esempio, non puoi specificare un ARN, come `arn:aws: iam::111122223333:role/my-team/developers/role-name` . L'ARN deve essere `arn:aws: iam::111122223333:role/role-name` .
     - username: il nome utente all'interno di Kubernetes da associare al ruolo IAM.
     - groups: il gruppo o l'elenco dei gruppi Kubernetes a cui mappare il ruolo. Tale gruppo può essere un gruppo predefinito o un gruppo specificato in un `clusterrolebinding` o un `rolebinding`. Per ulteriori informazioni consultare Default roles and role bindings nella documentazione di Kubernetes.
   - Per aggiungere un utente IAM: le best practice IAM consigliano di concedere le autorizzazioni ai ruoli anziché agli utenti. Aggiungi i dettagli dell'utente alla sezione `mapUsers` della `ConfigMap`, in `data`. Aggiungi questa sezione se non esiste già nel file. Ogni voce supporta i seguenti parametri:
     - userarn: L'ARN; dell'utente IAM da aggiungere.
     - username: Il nome utente all'interno di Kubernetes da associare all'utente IAM.
     - groups: il gruppo o l'elenco dei gruppi Kubernetes a cui mappare l'utente. Tale gruppo può essere un gruppo predefinito o un gruppo specificato in un `clusterrolebinding` o un

`rolebinding`. Per ulteriori informazioni consultare Default roles and role bindings nella documentazione di Kubernetes.

3. Ad esempio, il blocco YAML seguente contiene:

- Una sezione `mapRoles` che mappa l'istanza del nodo IAM ai gruppi Kubernetes, in modo che i nodi possano registrarsi autonomamente con il cluster e il ruolo IAM `my-console-viewer-role` mappato a un gruppo Kubernetes in grado di visualizzare tutte le risorse Kubernetes per tutti i cluster. Per un elenco delle autorizzazioni del gruppo IAM e Kubernetes necessarie per il ruolo IAM `my-console-viewer-role`, consulta the section called "Autorizzazioni richieste".
- Una `mapUsers` sezione che mappa l'utente `admin` IAM dall' AWS account predefinito al gruppo `system:masters` Kubernetes e l'`my-user`utente di un AWS account diverso mappato a un gruppo Kubernetes in grado di visualizzare le risorse Kubernetes per uno spazio dei nomi specifico. Per un elenco delle autorizzazioni del gruppo IAM e Kubernetes necessarie per l'utente IAM `my-user`, consulta the section called "Autorizzazioni richieste".

  Aggiungi o rimuovi righe in base alle esigenze e sostituisci tutti i valori di esempio con quelli desiderati.

```
# 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. Salva il file ed esci dall'editor di testo.

Applica la **aws-auth ConfigMap** al cluster

La `aws-auth` `ConfigMap` viene creata e applicata automaticamente al cluster quando si crea un gruppo di nodi gestito o quando si crea un gruppo di nodi utilizzando `eksctl`. È inizialmente creata per consentire ai nodi di unirsi al cluster, ma questa `ConfigMap` può essere usata anche per aggiungere l'accesso RBAC (controllo degli accessi basato sul ruolo) a principali IAM. Se non hai avviato nodi autogestiti e non hai applicato la `aws-auth` `ConfigMap` al cluster, puoi farlo utilizzando la procedura seguente.

1. Controlla per vedere se è già stata applicata la `aws-auth` `ConfigMap`.

   ```
   kubectl describe configmap -n kube-system aws-auth
   ```

   Se ricevi un errore di tipo "`Error from server (NotFound): configmaps "aws-auth" not found`", procedi con le fasi seguenti per applicare lo stock `ConfigMap`.

2. Scarica, modifica e applica la mappa di configurazione dell'autenticatore. AWS

   a. Scarica la mappa di configurazione.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
   aws-auth-cm.yaml
   ```

   b. Nel file `aws-auth-cm.yaml`, imposta il `rolearn` sul nome della risorsa Amazon (ARN) del ruolo IAM associato ai nodi. Per eseguire questa operazione, utilizza un editor di testo o sostituisci *`my-node-instance-role`* eseguendo il comando seguente:

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
   role|' aws-auth-cm.yaml
   ```

   Non modificare altre righe in questo file.

> **Important**
>
> L'ARN del ruolo non può includere un percorso, ad esempio `role/my-team/developers/my-role`. Il formato dell'ARN deve essere `arn:aws:iam::111122223333:role/my-role` . In questo esempio, `my-team/developers/` deve essere rimosso.

Puoi ispezionare gli output dello AWS CloudFormation stack per i tuoi gruppi di nodi e cercare i seguenti valori:

- InstanceRoleARN — Per i gruppi di nodi creati con `eksctl`
- NodeInstanceRole— Per i gruppi di nodi creati con i AWS CloudFormation modelli forniti da Amazon EKS nel Console di gestione AWS

c. Applica la configurazione. L'esecuzione di questo comando potrebbe richiedere alcuni minuti.

```
kubectl apply -f aws-auth-cm.yaml
```

> **Note**
>
> Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso negato o non autorizzato (`kubectl`)" nell'argomento relativo alla risoluzione dei problemi.

3. Guarda lo stato dei nodi e attendi che raggiungano lo stato `Ready`.

```
kubectl get nodes --watch
```

Inserisci `Ctrl+C` per tornare a un prompt della shell.

Concedere agli utenti l'accesso a Kubernetes con un provider OIDC esterno

Amazon EKS supporta l'utilizzo di provider di identità OpenID Connect (OIDC) come metodo per autenticare gli utenti nel cluster. I provider di identità OIDC possono essere utilizzati con o in alternativa a AWS Identity and Access Management (IAM). Per ulteriori informazioni sull'utilizzo di IAM, consultare the section called "Accesso API Kubernetes". Dopo aver configurato l'autenticazione

per il cluster, è possibile creare `roles` e `clusterroles` Kubernetes per assegnare autorizzazioni ai ruoli e quindi associare i ruoli alle identità utilizzando `rolebindings` e `clusterrolebindings` Kubernetes. Per ulteriori informazioni, consulta Using RBAC Authorization nella documentazione di Kubernetes.

- È possibile associare un provider di identità OIDC al cluster.
- Kubernetes non fornisce un provider di identità OIDC. È possibile utilizzare un provider di identità OIDC pubblico esistente oppure eseguire il proprio provider di identità. Per un elenco dei provider certificati, consultare OpenID Certification sul sito OpenID.
- L'URL dell'emittente del provider di identità OIDC deve essere accessibile pubblicamente, in modo che Amazon EKS possa individuare le chiavi di firma. Amazon EKS non supporta i provider di identità OIDC con certificati autofirmati.
- Puoi disabilitare l'autenticazione IAM nel cluster, perché sarà ancora necessaria per aggiungere i nodi a un cluster.
- Diversamente da un utente di provider di identità, un cluster Amazon EKS deve ancora essere creato da un principale IAM AWS . Questo perché il creatore del cluster interagisce con Amazon EKS APIs, anziché con APIs Kubernetes.
- Gli utenti autenticati dal provider di identità OIDC vengono elencati nel registro di controllo del cluster se i log sono attivati per il piano di controllo. CloudWatch Per ulteriori informazioni, consulta the section called "Abilitare o disabilitare i log del piano di controllo (control-plane)".
- Non è possibile accedere a Console di gestione AWS con un account di un provider OIDC. È possibile solo the section called "Accesso alle risorse di cluster" accedendo Console di gestione AWS con un account AWS Identity and Access Management.

Associazione di un provider di identità OIDC

Prima di associare un provider di identità OIDC al cluster, è necessario ottenere dal provider le seguenti informazioni:

URL dell'emittente

L'URL del provider di identità OIDC che consente al server API di individuare le chiavi di firma pubbliche per la verifica dei token. L'URL deve iniziare con `https://` e deve corrispondere alla richiesta `iss` nei token ID OIDC del provider. In conformità con lo standard OIDC, i componenti del percorso sono consentiti, ma i parametri di query non lo sono. In genere l'URL è costituito solo da un nome host, come `https://server.example.org` o `https://example.com`. Questo

URL dovrebbe puntare al livello sottostante `.well-known/openid-configuration` e dovrà essere accessibile pubblicamente tramite Internet.

ID client (noto anche come pubblico)

L'ID per l'applicazione client che effettua richieste di autenticazione al provider di identità OIDC.

È possibile associare un provider di identità utilizzando `eksctl` o il Console di gestione AWS.

Associazione di un provider di identità usando eksctl

1. Crea un file denominato `associate-identity-provider.yaml` con i seguenti contenuti. Sostituire i valori di esempio con i propri valori. I valori nella sezione `identityProviders` vengono ottenuti dal provider di identità OIDC. I valori sono obbligatori solo per il `name`, `type`, `issuerUrl`, e `clientId` impostazioni in `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**
>
> Non specificare `system:`, o qualsiasi parte di quella stringa, per `groupsPrefix` o `usernamePrefix`.

2. Creare il provider.

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. Per utilizzare `kubectl` per lavorare con il cluster e il provider di identità OIDC, consulta Using kubectl nella documentazione di Kubernetes.

Associare un provider di identità utilizzando la AWS console

1. Aprire la Console Amazon EKS.
2. Seleziona il cluster, poi seleziona la scheda Accesso.
3. Nella sezione Provider di identità OIDC, selezionare* Associa provider di identità*.
4. Nella pagina Associate provider di identità OIDC, inserire o selezionare le seguenti opzioni e quindi selezionare Associa.
   - In Nome, inserire un nome univoco per il provider.
   - Per URL dell'emittente, inserire l'URL per il provider. Questo URL deve essere accessibile in Internet.
   - Per ID client, inserire l'ID client del provider di identità OIDC (noto anche come audience).
   - Per Richiesta Nome Utente, inserire la richiesta da utilizzare come nome utente.
   - Per Richiesta di Gruppi, inserire la richiesta da utilizzare come gruppo dell'utente.
   - (Opzionale) Selezionare Opzioni avanzate, inserire o selezionare le seguenti informazioni.
     - Prefisso Nome utente: inserire un prefisso da anteporre alle richieste del nome utente. Il prefisso viene anteposto alle richieste di nome utente per evitare conflitti con i nomi esistenti. Se non si fornisce un valore e il nome utente è un valore diverso da `email`, il prefisso viene impostato per impostazione predefinita sul valore per URL dell'emittente. È possibile utilizzare il valore `-` per disabilitare tutti i prefissi. Non specificare `system:` o qualsiasi parte di quella stringa.
     - Prefisso Gruppi: inserire un prefisso da anteporre alle richieste di gruppo. Il prefisso viene anteposto alle richieste di gruppo per evitare conflitti con i nomi esistenti (ad

esempio `system: groups`). Ad esempio, il valore `oidc:` crea nomi di gruppo come `oidc:engineering` e `oidc:infra`. Non specificare `system:` o qualsiasi parte di quella stringa.

- Richieste necessarie: selezionare Aggiungi richiesta e inserire una o più coppie di valori chiave che descrivono le richieste necessarie nel token ID client. Le coppie descrivono le richieste necessarie nel Token ID. Se impostato, ogni richiesta viene verificata per essere presente nel token ID con un valore corrispondente.

  a. Per utilizzare `kubectl` per lavorare con il cluster e il provider di identità OIDC, consulta Using kubectl nella documentazione di Kubernetes.

Policy IAM di esempio

Se si desidera impedire che un provider di identità OIDC venga associato a un cluster, creare e associare le seguenti policy IAM agli account IAM degli amministratori Amazon EKS. Per ulteriori informazioni, consulta Creating IAM policies e Adding IAM identity permissions nella Guida per l'utente IAM e nelle Actions in Service Authorization Reference.

```
{
    "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": "*"
        }
    ]
}
```

La seguente policy di esempio consente l'associazione del provider di identità OIDC se il `clientID` è `kubernetes` e la `issuerUrl` è `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": "*"
        }
    ]
}
```

Dissociazione di un provider di identità OIDC dal cluster

Se si dissocia un provider di identità OIDC dal cluster, gli utenti inclusi nel provider non potranno più accedere al cluster. Tuttavia, è comunque possibile accedere al cluster con principali IAM.

1. Aprire la Console Amazon EKS.

2. Nella sezione Provider di identità OIDC, selezionare Dissocia, inserire il nome del provider di identità e quindi selezionare `Disassociate`.

Visualizza le risorse Kubernetes nel Console di gestione AWS

Puoi visualizzare le risorse Kubernetes implementate nel cluster utilizzando la Console di gestione AWS, Non puoi visualizzare le risorse Kubernetes con la CLI o AWS eksctl. Per visualizzare le risorse Kubernetes con uno strumento a riga di comando, utilizza kubectl.

> **Note**
>
> Per visualizzare la scheda Risorse e la sezione Nodi nella scheda Compute Console di gestione AWS, il principale IAM che stai utilizzando deve disporre di autorizzazioni IAM e Kubernetes specifiche. Per ulteriori informazioni, consulta the section called "Autorizzazioni richieste".

1. Aprire la Console Amazon EKS.
2. Nell'elenco Cluster, seleziona il cluster contenente le risorse Kubernetes da visualizzare.
3. Selezionare la scheda Risorse.
4. Seleziona un gruppo Resource type (Tipi di risorse) per il quale desideri visualizzare le risorse, ad esempio Workloads (Carichi di lavoro). Viene visualizzato un elenco dei tipi di risorse in tale gruppo.
5. Seleziona un tipo di risorsa, ad esempio Deployments (Implementazioni), nel gruppo Workloads (Carichi di lavoro). Viene visualizzata una descrizione del tipo di risorsa, un collegamento alla documentazione Kubernetes per ulteriori informazioni sul tipo di risorsa e un elenco di risorse simili implementate nel cluster. Se l'elenco è vuoto, nel cluster non sono presenti risorse appartenenti a quel tipo specifico.
6. Per visualizzare ulteriori informazioni su una risorsa, selezionarla. Prova gli esempi seguenti:
   - Seleziona il gruppo Workloads (Carichi di lavoro), scegli il tipo di risorsa Deployments (Implementazioni), quindi seleziona la risorsa coredns. Per impostazione predefinita, la selezione di una risorsa avviene in una Visualizzazione strutturata. Per alcuni tipi di risorse, nella Visualizzazione strutturata è presente una sezione Pod, Questa sezione elenca i pod gestiti dal carico di lavoro. È possibile selezionare qualsiasi pod presente in elenco per visualizzarne le informazioni. Non tutti i tipi di risorse visualizzano le informazioni nella Visualizzazione

strutturata. Selezionando Raw view (Visualizzazione non elaborata) nell'angolo in alto a destra della pagina della risorsa, viene visualizzata la risposta JSON completa dall'API Kubernetes per la risorsa.

- Seleziona il gruppo Cluster, quindi il tipo di risorsa Nodes (Nodi). Viene visualizzato un elenco di tutti i nodi del cluster. I nodi possono appartenere a qualsiasi tipo di nodo di Amazon EKS. Si tratta dello stesso elenco visualizzato nella sezione Nodes (Nodi) quando si seleziona la scheda Compute (Calcolo) del cluster. Seleziona una risorsa di nodo dall'elenco. Nella Vista strutturata è presente anche la sezione Pod Questa sezione mostra tutti i pod in esecuzione sul nodo.

Autorizzazioni richieste

Per visualizzare la scheda Risorse e la sezione Nodi nella scheda Compute Console di gestione AWS, il principale IAM che stai utilizzando deve disporre di autorizzazioni IAM e Kubernetes minime specifiche. È necessario che le autorizzazioni IAM e Kubernetes RBAC siano configurate correttamente. Completa i passaggi seguenti per assegnare le autorizzazioni richieste ai principali IAM.

1. Assicurati che `eks:AccessKubernetesApi` e le altre autorizzazioni IAM necessarie per visualizzare le risorse Kubernetes siano assegnati al principale IAM che stai utilizzando. Per ulteriori informazioni su come modificare le autorizzazioni per un principale IAM, consulta Controllo dell'accesso per i principali IAM nella Guida per l'utente di IAM. Per ulteriori informazioni su come modificare le autorizzazioni per un ruolo, consulta Modifica di una policy di autorizzazioni del ruolo (console) nella Guida per l'utente IAM.

   La seguente policy esemplificativa include le autorizzazioni necessarie affinché un principale possa visualizzare le risorse Kubernetes per tutti i cluster dell'account. Sostituiscilo con l'ID del tuo account. *111122223333* 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/*"
        }
    ]
}
```

Per visualizzare i nodi in cluster connessi, il ruolo IAM del connettore Amazon EKS dovrebbe essere in grado di rappresentare il principale nel cluster. Ciò consente ad Amazon EKS Connector di mappare il principale IAM a un utilizzatore Kubernetes.

2. Configura le autorizzazioni RBAC di Kubernetes utilizzando le voci di accesso EKS.

   Cosa sono gli EKS Access Entries?

   Le voci di accesso EKS sono un modo semplificato per concedere ai principali IAM (utenti e ruoli) l'accesso al tuo cluster Kubernetes. Invece di gestire manualmente le risorse RBAC di Kubernetes e le `aws-auth` ConfigMap, le voci di accesso gestiscono automaticamente la mappatura tra le autorizzazioni IAM e Kubernetes utilizzando le politiche gestite fornite da. AWS Per informazioni dettagliate the section called "Voci di accesso" sulle voci di accesso, consulta. Per informazioni sui criteri di accesso disponibili e sulle relative autorizzazioni, vedere Autorizzazioni dei criteri di accesso.

   Puoi allegare le autorizzazioni Kubernetes per accedere alle voci in due modi:

   - Usa una politica di accesso: le politiche di accesso sono modelli di autorizzazioni Kubernetes predefiniti gestiti da. AWS Questi forniscono set di autorizzazioni standardizzati per casi d'uso comuni.
   - Fai riferimento a un gruppo Kubernetes: se associ un'identità IAM a un gruppo Kubernetes, puoi creare risorse Kubernetes che concedono le autorizzazioni al gruppo. Per ulteriori informazioni, consulta Using RBAC Authorization nella documentazione di Kubernetes.

a. Crea una voce di accesso per il tuo principale IAM utilizzando la AWS CLI. Sostituisci *my-cluster* con il nome del cluster. Sostituisci *111122223333* con l'ID del tuo account.

```
aws eks create-access-entry \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role
```

Di seguito viene riportato un output di esempio.

```
{
    "accessEntry": {
        "clusterName": "my-cluster",
        "principalArn": "arn:aws: iam::111122223333:role/my-console-viewer-
role",
        "kubernetesGroups": [],
        "accessEntryArn": "arn:aws: eks:region-code:111122223333:access-
entry/my-cluster/role/111122223333/my-console-viewer-role/
abc12345-1234-1234-1234-123456789012",
        "createdAt": "2024-03-15T10:30:45.123000-07:00",
        "modifiedAt": "2024-03-15T10:30:45.123000-07:00",
        "tags": {},
        "username": "arn:aws: iam::111122223333:role/my-console-viewer-role",
        "type": "STANDARD"
    }
}
```

b. Associa una policy alla voce di accesso. Per visualizzare le risorse Kubernetes, utilizza: `AmazonEKSViewPolicy`

```
aws eks associate-access-policy \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role \
    --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy \
    --access-scope type=cluster
```

Di seguito viene riportato un output di esempio.

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws: iam::111122223333:role/my-console-viewer-role",
    "associatedAt": "2024-03-15T10:31:15.456000-07:00"
```

```
}
```

Per un accesso specifico allo spazio dei nomi, puoi estendere la policy a namespace specifici:

```
aws eks associate-access-policy \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role \
    --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy \
    --access-scope type=namespace,namespaces=default,kube-system
```

c. Verifica che la voce di accesso sia stata creata correttamente:

```
aws eks describe-access-entry \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role
```

d. Elenca le politiche associate per confermare l'associazione delle politiche:

```
aws eks list-associated-access-policies \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role
```

Di seguito viene riportato un output di esempio.

```
{
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws: eks::aws:cluster-access-policy/
AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster"
            },
            "associatedAt": "2024-03-15T10:31:15.456000-07:00",
            "modifiedAt": "2024-03-15T10:31:15.456000-07:00"
        }
    ]
}
```

CloudTrail visibilità

Durante la visualizzazione delle risorse Kubernetes, nei log verrà visualizzato il seguente nome dell'operazione: CloudTrail

- `AccessKubernetesApi`- Durante la lettura o la visualizzazione di risorse

Questo CloudTrail evento fornisce una traccia di controllo dell'accesso in lettura alle risorse Kubernetes.

> **Note**
>
> Questo nome dell'operazione viene visualizzato nei CloudTrail log solo a scopo di controllo. Non è un'azione IAM e non può essere utilizzata nelle dichiarazioni delle politiche IAM. Per controllare l'accesso in lettura alle risorse Kubernetes tramite le policy IAM, utilizza l'`eks:AccessKubernetesApi`autorizzazione come mostrato nella sezione. the section called "Autorizzazioni richieste"

Concedi AWS ai servizi l'accesso in scrittura a Kubernetes APIs

Autorizzazioni richieste

Per consentire ai AWS servizi di eseguire operazioni di scrittura sulle risorse Kubernetes nel tuo cluster Amazon EKS, devi concedere sia le `eks:AccessKubernetesApi` autorizzazioni che quelle IAM. `eks:MutateViaKubernetesApi`

Ad esempio, Amazon SageMaker HyperPod utilizza queste autorizzazioni per abilitare la distribuzione di modelli da SageMaker AI Studio. Per ulteriori informazioni, consulta Configurare le autorizzazioni JavaScript SDK opzionali nella Amazon SageMaker AI Developer Guide.

> **Important**
>
> Le operazioni di scrittura come la creazione, l'aggiornamento e l'eliminazione richiedono entrambe le autorizzazioni: se una delle due autorizzazioni è mancante, le operazioni di scrittura avranno esito negativo.

CloudTrail visibilità

Durante l'esecuzione di operazioni di scrittura sulle risorse Kubernetes, nei log verranno visualizzati nomi di operazioni specifici: CloudTrail

- `createKubernetesObject`- Durante la creazione di nuove risorse
- `updateKubernetesObject`- Quando si modificano risorse esistenti
- `deleteKubernetesObject`- Quando si rimuovono risorse

Questi CloudTrail eventi forniscono audit trail dettagliati di tutte le modifiche apportate alle risorse Kubernetes.

> **Note**
>
> Questi nomi di operazioni vengono visualizzati nei CloudTrail log solo a scopo di controllo. Non sono azioni IAM e non possono essere utilizzate nelle dichiarazioni politiche IAM. Per controllare l'accesso in scrittura alle risorse Kubernetes tramite le policy IAM, utilizza l'`eks:MutateViaKubernetesApi`autorizzazione come mostrato nella sezione. the section called “Autorizzazioni richieste”

Connettere kubectl a un cluster EKS creando un file kubeconfig

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

In questo argomento viene creato un file `kubeconfig` per il cluster (o ne viene aggiornato uno esistente).

Lo strumento della linea di comando `kubectl` utilizza le informazioni di configurazione nei file `kubeconfig` per comunicare con il server API di un cluster. Per ulteriori informazioni, consulta la sezione Organizing Cluster Access Using kubeconfig Files nella documentazione di Kubernetes.

Amazon EKS utilizza il comando `aws eks get-token` con `kubectl` per l'autenticazione del cluster. Per impostazione predefinita, la AWS CLI utilizza le stesse credenziali restituite con il seguente comando:

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

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Un ruolo o un utente IAM con l'autorizzazione a utilizzare l'azione API `eks:DescribeCluster` per il cluster che specifichi. Per ulteriori informazioni, consulta the section called "Policy basate sull'identità". Se utilizzi un'identità del tuo provider OpenID Connectper accedere al cluster, consulta la sezione Using kubectl nella documentazione di Kubernetes per creare o aggiornare il file `kube config`.

Creazione automatica del file **kubeconfig**

- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.

- Autorizzazione a utilizzare l'operazione API `eks:DescribeCluster` per il cluster specificato. Per ulteriori informazioni, consulta the section called "Policy basate sull'identità".

  1. Crea o aggiorna un file `kubeconfig` per il cluster. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster e sostituiscilo *`my-cluster`* con il nome del cluster.

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

     Per impostazione predefinita, il file di configurazione risultante viene creato nel percorso `kubeconfig` predefinito (`.kube`) nella home directory o unito a un file `config` esistente in quel percorso. È possibile specificare un altro percorso con l'opzione `--kubeconfig`.

     Puoi specificare un ARN del ruolo IAM con l'opzione `--role-arn` da utilizzare per l'autenticazione quando esegui comandi `kubectl`. Altrimenti, viene utilizzato il principale IAM nella catena di credenziali AWS CLI o SDK predefinita. È possibile visualizzare l'identità AWS CLI o SDK predefinita eseguendo il comando. `aws sts get-caller-identity`

     Per tutte le opzioni disponibili, esegui il *`aws eks update-kubeconfig help`* comando o consulta update-kubeconfig nella CLI AWS Command Reference.

  2. Prova la configurazione.

     ```
     kubectl get svc
     ```

     Di seguito viene riportato un output di esempio:

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

     Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso negato o non autorizzato (`kubectl`)" nell'argomento relativo alla risoluzione dei problemi.

Concessione ai carichi di lavoro Kubernetes dell'accesso a AWS utilizzando gli account di servizio Kubernetes

A Kubernetes service account provides an identity for processes that run in a Pod. For more information see Gestione degli account di servizio 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 "Credenziali con IRSA" or the section called "Pod Identity".

Token dell'account di servizio

La funzionalità BoundServiceAccountTokenVolume è abilitata per impostazione predefinita nelle versioni di Kubernetes. Questa funzionalità migliora la sicurezza dei token dell'account di servizio, consentendo ai carichi di lavoro in esecuzione su Kubernetes di richiedere token Web JSON associati a pubblico, tempo e chiavi. I token dell'account di servizio scadono entro un'ora, Nelle versioni precedenti di Kubernetes, i token non avevano scadenza. Di conseguenza, i client che si basano su questi token devono aggiornarli entro un'ora. Gli SDK client Kubernetes seguenti aggiornano automaticamente i token entro il periodo di tempo richiesto:

- Go versione `0.15.7` e successive
- Python versione `12.0.0` e successive
- Java versione `9.0.0` e successive
- JavaScript versione `0.10.3` e successive
- Ramo `master` di Ruby
- Haskell versione `0.3.0.0`
- Versione C# `7.0.5` e successive

Se il carico di lavoro utilizza una versione precedente del client, è necessario aggiornarla. Per consentire una migrazione agevole dei client verso i più recenti token dell'account di servizio con limite di tempo, Kubernetes proroga il periodo di scadenza dei token, rispetto a quello predefinito di un'ora. Per i cluster Amazon EKS, il periodo di scadenza è prorogato a 90 giorni. Il server API Kubernetes del cluster Amazon EKS rifiuta le richieste con token che sono più vecchi di 90 giorni. Consigliamo di controllare le applicazioni e le relative dipendenze per assicurarti che gli SDK del client Kubernetes corrispondano o siano successivi alle versioni elencate sopra.

Quando il server API riceve richieste con token più vecchie di un'ora, le annota nel log eventi di controllo API con `annotations.authentication.k8s.io/stale-token`. Il valore dell'annotazione è simile a quello riportato di seguito:

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold:
 4185802.
```

Se nel cluster è abilitata la registrazione del piano di controllo (control-plane), le annotazioni di trovano nei registri di controllo. Per identificare tutti i pod del cluster Amazon EKS che utilizzano token obsoleti, è possibile utilizzare la seguente query di Log Insights di Cloudwatch:

```
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
```

`subject` si riferisce all'account di servizio utilizzato dal pod. `elapsedtime` indica in secondi il tempo trascorso dalla lettura dell'ultimo token. Le richieste al server API vengono negate quando `elapsedtime` supera i 90 giorni (7.776.000 secondi). Per utilizzare una delle versioni elencate sopra e aggiornare automaticamente il token, devi aggiornare in modo proattivo l'SDK client Kubernetes delle applicazioni. Se il token dell'account di servizio utilizzato è prossimo ai 90 giorni e non si ha tempo sufficiente per aggiornare le versioni dell'SDK client prima della scadenza, è possibile terminare i pod esistenti e crearne di nuovi. Ciò comporta una ridefinizione del token dell'account di servizio, concedendo agli utenti altri 90 giorni per aggiornare gli SDK della versione client.

Se il pod fa parte di un'implementazione, è consigliabile terminare i pod eseguendo un rollout in modo da mantenere un'elevata disponibilità. Per eseguire questa operazione, utilizzare il comando seguente. Sostituisci *`my-deployment`* con il nome dell'implementazione.

```
kubectl rollout restart deployment/my-deployment
```

Componenti aggiuntivi del cluster

I componenti aggiuntivi del cluster seguenti sono stati aggiornati per utilizzare gli SDK client Kubernetes che riconfigurano automaticamente i token dell'account di servizio. Assicurati che le versioni elencate, o versioni successive, siano installate sul tuo cluster.

- Plugin helper per le metriche e plugin CNI di Amazon VPC per Kubernetes versione `1.8.0` e successive. Per verificare la versione attuale o aggiornarla, consulta the section called "CNI di Amazon VPC" e cni-metrics-helper.
- Versione CoredNS `1.8.4` e successive. Per verificare la versione attuale o aggiornarla, consulta the section called "CoreDNS".
- AWS Load Balancer Controller versione `2.0.0` e successive. Per verificare la versione attuale o aggiornarla, consulta the section called " AWS Controller Load Balancer".

- Versione `kube-proxy` attuale. Per verificare la versione attuale o aggiornarla, consulta the section called "kube-proxy".
- AWS per Fluent Bit versione `2.25.0` o successive. Per aggiornare la versione attuale, consulta Versioni su GitHub.
- Versione dell'immagine di Fluentd 1.14.6-1.2 o versione successiva e plugin di filtro Fluentd per la versione dei metadati Kubernetes 2.11.1 o successive.

Concedere le autorizzazioni di AWS Identity and Access Management ai carichi di lavoro sui cluster Amazon Elastic Kubernetes Service

Amazon EKS offre due modi per concedere le autorizzazioni di AWS Identity and Access Management ai carichi di lavoro eseguiti nei cluster Amazon EKS: ruoli IAM per gli account di servizio ed EKS Pod Identity.

Ruoli IAM per gli account di servizio

Ruoli IAM per gli account di servizio (IRSA) configura le applicazioni Kubernetes in esecuzione su AWS con autorizzazioni IAM granulari per accedere a varie altre risorse AWS come bucket Amazon S3, tabelle Amazon DynamoDB e altro ancora. Puoi eseguire più applicazioni contemporaneamente nello stesso cluster Amazon EKS e assicurarti che ogni applicazione disponga solo del set minimo di autorizzazioni di cui ha bisogno. IRSA è stato creato per supportare varie opzioni di implementazione Kubernetes supportate da AWS, come Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service su AWS, e cluster autogestiti su istanze Amazon EC2. Pertanto, IRSA è stato creato utilizzando servizi AWS fondamentali come IAM e non ha assunto alcuna dipendenza diretta dal servizio Amazon EKS e dall'API EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

EKS Pod Identity

EKS Pod Identity offre agli amministratori dei cluster un flusso di lavoro semplificato per l'autenticazione delle applicazioni al fine di accedere a varie altre risorse AWS come bucket Amazon S3, tabelle Amazon DynamoDB e altro ancora. EKS Pod Identity è solo per EKS e, di conseguenza, semplifica il modo in cui gli amministratori dei cluster possono configurare le applicazioni Kubernetes per ottenere le autorizzazioni IAM. Queste autorizzazioni possono ora essere facilmente configurate con meno passaggi direttamente tramite Console di gestione AWS, API EKS e AWS CLI e non è necessario eseguire alcuna operazione all'interno del cluster su nessun oggetto Kubernetes. Gli amministratori del cluster non devono passare dai servizi EKS ai servizi IAM o utilizzare operazioni IAM privilegiate per configurare le autorizzazioni richieste

dalle applicazioni. I ruoli IAM possono ora essere utilizzati su più cluster senza la necessità di aggiornare la policy di attendibilità dei ruoli durante la creazione di nuovi cluster. Le credenziali IAM fornite da EKS Pod Identity includono tag di sessione dei ruoli, con attributi come nome del cluster, namespace, nome dell'account di servizio. I tag di sessione di ruolo consentono agli amministratori di creare un singolo ruolo che può funzionare su più account di servizio, permettendo l'accesso alle risorse AWS in base ai tag corrispondenti. Per ulteriori informazioni, consulta the section called "Pod Identity".

Confronto tra EKS Pod Identity e IRSA

A un livello superiore, sia EKS Pod Identity che IRSA concedono autorizzazioni IAM alle applicazioni in esecuzione su cluster Kubernetes. Ma sono fondamentalmente diversi nel modo in cui vengono configurati, nei limiti supportati e nelle funzionalità abilitate. Di seguito, confrontiamo alcuni degli aspetti chiave di entrambe le soluzioni.

> **Note**
>
> AWS consiglia di utilizzare EKS Pod Identity per concedere l'accesso alle risorse AWS dei pod ogni volta che è possibile. Per ulteriori informazioni, consulta the section called "Pod Identity".

Attributo	EKS Pod Identity	IRSA
Estensibilità dei ruoli	È necessario configurare ogni ruolo una volta per instaurare un rapporto di attendibilità con il principale di servizio Amazon EKS introdotto di recente, `pods.eks.amazonaws.com` . Dopo questo passaggio una tantum, non è necessario aggiornare la policy di attendibilità del ruolo ogni volta che è utilizzato in un nuovo cluster.	È necessario aggiornare la policy di attendibilità del ruolo IAM con il nuovo endpoint del provider OIDC del cluster EKS ogni volta che si desidera utilizzare il ruolo in un nuovo cluster.

Attributo	EKS Pod Identity	IRSA
Scalabilità del cluster	EKS Pod Identity non richiede agli utenti di configurare il provider OIDC di IAM, quindi questo limite non si applica.	Ciascun cluster EKS ha un URL emittente OpenID Connect (OIDC) associato ad esso. Per utilizzare IRSA, è necessario creare un provider OpenID Connect univoco per ogni cluster EKS in IAM. IAM ha un limite globale predefinito di 100 provider OIDC per ciascun account AWS. Se prevedi di avere più di 100 cluster EKS per ciascun account AWS con IRSA, raggiungerai il limite del provider OIDC di IAM.

Attributo	EKS Pod Identity	IRSA
Scalabilità dei ruoli	EKS Pod Identity non richiede agli utenti di definire una relazione di attendibilità tra il ruolo IAM e l'account di servizio nella policy di attendibilità, quindi questo limite non è applicato.	In IRSA, definisci la relazione di attendibilità tra un ruolo IAM e un account di servizio nella policy di attendibilità del ruolo. Per impostazione predefinita, la lunghezza della dimensione della policy di attendibilità è `2048`. Ciò significa che in genere è possibile definire 4 relazioni di attendibilità in un'unica policy di attendibilità. Sebbene sia possibile aumentare il limite di lunghezza della policy di attendibilità, in genere si è limitati a un massimo di 8 relazioni di attendibilità all'interno di una singola policy di attendibilità.
Utilizzo della quota API di STS	EKS Pod Identity semplifica l'invio delle credenziali AWS ai pod e non richiede il codice per effettuare chiamate direttamente con il AWS Security Token Service (STS). Il servizio EKS gestisce l'assunzione dei ruoli e fornisce le credenziali alle applicazioni scritte utilizzando AWS SDK nei pod senza che i pod comunichino con AWS STS o utilizzino una quota API di STS.	In IRSA, le applicazioni scritte utilizzando AWS SDK nei pod utilizzano i token per chiamare l'API `AssumeRoleWithWebIdentity` su AWS Security Token Service (STS). A seconda della logica del codice di AWS SDK, è possibile che il codice effettui chiamate non necessarie a AWS STS e riceva errori di limitazione (della larghezza di banda della rete).

Attributo	EKS Pod Identity	IRSA
Riutilizzabilità dei ruoli	Le credenziali temporane e AWS STS fornite da EKS Pod Identity includono tag di sessione dei ruoli, come nome del cluster, namespace e nome dell'account di servizio. I tag di sessione dei ruoli permettono agli amministratori di creare un singolo ruolo IAM che può essere utilizzato con più account di servizio, con diverse autorizzazioni effettive , consentendo l'accesso alle risorse AWS in base ai tag ad essi collegati. Questo ruolo è chiamato anche controllo degli accessi basato su attributi (ABAC). Per ulteriori informazi oni, consulta the section called "Concessione dell'accesso dei pod".	I tag di sessione AWS STS non sono supportati. È possibile riutilizzare un ruolo tra i cluster, ma ogni pod riceve tutte le autorizzazioni del ruolo.
Ambienti supportati	EKS Pod Identity è disponibile solo su Amazon EKS.	IRSA può essere utilizzat o su cluster Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service su AWS e su cluster Kubernetes autogestiti su istanze Amazon EC2.

Attributo	EKS Pod Identity	IRSA
Versioni EKS supportate	Tutte le versioni del cluster EKS supportate. Per le versioni delle piattaforme specifiche, consulta the section called "Versioni del cluster EKS Pod Identity".	Tutte le versioni del cluster EKS supportate.

Ruoli IAM per gli account di servizio

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

Le applicazioni nei contenitori di un Pod possono utilizzare un AWS SDK o la AWS CLI per effettuare richieste API AWS ai servizi AWS utilizzando le autorizzazioni Identity and Access Management (IAM). Le applicazioni devono firmare le proprie richieste AWS API con credenziali. AWS I ruoli IAM per gli account di servizio (IRSA) offrono la possibilità di gestire le credenziali per le applicazioni, in modo simile al modo in cui i profili di EC2 istanza Amazon forniscono le credenziali alle istanze Amazon. EC2 Invece di creare e distribuire AWS le tue credenziali ai contenitori o utilizzare il ruolo dell' EC2 istanza Amazon, associ un ruolo IAM a un account di servizio Kubernetes e configuri i tuoi Pods per utilizzare l'account di servizio. Non puoi utilizzare i ruoli IAM per gli account di servizio con cluster locali per Amazon EKS on AWS Outposts.

La caratteristica dei ruoli IAM per gli account di servizio offre i seguenti vantaggi:

- Privilegio minimo: è possibile definire l'ambito delle autorizzazioni IAM per un account di servizio; solo i pod che utilizzano tale account avranno accesso alle autorizzazioni definite. Questa caratteristica elimina anche la necessità di soluzioni di terze parti, ad esempio `kiam` o `kube2iam`.
- Isolamento delle credenziali: quando l'accesso all'Amazon EC2 Instance Metadata Service (IMDS) è limitato, i contenitori di un Pod possono recuperare solo le credenziali per il ruolo IAM associato all'account di servizio utilizzato dal contenitore. Un container non ha mai accesso alle credenziali utilizzate da altri container in altri pod. Se IMDS non è soggetto a restrizioni, i container del pod hanno accesso anche al ruolo IAM del nodo Amazon EKS potrebbero essere in grado di accedere

alle credenziali dei ruoli IAM di altri pod sullo stesso nodo. Per ulteriori informazioni, consulta Limitazione dell'accesso al profilo dell'istanza assegnato al nodo worker.

> **Note**
>
> I pod configurati con `hostNetwork: true` avranno sempre accesso IMDS, ma la AWS SDKs CLI e la CLI utilizzeranno le credenziali IRSA quando abilitate.

- Verificabilità: la registrazione degli accessi e degli eventi è disponibile per garantire un controllo retrospettivo. AWS CloudTrail

> **Important**
>
> I container non costituiscono un limite di sicurezza e l'uso dei ruoli IAM per gli account di servizio non cambia questa situazione. I pod assegnati allo stesso nodo condivideranno un kernel e potenzialmente altre risorse a seconda della configurazione del pod. Sebbene i pod in esecuzione su nodi separati siano isolati a livello di elaborazione, esistono applicazioni di nodo che dispongono di autorizzazioni aggiuntive nell'API Kubernetes oltre all'ambito di una singola istanza. Alcuni esempi sono `kubelet`, `kube-proxy`, i driver di archiviazione CSI o le tue applicazioni Kubernetes.

Abilita i ruoli IAM per gli account di servizio completando le seguenti procedure:

1. Creazione di un provider OIDC IAM per il cluster: questa operazione viene completata solo una sola volta per cluster.

   > **Note**
   >
   > Se è stato abilitato l'endpoint VPC EKS, non è possibile accedere all'endpoint del servizio OIDC EKS dall'interno di tale VPC. Di conseguenza, le operazioni come la creazione di un provider OIDC con `eksctl` nel VPC non funzioneranno e comporteranno un timeout durante il tentativo di richiesta di `https://oidc.eks.region.amazonaws.com`. Segue un messaggio di errore di esempio:

```
server cant find oidc.eks.region.amazonaws.com: NXDOMAIN
```

Per completare questo passaggio, puoi eseguire il comando all'esterno del VPC, ad esempio in AWS CloudShell o su un computer connesso a Internet. In alternativa, puoi creare un resolver condizionale di tipo split-horizon nel VPC, come il risolutore Route 53, per utilizzare un resolver diverso per l'URL emittente OIDC e non utilizzare il DNS VPC. Per un esempio di inoltro condizionale in CoredNS, consulta la richiesta di funzionalità Amazon EKS su. GitHub

2. Assegna i ruoli IAM agli account del servizio Kubernetes: completa questa procedura per ogni set univoco di autorizzazioni che desideri abbia un'applicazione.
3. Configura i Pod per utilizzare un account di servizio Kubernetes: completa questa procedura per ogni Pod che deve accedere ai servizi. AWS
4. Usa IRSA con l' AWS SDK: conferma che il carico di lavoro utilizzi un AWS SDK di una versione supportata e che il carico di lavoro utilizzi la catena di credenziali predefinita.

Informazioni di base su IAM, Kubernetes e OpenID Connect (OIDC)

Nel 2014, AWS Identity and Access Management ha aggiunto il supporto per le identità federate utilizzando OpenID Connect (OIDC). Questa funzionalità consente di autenticare le chiamate AWS API con i provider di identità supportati e di ricevere un token web JSON (JWT) OIDC valido. Puoi passare questo token all'operazione `AssumeRoleWithWebIdentity` API AWS STS e ricevere credenziali di ruolo temporaneo IAM. Puoi utilizzare queste credenziali per interagire con qualsiasi AWS servizio, inclusi Amazon S3 e DynamoDB.

Ogni token JWT è firmato da una coppia di chiavi di firma. Le chiavi vengono fornite dal provider OIDC gestito da Amazon EKS e la chiave privata ruota ogni 7 giorni. Amazon EKS conserva le chiavi pubbliche fino alla loro scadenza. Se si connettono client OIDC esterni, si tenga presente che è necessario aggiornare le chiavi di firma prima della scadenza della chiave pubblica. Scopri Fetch signing keys to validate OIDC tokens.

Kubernetes utilizza da tempo gli account di servizio come sistema di identità interno. I pod possono eseguire l'autenticazione con il server API Kubernetes utilizzando un token con montaggio automatico (che è un token JWT non OIDC) che solo il server API Kubernetes può convalidare. Questi token legacy dell'account di servizio non hanno scadenza e la rotazione della chiave di firma è un processo difficile. Nella versione `1.12` di Kubernetes, è stato aggiunto il supporto per una nuova

funzionalità `ProjectedServiceAccountToken`. Questa funzionalità è un token Web OIDC JSON che contiene anche l'identità dell'account del servizio e supporta un pubblico configurabile.

Amazon EKS ora ospita un endpoint di rilevamento OIDC pubblico per ogni cluster, contenente le chiavi di firma per i token Web `ProjectedServiceAccountToken` JSON, in modo che i sistemi esterni, ad esempio IAM, possano convalidare e accettare i token OIDC emessi da Kubernetes.

Per creare un provider di identità IAM OIDC per il cluster

Il cluster ha un URL emittente OpenID Connect (OIDC) associato. Per utilizzare i ruoli AWS Identity and Access Management (IAM) per gli account di servizio, deve esistere un provider IAM OIDC per l'URL dell'emittente OIDC del cluster.

Prerequisiti

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Un file `kubectl config` esistente che contiene la configurazione del cluster. Per creare un file `kubectl config`, consulta the section called "Accedere al cluster con kubectl".

È possibile creare un gestore OIDC IAM per il cluster utilizzando `eksctl` o la Console di gestione AWS.

Creare un gestore OIDC (eksctl)

1. La versione `0.215.0` o quelle successive dello strumento a riga di comando `eksctl` deve essere installata sul dispositivo o nella AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Determinare l'ID dell'emittente OIDC per il cluster.

   Recuperare gli ID dell'emittente OIDC del cluster e archiviarlo in una variabile. Sostituire `<my-cluster>` con il proprio valore.

   ```
   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. Determinare se un gestore OIDC IAM con l'ID del cluster è già presente nell'account.

   ```
   aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
   ```

   Se viene restituito un output, il cluster dispone già di un gestore OIDC IAM ed è possibile saltare al passaggio successivo. Se non viene restituito alcun output, devi creare un provider OIDC IAM per il cluster.
4. Creare un provider di identità OIDC IAM per il cluster con il comando seguente.

   ```
   eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
   ```

   > **Note**
   >
   > Se è stato abilitato l'endpoint VPC EKS, non è possibile accedere all'endpoint del servizio OIDC EKS dall'interno di tale VPC. Di conseguenza, operazioni come la creazione di un gestore OIDC con `eksctl` nel VPC non funzioneranno e comporteranno un timeout. Segue un messaggio di errore di esempio:
   >
   > ```
   > ** server cant find oidc.eks.<region-code>.amazonaws.com: NXDOMAIN
   > ```

Per completare questo passaggio, puoi eseguire il comando all'esterno del VPC, ad esempio in AWS CloudShell o su un computer connesso a Internet. In alternativa, puoi creare un resolver condizionale di tipo split-horizon nel VPC, come il risolutore Route 53, per utilizzare un resolver diverso per l'URL emittente OIDC e non utilizzare il DNS VPC. Per un esempio di inoltro condizionale in CoredNS, consulta la richiesta di funzionalità Amazon EKS su. GitHub

Crea un provider OIDC (Console)AWS

1. Aprire la Console Amazon EKS.
2. Nel riquadro a sinistra, seleziona Cluster, quindi seleziona il nome del cluster nella pagina Cluster.
3. Nella sezione Dettagli della scheda Panoramica, annota il valore dell'URL del provider OpenID Connect.
4. Apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
5. Nel pannello di navigazione, scegli Identity Providers (Provider di identità) in Access management (Gestione accesso). Se un Provider nell'elenco corrisponde all'URL del cluster, significa che si dispone già di un provider per il cluster. Se nell'elenco non è presente un gestore che corrisponde all'URL del cluster, è necessario crearne uno.
6. Per creare un provider, selezionare Aggiungi provider.
7. Per Tipo di provider, selezionare OpenID Connect.
8. Per URL provider, inserire l'URL del gestore OIDC del cluster.
9. Per Destinatari, inserisci `sts.amazonaws.com`
10. (Facoltativo) Aggiungere eventuali tag, ad esempio un tag per identificare quale cluster è assegnato a questo gestore.
11. Scegli Aggiungi provider.

Fase successiva: the section called "Assegnazione di un ruolo IAM"

Assegnare ruoli IAM agli account di servizio Kubernetes

Questo argomento spiega come configurare un account di servizio Kubernetes per assumere un ruolo AWS Identity and Access Management (IAM). Tutti i Pod configurati per utilizzare l'account di servizio possono quindi accedere a qualsiasi AWS servizio a cui il ruolo dispone delle autorizzazioni per accedere.

Prerequisiti

- Un cluster esistente. Se non ne possiedi già uno, crealo seguendo una delle guide in Nozioni di base.
- Creare un provider di identità OpenID Connect (OIDC) IAM per il cluster. Per sapere se disponi di un account o, per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Un file `kubectl config` esistente che contiene la configurazione del cluster. Per creare un file `kubectl config`, consulta the section called "Accedere al cluster con kubectl".

Fase 1: creare la policy IAM

Se desideri associare una policy IAM esistente al ruolo IAM, passa al prossimo passaggio.

1. Creare una policy IAM Puoi creare la tua policy o copiare una policy AWS gestita che già concede alcune delle autorizzazioni di cui hai bisogno e personalizzarla in base alle tue esigenze specifiche. Per ulteriori informazioni, consulta Creating IAM policies nella Guida per l'utente di IAM.
2. Crea un file che includa le autorizzazioni per i AWS servizi a cui desideri che i tuoi Pod accedano. Per un elenco di tutte le azioni per tutti i AWS servizi, consulta il Service Authorization Reference.

   Puoi eseguire il comando seguente per creare un file policy di esempio che consenta l'accesso in sola lettura a un bucket Amazon S3. Facoltativamente, puoi archiviare le informazioni di configurazione o uno script di bootstrap in questo bucket; i container nel pod possono leggere il file dal bucket e caricarlo nell'applicazione. Se desideri creare questa policy di esempio, copia i

seguenti contenuti sul dispositivo. Sostituiscilo *my-pod-secrets-bucket* con il nome del tuo bucket ed esegui il comando.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

3. Creare la policy IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

Fase 2: creare e associare il ruolo IAM

Creare un ruolo IAM e associarlo a un account di servizio Kubernetes. È possibile utilizzare uno dei due `eksctl` o la AWS CLI.

Creare e associare il ruolo (eksctl)

Questo comando `eksctl` crea un account di servizio Kubernetes nel namespace specificato, crea un ruolo IAM (se non esiste) con il nome specificato, allega un ARN della policy IAM esistente al ruolo e annota l'account di servizio con l'ARN del ruolo IAM. Assicurati di sostituire i valori segnaposto di esempio in questo comando con i valori specifici. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `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 il ruolo o l'account del servizio esiste già, il comando precedente potrebbe non riuscire. `eksctl` ha diverse opzioni che puoi fornire in queste situazioni. Per ulteriori informazioni, esegui `eksctl create iamserviceaccount --help`.

Creazione e associazione di ruoli (AWS CLI)

Se disponi di un account di servizio Kubernetes esistente che deve assumere un ruolo IAM, puoi saltare questa fase.

1. Creare un account di servizio Kubernetes. Copia i seguenti contenuti sul dispositivo. Sostituiscilo *`my-service-account`* con il nome desiderato e *`default`* con un namespace diverso, se necessario. Se si modifica*`default`*, lo spazio dei nomi deve già esistere.

```
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. Imposta l'ID AWS del tuo account su una variabile di ambiente con il seguente comando.

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

3. Imposta il provider di identità OIDC del cluster su una variabile di ambiente con il comando seguente. Sostituisci *`my-cluster`* con il nome del 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. Imposta le variabili per lo spazio dei nomi e il nome dell'account del servizio. Sostituisci *`my-service-account`* con l'account del servizio Kubernetes che desideri assumere il ruolo. Sostituisci *`default`* con lo spazio dei nomi dell'account del servizio.

```
export namespace=default
```

```
export service_account=my-service-account
```

5. Per creare una policy di attendibilità del ruolo IAM, esegui il comando seguente. Se desideri consentire a tutti gli account di servizio all'interno di un namespace di utilizzare il ruolo, copia i seguenti contenuti sul dispositivo. Sostituisci *`StringEquals`* con `StringLike` e sostituisci *`$service_account`* con. * Puoi aggiungere più voci nelle condizioni `StringEquals` e `StringLike` riportate di seguito per utilizzare più account di servizio o namespace con il ruolo. Per consentire ai ruoli di un AWS account diverso da quello in cui si trova il cluster di assumere il ruolo, consulta the section called "IAM multi-account" per ulteriori informazioni.

```
{
  "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. Crea il ruolo. Sostituisci *`my-role`* con un nome per il ruolo IAM e *`my-role-description`* con una descrizione per il tuo ruolo.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

7. Allegare una policy IAM al ruolo. Sostituisci *`my-role`* con il nome del ruolo IAM e *`my-policy`* con il nome di una policy esistente che hai creato.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws: iam::
$account_id:policy/my-policy
```

8. Annota l'account del servizio con il nome della risorsa Amazon (ARN) del ruolo IAM che desideri che l'account del servizio assuma. Sostituisci *my-role* con il nome del ruolo IAM esistente. Supponiamo di aver consentito a un ruolo di un AWS account diverso da quello in cui si trova il cluster di assumere il ruolo in un passaggio precedente. Quindi, assicurati di specificare l' AWS account e il ruolo dell'altro account. Per ulteriori informazioni, consulta the section called "IAM multi-account".

```
kubectl annotate serviceaccount -n $namespace $service_account eks.amazonaws.com/
role-arn=arn:aws: iam::$account_id:role/my-role
```

9. (Facoltativo) Configura l'endpoint del servizio AWS Security Token per un account di servizio. AWS consiglia di utilizzare un endpoint AWS STS regionale anziché un endpoint globale. Ciò riduce la latenza, fornisce una ridondanza integrata e aumenta la validità del token di sessione.

Fase 3: confermare la configurazione

1. Confermare che la policy di attendibilità del ruolo IAM sia configurata correttamente.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

   Di seguito viene riportato un output di esempio.

```
{
    "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. Conferma che la policy che hai associato al tuo ruolo in un passaggio precedente sia associata al ruolo.

```
aws iam list-attached-role-policies --role-name my-role --query
 "AttachedPolicies[].PolicyArn" --output text
```

Di seguito viene riportato un output di esempio.

```
 arn:aws: iam::111122223333:policy/my-policy
```

3. Imposta una variabile per memorizzare il nome della risorsa Amazon (ARN) della policy che desideri utilizzare. Sostituiscilo *my-policy* con il nome della politica per cui desideri confermare le autorizzazioni.

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. Visualizza la versione predefinita della policy.

```
aws iam get-policy --policy-arn $policy_arn
```

Di seguito viene riportato un output di esempio.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws: iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. Visualizza il contenuto della policy per assicurarti che includa tutte le autorizzazioni di cui il pod ha bisogno. Se necessario, sostituiscilo *1* nel comando seguente con la versione restituita nell'output precedente.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

Di seguito viene riportato un output di esempio.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Se hai creato la policy di esempio in un passaggio precedente, il risultato è lo stesso. Se hai creato una politica diversa, il *example* contenuto è diverso.

6. Confermare che l'account di servizio Kubernetes sia annotato con il ruolo.

```
kubectl describe serviceaccount my-service-account -n default
```

Di seguito viene riportato un output di esempio.

```
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
[...]
```

Fasi successive

- the section called "Assegnazione a pod"

Configurazione di pod per utilizzare un account di servizio Kubernetes

Se un Pod deve accedere ai AWS servizi, devi configurarlo per utilizzare un account di servizio Kubernetes. L'account di servizio deve essere associato a un ruolo AWS Identity and Access Management (IAM) con le autorizzazioni per accedere ai servizi AWS .

- Un cluster esistente. Se non se ne possiede già uno, crearlo utilizzando una delle guide in Nozioni di base.
- Creare un provider di identità OpenID Connect (OIDC) IAM per il cluster. Per sapere se disponi di un account o, per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
- Un account di servizio Kubernetes esistente associato a un ruolo IAM. L'account del servizio deve essere annotato con il nome della risorsa Amazon (ARN) del ruolo IAM. Al ruolo deve essere associata una policy IAM che contenga le autorizzazioni che desideri che i tuoi Pod abbiano per utilizzare i servizi. AWS Per ulteriori informazioni su come creare e configurare l'account e il ruolo del servizio, consulta the section called "Assegnazione di un ruolo IAM".
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Un file `kubectl config` esistente che contiene la configurazione del cluster. Per creare un file `kubectl config`, consulta the section called "Accedere al cluster con kubectl".

  1. Utilizza il seguente comando per creare un manifesto di implementazione per implementare un pod con cui confermare la configurazione. Sostituire i valori di esempio con i propri valori.

```
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. Implementa il file manifesto al cluster.

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

3. Verifica che le variabili di ambiente richieste esistano per i pod.

   a. Visualizza i pod distribuiti con l'implementazione nella fase precedente.

   ```
   kubectl get pods | grep my-app
   ```

   Di seguito viene riportato un output di esempio.

   ```
   my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
   ```

   b. Visualizza l'ARN del ruolo IAM che il pod sta utilizzando.

   ```
   kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
   ```

   Di seguito viene riportato un output di esempio.

   ```
   AWS_ROLE_ARN: arn:aws: iam::111122223333:role/my-role
   ```

Il ruolo ARN deve corrispondere al ruolo ARN con cui hai annotato l'account del servizio esistente. Per ulteriori informazioni sull'annotazione dell'account del servizio, consulta the section called "Assegnazione di un ruolo IAM".

c. Conferma che il pod abbia un file di token di identità web montato.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_WEB_IDENTITY_TOKEN_FILE:
```

Di seguito viene riportato un output di esempio.

```
AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/serviceaccount/
token
```

`kubelet` richiede e memorizza il token per conto del pod. Per impostazione predefinita, la `kubelet` aggiorna il token se è più vecchio dell'80% del suo tempo totale di vita o se il token è più vecchio di 24 ore. È possibile modificare la durata di scadenza per qualsiasi account, ad eccezione dell'account di servizio predefinito, con le impostazioni nelle specifiche del pod. Per ulteriori informazioni, consultare Proiezione Volume Token Acount di Servizio nella documentazione di Kubernetes.

Il Webhook di Pod Identity di Amazon EKS sugli orologi del cluster per pod che utilizzano un account di servizio con la seguente annotazione:

```
eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/my-role
```

Il webhook applica le variabili di ambiente precedenti a questi pod. Il cluster non deve utilizzare il webhook per configurare le variabili di ambiente e i montaggi del file di token. È possibile configurare manualmente i pod per ottenere queste variabili di ambiente. Le versioni supportate dell' AWS SDK cercano innanzitutto queste variabili di ambiente nel provider della catena di credenziali. Le credenziali del ruolo sono utilizzate per i pod che soddisfano questi criteri.

4. Verifica che i tuoi Pod possano interagire con i AWS servizi utilizzando le autorizzazioni assegnate nella policy IAM associata al tuo ruolo.

> **Note**
>
> Quando un Pod utilizza AWS le credenziali di un ruolo IAM associato a un account di servizio, la AWS CLI o SDKs altro nei contenitori per quel Pod utilizza le credenziali fornite da quel ruolo. Se non si limita l'accesso alle credenziali fornite al ruolo IAM del nodo di Amazon EKS, il pod ha comunque accesso a tali credenziali. Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Se il tuo pod non riesce a interagire con i servizi come previsto, completa i seguenti passaggi per confermare che tutto sia configurato correttamente.

a. Verifica che i tuoi Pod utilizzino una versione AWS SDK che supporti l'assunzione di un ruolo IAM tramite un file di token di identità web OpenID Connect. Per ulteriori informazioni, consulta the section called "SDK supportati".

b. Conferma che l'implementazione stia utilizzando l'account del servizio.

```
kubectl describe deployment my-app | grep "Service Account"
```

Di seguito viene riportato un output di esempio.

```
Service Account:  my-service-account
```

c. Se i tuoi pod non riescono ancora ad accedere ai servizi, controlla i passaggi descritti in Assign IAM roles to Kubernetes service accounts per confermare che il ruolo e l'account del servizio siano configurati correttamente.

Configurare l'endpoint del Servizio di token di sicurezza AWS per un account di servizio

Se si utilizza un account di servizio Kubernetes con ruoli IAM per gli account di servizio, è possibile configurare il tipo di endpoint del Servizio di token di sicurezza AWS utilizzato dall'account di servizio.

AWS consiglia di utilizzare gli endpoint AWS STS Regionali anziché l'endpoint globale. Ciò riduce la latenza, fornisce una ridondanza integrata e aumenta la validità del token di sessione. Il Servizio di token di sicurezza AWS deve essere attivo nella Regione AWS in cui viene eseguito il pod. Inoltre, l'applicazione deve avere una ridondanza incorporata per una Regione AWS diversa nel caso di un

guasto del servizio nella Regione AWS iniziale. Per ulteriori informazioni, consultare Managing AWS STS in an AWS Region nella Guida per l'utente di IAM.

- Un cluster esistente. Se non se ne possiede già uno, crearlo utilizzando una delle guide in Nozioni di base.
- Provider OIDC IAM esistente per il cluster. Per ulteriori informazioni, consulta the section called "Gestore di identità OIDC IAM".
- Un account del servizio Kubernetes esistente configurato per l'uso con la funzionalità Ruoli IAM di Amazon EKS per gli account di servizio.

Tutti gli esempi seguenti utilizzano l'account di servizio Kubernetes aws-node utilizzato dal plugin CNI di Amazon VPC. I *valori di esempio* possono essere sostituiti con gli account di servizio, i pod, i namespace e altre risorse personalizzate.

1. Selezionare un pod che utilizza un account di servizio per il quale si desidera modificare l'endpoint. Determinare in quale Regione AWS viene eseguito il pod. Sostituire *aws-node-6mfgv* con il namespace e *kube-system* con il namespace del pod.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
   ```

   Di seguito viene riportato un output di esempio:

   ```
   ip-192-168-79-166.us-west-2/192.168.79.166
   ```

   Nell'output precedente, il pod è in esecuzione su un nodo nella Regione AWS us-west-2.
2. Determinare il tipo di endpoint utilizzato dall'account di servizio del pod.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
   ```

   Di seguito viene riportato un output di esempio:

   ```
   AWS_STS_REGIONAL_ENDPOINTS: regional
   ```

   Se l'endpoint attuale è globale, l'output restituisce `global`. Se non viene restituito alcun output, il tipo di endpoint predefinito è in uso e non è stato sovrascritto.

3. Se le versioni del cluster e della piattaforma corrispondono o sono successive a quelle elencate nella tabella, puoi modificare il tipo di endpoint utilizzato dall'account del servizio dal tipo predefinito con un tipo diverso tramite uno dei comandi seguenti. Sostituisci *aws-node* con il nome dell'account del servizio e *kube-system* con lo spazio dei nomi dell'account del servizio.

   - Se il tipo di endpoint predefinito o attuale è globale e vuoi modificarlo in regionale:

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
     ```

     Se si utilizzano i ruoli IAM per gli account di servizio per generare URL S3 prefirmati nell'applicazione in esecuzione nei container dei pod, il formato dell'URL per gli endpoint Regionali è simile al seguente esempio:

     ```
     https://bucket.s3.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
     ```

   - Se il tipo di endpoint predefinito o attuale è regionale e vuoi modificarlo in globale:

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
     ```

     Se l'applicazione sta effettuando esplicitamente richieste agli endpoint AWS STS globali e non si sostituisce il comportamento predefinito di utilizzo degli endpoint Regionali in cluster Amazon EKS, le richieste avranno esito negativo e restituiranno un codice di errore. Per ulteriori informazioni, consulta the section called "I container dei pod riceveranno il seguente errore: `An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`".

     Se si utilizzano i ruoli IAM per gli account di servizio per generare URL S3 prefirmati nell'applicazione in esecuzione nei container dei pod, il formato dell'URL per gli endpoint globali è simile al seguente esempio:

     ```
     https://bucket.s3.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
     ```

   Se si dispone di un sistema di automazione che prevede un determinato formato per l'URL prefirmato o se l'applicazione o le dipendenze a valle che utilizzano gli URL prefirmati hanno

aspettative per la Regione AWS di destinazione, apportare le modifiche necessarie per utilizzare l'endpoint AWS STS appropriato.

4. Eliminare e ricreare i pod esistenti associati all'account di servizio per applicare le variabili di ambiente delle credenziali. Il web hook di modifica non le applica ai pod già in esecuzione. È possibile sostituire *Pods*, *kube-system* e *-l k8s-app=aws-node* con le informazioni dei pod per cui viene impostata l'annotazione.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

5. Verificare che tutti i pod siano stati riavviati.

```
kubectl get Pods -n kube-system -l k8s-app=aws-node
```

6. Visualizzare le variabili di ambiente per uno dei pod. Verifica che il valore `AWS_STS_REGIONAL_ENDPOINTS` sia quello che hai impostato in una fase precedente.

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

Di seguito viene riportato un output di esempio:

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Eseguire l'autenticazione a un altro account con IRSA

possibile configurare le autorizzazioni IAM multi-account creando un gestore di identità dal cluster di un altro account o utilizzando operazioni `AssumeRole` concatenate. Negli esempi seguenti, l'account A possiede un cluster Amazon EKS che supporta i ruoli IAM per gli account di servizio. I pod in esecuzione sul cluster devono assumere le autorizzazioni IAM dall'account B.

Example Creare un gestore di identità dal cluster di un altro account

Example

In questo esempio, l'account A fornisce all'account B l'URL dell'emittente OpenID Connect (OIDC) dal relativo cluster. L'account B segue le istruzioni indicate alle pagine Create an IAM OIDC provider for your cluster e the section called "Assegnazione di un ruolo IAM" utilizzando l'URL dell'emittente OIDC dal cluster dell'account A. Quindi, un amministratore del cluster annota l'account di servizio nel cluster dell'Account A per utilizzare il ruolo dell'Account B (). *444455556666*

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::444455556666:role/account-b-role
```

Example Utilizzo di operazioni **`AssumeRole`** concatenate

Example

In questo esempio, l'account B crea una policy IAM con le autorizzazioni da assegnare ai pod nel cluster dell'account A. L'account B (*444455556666*) associa tale policy a un ruolo IAM con una relazione di fiducia che consente `AssumeRole` le autorizzazioni per l'Account A (). *111122223333*

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

L'account A crea un ruolo con una policy di attendibilità in grado di ottenere le credenziali dal gestore di identità creato con l'indirizzo dell'emittente OIDC del 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"
        }
      }
    }
  ]
}
```

L'account A allega una policy a tale ruolo con le seguenti autorizzazioni per assumere il ruolo creato dall'account B.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::444455556666:role/account-b-role"
        }
    ]
}
```

Il codice dell'applicazione per i pod che assumono il ruolo dell'account B utilizza due profili: `account_b_role` e `account_a_role`. Il profilo `account_b_role` utilizza il profilo `account_a_role` come origine. Per la AWS CLI, il `~/.aws/config` file è simile al seguente.

```
[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
```

Per specificare profili concatenati per altri AWS SDKs, consulta la documentazione dell'SDK che stai utilizzando. Per ulteriori informazioni, consulta Strumenti su cui costruire. AWS

Usa IRSA con AWS SDK

Utilizzo delle credenziali

Per utilizzare le credenziali dei ruoli IAM per gli account di servizio (IRSA), il codice può usare qualsiasi AWS SDK per creare un client per un servizio AWS con un SDK e, per impostazione predefinita, l'SDK cercherà le credenziali per AWS Identity and Access Management da utilizzare in una catena di posizioni. I ruoli IAM per le credenziali degli account di servizio vengono utilizzati se non specifichi un provider di credenziali quando crei il client o inizializzi in altro modo l'SDK.

Questo perché i ruoli IAM per gli account di servizio sono stati aggiunti come passaggio nella catena di credenziali predefinita. Se i carichi di lavoro attualmente utilizzano credenziali che si trovano all'inizio della catena di credenziali, tali credenziali continueranno a essere utilizzate anche se configuri ruoli IAM per gli account di servizio per lo stesso carico di lavoro.

L'SDK scambia automaticamente il token OIDC dell'account di servizio con credenziali temporanee di Servizio di token di sicurezza AWS utilizzando l'azione `AssumeRoleWithWebIdentity`. Amazon EKS e questa azione SDK continuano a ruotare le credenziali temporanee rinnovandole prima della scadenza.

Quando utilizzi Ruoli IAM per gli account di servizio, i container nei tuoi pod devono utilizzare una versione AWS SDK che supporta l'assunzione di un ruolo IAM tramite un file di token di identità web OpenID Connect. Accertati di utilizzare le seguenti versioni, o successive, per il tuo SDK AWS:

- Java (versione 2) — 2.10.11
- Java – 1.12.782
- AWS SDK per Go v1 – 1.23.13
- AWS SDK per Go v2: supporto per tutte le versioni
- Python (Boto3) — 1.9.220
- Python (botocore) — 1.12.200
- AWS CLI - 1.16.232
- Nodo: 2.525.0 e 3.27.0
- Ruby — 3.58.0
- C++ — 1.7.174
- .NET: 3,3.659.1 – Devi includere anche `AWSSDK.SecurityToken`.
- PHP — 3.110.7

Molti componenti aggiuntivi popolari di Kubernetes, come Cluster Autoscaler, il comando per instrada il traffico Internet col bilanciatore del carico AWS e il plug-in CNI di Amazon VPC per Kubernetes supportano i ruoli IAM per gli account di servizio.

Per assicurarti che l'SDK utilizzato sia supportato, quando crei i container segui la procedura di installazione per il tuo SDK preferito disponibile in Strumenti per costruire su AWS.

Considerazioni

Java

Quando si utilizza Java, è necessario includere il modulo `sts` nel percorso della classe. Per ulteriori informazioni, consulta WebIdentityTokenFileCredentialsProvider nei documenti SDK Java.

Recupera le chiavi di firma per convalidare i token OIDC

Kubernetes rilascia un `ProjectedServiceAccountToken` a ciascun account di servizio Kubernetes. Questo token è un token OIDC, che è un tipo di token web JSON (JWT). Amazon EKS ora ospita un endpoint pubblico OIDC per ogni cluster che contiene le chiavi di firma per i token in modo che i sistemi esterni possano convalidarlo.

Per convalidare un `ProjectedServiceAccountToken`, devi recuperare le chiavi di firma pubbliche OIDC, chiamate anche set di chiavi Web JSON (JWKS). Utilizza queste chiavi nell'applicazione per convalidare il token. Ad esempio, puoi usare PyJWT Python library per convalidare i token utilizzando queste chiavi. Per ulteriori informazioni su `ProjectedServiceAccountToken`, consultare the section called "Informazioni di base su IAM, Kubernetes e OpenID Connect (OIDC)".

Prerequisiti

- Un provider OpenID Connect (OIDC) esistente di AWS Identity and Access Management (IAM) per il cluster. Per determinare se disponi già di un provider IAM o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
- AWS CLI: uno strumento a riga di comando per usare i servizi AWS, tra cui Amazon EKS. Per ulteriori informazioni, consulta Installing nella Guida per l'utente dell'interfaccia a riga di comando AWS. Dopo aver installato AWS CLI, si consiglia di procedere anche con la configurazione. Per ulteriori informazioni, consulta Quick configuration with aws configure nella Guida per l'utente di AWS per l'interfaccia a guida di comando.

Procedura

1. Recupera l'URL OIDC del cluster Amazon EKS utilizzando 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. Recupera la chiave di firma pubblica usando curl o uno strumento simile. Il risultato è un JSON Web Key Set (JWKS).

> **Important**
>
> Amazon EKS limita le chiamate all'endpoint OIDC. È necessario caricare nella cache la chiave di firma pubblica. Rispetta l'intestazione `cache-control` inclusa nella risposta.

> **Important**
>
> Amazon EKS ruota la chiave di firma OIDC ogni sette giorni.

```
$ 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"}]
```

Informazioni su come EKS Pod Identity consente ai pod di accedere ai servizi AWS

Le applicazioni nei container di un pod possono utilizzare un AWS SDK o AWS CLI per effettuare richieste API ai servizi AWS utilizzando le autorizzazioni AWS Identity and Access Management (IAM). Le applicazioni devono firmare le proprie richieste API AWS con le credenziali AWS.

Le associazioni EKS Pod Identity forniscono la possibilità di gestire le credenziali per le applicazioni, analogamente al modo in cui i profili di istanza Amazon EC2 forniscono le credenziali alle istanze Amazon EC2. Invece di creare e distribuire le tue credenziali AWS ai container o utilizzare il ruolo dell'istanza Amazon EC2, associ un ruolo IAM a un account di servizio Kubernetes e configuri i tuoi pod per utilizzare l'account di servizio.

Ogni associazione EKS Pod Identity associa un ruolo a un account di servizio in uno spazio dei nomi nel cluster specificato. Se hai la stessa applicazione in più cluster, puoi creare associazioni identiche in ogni cluster senza modificare la policy di attendibilità del ruolo.

Se un pod utilizza un account di servizio con un'associazione, Amazon EKS imposta le variabili di ambiente nei container del pod. Le variabili di ambiente configurano AWS SDK, inclusa AWS CLI, per utilizzare le credenziali di EKS Pod Identity.

Vantaggi delle associazioni EKS Pod Identity

Le associazioni EKS Pod Identity offrono i seguenti vantaggi:

- Privilegio minimo: è possibile definire l'ambito delle autorizzazioni IAM per un account di servizio; solo i pod che utilizzano tale account avranno accesso alle autorizzazioni definite. Questa caratteristica elimina anche la necessità di soluzioni di terze parti, ad esempio `kiam` o `kube2iam`.
- Isolamento delle credenziali: quando l'accesso al servizio di metadati di istanza (IMDS) di Amazon EC2 è limitato, un container del pod può recuperare solo le credenziali per il ruolo IAM associato all'account del servizio che utilizza. Un container non ha mai accesso alle credenziali utilizzate da altri container in altri pod. Se IMDS non è soggetto a restrizioni, i container del pod hanno accesso anche al ruolo IAM del nodo Amazon EKS e i container potrebbero essere in grado di accedere alle credenziali dei ruoli IAM di altri pod sullo stesso nodo. Per ulteriori informazioni, consulta Limitazione dell'accesso al profilo dell'istanza assegnato al nodo worker.

> **Note**
>
> I pod configurati con `hostNetwork: true` avranno sempre accesso IMDS, ma AWS SDK e CLI utilizzeranno le credenziali di Pod Identity quando abilitati.

- Verificabilità: accesso e registrazione degli eventi sono disponibili tramite AWS CloudTrail per garantire controlli retrospettivi.

> **Important**
>
> I container non sono un limite di sicurezza e l'uso di Pod Identity non modifica questo aspetto. I pod assegnati allo stesso nodo condivideranno un kernel e potenzialmente altre risorse a seconda della configurazione del pod. Sebbene i pod in esecuzione su nodi separati siano isolati a livello di elaborazione, esistono applicazioni di nodo che dispongono di autorizzazioni aggiuntive nell'API Kubernetes oltre all'ambito di una singola istanza. Alcuni esempi sono `kubelet`, `kube-proxy`, driver di archiviazione CSI o le tue applicazioni Kubernetes.

EKS Pod Identity è un metodo più semplice di the section called "Credenziali con IRSA", in quanto non utilizza provider di identità OIDC. EKS Pod Identity presenta i seguenti miglioramenti:

- Operazioni indipendenti: in molte organizzazioni, la creazione di provider di identità OIDC è una responsabilità di team diversi rispetto all'amministrazione dei cluster Kubernetes. EKS Pod Identity offre una netta separazione dei compiti, in cui tutta la configurazione delle associazioni EKS Pod Identity viene eseguita in Amazon EKS e tutta la configurazione delle autorizzazioni IAM viene eseguita in IAM.
- Riusabilità: EKS Pod Identity utilizza un singolo principale IAM anziché principali separati per ogni cluster utilizzato dai ruoli IAM per gli account di servizio. L'amministratore IAM aggiunge il seguente principale alla policy di attendibilità di qualsiasi ruolo per renderlo utilizzabile da parte delle associazioni EKS Pod Identity.

```
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            }
```

- Scalabilità: ciascun set di credenziali temporanee è assunto dal servizio di autenticazione EKS in EKS Pod Identity, anziché da ciascun AWS SDK eseguito in ciascun pod. Quindi, si esegue Amazon EKS Pod Identity Agent su ogni nodo rilasciando le credenziali agli SDK. Pertanto, il carico si riduce a una volta per ciascun nodo e non è duplicato in ciascun pod. Per ulteriori dettagli del processo, consulta the section called "Come funziona".

Per ulteriori informazioni su come confrontare le due alternative, consulta the section called "Accesso ai carichi di lavoro a AWS ".

Panoramica sulla configurazione delle associazioni EKS Pod Identity

Attiva le associazioni EKS Pod Identity completando le seguenti procedure:

1. the section called "Configurazione dell'agente": completi questa procedura una sola volta per ogni cluster. Non è necessario completare questo passaggio se la modalità automatica EKS è abilitata sul cluster.
2. the section called "Assegnazione di un ruolo IAM": completa questa procedura per ciascun set univoco di autorizzazioni che desideri abbia un'applicazione.
3. the section called "Account di servizio dei pod": completa questa procedura per ciascun pod che deve accedere ai servizi AWS.

4. the section called "SDK supportati": verifica che il carico di lavoro utilizzi un AWS SDK di una versione supportata e che il carico di lavoro utilizzi la catena di credenziali predefinita.

Limiti

- Sono supportate fino a 5.000 associazioni EKS Pod Identity per cluster per mappare i ruoli IAM agli account di servizio Kubernetes.

Considerazioni

- Associazione del ruolo IAM: ogni account di servizio Kubernetes in un cluster può essere associato a un ruolo IAM dello stesso account AWS del cluster. Per modificare il ruolo, modifica l'associazione EKS Pod Identity. Per l'accesso multi-account, delega l'accesso al ruolo utilizzando i ruoli IAM. Per ulteriori informazioni, consulta Delegare l'accesso tra account AWS tramite i ruoli IAM nella Guida per l'utente IAM.
- Agente EKS Pod Identity: l'agente Pod Identity è necessario per utilizzare EKS Pod Identity. L'agente funziona come Kubernetes `DaemonSet` sui nodi del cluster, fornendo credenziali solo ai pod sullo stesso nodo. Utilizza `hostNetwork` del nodo, la porta occupante `80` e `2703` sull'indirizzo locale del collegamento (`169.254.170.23` per IPv4, `[fd00:ec2::23]` per IPv6). Se IPv6 è disabilitato nel cluster, disabilita IPv6 per l'agente Pod Identity. Per ulteriori informazioni, consulta Disabilitare IPv6 nell'agente EKS Pod Identity.
- Coerenza finale: le associazioni EKS Pod Identity alla fine sono coerenti, con potenziali ritardi di diversi secondi dopo le chiamate API. Evita di creare o aggiornare associazioni in percorsi di codice critici e ad alta disponibilità. Esegui, invece, queste azioni in routine di inizializzazione o configurazione separate e meno frequenti. Per ulteriori informazioni, consulta Gruppi di sicurezza per pod nella Guida alle best practice EKS.
- Considerazioni sui proxy e sui gruppi di sicurezza: per i pod che utilizzano un proxy, aggiungi `169.254.170.23` (IPv4) e `[fd00:ec2::23]` (IPv6) alle variabili di ambiente `no_proxy/NO_PROXY` per evitare richieste non riuscite all'agente EKS Pod Identity. Se utilizzi i gruppi di sicurezza per pods con il CNI di AWS VPC, imposta il flag `ENABLE_POD_ENI` su "vero" e il flag `POD_SECURITY_GROUP_ENFORCING_MODE` su "standard". Per ulteriori informazioni, consulta Assegnare di gruppi di sicurezza ai singoli pod.

Versioni del cluster EKS Pod Identity

Per utilizzare le associazioni EKS Pod Identity, il cluster deve avere una versione della piattaforma uguale o successiva a quella elencata nella tabella seguente o una versione Kubernetes successiva a quelle elencate nella tabella. Per trovare la versione consigliata di Amazon EKS Pod Identity Agent per una versione Kubernetes, consulta the section called "Verifica la compatibilità".

Versione di Kubernetes	Versione della piattaforma
Versioni di Kubernetes non elencate	Tutte le versioni delle piattaforma supportate
`1.28`	`eks.4`

Restrizioni di EKS Pod Identity

Le associazioni EKS Pod Identity sono disponibili in:

- Versioni del cluster Amazon EKS elencate nell'argomento precedente the section called "Versioni del cluster EKS Pod Identity".
- Nodi worker nel cluster che sono istanze Amazon EC2 di Linux.

Le associazioni EKS Pod Identity non sono disponibili in quanto segue:

- AWS Outposts.
- Amazon EKS Anywhere.
- Cluster Kubernetes creati ed eseguiti su Amazon EC2. I componenti di EKS Pod Identity sono disponibili solo su Amazon EKS.

Non è possibile utilizzare le associazioni EKS Pod Identity con:

- Pod che vengono eseguiti ovunque, tranne che su istanze Amazon EC2 di Linux. Pod Linux e Windows che sono eseguiti su AWS Fargate (Fargate) non sono supportati. I pod che vengono eseguiti su istanze Amazon EC2 di Windows non sono supportati.

Comprensione di come funziona EKS Pod Identity

Le associazioni Amazon EKS Pod Identity offrono la possibilità di gestire le credenziali per le applicazioni, in modo simile a come i profili di istanza di Amazon EC2 forniscono le credenziali alle istanze Amazon EC2.

Amazon EKS Pod Identity fornisce le credenziali per i tuoi carichi di lavoro con un'API EKS Auth aggiuntiva e un pod di agenti che viene eseguito su ogni nodo.

Nei componenti aggiuntivi, come i componenti aggiuntivi di Amazon EKS e i controller, gli operatori e altri componenti aggiuntivi autogestiti, l'autore deve aggiornare il software per utilizzare gli SDK AWS più recenti. Per l'elenco della compatibilità tra EKS Pod Identity e i componenti aggiuntivi prodotti da Amazon EKS, consulta la sezione precedente the section called "Restrizioni di EKS Pod Identity".

Utilizzo delle associazioni EKS Pod Identity nel codice

Nel codice, puoi utilizzare gli SDK AWS per accedere ai servizi AWS. Scrivi il codice per creare un client per un servizio AWS con un SDK e, per impostazione predefinita, l'SDK cerca in una catena di posizioni le credenziali per AWS Identity and Access Management da utilizzare. Dopo aver trovato credenziali valide, la ricerca viene interrotta. Per ulteriori informazioni sulle posizioni predefinite utilizzate, consulta Catena di fornitori di credenziali nella Guida di riferimento agli SDK e agli strumenti AWS.

Le associazioni EKS Pod Identity sono state aggiunte al provider di credenziali del container, che viene cercato in un passaggio nella catena di credenziali predefinita. Se i carichi di lavoro utilizzano attualmente credenziali che si trovano all'inizio della catena di credenziali, quest'ultime continuano a essere utilizzate anche se configuri un'associazione EKS Pod Identity per lo stesso carico di lavoro. In questo modo è possibile migrare in sicurezza da altri tipi di credenziali creando l'associazione prima di rimuovere le vecchie credenziali.

Il provider di credenziali del container assegna credenziali temporanee da un agente che viene eseguito su ogni nodo. In Amazon EKS, l'agente è Amazon EKS Pod Identity Agent e su Amazon Elastic Container Service l'agente è `amazon-ecs-agent`. Gli SDK utilizzano variabili di ambiente per individuare l'agente a cui connettersi.

Al contrario, i ruoli IAM per gli account di servizio forniscono un token di identità Web che AWS SDK deve scambiare con AWS Security Token Service utilizzando `AssumeRoleWithWebIdentity`.

Come funziona un agente EKS Pod Identity con un pod

1. Quando Amazon EKS avvia un nuovo pod che utilizza un account di servizio con un'associazione EKS Pod Identity, il cluster aggiunge il seguente contenuto al manifesto:

```
    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. Kubernetes seleziona su quale nodo eseguire il pod. Quindi, Amazon EKS Pod Identity Agent sul nodo utilizza l'azione AssumeRoleForPodIdentity per recuperare le credenziali temporanee dall'API EKS Auth.
3. EKS Pod Identity Agent rende disponibili queste credenziali per gli SDK AWS che esegui all'interno dei container.
4. Utilizza l'SDK nell'applicazione senza specificare un provider di credenziali per utilizzare la catena di credenziali predefinita. In alternativa, specifica il provider di credenziali del container. Per ulteriori informazioni sulle posizioni predefinite utilizzate, consulta Catena di fornitori di credenziali nella Guida di riferimento agli SDK e agli strumenti AWS.
5. L'SDK utilizza le variabili di ambiente per connettersi all'EKS Pod Identity Agent e recuperare le credenziali.

> **Note**
>
> Se i carichi di lavoro attualmente utilizzano credenziali che si trovano all'inizio della catena di credenziali, tali credenziali continueranno a essere utilizzate anche se configuri un'associazione EKS Pod Identity per lo stesso carico di lavoro.

Configurazione dell'agente Amazon EKS Pod Identity

Le associazioni Amazon EKS Pod Identity offrono la possibilità di gestire le credenziali per le tue applicazioni, in modo simile al modo in cui i profili di EC2 istanza Amazon forniscono le credenziali alle istanze Amazon EC2 .

Amazon EKS Pod Identity fornisce le credenziali per i tuoi carichi di lavoro con un'API EKS Auth aggiuntiva e un pod di agenti che viene eseguito su ogni nodo.

> **Tip**
>
> Non è necessario installare l'agente EKS Pod Identity sui cluster in modalità automatica EKS. Questa funzionalità è integrata in modalità automatica EKS.

Considerazioni

- Per impostazione predefinita, l'agente EKS Pod Identity è preinstallato sui cluster in modalità automatica EKS. Per ulteriori informazioni, consulta Modalità automatica EKS.
- Per impostazione predefinita, l'agente EKS Pod Identity ascolta su un indirizzo `IPv4` e sull'indirizzo `IPv6` perché i pod richiedano le credenziali. L'agente utilizza l'indirizzo IP di loopback (localhost) `169.254.170.23` per `IPv4` e l'indirizzo IP localhost `[fd00:ec2::23]` per `IPv6`.
- Se disabiliti gli indirizzi `IPv6` o impedisci in altro modo gli indirizzi IP `IPv6` localhost, l'agente non può avviarsi. Per avviare l'agente su nodi che non possono utilizzare `IPv6`, segui i passaggi indicati in the section called "Disabilitazione di IPv6" per disabilitare la configurazione `IPv6`.

Creazione di Amazon EKS Pod Identity Agent

Prerequisiti dell'agente

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base. La versione del cluster e la versione della piattaforma devono essere uguali o successive alle versioni elencate nelle versioni del cluster EKS Pod Identity.
- Il ruolo del nodo dispone delle autorizzazioni per consentire all'agente di eseguire l'azione `AssumeRoleForPodIdentity` nell'API EKS Auth. Puoi utilizzare la politica AWS gestita: Amazon EKSWorker NodePolicy o aggiungere una politica personalizzata simile alla seguente:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity"
            ],
            "Resource": "*"
        }
    ]
}
```

  Questa azione può essere limitata dai tag per ridurre i ruoli che possono essere assunti dai pod che utilizzano l'agente.
- I nodi possono raggiungere e scaricare immagini da Amazon ECR. L'immagine del container per il componente aggiuntivo si trova nei registri elencati in Visualizza registri delle immagini del container Amazon per i componenti aggiuntivi Amazon EKS.

  Tieni presente che puoi modificare la posizione dell'immagine e fornire `imagePullSecrets` componenti aggiuntivi EKS nelle impostazioni di configurazione opzionali nella e nella `--configuration-values` AWS CLI. Console di gestione AWS
- I nodi possono raggiungere l'API Amazon EKS Auth. Per i cluster privati, è necessario l'ingresso dell'`eks-auth`endpoint. AWS PrivateLink

Agente di configurazione con console AWS

1. Aprire la Console Amazon EKS.

2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi scegli il nome del cluster per cui configurare il componente aggiuntivo di EKS Pod Identity Agent.
3. Selezionare la scheda Componenti aggiuntivi.
4. Scegli Ottieni altri componenti aggiuntivi.
5. Seleziona la casella nella parte superiore destra di quella del componente aggiuntivo relativo a EKS Pod Identity Agent e scegli Avanti.
6. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona una versione qualsiasi nell'elenco a discesa Versione.
7. (Facoltativo) Espandi Impostazioni di configurazione facoltative per inserire una configurazione aggiuntiva. Ad esempio, puoi fornire una posizione alternativa per l'immagine del container e `ImagePullSecrets`. Lo Schema JSON con le chiavi accettate sono mostrate in Schema di configurazione del componente aggiuntivo.

   Inserisci le chiavi e i valori di configurazione in Valori di configurazione.
8. Scegli Next (Successivo).
9. Verifica che i pod di EKS Pod Identity Agent siano in esecuzione sul cluster.

   ```
   kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
   ```

   Di seguito viene riportato un output di esempio.

   ```
   eks-pod-identity-agent-gmqp7                                          1/1     Running
      1 (24h ago)   24h
   eks-pod-identity-agent-prnsh                                          1/1     Running
      1 (24h ago)   24h
   ```

   Ora puoi utilizzare le associazioni EKS Pod Identity nel cluster. Per ulteriori informazioni, consulta the section called "Assegnazione di un ruolo IAM".

Agente di configurazione con AWS CLI

1. Esegui il seguente comando AWS CLI. Sostituisci `my-cluster` con il nome del cluster.

   ```
   aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --
   addon-version v1.0.0-eksbuild.1
   ```

> **Note**
>
> L'agente EKS Pod Identity non utilizza `service-account-role-arn` per ruoli IAM per account di servizio. È necessario fornire a EKS Pod Identity Agent le autorizzazioni nel ruolo del nodo.

2. Verifica che i pod di EKS Pod Identity Agent siano in esecuzione sul cluster.

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

Di seguito viene riportato un output di esempio.

```
eks-pod-identity-agent-gmqp7                                          1/1     Running
   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1     Running
   1 (24h ago)   24h
```

Ora puoi utilizzare le associazioni EKS Pod Identity nel cluster. Per ulteriori informazioni, consulta the section called "Assegnazione di un ruolo IAM".

Assegnazione di un ruolo IAM a un account di servizio Kubernetes

Questo argomento spiega come configurare un account di servizio Kubernetes per assumere un ruolo di AWS Identity and Access Management (IAM) con EKS Pod Identity. Tutti i Pod configurati per utilizzare l'account di servizio possono quindi accedere a qualsiasi AWS servizio a cui il ruolo dispone delle autorizzazioni di accesso.

Per creare un'associazione EKS Pod Identity, è sufficiente un solo passaggio; si crea l'associazione in EKS tramite AWS CLI AWS CloudFormation e altri strumenti. Console di gestione AWS AWS SDKs Non sono presenti dati o metadati sulle associazioni all'interno del cluster in nessun oggetto Kubernetes e non si aggiungono annotazioni agli account di servizio.

Prerequisiti

- Un cluster esistente. Se non se ne possiede già uno, crearlo seguendo una delle guide in Nozioni di base.
- Il principale IAM che sta creando l'associazione deve avere `iam:PassRole`.

- La versione più recente della AWS CLI installata e configurata sul dispositivo o. AWS CloudShell È possibile verificare la versione corrente con `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Un file `kubectl config` esistente che contiene la configurazione del cluster. Per creare un file `kubectl config`, consulta the section called "Accedere al cluster con kubectl".

Crea un'associazione Pod Identity (AWS Console)

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui configurare il componente aggiuntivo di EKS Pod Identity Agent.
3. Scegli la scheda Accesso.
4. In Associazioni Pod Identity, scegli Crea.
5. Per Ruolo IAM, seleziona il ruolo IAM con le autorizzazioni che desideri assegnare al carico di lavoro.

> **Note**
>
> L'elenco contiene solo i ruoli con la seguente policy di attendibilità che consente a EKS Pod Identity di utilizzarli.

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

```
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`: EKS Pod Identity utilizza `AssumeRole` per assumere il ruolo IAM prima di passare le credenziali temporanee ai pod.

`sts:TagSession`— EKS Pod Identity utilizza `TagSession` per includere i tag di sessione nelle richieste a AWS STS.

È possibile utilizzare questi tag nelle chiavi di condizione nella policy di attendibilità per limitare gli account di servizio, i namespace e i cluster che possono utilizzare questo ruolo.

Per un elenco di chiavi di condizione di Amazon EKS, consultare Condizioni per Amazon Elastic Kubernetes Service in Service Authorization Reference. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consultare Operazioni definite da Amazon Elastic Kubernetes Service.

6. Per Namespace Kubernetes, seleziona il namespace Kubernetes che contiene l'account di servizio e il carico di lavoro. Facoltativamente, è possibile specificare un namespace che non esiste nel cluster.
7. Per l'account del servizio Kubernetes, seleziona l'account del servizio Kubernetes da utilizzare. Il manifesto del carico di lavoro Kubernetes deve specificare questo account di servizio. Facoltativamente, è possibile specificare un account di servizio in base al nome che non è presente nel cluster.
8. (Facoltativo) Seleziona Disabilita i tag di sessione per disabilitare i tag di sessione predefiniti che Pod Identity aggiunge automaticamente quando assume il ruolo.
9. (Facoltativo) Attiva Configure session policy per configurare una policy IAM per applicare restrizioni aggiuntive a questa associazione Pod Identity oltre alle autorizzazioni definite nella policy IAM allegata al ruolo IAM.

> **Note**
>
> Una policy di sessione può essere applicata solo quando l'impostazione Disabilita i tag di sessione è selezionata.

10. (Facoltativo) Per Tag, scegli Aggiungi tag per aggiungere metadati in una coppia chiave-valore. Questi tag vengono applicati all'associazione e possono essere utilizzati nelle policy IAM.

    Puoi ripetere questo passaggio per aggiungere più tag.

11. Scegli Create (Crea).

Creare un'associazione Pod Identity (AWS CLI)

1. Se desideri associare una policy IAM esistente al ruolo IAM, passa al prossimo passaggio.

   Creare una policy IAM Puoi creare la tua politica o copiare una politica AWS gestita che concede già alcune delle autorizzazioni di cui hai bisogno e personalizzarla in base alle tue esigenze specifiche. Per ulteriori informazioni, consulta Creating IAM policies nella Guida per l'utente di IAM.

   a. Crea un file che includa le autorizzazioni per i AWS servizi a cui desideri che i tuoi Pod accedano. Per un elenco di tutte le azioni per tutti i AWS servizi, consulta il Service Authorization Reference.

      Puoi eseguire il comando seguente per creare un file policy di esempio che consenta l'accesso in sola lettura a un bucket Amazon S3. Facoltativamente, puoi archiviare le informazioni di configurazione o uno script di bootstrap in questo bucket; i container nel pod possono leggere il file dal bucket e caricarlo nell'applicazione. Se desideri creare questa policy di esempio, copia i seguenti contenuti sul dispositivo. Sostituiscilo *my-pod-secrets-bucket* con il nome del tuo bucket ed esegui il comando.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
```

```
}
```

b. Creare la policy IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-
policy.json
```

2. Crea un ruolo IAM e associalo a un account del servizio Kubernetes.

a. Se disponi di un account di servizio Kubernetes esistente che desideri assuma un ruolo IAM, puoi saltare questo passaggio.

Creare un account di servizio Kubernetes. Copia i seguenti contenuti sul dispositivo. Sostituiscilo *my-service-account* con il nome desiderato e *default* con un namespace diverso, se necessario. Se si modifica*default*, lo spazio dei nomi deve già esistere.

```
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
```

Eseguire il seguente comando seguente.

```
kubectl apply -f my-service-account.yaml
```

b. Per creare una policy di attendibilità del ruolo IAM, esegui il comando seguente.

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

```
                "sts:TagSession"
            ]
        }
    ]
}
```

c. Crea il ruolo. Sostituisci *my-role* con un nome per il ruolo IAM e *my-role-description* con una descrizione per il tuo ruolo.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://
trust-relationship.json --description "my-role-description"
```

d. Allegare una policy IAM al ruolo. Sostituisci *my-role* con il nome del ruolo IAM e *my-policy* con il nome di una policy esistente che hai creato.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:
 iam::111122223333:policy/my-policy
```

> **Note**
>
> A differenza dei ruoli IAM per gli account di servizio, EKS Pod Identity non utilizza l'annotazione sull'account di servizio.

e. Esegui il comando seguente per creare l'associazione. Sostituiscilo `my-cluster` con il nome del cluster, sostituiscilo *my-service-account* con il nome desiderato e *default* con un namespace diverso, se necessario.

```
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
```

Di seguito viene riportato un output di esempio.

```
{
    "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**
>
> È possibile specificare un namespace e un account di servizio in base al nome che non esiste nel cluster. È necessario creare il namespace, l'account di servizio e il carico di lavoro che utilizza l'account di servizio affinché l'associazione EKS Pod Identity funzioni.

Confermare la configurazione

1. Conferma che la policy di attendibilità del ruolo IAM sia configurata correttamente.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

Di seguito viene riportato un output di esempio.

```
{
    "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. Conferma che la policy che hai associato al tuo ruolo in un passaggio precedente sia associata al ruolo.

```
aws iam list-attached-role-policies --role-name my-role --query
 'AttachedPolicies[].PolicyArn' --output text
```

   Di seguito viene riportato un output di esempio.

```
 arn:aws: iam::111122223333:policy/my-policy
```

3. Imposta una variabile per memorizzare il nome della risorsa Amazon (ARN) della policy che desideri utilizzare. Sostituiscilo *my-policy* con il nome della policy per cui desideri confermare le autorizzazioni.

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. Visualizza la versione predefinita della policy.

```
aws iam get-policy --policy-arn $policy_arn
```

   Di seguito viene riportato un output di esempio.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws: iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. Visualizza il contenuto della policy per assicurarti che includa tutte le autorizzazioni di cui il pod ha bisogno. Se necessario, sostituiscilo *1* nel comando seguente con la versione restituita nell'output precedente.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

   Di seguito viene riportato un output di esempio.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Se hai creato la policy di esempio in un passaggio precedente, il risultato è lo stesso. Se hai creato una politica diversa, il *example* contenuto è diverso.

Fasi successive

the section called "Account di servizio dei pod"

Accedi alle AWS risorse utilizzando i ruoli IAM di EKS Pod Identity Target

Quando esegui applicazioni su Amazon Elastic Kubernetes Service (Amazon EKS), potresti dover accedere AWS a risorse esistenti in account diversi. AWS Questa guida mostra come configurare l'accesso su più account utilizzando EKS Pod Identity, che consente ai pod Kubernetes di accedere ad altre risorse utilizzando i ruoli di destinazione. AWS

Prerequisiti

Prima di iniziare, assicurati di aver completato i passaggi seguenti:

- Configurazione dell'agente Amazon EKS Pod Identity
- Creazione di un ruolo di EKS Pod Identity

Come funziona

Pod Identity consente alle applicazioni del cluster EKS di accedere alle AWS risorse tra gli account attraverso un processo chiamato concatenamento dei ruoli.

Quando crei un'associazione Pod Identity, puoi fornire due ruoli IAM: un ruolo EKS Pod Identity nello stesso account del cluster EKS e un ruolo IAM Target dall'account contenente AWS le risorse a

cui desideri accedere (come i bucket S3 o i database RDS). Il ruolo EKS Pod Identity deve trovarsi nell'account del cluster EKS a causa PassRole dei requisiti IAM, mentre il ruolo IAM di Target può essere in qualsiasi account. AWS PassRole consente a un' AWS entità di delegare l'assunzione del ruolo a un altro servizio. EKS Pod Identity utilizza PassRole per connettere un ruolo a un account di servizio Kubernetes, richiedendo che sia il ruolo che l'identità che lo trasmette si trovino nello stesso AWS account del cluster EKS. Quando il pod dell'applicazione deve accedere alle AWS risorse, richiede le credenziali a Pod Identity. Quindi, Pod Identity esegue automaticamente due assunzione di ruolo in sequenza: prima assume il ruolo EKS Pod Identity, poi utilizza tali credenziali per assumere il ruolo IAM di destinazione. Questo processo fornisce al pod credenziali temporanee con le autorizzazioni definite nel ruolo di destinazione, consentendo l'accesso sicuro alle risorse di altri account. AWS

Considerazioni sul caching

Per via dei meccanismi del caching, gli aggiornamenti per un ruolo IAM in un'associazione Pod Identity esistente potrebbero non avere effetto immediato nei pod in esecuzione sul cluster EKS. L'agente Pod Identity memorizza nella cache le credenziali IAM in base alla configurazione dell'associazione al momento del recupero delle credenziali. Se l'associazione include solo un ruolo EKS Pod Identity e nessun ruolo IAM di destinazione, le credenziali memorizzate nella cache durano 6 ore. Se l'associazione include sia l'ARN del ruolo EKS Pod Identity che un ruolo IAM di destinazione, le credenziali memorizzate nella cache durano 59 minuti. La modifica di un'associazione esistente, ad esempio l'aggiornamento dell'ARN del ruolo EKS Pod Identity o l'aggiunta di un ruolo IAM di destinazione, non reimposta la cache esistente. Di conseguenza, l'agente non riconoscerà gli aggiornamenti fino all'aggiornamento delle credenziali memorizzate nella cache. Per applicare prima le modifiche, è possibile ricreare i pod esistenti; in caso contrario, sarà necessario attendere la scadenza della cache.

Passaggio 1: creazione e associazione di un ruolo IAM di destinazione

In questo passaggio, stabilirai una catena di affidabilità sicura creando e configurando un ruolo IAM di destinazione. A titolo dimostrativo, creeremo un nuovo ruolo Target IAM per stabilire una catena di fiducia tra due AWS account: il ruolo EKS Pod Identity (ad esempio`eks-pod-identity-primary-role`) nell' AWS account del cluster EKS ottiene l'autorizzazione ad assumere il ruolo Target IAM (ad esempio`eks-pod-identity-aws-resources`) nel tuo account di destinazione, abilitando l'accesso a AWS risorse come i bucket Amazon S3.

Creazione del ruolo IAM di destinazione

1. Apri la console IAM Amazon.

2. Nella barra di navigazione in alto, verifica di aver effettuato l'accesso all'account contenente AWS le risorse (come i bucket S3 o le tabelle DynamoDB) per il tuo ruolo IAM di Target.
3. Nel pannello di navigazione a sinistra, seleziona Ruoli.
4. Scegli il pulsante Crea ruolo, quindi l' AWS account nella sezione «Tipo di entità affidabile».
5. Scegli Altro AWS account, inserisci il tuo numero di AWS account (l'account in cui esiste il tuo ruolo EKS Pod Identity), quindi scegli Avanti.
6. Aggiungi le politiche di autorizzazione che desideri associare al ruolo (ad esempio, AmazonS3FullAccess), quindi scegli Avanti.
7. Immetti un nome ruolo, come `MyCustomIAMTargetRole`, quindi scegli Crea ruolo.

Aggiorna la policy di affidabilità di un ruolo IAM di destinazione

1. Dopo aver creato il ruolo, ritornerai all'elenco Ruoli. Trova e seleziona il nuovo ruolo creato nel passaggio precedente (ad esempio, `MyCustomIAMTargetRole`).
2. Seleziona la scheda Relazioni di attendibilità.
3. Fai clic su Modifica policy di affidabilità sul lato destro.
4. Nell'editor della policy, sostituisci il file JSON predefinito con la tua policy di affidabilità. Sostituisci i valori segnaposto per il nome del ruolo e `111122223333` nell'ARN del ruolo IAM con l'ID dell' AWS account che ospita il tuo cluster EKS. Opzionalmente, puoi anche utilizzare PrincipalTags la policy di fiducia dei ruoli per autorizzare solo account di servizio specifici di un determinato cluster e namespace ad assumere il tuo ruolo di destinazione. Esempio:

```
{
  "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"
        }
      }
    }
  ]
}
```

La politica di cui sopra consente al ruolo `eks-pod-identeity-primary-role` dell' AWS account 111122223333 con i pertinenti EKS Pod Identity Session Tag di assumere questo ruolo.

Se hai Disabilitato i tag di sessione nella tua EKS Pod Identity, EKS Pod Identity imposta anche `sts:ExternalId` con le informazioni sul cluster, sul namespace e sull'account di servizio di un pod quando assume un ruolo di destinazione.

```
{
  "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"
    }
  ]
}
```

La policy di cui sopra aiuta a garantire che solo il cluster, il namespace e l'account di servizio previsti possano assumere il ruolo di destinazione.

Aggiorna la policy di autorizzazione per il ruolo EKS Pod Identity

In questo passaggio, aggiornerai la policy di autorizzazione del ruolo EKS Pod Identity associato al tuo cluster Amazon EKS aggiungendo l'ARN del ruolo IAM di destinazione come risorsa.

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster EKS.
3. Scegliere la scheda Accesso.
4. In Associazioni Pod Identity, seleziona il tuo ruolo EKS Pod Identity.
5. Scegli Autorizzazioni, scegli Aggiungi autorizzazioni, quindi Crea policy in linea.
6. Scegli JSON sul lato destro.
7. Nell'editor della policy, sostituisci il file JSON predefinito con la tua policy di autorizzazione. Sostituisci il valore segnaposto per il nome del ruolo e `222233334444` nell'ARN del ruolo IAM con il tuo ruolo IAM di destinazione. Esempio:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Resource": "arn:aws:iam::222233334444:role/eks-pod-identity-aws-resources"
        }
    ]
```

```
}
```

Passaggio 2: associazione del ruolo IAM di destinazione a un account di servizio Kubernetes

In questo passaggio, creerai un'associazione tra il ruolo IAM di destinazione e l'account del servizio Kubernetes nel tuo cluster EKS.

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster a cui aggiungere l'associazione.
3. Scegli la scheda Accesso.
4. In Associazioni Pod Identity, scegli Crea.
5. Scegli il ruolo EKS Pod Identity nel ruolo IAM per i tuoi carichi di lavoro da assumere.
6. Scegli il ruolo IAM di destinazione nel ruolo IAM di destinazione che sarà assunto dal ruolo EKS Pod Identity.
7. Nel campo del namespace Kubernetes, inserisci il nome del namespace in cui desideri creare l'associazione (ad esempio, `my-app-namespace`). Ciò definisce dove risiede l'account del servizio.
8. Nel campo dell'account del servizio Kubernetes, inserisci il nome dell'account di servizio (ad esempio, `my-service-account`) che utilizzerà le credenziali IAM. Ciò collega il ruolo IAM all'account del servizio.
9. (Facoltativo) Seleziona Disabilita i tag di sessione per disabilitare i tag di sessione predefiniti che Pod Identity aggiunge automaticamente quando assume il ruolo.
10. (Facoltativo) Attiva Configure session policy per configurare una policy IAM per applicare restrizioni aggiuntive a questa associazione di identità Pod oltre alle autorizzazioni definite nella policy IAM allegata al ruolo IAM di Target.

> **Note**
>
> 1. Una policy di sessione può essere applicata solo quando l'impostazione Disabilita i tag di sessione è selezionata. 2. Se specifichi una policy di sessione, le restrizioni della policy si applicano alle autorizzazioni del ruolo IAM di Target e non al ruolo IAM associato a questa associazione Pod Identity.

11. Scegli Crea per creare l'associazione.

Configura i pod per accedere ai AWS servizi con account di servizio

Se un Pod deve accedere ai AWS servizi, devi configurarlo per utilizzare un account di servizio Kubernetes. L'account di servizio deve essere associato a un ruolo AWS Identity and Access Management (IAM) con le autorizzazioni per accedere ai servizi AWS .

- Un cluster esistente. Se non se ne possiede già uno, crearlo utilizzando una delle guide in Nozioni di base.
- Un account di servizio Kubernetes esistente e un'associazione EKS Pod Identity che associa l'account di servizio a un ruolo IAM. Al ruolo deve essere associata una policy IAM che contenga le autorizzazioni che desideri che i tuoi Pod abbiano per utilizzare i servizi. AWS Per ulteriori informazioni su come creare e configurare l'account e il ruolo del servizio, consulta the section called "Assegnazione di un ruolo IAM".
- La versione più recente della AWS CLI installata e configurata sul dispositivo o. AWS CloudShell È possibile verificare la versione corrente con `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Un file `kubectl config` esistente che contiene la configurazione del cluster. Per creare un file `kubectl config`, consulta the section called "Accedere al cluster con kubectl".

1. Utilizza il seguente comando per creare un manifesto di implementazione per implementare un pod con cui confermare la configurazione. Sostituire i valori di esempio con i propri valori.

```
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. Implementa il file manifesto al cluster.

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

3. Verifica che le variabili di ambiente richieste esistano per i pod.
   a. Visualizza i pod distribuiti con l'implementazione nella fase precedente.

```
kubectl get pods | grep my-app
```

   Di seguito viene riportato un output di esempio.

```
my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
```

   b. Verifica che il pod abbia il file di token dell'account di servizio montato.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
 AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
```

   Di seguito viene riportato un output di esempio.

```
AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:  /var/run/secrets/
pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
```

4. Verifica che i tuoi Pod possano interagire con i AWS servizi utilizzando le autorizzazioni che hai assegnato nella policy IAM associata al tuo ruolo.

> **Note**
>
> Quando un Pod utilizza AWS le credenziali di un ruolo IAM associato a un account di servizio, la AWS CLI o SDKs altro nei contenitori per quel Pod utilizza le credenziali fornite da quel ruolo. Se non si limita l'accesso alle credenziali fornite al ruolo IAM del nodo di Amazon EKS, il pod ha comunque accesso a tali credenziali. Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Se il tuo pod non riesce a interagire con i servizi come previsto, completa i seguenti passaggi per confermare che tutto sia configurato correttamente.

a. Verifica che i tuoi Pod utilizzino una versione AWS SDK che supporti l'assunzione di un ruolo IAM tramite un'associazione EKS Pod Identity. Per ulteriori informazioni, consulta the section called "SDK supportati".

b. Conferma che l'implementazione stia utilizzando l'account del servizio.

```
kubectl describe deployment my-app | grep "Service Account"
```

Di seguito viene riportato un output di esempio.

```
Service Account:  my-service-account
```

Concedi ai Pods l'accesso alle AWS risorse in base ai tag

Il controllo degli accessi basato su attributi (ABAC) concede diritti agli utenti attraverso policy che combinano gli attributi. EKS Pod Identity allega i tag alle credenziali temporanee di ciascun pod con attributi come nome cluster, namespace e nome account di servizio. Questi tag di sessione di ruolo consentono agli amministratori di creare un singolo ruolo che può funzionare su più account di servizio, consentendo l'accesso alle risorse in base ai tag corrispondenti. AWS Aggiungendo il supporto per i tag di sessione dei ruoli, puoi imporre limiti di sicurezza più rigorosi tra i cluster e i carichi di lavoro all'interno dei cluster, riutilizzando al contempo gli stessi ruoli IAM e le stesse policy IAM.

Policy esemplificativa con tag

Di seguito è riportato un esempio di policy IAM che concede a `s3:GetObject` autorizzazioni quando l'oggetto corrispondente è etichettato con il nome del cluster 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}"
                }
            }
        }
    ]
}
```

Abilitare o disabilitare i tag di sessione

EKS Pod Identity aggiunge un set predefinito di tag di sessione quando assume il ruolo. Questi tag di sessione consentono agli amministratori di creare un singolo ruolo in grado di funzionare su più risorse, consentendo l'accesso alle AWS risorse in base ai tag corrispondenti.

Disattivare i tag di sessione

I tag di sessione sono abilitati automaticamente con EKS Pod Identity: non è richiesta alcuna azione da parte dell'utente. Per impostazione predefinita, EKS Pod Identity allega una serie

di tag predefiniti alla sessione. Per fare riferimento a tali tag nelle politiche, utilizza la sintassi `${aws:PrincipalTag/` seguita dalla chiave del tag. Ad esempio, `${aws:PrincipalTag/kubernetes-namespace}`.

- `eks-cluster-arn`
- `eks-cluster-name`
- `kubernetes-namespace`
- `kubernetes-service-account`
- `kubernetes-pod-name`
- `kubernetes-pod-uid`

Disattiva i tag di sessione

AWS comprime i criteri di sessione in linea, i criteri ARNs gestiti e i tag di sessione in un formato binario compresso con un limite separato. Se ricevi un errore `PackedPolicyTooLarge` che indica che il formato binario compresso ha superato il limite di dimensione, puoi tentare ridurlo disattivando i tag di sessione aggiunti da EKS Pod Identity. Per disattivare questi tag di sessione, procedi nel seguente modo:

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster che desideri modificare.
3. Scegliere la scheda Accesso.
4. Nelle associazioni Pod Identity, scegli l'ID di associazione che desideri modificare in ID di associazione, quindi scegli Modifica.
5. In Tag di sessione, scegli Disattiva i tag di sessione.
6. Scegli Save changes (Salva modifiche).

Tag tra account

Tutti i tag di sessione aggiunti da EKS Pod Identity sono transitivi; le chiavi e i valori dei tag vengono passati a tutte le azioni `AssumeRole` utilizzate dai carichi di lavoro per cambiare ruolo in un altro account. È possibile utilizzare questi tag nelle policy di altri account per limitare l'accesso in scenari tra account. Per ulteriori informazioni, consulta Concatenamento di ruoli con i tag di sessione nella Guida per l'utente di IAM.

Tag personalizzati

EKS Pod Identity non può aggiungere tag personalizzati supplementari all'azione `AssumeRole` che esegue. Tuttavia, i tag applicati al ruolo IAM sono sempre disponibili nello stesso formato: `${aws:PrincipalTag/` seguito dalla chiave, ad esempio `${aws:PrincipalTag/MyCustomTag}`.

> **Note**
>
> I tag aggiunti alla sessione tramite la richiesta `sts:AssumeRole` hanno la precedenza in caso di conflitto. Ad esempio, supponiamo che:
>
> - Amazon EKS aggiunge una chiave `eks-cluster-name` e un valore `my-cluster` alla sessione quando EKS assume il ruolo di cliente e
> - Hai aggiunto anche un tag `eks-cluster-name` al ruolo IAM con valore `my-own-cluster`.
>
> In questo caso, il primo ha la precedenza e il valore del tag per `eks-cluster-name` sarà `my-cluster`.

Utilizzo di Pod Identity con SDK AWS

Utilizzo delle credenziali EKS Pod Identity

Per utilizzare le credenziali di un'associazione EKS Pod Identity, il codice può utilizzare qualsiasi SDK AWS per creare un client per un servizio AWS con un SDK e, per impostazione predefinita, l'SDK cerca le credenziali AWS Identity and Access Management da utilizzare in una catena di posizioni. Si utilizzano le credenziali EKS Pod Identity se non si specifica un provider di credenziali quando si crea il client o si inizializza in altro modo l'SDK.

Questo avviene perché le associazioni EKS Pod Identity sono state aggiunte al provider di credenziali del container, che viene cercato in un passaggio nella catena di credenziali predefinita. Se i carichi di lavoro utilizzano attualmente credenziali che si trovano all'inizio della catena di credenziali, quest'ultime continuano a essere utilizzate anche se configuri un'associazione EKS Pod Identity per lo stesso carico di lavoro.

Per ulteriori informazioni sul funzionamento delle associazioni EKS Pod Identity, consulta the section called "Come funziona".

Quando utilizzi Ulteriori informazioni su come EKS Pod Identity concede ai pod l'accesso ai servizi AWS, i container nei tuoi pod devono utilizzare una versione di AWS SDK che supporta l'assunzione di un ruolo IAM dall'agente EKS Pod Identity. Accertati di utilizzare le seguenti versioni, o successive, per il tuo SDK AWS:

- Java (Versione 2) – 2.21.30
- Java – 1.12.746
- Vai a v1 – v1.47.11
- Vai a 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 – v3.458.0
- 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

Per assicurarti che l'SDK utilizzato sia supportato, quando crei i container segui la procedura di installazione per il tuo SDK preferito disponibile in Strumenti per costruire su AWS.

Per un elenco di componenti aggiuntivi che supportano EKS Pod Identity, consulta the section called "Riferimento di supporto Pod Identity".

Disabilitazione di **IPv6** nell'agente EKS Pod Identity

Console di gestione AWS

1. Per disabilitare IPv6 nell'agente EKS Pod Identity, aggiungi la seguente configurazione alle Impostazioni di configurazione opzionali del componente aggiuntivo EKS.
   a. Aprire la Console Amazon EKS.
   b. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui configurare il componente aggiuntivo.
   c. Seleziona la scheda Componenti aggiuntivi.
   d. Seleziona la casella nella parte superiore destra di quella del componente aggiuntivo relativo all'agente EKS Pod Identity, quindi scegli Modifica.
   e. Nella pagina Configura agente EKS Pod Identity:
      i. Seleziona la Versione da utilizzare. Si consiglia di mantenere la stessa versione rispetto al passaggio precedente e di aggiornare la versione e la configurazione con operazioni separate.
      ii. Scegli Impostazioni di configurazione facoltative.
      iii. Inserisci la chiave JSON `"agent":` e il valore di un oggetto JSON annidato con una chiave `"additionalArgs":` in Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. Se questa chiave e questo valore sono gli unici dati nella casella di testo, racchiudi la chiave e il valore tra parentesi graffe `{ }`. Di seguito è riportato l'esempio dell'abilitazione della policy di rete:

```
{
    "agent": {
        "additionalArgs": {
            "-b": "169.254.170.23"
        }
    }
}
```

      Questa configurazione imposta l'indirizzo IPv4 come unico indirizzo utilizzato dall'agente.
   f. Per applicare la nuova configurazione sostituendo i pod dell'agente EKS Pod Identity, scegli Salva modifiche.

      Amazon EKS applica le modifiche ai componenti aggiuntivi EKS utilizzando unrollout di Kubernetes DaemonSet per l'agente EKS Pod Identity. È possibile monitorare lo stato del

rollout nella Cronologia degli aggiornamenti del componente aggiuntivo nella Console di gestione AWS e con `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`.

`kubectl rollout` dispone dei seguenti comandi:

```
$ 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 il rollout richiede troppo tempo, Amazon EKS annullerà l'operazione e un messaggio contenente il tipo di Aggiornamento del componente aggiuntivo e lo stato Non riuscito sarà aggiunto alla Cronologia degli aggiornamenti del componente aggiuntivo. Per esaminare eventuali problemi, inizia dalla cronologia del rollout ed esegui `kubectl logs` su un pod dell'agente EKS Pod Identity per visualizzare i log dell'agente EKS Pod Identity.

2. Se la nuova voce nella Cronologia degli aggiornamenti ha lo stato Riuscito, il rollout è stato completato e il componente aggiuntivo utilizza la nuova configurazione in tutti i pod dell'agente EKS Pod Identity.

CLI AWS

1. Per disabilitare `IPv6` nell'agente EKS Pod Identity, aggiungi la seguente configurazione ai valori di configurazione del componente aggiuntivo EKS.

   Esegui il seguente comando AWS CLI. Sostituisci `my-cluster` con il nome del cluster e l'ARN del ruolo IAM con il ruolo che stai utilizzando.

   ```
   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"}}}'
   ```

   Questa configurazione imposta l'indirizzo `IPv4` come unico indirizzo utilizzato dall'agente.

Amazon EKS applica le modifiche ai componenti aggiuntivi EKS utilizzando un rollout di DaemonSet Kubernetes per l'agente EKS Pod Identity. È possibile monitorare lo stato del rollout nella Cronologia degli aggiornamenti del componente aggiuntivo nella Console di gestione AWS e con `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`.

`kubectl rollout` dispone dei seguenti comandi:

```
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 il rollout richiede troppo tempo, Amazon EKS annullerà l'operazione e un messaggio contenente il tipo di Aggiornamento del componente aggiuntivo e lo stato Non riuscito sarà aggiunto alla Cronologia degli aggiornamenti del componente aggiuntivo. Per esaminare eventuali problemi, inizia dalla cronologia del rollout ed esegui `kubectl logs` su un pod dell'agente EKS Pod Identity per visualizzare i log dell'agente EKS Pod Identity.

Creazione di un ruolo IAM con la policy di fiducia richiesta da EKS Pod Identity

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

```
    ]
}
```

`sts:AssumeRole`

EKS Pod Identity utilizza `AssumeRole` per assumere il ruolo IAM prima di passare le credenziali temporanee ai pod.

`sts:TagSession`

EKS Pod Identity utilizza `TagSession` per includere i tag di sessione nelle richieste a STS. AWS

Impostazione delle condizioni

È possibile utilizzare questi tag nelle chiavi di condizione nella policy di attendibilità per limitare gli account di servizio, i namespace e i cluster che possono utilizzare questo ruolo. Per l'elenco dei tag di richiesta aggiunti da Pod Identity, consulta the section called "Abilitare o disabilitare i tag di sessione".

Ad esempio, è possibile limitare i pod che possono assumere il ruolo di un ruolo IAM di Pod Identity a `ServiceAccount` e `Namespace` specifici con la seguente policy di fiducia con l'aggiunta di `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"
                    ]
                }
            }
        }
    ]
}
```

Per un elenco di chiavi di condizione di Amazon EKS, consultare Condizioni per Amazon Elastic Kubernetes Service in Service Authorization Reference. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consultare Operazioni definite da Amazon Elastic Kubernetes Service.

Gestire le risorse di elaborazione utilizzando i nodi

Un nodo Kubernetes è una macchina che esegue applicazioni containerizzate. Ogni nodo include i seguenti componenti:

- Runtime del container: software responsabile dell'esecuzione dei container.
- kubelet: garantisce che i container siano integri e funzionanti nel proprio pod associato.
- kube-proxy: gestisce le regole della rete che consentono la comunicazione con i pod.

Per ulteriori informazioni, consulta Nodi nella documentazione Kubernetes.

Il cluster Amazon EKS può pianificare pod su qualsiasi combinazione di nodi gestiti dalla modalità automatica EKS, nodi autogestiti, gruppi di nodi gestiti da Amazon EKS, AWS Fargate e Amazon EKS Hybrid Nodes. Per ulteriori informazioni sui nodi implementati nel cluster, consulta the section called "Accesso alle risorse di cluster".

> **Note**
>
> A eccezione dei nodi ibridi, i nodi devono trovarsi nello stesso VPC delle sottoreti selezionate al momento della creazione del cluster, ma non necessariamente nelle stesse sottoreti.

Confronto delle opzioni di calcolo

Nella tabella seguente vengono forniti diversi criteri per valutare quali opzioni sono più adatte alle proprie esigenze. I nodi autogestiti sono un'opzione alternativa che supporta tutti i criteri elencati, ma che richiede molta più manutenzione manuale. Per ulteriori informazioni, consulta the section called "Nodi autogestiti".

> **Note**
>
> Bottlerocket presenta alcune differenze specifiche rispetto alle informazioni generali contenute in questa tabella. Per ulteriori informazioni, consultare la documentazione relativa a Bottlerocket su GitHub.

Criteri	Gruppi di nodi gestiti EKS	Modalità automatica EKS	Amazon EKS Hybrid Nodes
Può essere implement ato in AWS Outposts	No	No	No
Può essere implement ato nella zona locali AWS	Sì	No	No
Può eseguire container che richiedono Windows	Sì	No	No
Può eseguire container che richiedono Linux	Sì	Sì	Sì
Può eseguire carichi di lavoro che richiedono il chip Inferentia	Sì – Solo nodi Amazon Linux	Sì	No
Può eseguire carichi di lavoro che richiedono una GPU	Sì – Solo nodi Amazon Linux	Sì	Sì
Può eseguire carichi di lavoro che richiedono processori Arm	Sì	Sì	Sì
Può eseguire AWS Bottlerocket	Sì	Sì	No
I pod condivido no risorse di CPU, memoria, archiviaz	Sì	Sì	Sì

Criteri	Gruppi di nodi gestiti EKS	Modalità automatica EKS	Amazon EKS Hybrid Nodes
ione e rete con altri pod.			
Deve implementare e gestire le istanze Amazon EC2	Sì	No - Ulteriori informazioni sulle istanze gestite di EC2	Sì - Le macchine fisiche on-premises o virtuali sono autogestite con strumenti a scelta.
È necessario proteggere, mantenere e applicare patch al sistema operativo delle istanze Amazon EC2	Sì	No	Sì - Il sistema operativo in esecuzione sulle macchine fisiche o virtuali è autogestito con strumenti a scelta.
Può fornire argomenti di bootstrap all'implementazione di un nodo, come argomenti kubelet aggiuntivi.	Sì - Utilizzando `eksctl` o un modello di avvio con un'AMI personalizzata.	No - Utilizzare `NodeClass` per configurare i nodi	Sì - È possibile personalizzare gli argomenti di bootstrap con nodeadm. Consultare the section called "nodeadm dei nodi ibridi".
Può assegnare indirizzi IP ai pod da un intervallo CIDR diverso rispetto all'indirizzo IP assegnato al nodo.	Sì: utilizzando un modello di avvio con un'AMI personalizzata. Per ulteriori informazioni, consulta the section called "Modelli di avvio".	No	Sì - Consultare the section called "Configurazione della CNI".
Puoi eseguire SSH nel nodo	Sì	No - Informazioni su come risolvere i problemi dei nodi	Sì

Criteri	Gruppi di nodi gestiti EKS	Modalità automatica EKS	Amazon EKS Hybrid Nodes
Puoi implementare un'AMI personalizzata nei nodi	Sì – Utilizzo di un modello di avvio	No	Sì
Può implementare un CNI personalizzato nei nodi	Sì – Utilizzando un modello di avvio con un'AMI personalizzata	No	Sì
É necessario aggiornare l'AMI del nodo per conto proprio	Sì - Se è stata implementata un'AMI ottimizzata per Amazon EKS, sarà inviata una notifica nella console Amazon EKS quando gli aggiornamenti sono disponibili. È possibile eseguire l'aggiornamento con un clic nella console. Se è stata implementata un'AMI personalizzata, non sarà inviata una notifica nella console Amazon EKS quando gli aggiornamenti sono disponibili. È necessario eseguire l'aggiornamento per conto proprio.	No	Sì - Il sistema operativo eseguito sulle macchine fisiche o virtuali è autogesti to con strumenti a scelta. Consultare the section called "Preparazione del sistema operativo".

Criteri	Gruppi di nodi gestiti EKS	Modalità automatica EKS	Amazon EKS Hybrid Nodes
È necessario aggiornare la versione del nodo Kubernetes per conto proprio	Sì - Se è stata implementata un'AMI ottimizzata per Amazon EKS, sarà inviata una notifica nella console Amazon EKS quando gli aggiornamenti sono disponibili. È possibile eseguire l'aggiornamento con un clic nella console. Se è stata implementata un'AMI personalizzata, non sarà inviata una notifica nella console Amazon EKS quando gli aggiornamenti sono disponibili. È necessario eseguire l'aggiornamento per conto proprio.	No	Sì - Gli aggiornamenti dei nodi ibridi vengono gestiti con strumenti a scelta o con `nodeadm`. Consultare the section called "Aggiornamento dei nodi ibridi".
Può utilizzare lo spazio di archiviazione Amazon EBS con i pod	Sì	Sì, come funzionalità integrata. Informazioni su come creare una classe di archiviazione.	No
Può utilizzare lo spazio di archiviazione Amazon EFS con i pod	Sì	Sì	No

Criteri	Gruppi di nodi gestiti EKS	Modalità automatica EKS	Amazon EKS Hybrid Nodes
Può utilizzare lo spazio di archiviazione Amazon FSx per Lustre con i pod	Sì	Sì	No
Può utilizzare Network Load Balancer per i servizi	Sì	Sì	Sì - È necessario utilizzare il tipo di destinazione `ip`.
I pod possono essere eseguiti in una sottorete pubblica	Sì	Sì	No - I pod vengono eseguiti in un ambiente on-premises.
Può assegnare diversi gruppi di sicurezza VPC a singoli pod	Sì – Solo nodi Linux	No	No
Può eseguire Kubernetes DaemonSets	Sì	Sì	Sì
Supporto per `HostPort` e `HostNetwork` nel manifesto pod	Sì	Sì	Sì
Disponibilità nelle regioni AWS	Tutte le regioni supportate da Amazon EKS	Tutte le Regioni Amazon EKS supportate	Tutte le Regioni supportate da Amazon EKS, a eccezione delle Regioni AWS GovCloud (Stati Uniti) e le Regioni della Cina.

Criteri	Gruppi di nodi gestiti EKS	Modalità automatica EKS	Amazon EKS Hybrid Nodes
Può eseguire container su host dedicati Amazon EC2	Sì	No	No
Prezzi	Costo dell'istanza Amazon EC2 che esegue più pod. Per ulteriori informazioni, consulta Prezzi di Amazon EC2.	Quando la modalità automatica EKS è abilitata nel cluster, viene addebitata una tariffa separata per le istanze avviate utilizzando la funzionalità di calcolo della modalità automatica, in aggiunta ai costi standard delle istanze EC2. L'importo varia a seconda del tipo di istanza avviata e della Regione AWS in cui si trova il cluster. Per ulteriori informazioni, consultare Prezzi di Amazon EKS.	Costo dei nodi ibridi in vCPU all'ora. Per ulteriori informazioni, consultare Prezzi di Amazon EKS.

Semplifica il ciclo di vita dei nodi con gruppi di nodi gestiti

I gruppi di nodi gestiti di Amazon EKS automatizzano il provisioning e la gestione del ciclo di vita dei nodi (EC2 istanze Amazon) per i cluster Amazon EKS Kubernetes.

Con i gruppi di nodi gestiti di Amazon EKS, non è necessario effettuare il provisioning o registrare separatamente EC2 le istanze Amazon che forniscono capacità di calcolo per eseguire le applicazioni Kubernetes. È possibile creare, aggiornare o terminare automaticamente i nodi per il cluster con una

singola operazione. Gli aggiornamenti e le interruzioni dei nodi svuotano automaticamente i nodi per garantire che le applicazioni rimangano disponibili.

Ogni nodo gestito viene fornito come parte di un gruppo Amazon EC2 Auto Scaling gestito per te da Amazon EKS. Ogni risorsa, incluse le istanze e i gruppi Auto Scaling, viene eseguita all'interno AWS del tuo account. Ogni gruppo di nodi viene eseguito su più zone di disponibilità definite.

I gruppi di nodi gestiti possono anche sfruttare opzionalmente la riparazione automatica, che monitora continuamente lo stato di integrità dei nodi. In pratica, reagisce automaticamente ai problemi rilevati e sostituisce i nodi quando possibile. Ciò aiuta la disponibilità complessiva del cluster con un intervento manuale minimo. Per ulteriori informazioni, consulta the section called "Integrità del nodo".

Puoi aggiungere un gruppo di nodi gestiti a cluster nuovi o esistenti utilizzando la console Amazon EKS, la AWS CLI `eksctl` AWS , l'API o l'infrastruttura come strumenti di codice, tra cui. AWS CloudFormation I nodi avviati come parte di un gruppo di nodi gestiti vengono automaticamente taggati per l'individuazione automatica tramite Cluster Autoscaler di Kubernetes. É possibile utilizzare il gruppo di nodi per applicare le etichette Kubernetes ai nodi e aggiornarle in qualsiasi momento.

Non ci sono costi aggiuntivi per l'utilizzo dei gruppi di nodi gestiti di Amazon EKS, paghi solo per le AWS risorse fornite. Questi includono EC2 istanze Amazon, volumi Amazon EBS, orari dei cluster Amazon EKS e qualsiasi altra AWS infrastruttura. Non sono previste tariffe minime né impegni anticipati.

Per iniziare a utilizzare un nuovo cluster Amazon EKS e un gruppo di nodi gestiti, vedi the section called "Crea cluster (console e CLI)".

Per aggiungere un gruppo di nodi gestiti a un cluster esistente, vedi the section called "Crea".

Concetti sui gruppi di nodi gestiti

- I gruppi di nodi gestiti di Amazon EKS creano e gestiscono EC2 istanze Amazon per te.
- Ogni nodo gestito viene fornito come parte di un gruppo Amazon EC2 Auto Scaling gestito per te da Amazon EKS. Inoltre, tutte le risorse, incluse EC2 le istanze Amazon e i gruppi di Auto Scaling, vengono eseguite all'interno AWS del tuo account.
- Il gruppo Auto Scaling di un gruppo di nodi gestiti si estende su ogni sottorete specificata al momento della creazione del gruppo.
- Amazon EKS applica tag alle risorse del gruppo di nodi gestiti in modo che siano configurate per utilizzare Cluster Autoscaler di Kubernetes.

> **Important**
>
> Se si esegue un'applicazione stateful in più zone di disponibilità supportate dai volumi Amazon EBS e che utilizza il Cluster Autoscaler di Kubernetes, è necessario configurare più gruppi di nodi, ognuno dei quali definito per una singola zona di disponibilità. Inoltre, è necessario abilitare la funzionalità `--balance-similar-node-groups`.

- È possibile utilizzare un modello di avvio personalizzato per un maggiore livello di flessibilità e personalizzazione durante l'implementazione dei nodi gestiti. Ad esempio, è possibile specificare argomenti `kubelet` aggiuntivi e utilizzare un'AMI personalizzata. Per ulteriori informazioni, consulta the section called "Modelli di avvio". Se non si utilizza un modello di avvio personalizzato quando si crea per la prima volta un gruppo di nodi gestiti, è disponibile un modello di avvio generato automaticamente. Non modificare manualmente questo modello generato automaticamente o si verificheranno errori.
- Amazon EKS segue il modello di responsabilità condivisa CVEs e le patch di sicurezza sui gruppi di nodi gestiti. Quando i nodi gestiti eseguono un'AMI ottimizzata per Amazon EKS, Amazon EKS è responsabile della creazione di versioni con patch dell'AMI quando vengono segnalati bug o problemi. A questo scopo vengono pubblicate delle correzioni. Sei invece responsabile della distribuzione di queste versioni AMI con patch ai gruppi di nodi gestiti. Quando i nodi gestiti eseguono un'AMI personalizzata, sei responsabile della creazione di versioni con patch dell'AMI quando vengono segnalati bug o problemi, oltre che della distribuzione dell'AMI. Per ulteriori informazioni, consulta the section called "Aggiornamento".
- I gruppi di nodi gestiti Amazon EKS possono essere avviati in sottoreti pubbliche e private. Se si avvia un gruppo di nodi gestiti in una sottorete pubblica in data 22 aprile 2020 o successiva, sarà necessario che la sottorete abbia `MapPublicIpOnLaunch` impostato su vero affinché le istanze possano partecipare correttamente a un cluster. Se la sottorete pubblica è stata creata utilizzando `eksctl` o i AWS CloudFormation modelli forniti da Amazon EKS a partire dal 26 marzo 2020, questa impostazione è già impostata su true. Se le sottoreti pubbliche sono state create prima del 26 marzo 2020 devi modificare manualmente l'impostazione. Per ulteriori informazioni, consulta Modifica dell'attributo di IPv4 indirizzamento pubblico per la sottorete.
- Quando implementi un gruppo di nodi gestito in sottoreti private, devi assicurarti che possa accedere ad Amazon ECR per estrarre le immagini dei container. È possibile farlo collegando un gateway NAT alla tabella di routing della sottorete o aggiungendo i seguenti endpoint VPC AWS PrivateLink :
  - Interfaccia endpoint dell'API di Amazon ECR: `com.amazonaws.region-code.ecr.api`

  - Interfaccia endpoint dell'API del registro Docker di Amazon ECR: `com.amazonaws.region-code.ecr.dkr`
  - Endpoint del gateway Amazon S3: `com.amazonaws.region-code.s3`

  Per altri servizi ed endpoint di uso comune, consulta la sezione the section called "Cluster privati".
- I gruppi di nodi gestiti non possono essere distribuiti su AWS Outposts o in AWS Wavelength. È possibile creare gruppi di nodi gestiti in AWS Local Zones. Per ulteriori informazioni, consulta the section called " AWSZone locali ".
- È possibile creare più gruppi di nodi gestiti all'interno di un singolo cluster. Ad esempio, puoi creare un gruppo di nodi con l'AMI Linux standard ottimizzata per Amazon EKS per alcuni carichi di lavoro e un altro gruppo con la variante GPU per i carichi di lavoro che richiedono il supporto della GPU.
- Se il tuo gruppo di nodi gestiti rileva un errore di controllo dello stato delle EC2 istanze Amazon, Amazon EKS restituisce un codice di errore per aiutarti a diagnosticare il problema. Per ulteriori informazioni, consulta the section called "Codici di errore dei gruppi di nodi gestiti".
- Amazon EKS aggiunge etichette Kubernetes alle istanze del gruppo di nodi gestiti. Queste etichette fornite da Amazon EKS hanno il prefisso `eks.amazonaws.com`.
- Amazon EKS svuota automaticamente i nodi utilizzando l'API Kubernetes durante le interruzioni o gli aggiornamenti.
- I budget di interruzione del pod non vengono rispettati quando si termina un nodo con `AZRebalance` o si riduce il numero di nodi desiderato. Queste operazioni cercano di espellere pod dal nodo. Tuttavia, se trascorrono più di 15 minuti, il nodo viene terminato a prescindere dal fatto che tutti i pod sul nodo siano stati terminati. Per prolungare il periodo fino alla terminazione del nodo, aggiungi un hook del ciclo di vita al gruppo con scalabilità automatica. Per ulteriori informazioni, consulta Add lifecycle hook nella Amazon EC2 Auto Scaling User Guide.
- Per eseguire correttamente il processo di scarico dopo aver ricevuto una notifica di interruzione spot o una notifica di ribilanciamento della capacità, `CapacityRebalance` deve essere impostato su `true`.
- L'aggiornamento di gruppi di nodi gestiti rispettano i budget di interruzione dei pod impostati per i pod. Per ulteriori informazioni, consulta the section called "Dettagli sull'aggiornamento del comportamento".
- I gruppi di nodi gestiti Amazon EKS non prevedono costi aggiuntivi. Paghi solo per le AWS risorse che fornisci.
- Se si desidera crittografare i volumi Amazon EBS per i nodi, è possibile implementare i nodi utilizzando un modello di avvio. Per implementare nodi gestiti con volumi Amazon EBS crittografati

senza utilizzare un modello di avvio, crittografare tutti i nuovi volumi Amazon EBS creati nell'account. Per ulteriori informazioni, consulta Encryption by default nella Amazon EC2 User Guide.

Tipi di capacità del gruppo di nodi gestiti

Quando si crea un gruppo di nodi gestito, è possibile scegliere il tipo di capacità su on demand o Spot. Amazon EKS implementa un gruppo di nodi gestiti con un gruppo Amazon EC2 Auto Scaling che contiene solo istanze On-Demand o solo istanze Amazon EC2 Spot. All'interno di un singolo cluster Kubernetes, è possibile pianificare i pod per le applicazioni tolleranti ai guasti sia per i gruppi di nodi gestiti Spot che per i gruppi di nodi on-demand. Per impostazione predefinita, un gruppo di nodi gestiti distribuisce istanze Amazon EC2 On-Demand.

On demand

Con Istanze on demand, sono previsti costi per la capacità di calcolo entro la seconda ora senza impegni a lungo termine.

Per impostazione predefinita, se non specifichi un Tipo capacità, per il gruppo di nodi gestiti viene eseguito il provisioning con Istanze on demand. Un gruppo di nodi gestito configura un gruppo Amazon EC2 Auto Scaling per tuo conto con le seguenti impostazioni applicate:

- La strategia di allocazione per il provisioning della capacità On-Demand è impostata su `prioritized`. Il gruppo di nodi gestiti utilizza l'ordine dei tipi di istanze approvata dall'API per determinare quale tipo di istanza utilizzare prima per soddisfare la capacità on demand. Ad esempio, è possibile specificare tre tipi di istanza nell'ordine seguente: `c5.large`, `c4.large`, e `c3.large`. Quando le istanze on demand vengono avviate, il gruppo di nodi gestiti soddisfa la capacità on demand a partire da `c5.large`, quindi `c4.large`, e infine `c3.large`. Per ulteriori informazioni, consulta il gruppo Amazon EC2 Auto Scaling nella Amazon Auto EC2 Scaling User Guide.
- Amazon EKS aggiunge la seguente etichetta Kubernetes a tutti i nodi del gruppo di nodi gestito che specifica il tipo di capacità: `eks.amazonaws.com/capacityType: ON_DEMAND`. È possibile utilizzare questa etichetta sui nodi on demand per pianificare applicazioni con stato o fault intolerant.

Spot

Le istanze Amazon EC2 Spot sono una EC2 capacità Amazon inutilizzata che offre forti sconti sui prezzi On-Demand. Le istanze Amazon EC2 Spot possono essere interrotte con un avviso di interruzione di due minuti quando è EC2 necessario ripristinare la capacità. Per ulteriori informazioni, consulta le istanze Spot nella Amazon EC2 User Guide. Puoi configurare un gruppo di nodi gestiti con Amazon EC2 Spot Instances per ottimizzare i costi per i nodi di calcolo in esecuzione nel tuo cluster Amazon EKS.

Per utilizzare istanze spot all'interno di un gruppo di nodi gestiti, creare un gruppo di nodi gestiti impostando il tipo di capacità come `spot`. Un gruppo di nodi gestito configura un gruppo Amazon EC2 Auto Scaling per tuo conto applicando le seguenti best practice Spot:

- Per garantire che il provisioning dei nodi Spot venga eseguito nei pool di capacità Spot ottimali, la strategia di allocazione è impostata su uno dei seguenti valori:
  - `price-capacity-optimized` (PCO): quando si creano nuovi gruppi di nodi in un cluster con Kubernetes versione `1.28` o successive, la strategia di allocazione è impostata su `price-capacity-optimized`. Tuttavia, la strategia di allocazione non viene modificata per i gruppi di nodi già creati con `capacity-optimized` prima che i gruppi di nodi gestiti da Amazon EKS inizino a supportare PCO.
  - `capacity-optimized` (CO): quando si creano nuovi gruppi di nodi in un cluster con Kubernetes versione `1.27` o precedenti, la strategia di allocazione è impostata su `capacity-optimized`.

  Per aumentare il numero di pool di capacità Spot disponibili da cui allocare le capacità, configurare un gruppo di nodi gestiti per l'utilizzo di più tipi di istanza.
- Amazon EC2 Spot Capacity Rebalancing è abilitato in modo che Amazon EKS possa drenare e ribilanciare in modo corretto i nodi Spot per ridurre al minimo l'interruzione delle applicazioni quando un nodo Spot è a elevato rischio di interruzione. Per ulteriori informazioni, consulta Amazon EC2 Auto Scaling Capacity Rebalancing nella Amazon Auto Scaling EC2 User Guide.
  - Quando un nodo Spot riceve un consiglio di ribilanciamento, Amazon EKS tenta automaticamente di avviare un nuovo nodo Spot sostitutivo.
  - Se un avviso di interruzione Spot di due minuti arriva prima che il nodo Spot sostitutivo si trovi in uno stato `Ready`, Amazon EKS inizia a svuotare il nodo Spot che ha ricevuto il suggerimento di ribilanciamento. Amazon EKS svuota il nodo nel modo migliore possibile. Di conseguenza, non c'è alcuna garanzia che Amazon EKS attenda che il nodo sostitutivo si unisca al cluster prima di svuotare il nodo esistente.

  - Quando un nodo Spot sostitutivo viene avviato ed è allo stato `Ready` su Kubernetes, Amazon EKS isola e svuota il nodo Spot che ha ricevuto il suggerimento di ribilanciamento. L'isolamento del nodo Spot assicura che il controller di servizio non invii nuove richieste a questo nodo Spot. Il nodo viene rimosso anche dall'elenco di nodi Spot sani e attivi. Lo svuotamento del nodo Spot assicura che i pod in esecuzione vengano espulsi in modo corretto.
- Amazon EKS aggiunge la seguente etichetta Kubernetes a tutti i nodi del gruppo di nodi gestito che specifica il tipo di capacità: `eks.amazonaws.com/capacityType: SPOT`. È possibile utilizzare questa etichetta per pianificare applicazioni fault tolerant sui nodi Spot.

> **Important**
>
> EC2 emette un avviso di interruzione Spot due minuti prima della chiusura dell'istanza Spot. Tuttavia, i Pod sui nodi Spot potrebbero non ricevere l'intera finestra di 2 minuti necessaria per uno spegnimento regolare. Quando EC2 emette l'avviso, c'è un ritardo prima che Amazon EKS inizi a sfrattare i Pods. Gli sfratti avvengono in sequenza per proteggere il server dell'API Kubernetes, quindi durante più reclami Spot simultanei, alcuni Pod potrebbero ricevere avvisi di sfratto ritardato. I Pod possono essere chiusi forzatamente senza ricevere segnali di terminazione, in particolare su nodi ad alta densità di Pod, durante reclami simultanei o quando si utilizzano periodi di tolleranza di terminazione prolungati. Per i carichi di lavoro Spot, consigliamo di progettare le applicazioni in modo da tollerare le interruzioni, utilizzando periodi di tolleranza di terminazione pari o inferiori a 30 secondi, evitando agganci PreStop di lunga durata e monitorando le metriche di sfratto dei Pod per comprendere i periodi di tolleranza effettivi nei cluster. Per i carichi di lavoro che richiedono una terminazione corretta e garantita, consigliamo invece di utilizzare la capacità On-Demand.

Quando si decide se implementare un gruppo di nodi con capacità On-Demand o Spot, è necessario considerare le seguenti condizioni:

- Le istanze Spot sono consigliate per applicazioni senza stato, fault tolerant e flessibili. Questi includono carichi di lavoro di formazione in batch e machine learning, big data ETLs come Apache Spark, applicazioni di elaborazione delle code ed endpoint API stateless. Poiché Spot è una EC2 capacità Amazon inutilizzata, che può cambiare nel tempo, ti consigliamo di utilizzare la capacità Spot per carichi di lavoro tolleranti alle interruzioni. Più specificamente, la capacità Spot è adatta per carichi di lavoro in grado di tollerare periodi in cui la capacità richiesta non è disponibile.

- Consigliamo di utilizzare On-demand per le applicazioni che sono fault intolerant. Ciò include strumenti di gestione dei cluster come strumenti di monitoraggio e operativi, implementazioni che richiedono `StatefulSets` e applicazioni con stato, come database.
- Per ottimizzare la disponibilità delle applicazioni durante l'utilizzo di istanze Spot, è consigliabile configurare un gruppo di nodi gestiti Spot per l'utilizzo di più tipi di istanza. Quando si utilizzano più tipi di istanza, si consiglia di applicare le seguenti regole:
  - All'interno di un gruppo di nodi gestiti, se utilizzi Cluster Autoscaler, ti consigliamo di utilizzare un set flessibile di tipi di istanza con la stessa quantità di vCPU e risorse di memoria. Questo per garantire che i nodi del cluster vengano dimensionati come previsto. Ad esempio, se hai bisogno di quattro v CPUs e otto GiB di memoria,,`c3.xlarge`,`c4.xlarge`, `c5.xlarge` `c5d.xlarge` `c5a.xlarge`,`c5n.xlarge`, o altri tipi di istanza simili.
  - Per migliorare la disponibilità delle applicazioni, si consiglia di implementare più gruppi di nodi gestiti Spot. Per questo, ogni gruppo dovrebbe utilizzare un set flessibile di tipi di istanza con le stesse risorse vCPU e memoria. Ad esempio, se hai bisogno di memoria 4 v CPUs e 8 GiB, ti consigliamo di creare un gruppo di nodi gestiti con`c3.xlarge`,,,`c4.xlarge`, `c5.xlarge` `c5d.xlarge` `c5a.xlarge`,`c5n.xlarge`, o altri tipi di istanze simili e un secondo gruppo di nodi gestiti con`m3.xlarge`,,, `m4.xlarge` `m5.xlarge` `m5d.xlarge`,`m5a.xlarge`, `m5n.xlarge` o altri tipi di istanze simili.
  - Quando si implementa il gruppo di nodi con il tipo di capacità Spot che utilizza un modello di avvio personalizzato, utilizza l'API per passare più tipi di istanza. Non passare un singolo tipo di istanza tramite il modello di avvio. Per ulteriori informazioni sull'implementazione di un gruppo di nodi tramite un modello di avvio, vedere the section called "Modelli di avvio".

Creare un gruppo di nodi gestiti per il cluster

In questo argomento viene descritto come avviare gruppi di nodi gestiti Amazon EKS per nodi che si registrano con il cluster Amazon EKS. Dopo che i nodi vengono aggiunti al cluster, puoi implementare applicazioni Kubernetes per gli stessi.

Se si avvia un gruppo di nodi gestiti Amazon EKS per la prima volta, consigliamo invece di seguire una delle nostre guide in Nozioni di base. Queste guide forniscono procedure dettagliate per la creazione di un cluster Amazon EKS con nodi.

Important

- I nodi Amazon EKS sono istanze Amazon EC2 standard. La fatturazione viene effettuata in base ai normali prezzi Amazon EC2. Per ulteriori informazioni, consulta Prezzi di Amazon EC2.
- Non puoi creare nodi gestiti in una AWS regione in cui AWS Outposts o Wavelength AWS sono abilitati. È possibile, invece creare nodi autogestiti. Per ulteriori informazioni, consultare the section called "Amazon Linux", the section called "Windows" e the section called "Portabottiglie". Puoi anche creare un gruppo di nodi Amazon Linux autogestito su un Outpost. Per ulteriori informazioni, consulta the section called "Nodi".
- Se non si specifica un ID AMI per il file `bootstrap.sh` incluso in Linux o Bottlerocket ottimizzato per Amazon EKS, i gruppi di nodi gestiti impongono un numero massimo sul valore di `maxPods`. Per le istanze con meno di 30 vCPUs, il numero massimo è. `110` Per le istanze con più di 30 vCPUs, il numero massimo salta a. `250` Questa applicazione ha la precedenza su altre `maxPods` configurazioni, tra cui. `maxPodsExpression` Per ulteriori informazioni su come `maxPods` viene determinato e su come personalizzarlo, vedere. the section called "Come viene determinato MaxPods"

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta the section called "Creazione di un cluster ".
- Un ruolo IAM esistente per i nodi da utilizzare. Per crearne uno, consulta the section called "Ruolo IAM del nodo". Se questo ruolo non prevede nessuna delle policy per VPC CNI, per i pod VPC CNI è necessario il ruolo separato riportato di seguito.
- (Facoltativo, ma consigliato) Il componente aggiuntivo plug-in CNI di Amazon VPC per Kubernetes configurato con il proprio ruolo IAM a cui è allegata la policy IAM necessaria. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
- Familiarità con le considerazioni presenti alla pagina Choose an optimal Amazon EC2 node instance type. A seconda del tipo di istanza scelto, potrebbero esserci ulteriori prerequisiti per il cluster e il VPC.
- Per aggiungere un gruppo di nodi gestiti da Windows, è prima necessario abilitare il supporto Windows per il cluster. Per ulteriori informazioni, consulta the section called "Abilitazione del supporto Windows".

È possibile creare un gruppo di nodi gestito con uno di questi due strumenti:

- the section called "eksctl"
- the section called "Console di gestione AWS"

eksctl

Creare un gruppo di nodi gestito con eksctl

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

1. (Facoltativo) Se la policy IAM gestita AmazonEKS_CNI_Policy è collegata al ruolo IAM del nodo Amazon EKS, si consiglia, invece, di assegnarla a un ruolo IAM associato all'account di servizio Kubernetes `aws-node`. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
2. Creare un gruppo di nodi gestiti con o senza utilizzare un modello di avvio personalizzato. La specifica manuale di un modello di avvio consente una maggiore personalizzazione di un gruppo di nodi. Ad esempio, può consentire l'implementazione di un'AMI personalizzata o la presenza di argomenti sullo script `boostrap.sh` in un'AMI ottimizzata per Amazon EKS. Per avere un elenco completo di tutte le opzioni e le impostazioni predefinite disponibili, inserisci il comando seguente.

   ```
   eksctl create nodegroup --help
   ```

   Nel comando seguente, sostituisci *`my-cluster`* con il nome del tuo cluster e sostituisci *`my-mng`* con il nome del gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura.

   > **Important**
   >
   > Se non si utilizza un modello di avvio personalizzato durante la prima creazione di un gruppo di nodi gestiti, non utilizzarne uno per il gruppo di nodi in un secondo momento. Se non è stato specificato un modello di avvio personalizzato, il sistema genera

automaticamente un modello di avvio che non è consigliabile modificare manualmente. La modifica manuale di questo modello di avvio generato in automatico potrebbe causare errori.

Senza un modello di avvio

`eksctl` crea un modello di avvio Amazon EC2 predefinito nell'account e implementa il gruppo di nodi utilizzando un modello di avvio creato in base alle opzioni specificate. Prima di specificare un valore per `--node-type`, consulta the section called "Tipi di istanza Amazon EC2".

Sostituisci *`ami-family`* con una parola chiave consentita. Per ulteriori informazioni, consulta Setting the node AMI Family (Impostazione della famiglia AMI dei nodi) nella documentazione di `eksctl`. Sostituisci *`my-key`* con il nome della coppia di chiavi Amazon EC2 o della chiave pubblica. Questa chiave viene utilizzata per eseguire il SSH nei nodi dopo il loro avvio.

> **Note**
>
> Per Windows, questo comando non abilita SSH. Associa, invece, la coppia di chiavi Amazon EC2 all'istanza e ti consente di eseguire il protocollo RDP nell'istanza.

Se non si dispone di una coppia di chiavi Amazon EC2, è possibile crearla nella Console di gestione AWS. Per informazioni su Linux, consultare Amazon EC2 key pairs and Linux instances nella Guida per l'utente di Amazon EC2. Per informazioni su Windows, consultare Amazon EC2 key pairs and Windows instances nella Guida per l'utente di Amazon EC2.

Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:

- Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
- Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Se si desidera bloccare l'accesso del pod a IMDS, aggiungere l'opzione `--disable-pod-imds` al seguente comando.

```
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
```

Le istanze possono opzionalmente assegnare un numero di indirizzi IP significativamente superiore ai pod, assegnare indirizzi IP ai pod da un intervallo CIDR diverso da quello dell'istanza ed essere distribuite in un cluster senza accesso a Internet. Per ulteriori informazioni, consultare the section called "Aumentare gli indirizzi IP", the section called "Rete personalizzata", e the section called "Cluster privati" per ulteriori opzioni da aggiungere al comando precedente.

I gruppi di nodi gestiti calcolano e applicano un singolo valore per il numero massimo di pod che possono essere eseguiti su ogni nodo del gruppo in base al tipo di istanza. Se si crea un gruppo di nodi con tipi di istanza diversi, il valore più piccolo calcolato in tutti i tipi di istanza viene applicato come numero massimo di pod che possono essere eseguiti su ogni tipo di istanza nel gruppo di nodi. I gruppi di nodi gestiti calcolano il valore utilizzando lo script a cui si fa riferimento nella pagina Amazon EKS recommended maximum Pods for each Amazon EC2 instance type.

Con un modello di lancio

Il modello di avvio deve esistere già e deve soddisfare i requisiti specificati nella pagina Launch template configuration basics. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:

- Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
- Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Se si desidera bloccare l'accesso del pod a IMDS, specificare le impostazioni necessarie nel modello di avvio.

a. Copia i seguenti contenuti sul dispositivo. Sostituisci i valori di esempio e quindi esegui il comando modificato per creare il file. `eks-nodegroup.yaml` Diverse impostazioni specificate durante l'implementazione senza un modello di avvio vengono spostate nel modello di avvio. Se non si specifica una `version`, viene utilizzata la versione predefinita del modello.

```
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
```

Per un elenco completo delle impostazioni del file di configurazione `eksctl`, consulta Schema dei file config nella documentazione di `eksctl`. Le istanze possono opzionalmente assegnare un numero di indirizzi IP significativamente superiore ai pod, assegnare indirizzi IP ai pod da un intervallo CIDR diverso da quello dell'istanza ed essere implementate in un cluster senza accesso a Internet in uscita. Per ulteriori informazioni, consultare the section called "Aumentare gli indirizzi IP", the section called "Rete personalizzata", e the section called "Cluster privati" per altre opzioni da aggiungere al file di configurazione.

Se non è stato specificato un ID AMI nel modello di avvio, i gruppi di nodi gestiti calcolano e applicano un singolo valore per il numero massimo di pod che possono essere eseguiti su ogni nodo del gruppo in base al tipo di istanza. Se si crea un gruppo di nodi con tipi di istanza diversi, il valore più piccolo calcolato in tutti i tipi di istanza viene applicato come numero massimo di pod che possono essere eseguiti su ogni tipo di istanza nel gruppo di nodi. I gruppi di nodi gestiti calcolano il valore utilizzando lo script a cui si fa riferimento nella pagina Amazon EKS recommended maximum Pods for each Amazon EC2 instance type.

Se nel modello di avvio è stato specificato un ID AMI, specificare il numero massimo di pod che possono essere eseguiti su ciascun nodo del gruppo se si utilizza una rete personalizzata o se si vuole aumentare il numero di indirizzi IP assegnati all'istanza. Per ulteriori informazioni, consulta the section called "Per ogni tipo di istanza Amazon EC2, Amazon EKS consiglia un numero massimo di pod".

b. Implementare il gruppo di nodi mediante il comando seguente.

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

Console di gestione AWS

Creare un gruppo di nodi gestiti utilizzando la Console di gestione AWS

1. Attendi che lo stato del cluster risulti `ACTIVE`. Non è possibile creare un gruppo di nodi gestiti per un cluster non ancora `ACTIVE`.
2. Aprire la Console Amazon EKS.
3. Scegli il nome del cluster in cui desideri creare un gruppo di nodi gestiti.
4. Seleziona la scheda Compute (Calcolo).
5. Scegli Add node group (Aggiungi gruppo di nodi).
6. Nella pagina Configura il gruppo di nodi, compila i parametri di conseguenza e quindi scegli Successivo.
   - Nome: inserisci un nome univoco per il gruppo di nodi gestiti. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura.
   - Ruolo IAM del nodo: scegli il ruolo dell'istanza del nodo da utilizzare con il gruppo di nodi. Per ulteriori informazioni, consulta the section called "Ruolo IAM del nodo".

   > **Important**
   > - Non è possibile usare lo stesso ruolo utilizzato per creare i cluster.
   > - Consigliamo di utilizzare un ruolo che non è attualmente in uso da alcun gruppo di nodi autogestiti. In caso contrario, prevedi l'utilizzo di un nuovo gruppo di nodi autogestiti. Per ulteriori informazioni, consulta the section called "Elimina".

   - Usa modello di avvio (Facoltativo): scegli se desideri utilizzare un modello di avvio esistente. Seleziona un Launch Template Name (Nome del modello di avvio). Quindi, seleziona una Launch template version (Versione del modello di avvio). Se non si seleziona una versione, Amazon EKS utilizza la versione predefinita del modello. I modelli di avvio permettono una maggiore personalizzazione del gruppo di nodi, ad esempio consentendo di implementare

un'AMI personalizzata, assegnare un numero significativamente superiore di indirizzi IP ai pod, assegnare indirizzi IP ai pod da un intervallo CIDR diverso da quello dell'istanza e implementare i nodi in un cluster senza accesso a Internet in uscita. Per ulteriori informazioni, consultare the section called "Aumentare gli indirizzi IP", the section called "Rete personalizzata" e the section called "Cluster privati".

Il modello di avvio deve soddisfare i requisiti descritti nella pagina Customize managed nodes with launch templates. Se non si utilizza il proprio modello, l'API di Amazon EKS crea un modello di avvio Amazon EC2 predefinito nell'account e implementa il gruppo di nodi utilizzando il modello di avvio predefinito.

Se implementate i ruoli IAM per gli account di servizio, assegnate le autorizzazioni necessarie direttamente a ogni Pod che richiede l'accesso ai AWS servizi e nessun Pod del cluster richiede l'accesso a IMDS per altri motivi, come il recupero della AWS regione corrente, allora potete anche disabilitare l'accesso a IMDS per i Pod che non utilizzano la rete host in un modello di lancio. Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

- Etichette Kubernetes – (Facoltativo) È possibile scegliere di applicare etichette Kubernetes ai nodi del gruppo di nodi gestiti.
- Taint Kubernetes: (facoltativo) è possibile scegliere di applicare i taint Kubernetes ai nodi del gruppo di nodi gestiti. Nel menu Effetto sono disponibili le opzioni **`NoSchedule`**, **`NoExecute`** e **`PreferNoSchedule`**. Per ulteriori informazioni, consulta the section called "Impedire che i pod siano pianificati su nodi specifici".
- Tag: (facoltativo) è possibile scegliere di aggiungere tag al gruppo di nodi gestiti Amazon EKS. Questi tag non si propagano ad altre risorse del gruppo di nodi, ad esempio gruppi Auto Scaling o istanze. Per ulteriori informazioni, consulta the section called "Assegnazione di tag alle risorse".

7. Nella pagina Imposta configurazione di calcolo e dimensionamento, compila i parametri di conseguenza e quindi scegli Successivo.
   - Tipo di AMI: selezionare un tipo di AMI. Se stai distribuendo istanze Arm, assicurati di rivedere le considerazioni in Arm Amazon Linux AMIs ottimizzato per Amazon EKS prima della distribuzione.

     Se nella pagina precedente sono stati specificati un modello di avvio e un'AMI nel modello, non è possibile selezionare un valore. Viene visualizzato il valore del modello. L'AMI specificata nel modello deve soddisfare i requisiti indicati alla pagina Specifying an AMI.

- Tipo di capacità: seleziona un tipo di capacità. Per ulteriori informazioni sulla scelta di un tipo di capacità, consulta the section called "Tipi di capacità del gruppo di nodi gestiti". Non è possibile combinare diversi tipi di capacità all'interno dello stesso gruppo di nodi. Se desideri utilizzare entrambi i tipi di capacità, crea gruppi di nodi separati, ognuno con i propri tipi di capacità e istanza. GPUs Per informazioni sul provisioning e la scalabilità dei nodi di lavoro accelerati da GPU, consulta Reserve for managed node groups.
- Tipi di istanza: per impostazione predefinita, viene specificato uno o più tipi di istanza. Per rimuovere un tipo di istanza predefinito, seleziona il X sul lato destro del tipo di istanza. Scegli il tipo di istanza da utilizzare nel gruppo di nodi gestiti. Per ulteriori informazioni, consulta the section called "Tipi di istanza Amazon EC2".

  Nella console viene visualizzato un insieme di tipi di istanza di uso comune. Se devi creare un gruppo di nodi gestito con un tipo di istanza che non viene visualizzato`eksctl`, usa la AWS CLI o un SDK per creare il gruppo di nodi. AWS CloudFormation Se nella pagina precedente è stato specificato un modello di avvio, non è possibile selezionare un valore perché il tipo di istanza deve essere specificato nel modello di avvio. Viene visualizzato il valore del modello di avvio. Se è stato selezionato Spot per Tipo capacità, al fine di migliorare la disponibilità è consigliabile specificare più tipi di istanza.
- Dimensioni disco: inserire le dimensioni del disco (in GiB) da utilizzare per il volume root del nodo.

  Se nella pagina precedente è stato specificato un modello di avvio, non è possibile selezionare un valore perché deve essere specificato nel modello di avvio.
- Dimensione desiderata: specifica il numero corrente di nodi che il gruppo di nodi gestiti deve mantenere all'avvio.

  > **Note**
  >
  > Amazon EKS non aumenta o riduce orizzontalmente in automatico il gruppo di nodi. Tuttavia, è possibile configurare il Cluster Autoscaler di Kubernetes per eseguire questa operazione. Per ulteriori informazioni, consulta Cluster Autoscaler on. AWS
- Dimensione minima: specifica il numero minimo di nodi a cui il gruppo di nodi gestiti può essere ridotto.
- Dimensione massima: specifica il numero massimo di nodi a cui il gruppo di nodi gestiti può essere aumentato.

- Configurazione dell'aggiornamento dei gruppi di nodi: (facoltativo) puoi selezionare il numero o la percentuale di nodi da aggiornare in parallelo. Questi nodi non saranno disponibili durante l'aggiornamento. Per Numero massimo non disponibile, seleziona una delle seguenti opzioni e specifica un Valore:
  - Numero: seleziona e specifica il numero di nodi nel gruppo nodi che possono essere aggiornati in parallelo.
  - Percentuale: seleziona e specifica la percentuale di nodi nel gruppo di nodi che possono essere aggiornati in parallelo. Questa funzione è utile se disponi di un numero elevato di nodi nel gruppo di nodi.
- Configurazione di riparazione automatica del nodo: (facoltativo) se viene attivata la casella di controllo Abilita la riparazione automatica del nodo, Amazon EKS sostituirà automaticamente i nodi quando si verificano i problemi rilevati. Per ulteriori informazioni, consulta the section called "Integrità del nodo".

8. Nella pagina Specifica reti, compila i parametri opportunamente, quindi scegli Successivo.
   - Sottoreti: sceglii le sottoreti in cui avviare i nodi gestiti.

> **Important**
>
> Se si esegue un'applicazione stateful in più zone di disponibilità supportate dai volumi Amazon EBS e che utilizza il Cluster Autoscaler di Kubernetes, è necessario configurare più gruppi di nodi, ognuno dei quali definito per una singola zona di disponibilità. Inoltre, è necessario abilitare la funzionalità `--balance-similar-node-groups`.

> **Important**
>
> - Se si sceglie una sottorete pubblica e nel cluster è abilitato solo l'endpoint del server API pubblico, è necessario che la sottorete disponga di `MapPublicIPOnLaunch` impostato su `true` per consentire alle istanze di unirsi correttamente a un cluster. Se la sottorete è stata creata utilizzando `eksctl` o i AWS CloudFormation modelli forniti da Amazon EKS a partire dal 26 marzo 2020, allora questa impostazione è già impostata su. `true` Se le sottoreti sono state create con `eksctl` o i AWS CloudFormation modelli prima del 26 marzo 2020, devi modificare l'impostazione manualmente. Per ulteriori informazioni, vedere Modifica dell'attributo di IPv4 indirizzamento pubblico per la sottorete.

- Se si utilizza un modello di avvio e si specificano più interfacce di rete, Amazon EC2 non assegnerà automaticamente un indirizzo `IPv4` pubblico, anche se `MapPublicIpOnLaunch` è impostato su `true`. Affinché i nodi possano unirsi al cluster in questo scenario, è necessario abilitare l'endpoint del server API privato del cluster, oppure avviare i nodi in una sottorete privata con accesso Internet in uscita fornito attraverso un metodo alternativo, ad esempio un gateway NAT. Per ulteriori informazioni, consultare la pagina Amazon EC2 instance IP addressing della Guida per l'utente di Amazon EC2.

- Configurazione dell'accesso SSH ai nodi (facoltativo). L'abilitazione di SSH consente di connettersi alle istanze e raccogliere informazioni diagnostiche in caso di problemi. Consigliamo vivamente di abilitare l'accesso remoto quando crei un gruppo di nodi. Non è possibile abilitare l'accesso remoto dopo la creazione del gruppo di nodi.

  Se si sceglie di utilizzare un modello di avvio, questa opzione non viene visualizzata. Per abilitare l'accesso remoto ai nodi, specifica una coppia di chiavi nel modello di avvio e assicurati che la porta appropriata sia aperta ai nodi nei gruppi di sicurezza specificati nel modello di avvio. Per ulteriori informazioni, consulta the section called "Utilizzo di gruppi di sicurezza personalizzati".

  > **Note**
  >
  > Per Windows, questo comando non abilita SSH. Associa, invece, la coppia di chiavi Amazon EC2 all'istanza e ti consente di eseguire il protocollo RDP nell'istanza.

- Per Coppia di chiavi SSH (facoltativo), scegli una chiave SSH Amazon EC2 da utilizzare. Per informazioni su Linux, consultare Amazon EC2 key pairs and Linux instances nella Guida per l'utente di Amazon EC2. Per informazioni su Windows, consultare Amazon EC2 key pairs and Windows instances nella Guida per l'utente di Amazon EC2. Se si sceglie di utilizzare un modello di avvio, non è possibile selezionarne una. Quando viene fornita una chiave SSH Amazon EC2 per i gruppi di nodi che utilizzano Bottlerocket AMIs, viene abilitato anche il contenitore amministrativo. Per ulteriori informazioni, consulta Container amministratore su GitHub.

- Per Autorizzazione di accesso remoto SSH da, se desideri limitare l'accesso a istanze specifiche, seleziona i gruppi di sicurezza associati a tali istanze. Se non si selezionano gruppi di sicurezza specifici, l'accesso SSH è consentito da qualsiasi indirizzo di Internet (`0.0.0.0/0`).

9. Nella pagina Rivedi e crea, controlla la configurazione del gruppo di nodi gestiti e scegli Crea.

Se i nodi di lavoro non riescono a unirsi al cluster, consultare the section called "Impossibile aggiungere i nodi al cluster" nel capitolo dedicato alla risoluzione dei problemi.

10. Guarda lo stato dei nodi e attendi che raggiungano lo stato `Ready`.

```
kubectl get nodes --watch
```

11. (Solo nodi GPU) Se hai scelto un tipo di istanza GPU e un'AMI accelerata ottimizzata per Amazon EKS, devi applicare il plug-in del dispositivo NVIDIA per Kubernetes sul tuo cluster. DaemonSet Sostituiscilo *vX.X.X* con la versione di s-device-pluginNVIDIA/K8 desiderata prima di eseguire il comando seguente.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Installare i componenti aggiuntivi Kubernetes

Ora che si dispone di un cluster Amazon EKS funzionante con nodi, è possibile avviare l'installazione dei componenti aggiuntivi Kubernetes e l'implementazione di applicazioni al cluster. I seguenti argomenti della documentazione consentono di estendere la funzionalità del cluster.

- Il principale IAM in grado di creare il cluster è l'unico principale che può effettuare chiamate al server API Kubernetes tramite `kubectl` o la Console di gestione AWS. Se si desidera che altri principali IAM abbiano accesso al cluster, è necessario aggiungerli. Per ulteriori informazioni, consultare the section called "Accesso API Kubernetes" e the section called "Autorizzazioni richieste".
- Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
  - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
  - Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

  Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).
- Configurare il Cluster Autoscaler di Kubernetes per regolare automaticamente il numero di nodi nei gruppi di nodi.
- Implementare un'applicazione di esempio al cluster.

- Organizzare e monitorare le risorse del cluster con strumenti importanti per la gestione.

Aggiornamento del gruppo di nodi gestito per il cluster

Quando si avvia un aggiornamento di un gruppo di nodi gestiti, Amazon EKS aggiorna automaticamente i nodi, completando i passaggi elencati in Understand each phase of node updates. Se si utilizza un'AMI ottimizzata per Amazon EKS, Amazon EKS applica automaticamente le patch di sicurezza più recenti e gli aggiornamenti del sistema operativo ai nodi come parte dell'ultima versione di AMI.

Esistono diversi scenari in cui è utile aggiornare la versione o la configurazione del gruppo di nodi gestiti Amazon EKS:

- Hai aggiornato la versione di Kubernetes per il cluster Amazon EKS e vuoi aggiornare i nodi per utilizzare la stessa versione di Kubernetes.
- Una nuova versione dell'AMI è disponibile per il gruppo di nodi gestiti. Per ulteriori informazioni sulle versioni AMI, consultare le seguenti sezioni:
  - the section called "Ottenere informazioni sulla versione"
  - the section called "Bottlerocket"
  - the section called "Ottenere informazioni sulla versione"
- Per regolare il conteggio minimo, massimo o desiderato delle istanze nel gruppo di nodi gestiti.
- Per aggiungere o rimuovere le etichette Kubernetes dalle istanze del gruppo di nodi gestiti.
- Per aggiungere o rimuovere tag AWS dal gruppo di nodi gestiti.
- È necessario implementare una nuova versione di un modello di avvio con modifiche alla configurazione, ad esempio un'AMI personalizzata aggiornata.
- È stata implementata la versione `1.9.0` o successiva del componente aggiuntivo CNI di Amazon VPC, è stato abilitato il componente aggiuntivo per la delega del prefisso e nuove istanze di AWS Nitro System sono necessarie in un gruppo di nodi per supportare un numero significativamente maggiore di pod. Per ulteriori informazioni, consulta the section called "Aumentare gli indirizzi IP".
- Hai abilitato la delega del prefisso IP per i nodi di Windows e vuoi che le nuove istanze di AWS Nitro System supportino in un gruppo di nodi un numero significativamente maggiore di pod. Per ulteriori informazioni, consulta the section called "Aumentare gli indirizzi IP".

Se esiste una versione dell'AMI più recente per la versione Kubernetes del gruppo di nodi gestiti rispetto a quella in esecuzione nel gruppo di nodi, è possibile aggiornare quest'ultima per utilizzare la

nuova versione dell'AMI. Analogamente, se nel cluster è in esecuzione una versione di Kubernetes più recente rispetto al gruppo di nodi, è possibile aggiornare il gruppo di nodi per utilizzare la versione più recente dell'AMI che corrisponde alla versione di Kubernetes del cluster.

Quando un nodo in un gruppo di nodi gestiti è terminato a causa di un'operazione di dimensionamento o di un aggiornamento, i pod nel nodo sono svuotati per primi. Per ulteriori informazioni, consulta the section called "Dettagli sull'aggiornamento del comportamento".

Aggiornare la versione di un gruppo di nodi

È possibile aggiornare la versione di un gruppo di nodi con uno dei seguenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

La versione a cui si aggiorna non può essere successiva alla versione del piano di controllo.

`eksctl`

Aggiornare un gruppo di nodi gestito utilizzando **`eksctl`**

Aggiorna un gruppo di nodi gestiti alla stessa versione AMI più recente di Kubernetes attualmente implementata nei nodi con il comando seguente. Sostituisci ogni *`valore esemplificativo`* con i tuoi valori.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code
```

> **Note**
>
> Se stai aggiornando un gruppo di nodi implementato con un modello di avvio a una nuova versione del modello di avvio, aggiungi `--launch-template-version` *`version-number`* al precedente comando. Il modello di avvio deve soddisfare i requisiti descritti in Customize managed nodes with launch templates. Se il modello di avvio include un'AMI personalizzata, l'AMI deve soddisfare i requisiti in Specificare un'AMI. Quando si aggiorna il gruppo di nodi a una versione più recente del modello di avvio, tutti i nodi sono riciclati in modo da corrispondere alla nuova configurazione della versione del modello di avvio specificata.

Non è possibile aggiornare direttamente un gruppo di nodi implementato senza un modello di avvio a una nuova versione del modello di avvio. È invece necessario implementare un nuovo gruppo di nodi utilizzando il modello di avvio per aggiornare il gruppo di nodi a una nuova versione del modello di avvio.

È possibile aggiornare un gruppo di nodi alla stessa versione della versione Kubernetes del piano di controllo. Ad esempio, se hai un cluster che esegue Kubernetes `1.33`, puoi aggiornare i nodi che attualmente eseguono Kubernetes `1.32` alla versione `1.33` con il comando seguente.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code \
  --kubernetes-version=1.33
```

Console di gestione AWS

Aggiornare un gruppo di nodi gestito utilizzando Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegliere il cluster che contiene il gruppo di nodi da aggiornare.
3. Se almeno un gruppo di nodi dispone di un aggiornamento disponibile, nella parte superiore della pagina viene visualizzata una casella di notifica dell'aggiornamento disponibile. Se selezioni la scheda Calcolo, visualizzerai Aggiorna ora nella colonna Versione rilascio AMI nella tabella Gruppi di nodi per il gruppo di nodi per cui è disponibile un aggiornamento. Per aggiornare il gruppo di nodi, scegli Update now (Aggiorna ora).

   Non verrà visualizzata una notifica per i gruppi di nodi implementati con un'AMI personalizzata. Se i nodi vengono implementati con un'AMI personalizzata, completare la procedura seguente per implementare una nuova AMI personalizzata aggiornata.

   a. Crea una nuova versione dell'AMI.

   b. Creare una nuova versione del modello di avvio con il nuovo ID AMI.

   c. Aggiornamento dei i nodi alla nuova versione del modello di avvio.
4. Nella finestra di dialogo Update node group version (Aggiorna la versione del gruppo di nodi), attiva o disattiva le seguenti opzioni:

- Update node group version (Aggiorna la versione del gruppo di nodi): questa opzione non è disponibile se hai implementato un'AMI personalizzata o se l'AMI ottimizzata per Amazon EKS attualmente è disponibile nella versione più recente del cluster.
- Change launch template version (Modifica la versione del modello di avvio): questa opzione non è disponibile se il gruppo di nodi è implementato senza un modello di avvio personalizzato. È possibile aggiornare la versione del modello di avvio solo per un gruppo di nodi implementato con un modello di avvio personalizzato. Seleziona la Launch template version (Versione del modello di avvio) a cui eseguire l'aggiornamento del gruppo di nodi. Se il gruppo di nodi è configurato con un'AMI personalizzata, anche la versione selezionata deve specificare un'AMI. Quando si esegue l'aggiornamento a una versione più recente del modello di avvio, tutti i nodi vengono riciclati in modo da corrispondere alla nuova configurazione della versione del modello di avvio specificata.

5. Per Update strategy (Strategia aggiornamento), seleziona una delle seguenti opzioni:
   - Aggiornamento in sequenza: questa opzione rispetta i budget di interruzione del pod per il cluster. Gli aggiornamenti non riescono se c'è un problema di budget che interrompe il pod, che fa sì che Amazon EKS non riesca a svuotare correttamente i pod in esecuzione su questo gruppo di nodi.
   - Forza aggiornamento: questa opzione non rispetta i budget per le interruzioni dei pod. Gli aggiornamenti sono eseguiti indipendentemente dai problemi di budget di interruzione del pod forzando il riavvio dei nodi.
6. Scegli Aggiorna.

Modificare la configurazione di un gruppo di nodi

È possibile modificare parte della configurazione di un gruppo di nodi gestiti.

1. Aprire la Console Amazon EKS.
2. Scegliere il cluster che contiene il gruppo di nodi da modificare.
3. Seleziona la scheda Compute (Calcolo).
4. Seleziona il gruppo di nodi da modificare, e scegli Edit (Modifica).
5. (Facoltativo) Nella pagina Edit node group (Modifica gruppo di nodi), effettua le seguenti operazioni:
   a. Modifica la configurazione di scalabilità del gruppo di nodi.

- Dimensione desiderata: specifica il numero corrente di nodi che il gruppo di nodi gestiti deve mantenere.
- Dimensione minima: specifica il numero minimo di nodi a cui il gruppo di nodi gestiti può essere ridotto.
- Dimensione massima: specifica il numero massimo di nodi a cui il gruppo di nodi gestiti può essere aumentato orizzontalmente. Per il numero massimo di nodi supportati in un gruppo di nodi, vedere the section called "Quote di servizio".

b. (Facoltativo) Aggiungi o rimuovi etichette Kubernetes ai nodi nel gruppo di nodi. Le etichette mostrate qui sono solo le etichette applicate con Amazon EKS. Altre etichette, non mostrate qui, possono essere presenti sui nodi.

c. (Facoltativo) Aggiungi o rimuovi taint Kubernetes ai nodi nel gruppo di nodi. I taint aggiunti possono avere l'effetto di **`NoSchedule`**, **`NoExecute`**, oppure **`PreferNoSchedule`**. Per ulteriori informazioni, consulta the section called "Impedire che i pod siano pianificati su nodi specifici".

d. (Facoltativo) Aggiungi o rimuovi Tag dalla risorsa del gruppo di nodi. Questi tag vengono applicati solo al gruppo di nodi Amazon EKS. I tag dei gruppi di nodi non si propagano ad altre risorse come istanze o le sottoreti di Amazon EC2 nel gruppo di nodi.

e. (Facoltativo) Modifica la Configurazione dell'aggiornamento del gruppo di nodi. Seleziona un'opzione tra Numero o Percentuale.

- Numero: seleziona e specifica il numero di nodi nel gruppo nodi che possono essere aggiornati in parallelo. Questi nodi non saranno disponibili durante l'aggiornamento.
- Percentuale: selezionare e specificare la percentuale di nodi nel gruppo di nodi che possono essere aggiornati in parallelo. Questi nodi non saranno disponibili durante l'aggiornamento. Questa funzione è utile se si dispone di diversi nodi nel gruppo di nodi.

f. Al termine della modifica, scegli Salva modifiche.

> **Important**
>
> Quando si aggiorna la configurazione del gruppo di nodi, la modifica di `NodegroupScalingConfig` non rispetta i budget di interruzione pod (PDB). A differenza del processo di aggiornamento del gruppo di nodi (che drena i nodi e rispetta i PDB durante la fase di aggiornamento), l'aggiornamento della configurazione di scalabilità causa la chiusura immediata dei nodi tramite una chiamata di riduzione verticale del gruppo Auto Scaling (ASG). Ciò avviene senza considerare i PDB, indipendentemente dalla dimensione

di destinazione a cui stai ridimensionando verticalmente. Ciò significa che quando riduci `desiredSize` di un gruppo di nodi gestiti da Amazon EKS, i pod sono espulsi non appena i nodi sono terminati, senza onorare alcun PDB.

Comprendi ogni fase degli aggiornamenti dei nodi

La strategia di aggiornamento del nodo (worker) gestito di Amazon EKS ha quattro fasi diverse descritte nelle sezioni seguenti.

Fase di configurazione

La fase di configurazione prevede i seguenti passaggi:

1. Crea una nuova versione del modello di avvio Amazon EC2 per il gruppo Auto Scaling associato al gruppo di nodi. La nuova versione del modello di avvio utilizza l'AMI di destinazione o una versione del modello di avvio personalizzata per l'aggiornamento.
2. Il gruppo Auto Scaling viene aggiornato per utilizzare la versione più recente del modello di avvio.
3. Determina la quantità massima di nodi da aggiornare in parallelo utilizzando la proprietà `updateConfig` per il gruppo di nodi. Il massimo non disponibile ha una quota di 100 nodi. Il valore di default è un nodo. Per ulteriori informazioni, consulta la proprietà updateConfig nella Documentazione di riferimento delle API di Amazon EKS.

Fase di scalabilità

Quando si aggiornano i nodi in un gruppo di nodi gestiti, i nodi aggiornati vengono avviati nella stessa zona di disponibilità di quelli in fase di aggiornamento. Per garantire questo posizionamento, utilizziamo il ribilanciamento della zona di disponibilità di Amazon EC2. Per ulteriori informazioni, consulta Availability Zone Rebalancing nella Guida per l'utente di Amazon EC2 Auto Scaling. Per soddisfare questo requisito, è possibile avviare fino a due istanze per zona di disponibilità nel gruppo di nodi gestiti.

La fase di scalabilità prevede i seguenti passaggi:

1. Incrementa la dimensione massima e la dimensione desiderata del gruppo Auto Scaling in base alla scelta maggiore tra:
   - Fino a due volte il numero di zone di disponibilità in cui è stato implementato il gruppo Auto Scaling.

- Il massimo non disponibile per l'aggiornamento.

  Ad esempio, se il gruppo di nodi ha cinque zone di disponibilità e `maxUnavailable` è uno, il processo di aggiornamento può avviare al massimo 10 nodi. Tuttavia, quando `maxUnavailable` è 20 (o comunque superiore a 10), il processo avvia 20 nuovi nodi.

2. Dopo aver dimensionato il gruppo Auto Scaling, verifica se i nodi che utilizzano la configurazione più recente sono presenti nel gruppo di nodi. Questo passaggio ha esito positivo solo quando soddisfa i seguenti criteri:
   - Almeno un nuovo nodo viene avviato in ogni zona di disponibilità in cui esiste il nodo.
   - Ogni nuovo nodo deve essere nello stato `Ready`.
   - I nuovi nodi devono avere etichette applicate da Amazon EKS.

     Queste sono le etichette applicate da Amazon EKS sui nodi (worker) di un normale gruppo di nodi:

     - `eks.amazonaws.com/nodegroup-image=$amiName`
     - `eks.amazonaws.com/nodegroup=$nodeGroupName`

     Queste sono le etichette applicate da Amazon EKS sui nodi (worker) in un modello di avvio personalizzato o un gruppo di nodi AMI:

     - `eks.amazonaws.com/nodegroup-image=$amiName`
     - `eks.amazonaws.com/nodegroup=$nodeGroupName`
     - `eks.amazonaws.com/sourceLaunchTemplateId=$launchTemplateId`
     - `eks.amazonaws.com/sourceLaunchTemplateVersion=$launchTemplateVersion`
3. Contrassegna i nodi come non programmabili per evitare di programmare nuovi pod. Inoltre, etichetta i nodi con `node.kubernetes.io/exclude-from-external-load-balancers=true` per rimuovere i vecchi nodi dai sistemi di bilanciatore del carico prima di terminare i nodi.

Di seguito sono riportati i motivi noti che portano a un errore `NodeCreationFailure` in questa fase:

Capacità insufficiente nella zona di disponibilità

Esiste la possibilità che la zona di disponibilità non abbia capacità disponibile per i tipi di istanza richiesti. Si consiglia di configurare più tipi di istanze durante la creazione di un gruppo di nodi gestiti.

Limiti delle istanze EC2 dell'account

Potrebbe essere necessario aumentare il numero di istanze Amazon EC2 che il l'account può eseguire contemporaneamente utilizzando Service Quotas. Per ulteriori informazioni, consulta Service Quotas di EC2 nella Guida per l'utente di Amazon Elastic Compute Cloud per le istanze Linux.

Dati utente personalizzati

I dati utente personalizzati possono talvolta interrompere il processo di bootstrap. Questo scenario può portare al mancato avvio di `kubelet` sul nodo o sui nodi che non ricevono le etichette Amazon EKS previste. Per ulteriori informazioni, consulta the section called "Specifica di un'AMI".

Qualsiasi modifica che renda un nodo non integro o non pronto

La pressione del disco del nodo, la pressione della memoria e condizioni simili possono far sì che un nodo non assuma lo stato `Ready`.

Ogni nodo si avvia al massimo entro 15 minuti

Se un nodo impiega più di 15 minuti per avviarsi e unirsi al cluster, si verificherà il timeout dell'aggiornamento. Questo è il runtime totale per il bootstrap di un nuovo nodo misurato dal momento in cui è necessario un nuovo nodo al momento in cui si unisce al cluster. Quando si aggiorna un gruppo di nodi gestito, il contatore temporale si avvia non appena la dimensione del gruppo Auto Scaling aumenta.

Fase di aggiornamento

La fase di aggiornamento si comporta in due modi diversi, a seconda della strategia di aggiornamento. Esistono due strategie di aggiornamento: predefinita e minimale.

Consigliamo una strategia predefinita nella maggior parte degli scenari. Infatti, questa crea nuovi nodi prima di terminare quelli vecchi, in modo da mantenere la capacità disponibile durante la fase di aggiornamento. La strategia minimale è utile in scenari in cui si è limitati per quanto riguarda risorse o costi, ad esempio con acceleratori hardware come le GPU. Infatti, termina i vecchi nodi prima di creare quelli nuovi, in modo che la capacità totale non aumenti mai oltre la quantità configurata.

La strategia di aggiornamento predefinita prevede i seguenti passaggi:

1. Aumenta la quantità di nodi (numero desiderato) nel gruppo Auto Scaling, facendo sì che il gruppo di nodi crei nodi aggiuntivi.

2. Seleziona casualmente un nodo che deve essere aggiornato, fino al massimo non disponibile configurato per il gruppo di nodi.
3. Svuota i pod dal nodo. Se i pod non lasciano il nodo entro 15 minuti e non c'è un flag di forzatura, la fase di aggiornamento non riesce con errore `PodEvictionFailure`. Per questo scenario, è possibile applicare il flag di forzatura con la richiesta `update-nodegroup-version` di eliminare i pod.
4. Isola il nodo dopo che ogni pod è stato espulso e aspetta 60 secondi. Ciò consente al controller di servizio di non inviare nuove richieste a questo nodo, e lo rimuove dall'elenco di nodi attivi.
5. Invia una richiesta di interruzione al gruppo con scalabilità automatica per il nodo isolato.
6. Ripete i passaggi di aggiornamento precedenti fino a quando non vi sono nodi nel gruppo implementato con la versione precedente del modello di avvio.

La strategia di aggiornamento minimale prevede i seguenti passaggi:

1. All'inizio isola tutti i nodi del gruppo di nodi, in modo che il controller di servizio non invii nuove richieste a questi nodi.
2. Seleziona casualmente un nodo che deve essere aggiornato, fino al massimo non disponibile configurato per il gruppo di nodi.
3. Drena i Pod dai nodi selezionati. Se i pod non lasciano il nodo entro 15 minuti e non c'è un flag di forzatura, la fase di aggiornamento non riesce con errore `PodEvictionFailure`. Per questo scenario, puoi applicare il flag di forzatura con la richiesta `update-nodegroup-version` di eliminare i pod.
4. Dopo che ogni pod è stato rimosso e ha atteso 60 secondi, invia una richiesta di interruzione al gruppo Auto Scaling per i nodi selezionati. Il gruppo Auto Scaling crea nuovi nodi (lo stesso numero di nodi selezionati) per sostituire la capacità mancante.
5. Ripete i passaggi di aggiornamento precedenti fino a quando non vi sono nodi nel gruppo implementato con la versione precedente del modello di avvio.

Errori **`PodEvictionFailure`** durante la fase di aggiornamento

Di seguito sono riportati i motivi noti che portano a un errore `PodEvictionFailure` in questa fase:

PDB aggressivo

Sul pod è definito un PDB aggressivo oppure sono presenti più PDB che puntano allo stesso pod.

Implementazione che tollera tutti i taint

Una volta espulso ogni pod, il nodo dovrebbe essere vuoto in quanto oggetto di taint nei passaggi precedenti. Tuttavia, se l'implementazione tollera ogni taint, è più probabile che il nodo non sia vuoto, causando un errore di espulsione dal pod.

Fase di dimensionamento

La fase di dimensionamento diminuisce di uno la dimensione massima del gruppo Auto Scaling e la dimensione desiderata per tornare ai valori prima dell'avvio dell'aggiornamento.

Se il flusso di lavoro di aggiornamento determina che il Cluster Autoscaler stia dimensionando il gruppo di nodi durante la fase di dimensionamento del flusso di lavoro, esce immediatamente senza riportare il gruppo di nodi alle dimensioni originali.

Personalizzazione dei nodi gestiti con modelli di avvio

Per il massimo livello di personalizzazione, puoi distribuire nodi gestiti con il tuo modello di lancio in base ai passaggi di questa pagina. L'utilizzo di un modello di avvio consente funzionalità come fornire argomenti di bootstrap durante la distribuzione di un nodo (ad esempio, argomenti kubelet aggiuntivi), assegnare indirizzi IP ai Pod da un blocco CIDR diverso dall'indirizzo IP assegnato al nodo, distribuire la propria AMI personalizzata ai nodi o distribuire il proprio CNI personalizzato sui nodi.

Quando fornisci il tuo modello di avvio alla prima creazione di un gruppo di nodi gestito, in seguito avrai anche una maggiore flessibilità. Dopo aver implementato un gruppo di nodi gestiti con un modello di avvio personalizzato, potrai aggiornarlo in maniera iterativa con una versione diversa dello stesso modello. Quando si aggiorna il gruppo di nodi a una versione diversa del modello di avvio, tutti i nodi del gruppo vengono riciclati in modo da corrispondere alla nuova configurazione della versione del modello di avvio specificata.

I gruppi di nodi gestiti vengono sempre implementati con un modello di avvio da utilizzare con gruppo di Amazon EC2 Auto Scaling. Quando non è fornito un modello di avvio, l'API Amazon EKS ne crea automaticamente uno con i valori predefiniti nel tuo account. Tuttavia, consigliamo di non modificare i modelli di avvio generati automaticamente. Inoltre, i gruppi di nodi esistenti che non utilizzano un modello di avvio personalizzato non possono essere aggiornati direttamente. A tale scopo, sarà invece necessario creare un nuovo gruppo di nodi con un modello di avvio personalizzato.

Informazioni di base sulla configurazione del modello di avvio

Puoi creare un modello di lancio di Amazon EC2 Auto Scaling con la Console di gestione AWS CLI AWS o un SDK. AWS Per ulteriori informazioni, consulta Creazione di un modello di avvio per un gruppo con scalabilità automatica nella Guida per l'utente di Amazon EC2 Auto Scaling. Alcune impostazioni di un modello di avvio sono simili a quelle utilizzate per la configurazione dei nodi gestiti. Quando si implementa o si aggiorna un gruppo di nodi con un modello di avvio, è necessario specificare alcune impostazioni nella configurazione del gruppo di nodi o nel modello di avvio. Non specificare un'impostazione in entrambe le posizioni. Se un'impostazione è impostata dove non dovrebbe, operazioni come la creazione o l'aggiornamento di un gruppo di nodi non hanno esito positivo.

Nella tabella seguente sono elencate le impostazioni vietate in un modello di avvio. Elenca inoltre impostazioni simili, se disponibili, che sono necessarie per la configurazione del gruppo di nodi gestiti. Le impostazioni elencate sono quelle visualizzate nella console. Potrebbero avere nomi simili ma diversi nella AWS CLI e nell'SDK.

Modello di avvio – Vietato	Configurazione del gruppo di nodi Amazon EKS
Sottorete in Interfacce di rete (Aggiungi interfaccia di rete)	Sottoreti in Configurazione di rete del gruppo di nodi nella pagina Specifica reti
Profilo dell'istanza IAM in Dettagli avanzati	Ruolo IAM del nodo in Configurazione del gruppo di nodi nella pagina Configura gruppo di nodi
Comportamento di arresto e Interrompi - Iberna comportamento in Dettagli avanzati. Mantieni default Non includere nell'impostazione del modello di avvio nel modello di avvio per entrambe le impostazioni.	Nessun equivalente. Amazon EKS deve controllare il ciclo di vita dell'istanza, non il gruppo Auto Scaling.

Nella tabella seguente sono elencate le impostazioni vietate nella configurazione di un gruppo di nodi gestiti. Elenca inoltre impostazioni simili, se disponibili, richieste in un modello di avvio. Le impostazioni elencate sono quelle visualizzate nella console. Potrebbero avere nomi simili nella AWS CLI e nell'SDK.

Configurazione del gruppo di nodi Amazon EKS – Vietata	Modello di avvio

(Solo se è stata specificata un'AMI personalizzata in un modello di avvio) Tipo di AMI in Configurazione di calcolo del gruppo di nodi sulla pagina Imposta configurazione di calcolo e dimensionamento: la console mostra il messaggio Specificato nel modello di lancio e l'ID AMI specificato.
 	
Se nel modello di avvio non è stato specificato Applicazione e immagini OS (Amazon Machine Image), è possibile selezionare un'AMI nella configurazione del gruppo di nodi.
	Applicazione e immagini OS (Amazon Machine Image) in Contenuti del modello di avvio: è necessario specificare un ID se soddisfi uno dei seguenti requisiti:
	
	• É in esecuzione un'AMI personalizzata. Se si specifica un'AMI che non soddisfa i requisiti elencati in Specificare un'AMI, l'implementazione del gruppo di nodi non avrà esito positivo.
	• Necessità di fornire i dati utente per passare gli argomenti al file `bootstrap.sh` incluso nell'AMI ottimizzata per Amazon EKS. È possibile abilitare le istanze in modo che assegnino un numero significativamente più elevato di indirizzi IP ai pod, assegnare indirizzi IP ai pod da un intervallo CIDR diverso da quello dell'istanza o implementare un cluster privato senza accesso a Internet in uscita. Per ulteriori informazioni, consulta i seguenti argomenti:
	  • Assegnazione di più indirizzi IP ai nodi Amazon EKS con prefissi
	  • Implementazione dei pod in sottoreti alternative con reti personalizzate
	  • Implementazione di cluster privati con accesso limitato a Internet
	  • Specificare un AMI

Dimensioni del disco in Configurazione di calcolo del gruppo di nodi nella pagina Imposta configurazione di calcolo e dimensionamento:	Dimensioni in Archiviazione (volumi) (Aggiungi nuovo volume). È necessario specificarlo nel modello di avvio.

Configurazione del gruppo di nodi Amazon EKS – Vietata	Modello di avvio
la console mostra il messaggio Specificato nel modello di lancio.	
Coppia di chiavi SSH in Configurazione del gruppo di nodi nella pagina Specifica reti: la console mostra la chiave specificata nel modello di avvio o mostra il messaggio Non specificato nel modello di lancio.	Nome della coppia di chiavi in Coppia di chiavi (login).
Non è possibile specificare gruppi di sicurezza di origine a cui è consentito l'accesso remoto quando si utilizza un modello di avvio.	Gruppi di sicurezza in Impostazioni di rete per l'istanza o Gruppi di sicurezza in Interfacce di rete (Aggiungi interfaccia di rete), ma non entrambi. Per ulteriori informazioni, consulta the section called "Utilizzo di gruppi di sicurezza personalizzati".

Note

- Se si esegue l'implementazione di un gruppo di nodi utilizzando un modello di avvio, specificare uno o nessun Tipo di istanza in Contenuti del modello di avvio in un modello di avvio. In alternativa, è possibile specificare 0-20 tipi di istanza per Tipi di istanza nella pagina Impostare la configurazione di calcolo e dimensionamento della console. In alternativa, è possibile farlo utilizzando altri strumenti che utilizzano l'API Amazon EKS. Se specifichi un tipo di istanza in un modello di avvio e si utilizza tale modello di avvio per implementare il gruppo di nodi, non è possibile specificare alcun tipo di istanza nella console o utilizzare altri strumenti che sfruttino l'API Amazon EKS. Se non specifichi un tipo di istanza in un modello di avvio nella console o in altri strumenti che utilizzano l'API Amazon EKS, il tipo di istanza `t3.medium` sarà utilizzato. Se il gruppo di nodi utilizza il tipo di capacità Spot, è consigliabile specificare più tipi di istanza utilizzando la console. Per ulteriori informazioni, consulta the section called "Tipi di capacità del gruppo di nodi gestiti".
- Se i container che si implementano nel gruppo di nodi utilizzano Instance Metadata Service versione 2, assicurarsi di impostare il Limite hop di risposta metadati a 2 nel modello di

avvio. Per ulteriori informazioni, consulta Metadati e dati dell'utente delle istanze nella Guida per l'utente di Amazon EC2.

- I modelli di avvio non supportano la funzionalità `InstanceRequirements` che consente la selezione flessibile del tipo di istanza.

Assegnazione di tag a istanze Amazon EC2

É possibile utilizzare il parametro `TagSpecification` di un modello di avvio per specificare quali tag applicare alle istanze Amazon EC2 nel gruppo di nodi. L'entità IAM che chiama `CreateNodegroup` o `UpdateNodegroupVersion` APIs deve disporre delle autorizzazioni per `ec2:RunInstances` and `ec2:CreateTags` e i tag devono essere aggiunti al modello di lancio.

Utilizzo di gruppi di sicurezza personalizzati

É possibile utilizzare un modello di avvio per specificare i gruppi di sicurezza Amazon EC2 da applicare alle istanze del gruppo di nodi. Questo può essere nel parametro gruppi di sicurezza a livello di istanza o come parte dei parametri di configurazione dell'interfaccia di rete. Non è possibile avviare un'istanza da un modello di avvio che specifica sia il livello di istanza sia l'interfaccia di rete dei gruppi di sicurezza. Considerare le seguenti condizioni che si applicano all'utilizzo di gruppi di sicurezza personalizzati con gruppi di nodi gestiti:

- Quando si utilizza Console di gestione AWS, Amazon EKS consente solo modelli di avvio con una singola specifica di interfaccia di rete.
- Per impostazione predefinita, Amazon EKS applica il Gruppo di sicurezza del cluster alle istanze nel gruppo di nodi, per facilitare la comunicazione tra i nodi e il piano di controllo. Se specifichi gruppi di sicurezza personalizzati nel modello di avvio utilizzando entrambe le opzioni menzionate in precedenza, Amazon EKS non aggiunge il gruppo di sicurezza del cluster. Pertanto, è necessario assicurarsi che le regole in entrata e in uscita dei gruppi di sicurezza abilitino la comunicazione con l'endpoint del cluster. Se le regole del gruppo di sicurezza non sono corrette, i nodi (worker) non possono aggiungersi al cluster. Per ulteriori informazioni sulle regole del gruppo di sicurezza, consultare the section called “Requisiti relativi al gruppo di sicurezza”.
- Se è necessario l'accesso SSH alle istanze nel gruppo di nodi, includere un gruppo di sicurezza che consenta tale accesso.

Dati utente Amazon EC2

Il modello di avvio include una sezione per i dati utente personalizzati. Puoi specificare le impostazioni di configurazione per il tuo gruppo di nodi in questa sezione senza creare manualmente singole impostazioni personalizzate AMIs. Per ulteriori informazioni sulle impostazioni disponibili per Bottlerocket, consulta Using user data on. GitHub

É possibile fornire i dati utente Amazon EC2 nel tuo modello di avvio utilizzando `cloud-init` durante l'avvio delle istanze. Per ulteriori informazioni, consultare la documentazione cloud-init. I dati utente possono essere utilizzati per eseguire operazioni di configurazione comuni. Sono comprese le seguenti opzioni:

- Inclusione di utenti o gruppi.
- Installazione di pacchetti

I dati utente di Amazon EC2 nei modelli di avvio utilizzati con gruppi di nodi gestiti devono essere nel formato di archivio multiparte MIME per Amazon Linux AMIs e nel formato TOML per Bottlerocket. AMIs Questo perché i dati utente vengono uniti con i dati utente Amazon EKS necessari ai nodi per aderire al cluster. Non specificare alcun comando nei dati utente che avviano o modificano `kubelet`. Questa operazione viene eseguita come parte dei dati utente uniti da Amazon EKS. Certi parametri `kubelet`, come l'impostazione delle etichette sui nodi, possono essere configurati direttamente tramite l'API dei gruppi di nodi gestiti.

> **Note**
>
> Per ulteriori informazioni sulle personalizzazioni `kubelet` avanzate, tra cui l'avvio manuale o il passaggio a parametri di configurazione personalizzati, vedere the section called "Specifica di un'AMI". Amazon EKS non unisce i dati utente se un ID AMI viene specificato in un modello di avvio.

I seguenti dettagli forniscono ulteriori informazioni sulla sezione dei dati utente.

Dati utente di Amazon Linux 2

È possibile unire più blocchi di dati utente in un unico blocco, detto file MIME in più parti. Ad esempio, è possibile combinare un hook di avvio del cloud per la configurazione del daemon

Docker con uno script di shell dei dati utente che installa un pacchetto personalizzato. Un file MIME in più parti è composto dai seguenti elementi:

- Il tipo di contenuto e la dichiarazione di delimitazione della parte: `Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="`
- La dichiarazione della versione MIME: `MIME-Version: 1.0`
- Uno o più blocchi di dati utente, che contengono i seguenti elementi:
  - La delimitazione di apertura, che indica l'inizio di un blocco di dati utente: `--==MYBOUNDARY==`
  - La dichiarazione del tipo di contenuto per il blocco: `Content-Type: text/cloud-config; charset="us-ascii"`. Per ulteriori informazioni sui tipi di contenuto, consultare la documentazione di cloud-init.
  - Il contenuto dei dati utente, ad esempio un elenco di comandi shell o direttive `cloud-init`.
  - La delimitazione di chiusura, che indica la fine del file MIME in più parti: `--==MYBOUNDARY==--`

Di seguito è riportato un esempio di un file in più parti MIME che è possibile utilizzare per crearne uno.

```
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==--
```

Dati utente di Amazon Linux 2023

Amazon Linux 2023 (AL2023) introduce un nuovo processo di inizializzazione dei nodi `nodeadm` che utilizza uno schema di configurazione YAML. Se utilizzi gruppi di nodi autogestiti o un'AMI con un modello di avvio, ora dovrai fornire esplicitamente metadati del cluster aggiuntivi quando si crea un nuovo gruppo di nodi. Un esempio dei parametri minimi richiesti è come segue, dove ora `apiServerEndpoint`, `certificateAuthority` e il servizio `cidr` sono richiesti:

```
---
```

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

In genere questa configurazione è impostata nei dati utente, così com'è o incorporata in un documento MIME composto da più parti:

```
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--
```

Nel AL2, i metadati di questi parametri sono stati scoperti dalla chiamata `DescribeCluster` API di Amazon EKS. Nel frattempo AL2023, questo comportamento è cambiato poiché la chiamata API aggiuntiva rischia di rallentare durante l'up-up di nodi su larga scala. Questa modifica non ha effetto su di te se utilizzi gruppi di nodi gestiti senza un modello di avvio o se utilizzi Karpenter. Per ulteriori informazioni su `certificateAuthority` e sul servizio `cidr`, consulta `DescribeCluster` nel Riferimento di API Amazon EKS.

Ecco un esempio completo di dati AL2023 utente che combina uno script di shell per personalizzare il nodo (come l'installazione di pacchetti o la prememorizzazione nella cache delle immagini dei contenitori) con la configurazione richiesta. `nodeadm` Questo esempio mostra personalizzazioni comuni tra cui: * Installazione di pacchetti di sistema aggiuntivi * Pre-memorizzazione nella cache delle immagini dei container per migliorare il tempo di avvio del pod * Impostazione della configurazione del proxy HTTP * Configurazione dei flag `kubelet` per l'etichettatura dei nodi

```
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--
```

Dati utente Bottlerocket

I dati utente delle strutture Bottlerocket nel formato TOML. È possibile fornire i dati utente da unire con i dati utente forniti da Amazon EKS. Ad esempio, è possibile fornire ulteriori impostazioni `kubelet`.

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

Per ulteriori informazioni sulle impostazioni supportate, consultare la documentazione Bottlerocket. È possibile configurare etichette di nodi e taint nei dati utente. Consigliamo, tuttavia, di configurarle all'interno del gruppo di nodi. In questo caso, Amazon EKS applica queste configurazioni.

Quando i dati utente sono uniti, la formattazione non è mantenuta, ma il contenuto rimane invariato. La configurazione fornita nei dati utente sostituisce tutte le impostazioni configurate da Amazon EKS. Quindi, se imposti `settings.kubernetes.max-pods` o `settings.kubernetes.cluster-dns-ip`, questi valori nei dati utente sono applicati ai nodi.

Amazon EKS non supporta tutti i TOML validi. Di seguito è riportato un elenco di formati noti non supportati:

- Quote all'interno delle chiavi stimate: `'quoted "value"' = "value"`
- Quote evase nei valori: `str = "I'm a string. \"You can quote me\""`
- Galleggianti e numeri interi misti: `numbers = [ 0.1, 0.2, 0.5, 1, 2, 5 ]`
- Tipi misti nelle matrici: `contributors = ["foo@example.com", { name = "Baz", email = "baz@example.com" }]`
- Intestazioni tra parentesi con chiavi stimate: `[foo."bar.baz"]`

Dati utente Windows

I dati utente di Windows utilizzano i comandi. PowerShell Quando crei un gruppo di nodi gestiti, i dati utente personalizzati si combinano con i dati utente gestiti da Amazon EKS. PowerShell I tuoi comandi vengono prima, seguiti dai comandi gestiti per i dati utente, il tutto all'interno di un unico `<powershell></powershell>` tag.

> **Important**
>
> Durante la creazione di gruppi di nodi Windows, Amazon EKS aggiorna il file `aws-auth ConfigMap` per consentire ai nodi basati su Linux di unirsi al cluster. Il servizio non configura automaticamente le autorizzazioni per Windows AMIs. Se utilizzi i nodi Windows, dovrai gestire l'accesso tramite l'API della voce di accesso o aggiornando direttamente `aws-auth ConfigMap`. Per ulteriori informazioni, consulta the section called "Abilitazione del supporto Windows".

> **Note**
>
> Quando non è specificato alcun ID AMI nel modello di avvio, nei dati utente non utilizzare lo script bootstrap di Amazon EKS per Windows per configurare Amazon EKS.

Di seguito sono riportati dati utente di esempio.

```
<powershell>
Write-Host "Running custom user data script"
</powershell>
```

Specifica di un'AMI

Se si dispone di uno dei seguenti requisiti, specificare un ID AMI nel campo `ImageId` del modello di avvio. Selezionare il requisito di cui si dispone per ulteriori informazioni.

Fornisci i dati utente per passare argomenti al **`bootstrap.sh`** file incluso in un' Linux/Bottlerocket AMI ottimizzata per Amazon EKS

Il termine "bootstrapping" indica l'aggiunta di comandi che possono essere eseguiti all'avvio di un'istanza. Ad esempio, il bootstrap consente di utilizzare argomenti kubelet aggiuntivi. È possibile passare gli argomenti allo script `bootstrap.sh` utilizzando `eksctl` senza specificare un modello di avvio. Oppure è possibile farlo specificando le informazioni nella sezione dati utente di un modello di avvio.

eksctl senza specificare un modello di avvio

Crea un file denominato *my-nodegroup.yaml* con i seguenti contenuti. Sostituisci ogni *example value* con i valori in tuo possesso. Gli argomenti `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` sono facoltativi, ma grazie alla loro definizione lo script `bootstrap.sh` può evitare di effettuare una chiamata a `describeCluster`. Ciò è utile nelle configurazioni di cluster privati o nei cluster in cui si effettuano dimensionamenti frequenti dei nodi. Per ulteriori informazioni sullo `bootstrap.sh` script, consulta il file bootstrap.sh su GitHub.

- L'unico argomento richiesto è il nome del cluster (*my-cluster*).
- Per recuperare l'ID di un'AMI ottimizzata per `ami-`*1234567890abcdef0* , consulta le sezioni seguenti:
  - Recupera le AMI Amazon Linux consigliate IDs
  - Recupera l'AMI Bottlerocket consigliata IDs
  - Recupera l'AMI Microsoft Windows consigliata IDs
- Per recuperare il valore *certificate-authority* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Per recuperare il valore *api-server-endpoint* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- Il valore per `--dns-cluster-ip` è il tuo servizio CIDR con `.10` alla fine. Per recuperare il valore *service-cidr* per il cluster, esegui il comando seguente. Ad esempio, se il valore restituito è `ipv4 10.100.0.0/16`, il tuo valore è *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Questo esempio fornisce un argomento `kubelet` per impostare un valore `max-pods` personalizzato utilizzando lo script `bootstrap.sh` incluso nell'AMI ottimizzata per Amazon EKS. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura. Per assistenza nella scelta di *my-max-pods-value*, consulta the section called "Per ogni tipo di istanza Amazon EC2, Amazon EKS consiglia un numero massimo di pod". Per ulteriori informazioni su

come `maxPods` viene determinato quando si utilizzano i gruppi di nodi gestiti, vederethe section called "Come viene determinato MaxPods".

```
---
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
```

Per ogni opzione disponibile per il file `eksctl config`, consulta Schema del file config nella documentazione su `eksctl`. L'utility `eksctl` crea autonomamente un modello di avvio e popola i dati utente con i dati forniti nel file `config`.

Crea un gruppo di nodi con il comando seguente.

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

Dati utente in un modello di avvio

Specifica le seguenti informazioni nella sezione dati utente del modello di avvio. Sostituisci ogni *`example value`* con i valori in tuo possesso. Gli argomenti `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` sono facoltativi, ma grazie alla loro definizione lo script `bootstrap.sh` può evitare di effettuare una chiamata a `describeCluster`. Ciò è utile nelle

configurazioni di cluster privati o nei cluster in cui si effettuano dimensionamenti frequenti dei nodi. Per ulteriori informazioni sullo `bootstrap.sh` script, vedere il file bootstrap.sh su GitHub.

- L'unico argomento richiesto è il nome del cluster (*`my-cluster`*).
- Per recuperare il valore *`certificate-authority`* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Per recuperare il valore *`api-server-endpoint`* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- Il valore per `--dns-cluster-ip` è il tuo servizio CIDR con `.10` alla fine. Per recuperare il valore *`service-cidr`* per il cluster, esegui il comando seguente. Ad esempio, se il valore restituito è `ipv4 10.100.0.0/16`, il tuo valore è *`10.100.0.10`*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Questo esempio fornisce un argomento `kubelet` per impostare un valore `max-pods` personalizzato utilizzando lo script `bootstrap.sh` incluso nell'AMI ottimizzata per Amazon EKS. Per assistenza nella scelta di *`my-max-pods-value`*, consulta the section called "Per ogni tipo di istanza Amazon EC2, Amazon EKS consiglia un numero massimo di pod". Per ulteriori informazioni su come `maxPods` viene determinato quando si utilizzano i gruppi di nodi gestiti, vederethe section called "Come viene determinato MaxPods".

```
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==--
```

Fornire i dati dell'utente per passare gli argomenti al file **`Start-EKSBootstrap.ps1`** incluso nell'AMI Windows ottimizzata per Amazon EKS

Il termine "bootstrapping" indica l'aggiunta di comandi che possono essere eseguiti all'avvio di un'istanza. È possibile passare gli argomenti allo script `Start-EKSBootstrap.ps1` utilizzando `eksctl` senza specificare un modello di avvio. Oppure è possibile farlo specificando le informazioni nella sezione dati utente di un modello di avvio.

Per specificare un ID AMI Windows personalizzato, fai le seguenti considerazioni:

- Devi utilizzare un modello di avvio e fornire i comandi bootstrap richiesti nella sezione dei dati utente. Per recuperare l'ID Windows desiderato, puoi utilizzare la tabella in Crea nodi con Windows AMIs ottimizzato.
- Esistono vari limiti e condizioni. Ad esempio, è necessario aggiungere `eks:kube-proxy-windows` alla mappa di configurazione di AWS IAM Authenticator. Per ulteriori informazioni, consulta the section called "Limiti e condizioni quando si specifica un ID AMI".

Specifica le seguenti informazioni nella sezione dati utente del modello di avvio. Sostituisci ogni *`example value`* con i valori in tuo possesso. Gli argomenti `-APIServerEndpoint`, `-Base64ClusterCA` e `-DNSClusterIP` sono facoltativi, ma grazie alla loro definizione lo script `Start-EKSBootstrap.ps1` può evitare di effettuare una chiamata a `describeCluster`.

- L'unico argomento richiesto è il nome del cluster (*`my-cluster`*).
- Per recuperare il valore *`certificate-authority`* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- Per recuperare il valore *`api-server-endpoint`* per il cluster, esegui il comando seguente.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- Il valore per `--dns-cluster-ip` è il tuo servizio CIDR con `.10` alla fine. Per recuperare il valore *`service-cidr`* per il cluster, esegui il comando seguente. Ad esempio, se il valore restituito è `ipv4 10.100.0.0/16`, il tuo valore è *`10.100.0.10`*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- Per ulteriori argomenti, consulta.the section called "Parametri di configurazione dello script di bootstrap".

> **Note**
>
> Se utilizzi il servizio personalizzato CIDR, devi specificarlo utilizzando il parametro `-ServiceCIDR`. In caso contrario, la risoluzione DNS per i pod nel cluster avrà esito negativo.

```
<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>
```

Eseguire un'AMI personalizzata a causa di specifici requisiti di sicurezza, conformità o policy interne

Per ulteriori informazioni, consulta Amazon Machine Image (AMI) nella Guida per l'utente di Amazon EC2. La specifica di compilazione AMI Amazon EKS contiene risorse e script di configurazione per la creazione di un'AMI Amazon EKS personalizzata basata su Amazon Linux. Per ulteriori informazioni, consulta Specifica della build AMI di Amazon EKS su GitHub. Per creare AMIs installazioni personalizzate con altri sistemi operativi, consulta Amazon EKS Sample Custom AMIs on GitHub.

Non è possibile utilizzare riferimenti dinamici ai parametri per AMI IDs nei modelli di avvio utilizzati con gruppi di nodi gestiti.

Important

- Quando si specifica un'AMI, Amazon EKS non convalida la versione di Kubernetes incorporata nell'AMI rispetto alla versione del piano di controllo del cluster. Sei responsabile di garantire che la versione Kubernetes della tua AMI personalizzata sia conforme alla politica di modifica della versione di Kubernetes:
  - La `kubelet` versione sui tuoi nodi non deve essere più recente della versione del cluster
  - La `kubelet` versione sui tuoi nodi deve essere uguale o superiore a 3 versioni secondarie rispetto alla versione del cluster (per la versione Kubernetes `1.28` o successiva) o fino a 2 versioni minori rispetto alla versione del cluster (per la versione Kubernetes o precedente) `1.27`

    La creazione di gruppi di nodi gestiti con violazioni dell'asimmetria delle versioni può comportare:
  - I nodi non riescono a unirsi al cluster
  - Comportamento indefinito o incompatibilità delle API
  - Instabilità del cluster o errori del carico di lavoro
- Quando si specifica una AMI, Amazon EKS non unisce alcun dato utente. È l'utente, piuttosto, ad essere responsabile della fornitura dei comandi `bootstrap` richiesti per unire i nodi al cluster. Se i nodi non riescono a unirsi al cluster, le operazioni `CreateNodegroup` e `UpdateNodegroupVersion` di Amazon EKS non vengono eseguite con successo.

Limiti e condizioni quando si specifica un ID AMI

Di seguito sono riportati i limiti e le condizioni per la specifica di un ID AMI con gruppi di nodi gestiti:

- È necessario creare un nuovo gruppo di nodi per passare dalla specifica o meno di un ID AMI in un modello di avvio.
- Non riceverai una notifica nella console quando è disponibile una versione AMI più recente. Per aggiornare il gruppo di nodi a una versione AMI più recente, devi creare una nuova versione del modello di avvio con un ID AMI aggiornato. Quindi, è necessario aggiornare il gruppo di nodi con la nuova versione del modello di avvio.
- I seguenti campi non possono essere impostati nell'API se si specifica un ID AMI:
  - `amiType`

  - `releaseVersion`
  - `version`
- Se si specifica un ID AMI, qualsiasi set di `taints` nell'API viene applicato in modo asincrono. Per applicare i taint prima che un nodo si unisca al cluster, è necessario passare i taint a `kubelet` utilizzando il flag della riga di comando `--register-with-taints`. Per ulteriori informazioni, consulta kubelet nella documentazione Kubernetes.
- Quando specifichi un ID AMI personalizzato per i gruppi di nodi gestiti da Windows, aggiungilo `eks:kube-proxy-windows` alla mappa di configurazione di AWS IAM Authenticator. Questa API è necessaria per il funzionamento di DNS.
  a. Apri la mappa di configurazione di AWS IAM Authenticator per modificarla.

```
kubectl edit -n kube-system cm aws-auth
```

  b. Aggiungi questa voce all'elenco `groups` sotto ogni `rolearn` associato ai nodi Windows. La tua mappa di configurazione dovrebbe essere simile a aws-auth-cm-windows .yaml.

```
- eks:kube-proxy-windows
```

  c. Salva il file ed esci dall'editor di testo.
- Per ogni AMI che utilizza un modello di avvio personalizzato, l'impostazione predefinita `HttpPutResponseHopLimit` per i gruppi di nodi gestiti è impostata su 2.

Eliminare un gruppo di nodi gestito dal cluster

In questo argomento viene descritto come eliminare un gruppo di nodi gestiti Amazon EKS. Quando elimini un gruppo di nodi gestiti, Amazon EKS imposterà su zero la dimensione minima, massima e desiderata del tuo gruppo Auto Scaling. Questo fa sì che il gruppo di nodi venga dimensionato.

Prima che ogni istanza venga terminata, Amazon EKS invia un segnale per drenare quel nodo. Durante il processo di drenaggio, Kubernetes esegue le seguenti operazioni per ogni pod sul nodo: esegue tutti gli hook del `preStop` ciclo di vita configurati, invia `SIGTERM` segnali ai contenitori, quindi attende il corretto spegnimento. `terminationGracePeriodSeconds` Se il nodo non è stato svuotato dopo 5 minuti, Amazon EKS consente ad Auto Scaling di continuare la chiusura forzata dell'istanza. Dopo che tutte le istanze sono state terminate, il gruppo Auto Scaling viene eliminato.

> **Important**
>
> Se si elimina un gruppo di nodi gestiti che utilizza un ruolo IAM del nodo non impiegato da nessun altro gruppo di nodi gestiti nel cluster, il ruolo viene rimosso dalla `ConfigMap` `aws-auth`. Se i gruppi di nodi autogestiti nel cluster utilizzano lo stesso ruolo IAM del nodo, i nodi autogestiti passeranno allo stato `NotReady`. Inoltre, anche l'operazione del cluster viene interrotta. Per aggiungere una mappatura per il ruolo da utilizzare solo per i gruppi di nodi autogestiti, consultare la pagina the section called "Creare voci di accesso", se la versione della piattaforma del cluster corrisponde almeno alla versione minima elencata nella sezione dei prerequisiti della pagina Grant IAM users access to Kubernetes with EKS access entries. Se la versione della piattaforma è precedente alla versione minima richiesta per le voci di accesso, è possibile aggiungere nuovamente la voce alla `ConfigMap` `aws-auth`. Per ulteriori informazioni, digita `eksctl create iamidentitymapping --help` nel terminale.

È possibile creare un gruppo di nodi gestiti con i seguenti strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

`eksctl`

Eliminare un gruppo di nodi gestiti con **`eksctl`**

Inserire il seguente comando. Sostituisci ogni `<example value>` con i valori in tuo possesso.

```
eksctl delete nodegroup \
  --cluster <my-cluster> \
  --name <my-mng> \
  --region <region-code>
```

Per ulteriori opzioni, consulta Eliminazione e svuotamento dei gruppi di nodi nella documentazione di `eksctl`.

Console di gestione AWS

Eliminare un gruppo di nodi gestiti con Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nella pagina Cluster, scegli il cluster che contiene il gruppo di nodi da eliminare.
3. Nella pagina del cluster selezionata, scegli la scheda Calcolo.
4. Nella sezione Gruppi di nodi, scegliere il gruppo di nodi da eliminare. Scegli Elimina.
5. Nella finestra di dialogo di conferma di Eliminazione del gruppo di nodi, inserisci il nome del gruppo di nodi. Scegli Elimina.

AWS CLI

Eliminare un gruppo di nodi gestito con AWS CLI

1. Inserire il seguente comando. Sostituisci ogni `<example value>` con i valori in tuo possesso.

```
aws eks delete-nodegroup \
  --cluster-name <my-cluster> \
  --nodegroup-name <my-mng> \
  --region <region-code>
```

2. Se `cli_pager=` è impostato nella configurazione CLI, usa i tasti freccia sulla tastiera per scorrere l'output della risposta. Premere il tasto q al termine.

   Per ulteriori opzioni, consultate il `delete-nodegroup` comando nella AWS CLI Command Reference.

Gestione autonoma dei nodi con nodi autogestiti

Un cluster contiene uno o più nodi Amazon EC2 su cui sono pianificati i pod. I nodi Amazon EKS vengono eseguiti nell'account AWS e si connettono al piano di controllo del cluster tramite l'endpoint del server API del cluster. La loro fatturazione è basata sui prezzi Amazon EC2. Per ulteriori informazioni, consulta Prezzi di Amazon EC2.

Un cluster può contenere diversi gruppi di nodi. Ciascun gruppo di nodi contiene uno o più nodi implementati in un gruppo Amazon EC2 Auto Scaling. Il tipo di istanza dei nodi all'interno del gruppo

può variare, ad esempio quando si utilizza la selezione del tipo di istanza basata sugli attributi con Karpenter. Tutte le istanze di un gruppo di nodi devono utilizzare il ruolo IAM del nodo Amazon EKS.

Amazon EKS fornisce Amazon Machine Image (AMI) specializzate chiamate AMI ottimizzate per Amazon EKS. Le AMI sono configurate per funzionare con Amazon EKS. I loro componenti includono `containerd`, `kubelet` e l'AWS IAM Authenticator. L'AMI contiene anche uno script di bootstrap specializzato che consente di individuare e connettersi automaticamente al piano di controllo del cluster.

Se si limita l'accesso all'endpoint pubblico del cluster utilizzando blocchi CIDR, è consigliabile abilitare anche l'accesso agli endpoint privati. In questo modo i nodi possono comunicare con il cluster. Senza l'endpoint privato abilitato, i blocchi CIDR specificati per l'accesso pubblico devono includere le origini di uscita dal VPC. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

Per aggiungere nodi autogestiti al cluster Amazon EKS, consultare gli argomenti che seguono. Se si avviano manualmente i nodi autogestiti, è necessario aggiungere il seguente tag a ciascun nodo, assicurandosi al contempo che `<cluster-name>` corrisponda al cluster. Per ulteriori informazioni, consultare Aggiunta ed eliminazione di tag in una singola risorsa. Se si seguono i passaggi della guida, il tag richiesto viene aggiunto al nodo per conto dell'utente.

Chiave	Valore
`kubernetes.io/cluster/<cluster-name>`	`owned`

Important

I tag nel servizio di metadati di istanza (IMDS) di Amazon EC2 non sono compatibili con i nodi EKS. Quando i tag dei metadati delle istanze sono abilitati, è impedito l'uso di barre ("/") nei valori dei tag. Tale limitazione può causare errori nell'avvio delle istanze, in particolare quando si utilizzano strumenti di gestione dei nodi come Karpenter o un dispositivo di dimensionamento automatico dei cluster, poiché questi servizi si basano solo su tag contenenti barre per una corretta funzionalità.

Per ulteriori informazioni sui nodi da un prospettiva Kubernetes generale, consultare Nodi nella documentazione Kubernetes.

Argomenti

- Crea nodi Amazon Linux autogestiti
- Crea nodi autogestiti Bottlerocket
- Crea nodi Microsoft Windows autogestiti
- Creazione di nodi Ubuntu Linux autogestiti
- Aggiornamento dei nodi autogestiti per il tuo cluster

Crea nodi Amazon Linux autogestiti

In questo argomento viene descritto come avviare gruppi con scalabilità automatica di nodi Linux che si registrano con il cluster Amazon EKS. Dopo che i nodi vengono aggiunti al cluster, puoi implementare applicazioni Kubernetes per gli stessi. Puoi anche avviare nodi Amazon Linux autogestiti con `eksctl` o. Console di gestione AWS Se devi avviare nodi su AWS Outposts, consulta. the section called "Nodi"

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta the section called "Creazione di un cluster ". Se hai delle sottoreti nella AWS regione in cui hai abilitato AWS Outposts, AWS Wavelength o AWS Local Zones, tali sottoreti non devono essere state passate al momento della creazione del cluster.
- Un ruolo IAM esistente per i nodi da utilizzare. Per crearne uno, consulta the section called "Ruolo IAM del nodo". Se questo ruolo non prevede nessuna delle policy per VPC CNI, per i pod VPC CNI è necessario il ruolo separato riportato di seguito.
- (Facoltativo, ma consigliato) Il componente aggiuntivo plug-in CNI di Amazon VPC per Kubernetes configurato con il proprio ruolo IAM a cui è allegata la policy IAM necessaria. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
- Conoscenza delle considerazioni elencate in Scegli un tipo di istanza di EC2 nodo Amazon ottimale. A seconda del tipo di istanza scelto, potrebbero esserci ulteriori prerequisiti per il cluster e il VPC.

È possibile avviare nodi Linux autogestiti con uno dei seguenti modi:

- the section called "`eksctl`"

- the section called "Console di gestione AWS"

`eksctl`

Avvia nodi Linux autogestiti utilizzando **`eksctl`**

1. Installa la versione `0.215.0` o successiva dello strumento da riga di `eksctl` comando installato sul tuo dispositivo o. AWS CloudShell Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. (Facoltativo) Se la policy IAM gestita AmazonEKS_CNI_Policy è collegata al ruolo IAM del nodo Amazon EKS, ti consigliamo, invece, di assegnarla a un ruolo IAM associato all'account di servizio Kubernetes `aws-node`. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
3. Il comando seguente crea un gruppo di nodi in un cluster esistente. Sostituisci *`al-nodes`* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura. Sostituisci *`my-cluster`* con il nome del cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster. Sostituisci i *`example value`* rimanenti con i valori in tuo possesso. Per impostazione predefinita, i nodi vengono creati con la stessa versione Kubernetes del piano di controllo.

   Prima di scegliere un valore per`--node-type`, consulta Scegli un tipo di istanza Amazon EC2 node ottimale.

   Sostituiscilo *`my-key`* con il nome della tua coppia di EC2 chiavi Amazon o della tua chiave pubblica. Questa chiave viene utilizzata per eseguire il SSH nei nodi dopo il loro avvio. Se non disponi già di una coppia di EC2 chiavi Amazon, puoi crearne una in Console di gestione AWS. Per ulteriori informazioni, consulta le coppie di EC2 chiavi Amazon nella Amazon EC2 User Guide.

   Crea il tuo gruppo di nodi con il comando seguente.

   > **Important**
   >
   > Se desideri distribuire un gruppo di nodi nelle sottoreti AWS Outposts, Wavelength o Local Zone, ci sono altre considerazioni:

- Le sottoreti non devono essere state trasmesse al momento della creazione del cluster.
- È necessario creare il gruppo di nodi con un file di configurazione, specificando le sottoreti e `volumeType: gp2`. Per ulteriori informazioni, consulta Creazione di un gruppo di nodi da un file di configurazione e lo Schema del file config nella documentazione di `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
```

Per implementare un gruppo di nodi che:

- può assegnare un numero significativamente più elevato di indirizzi IP a pod rispetto alla configurazione predefinita, consulta the section called "Aumentare gli indirizzi IP".
- può assegnare gli indirizzi `IPv4` a pod da un altro intervallo CIDR rispetto a quello dell'istanza, consulta the section called "Rete personalizzata".
- può assegnare gli indirizzi `IPv6` a pod e servizi, consulta the section called "IPv6".
- non dispone di accesso a Internet in uscita, consulta the section called "Cluster privati".

  Per un elenco completo di tutte le opzioni e i valori predefiniti disponibili, immetti il comando seguente.

  ```
  eksctl create nodegroup --help
  ```

  Se i nodi di lavoro non riescono a unirsi al cluster, consulta the section called "Impossibile aggiungere i nodi al cluster" nel capitolo sulla risoluzione dei problemi.

  Di seguito viene riportato un output di esempio. Durante la creazione dei nodi vengono generate diverse righe. Una delle ultime righe di output è simile alla seguente riga di esempio.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

4. (Facoltativo) implementare un'applicazione di esempio per testare il cluster e i nodi Linux.
5. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati delle EC2 istanze Amazon (IMDS) per altri motivi, come il recupero della regione corrente. AWS

   Per ulteriori informazioni, consulta Limitazione dell'accesso al profilo dell'istanza assegnato al nodo worker.

Console di gestione AWS

Passaggio 1: avvia i nodi Linux autogestiti utilizzando la Console di gestione AWS

1. Scarica l'ultima versione del modello. AWS CloudFormation

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2025-11-26/
amazon-eks-nodegroup.yaml
```

2. Attendi che lo stato del cluster risulti `ACTIVE`. Se vengono avviati prima che il cluster sia attivo, i nodi non riescono a effettuare la registrazione al cluster e sarà necessario riavviarli.
3. Apri la AWS CloudFormation console.
4. Scegli Crea stack e quindi seleziona Con nuove risorse (standard).
5. In Specifica modello, seleziona Carica un file di modello e Scegli file.
6. Seleziona il file `amazon-eks-nodegroup.yaml` scaricato.
7. Seleziona Successivo.
8. Nella pagina Specifica i dettagli dello stack, immetti i parametri seguenti e scegli Successivo:
   - Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, è possibile chiamarlo *`my-cluster-nodes`*. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.

- ClusterName: inserisci il nome che hai usato per creare il cluster Amazon EKS. Questo nome deve corrispondere a quello del cluster o i nodi non verranno aggiunti al cluster.
- ClusterControlPlaneSecurityGroup: scegli il SecurityGroupsvalore dall' AWS CloudFormation output che hai generato quando hai creato il tuo VPC.

  Nella procedura seguente viene illustrata un'operazione per recuperare il gruppo applicabile.

  a. Aprire la Console Amazon EKS.

  b. Scegli il nome del cluster.

  c. Scegli la scheda Reti.

  d. Utilizza il valore dei gruppi di sicurezza aggiuntivi come riferimento quando selezioni dall'ClusterControlPlaneSecurityGroupelenco a discesa.
- ApiServerEndpoint: Inserite l'API Server Endpoint per il vostro cluster EKS. È possibile trovarlo nella sezione Dettagli della console EKS Cluster
- CertificateAuthorityData: Immettete i dati dell'Autorità di certificazione codificati in base 64, disponibili anche nella sezione Dettagli della console EKS Cluster.
- ServiceCidr: Inserisci l'intervallo CIDR utilizzato per allocare gli indirizzi IP ai servizi Kubernetes all'interno del cluster. Si trova nella scheda di rete della console del cluster EKS.
- AuthenticationMode: Seleziona la modalità di autenticazione in uso nel cluster EKS esaminando la scheda di accesso all'interno della console del cluster EKS.
- NodeGroupName: Inserisci un nome per il tuo gruppo di nodi. Questo nome può essere utilizzato in seguito per identificare il gruppo di nodi con dimensionamento automatico creato per i tuoi nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura.
- NodeAutoScalingGroupMinSize: Inserisci il numero minimo di nodi su cui il gruppo Auto Scaling del nodo può scalare.
- NodeAutoScalingGroupDesiredCapacity: Inserisci il numero di nodi desiderato su cui scalare quando viene creato lo stack.
- NodeAutoScalingGroupMaxSize: Inserisci il numero massimo di nodi su cui il gruppo Auto Scaling del nodo può scalare orizzontalmente.
- NodeInstanceType: scegli un tipo di istanza per i tuoi nodi. Per ulteriori informazioni, consulta the section called "Tipi di istanza Amazon EC2".
- NodeImageIdSSMParam: precompilato con il parametro Amazon EC2 Systems Manager di una recente AMI Amazon Linux 2023 ottimizzata per Amazon EKS per una versione variabile di Kubernetes. Se si desidera utilizzare una diversa versione secondaria di Kubernetes supportata

da Amazon EKS, è possibile sostituire *1.XX* con una versione supportata differente. Si consiglia di specificare la stessa versione Kubernetes del cluster.

Puoi anche sostituirlo *amazon-linux-2023* con un altro tipo di AMI. Per ulteriori informazioni, consulta the section called "Ottieni le ultime novità IDs".

> **Note**
>
> I nodi Amazon EKS AMIs sono basati su Amazon Linux. Puoi tenere traccia degli eventi di sicurezza o privacy per Amazon Linux 2023 nell'Amazon Linux Security Center o abbonarti al feed RSS associato. Gli eventi di sicurezza e privacy includono una panoramica del problema, quali sono i pacchetti interessati e come aggiornare le istanze per risolvere il problema.

- NodeImageId: (Facoltativo) Se utilizzi un'AMI personalizzata (anziché un'AMI ottimizzata per Amazon EKS), inserisci un ID AMI del nodo per la tua AWS regione. Se specifichi un valore qui, questo sostituisce tutti i valori nel NodeImageIdSSMParamcampo.
- NodeVolumeSize: Specificate la dimensione del volume root per i nodi, in GiB.
- NodeVolumeType: Specificate un tipo di volume root per i nodi.
- KeyName: inserisci il nome di una coppia di chiavi Amazon EC2 SSH che puoi usare per connetterti tramite SSH ai tuoi nodi dopo il loro avvio. Se non disponi già di una coppia di EC2 chiavi Amazon, puoi crearne una in Console di gestione AWS. Per ulteriori informazioni, consulta le coppie di EC2 chiavi Amazon nella Amazon EC2 User Guide.
- VpcId: inserisci l'ID per il VPC che hai creato.
- Sottoreti: scegli le sottoreti create per il VPC. Se hai creato il VPC utilizzando i passaggi descritti in Create an Amazon VPC for your Amazon EKS cluster, specifica solo le sottoreti private all'interno del VPC per avviare i nodi. È possibile visualizzare le sottoreti private aprendo ogni collegamento relativo alla sottorete dalla scheda Reti del cluster.

> **Important**
>
> - Se una qualsiasi delle sottoreti è pubblica, devi abilitare l'impostazione di assegnazione automatica degli indirizzi IP pubblici. Se l'impostazione non è abilitata per la sottorete pubblica, a tutti i nodi distribuiti in quella sottorete pubblica non verrà assegnato un indirizzo IP pubblico e non saranno in grado di comunicare con il cluster o altri servizi. AWS Se la sottorete è stata distribuita prima del 26 marzo 2020 utilizzando uno dei modelli AWS CloudFormation VPC di Amazon EKS o

utilizzando`eksctl`, l'assegnazione automatica degli indirizzi IP pubblici è disabilitata per le sottoreti pubbliche. Per informazioni su come abilitare l'assegnazione di indirizzi IP pubblici per una sottorete, consulta Modifica dell'attributo di indirizzamento pubblico per la sottorete. IPv4 Se il nodo viene distribuito in una sottorete privata, è in grado di comunicare con il cluster e altri AWS servizi tramite un gateway NAT.

- Se le sottoreti non hanno accesso a Internet, leggi con attenzione le considerazioni e i passaggi aggiuntivi descritti in Deploy private clusters with limited internet access.
- Se si AWS selezionano le sottoreti Outposts, Wavelength o Local Zone, le sottoreti non devono essere state passate al momento della creazione del cluster.

9. Seleziona le opzioni desiderate nella pagina Configura opzioni dello stack, quindi scegli Next (Avanti).
10. Seleziona la casella di controllo a sinistra di Riconosco che potrebbe creare risorse IAM. AWS CloudFormation , quindi scegli Create stack.
11. Al termine della creazione dello stack, selezionalo nella console e scegli Output. Se utilizzi le `EKS API` o le modalità di `EKS API and ConfigMap` autenticazione, questo è l'ultimo passaggio.
12. Se stai usando la modalità di `ConfigMap` autenticazione, registra il NodeInstanceRoleper il gruppo di nodi che è stato creato.

Fase 2: abilitazione dell'aggiunta di nodi al cluster

> **Note**
>
> I due passaggi seguenti sono necessari solo se si utilizza la modalità di autenticazione Configmap all'interno del cluster EKS. Inoltre, se hai avviato nodi all'interno di un VPC privato senza accesso a Internet in uscita, assicurati di consentire ai nodi di unirsi al cluster dall'interno del VPC.

1. Verifica se disponi già di una `ConfigMap` per `aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se ti viene mostrata una `ConfigMap` per `aws-auth`, aggiornala se necessario.

   a. Apri `ConfigMap` per la modifica.

```
kubectl edit -n kube-system configmap/aws-auth
```

b. Aggiungi una nuova voce `mapRoles`, se necessario. Impostate il `rolearn` valore sul NodeInstanceRolevalore registrato nella procedura precedente.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

c. Salva il file ed esci dall'editor di testo.

3. Se hai ricevuto un messaggio di errore che indica "`Error from server (NotFound): configmaps "aws-auth" not found`", applica lo `ConfigMap` di stock.

a. Scarica la mappa di configurazione.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

b. Nel `aws-auth-cm.yaml` file, impostate il `rolearn` NodeInstanceRolevalore sul valore registrato nella procedura precedente. Per eseguire questa operazione, utilizza un editor di testo o sostituisci *`my-node-instance-role`* eseguendo il comando seguente:

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
role|' aws-auth-cm.yaml
```

c. Applica la configurazione. L'esecuzione di questo comando potrebbe richiedere alcuni minuti.

```
kubectl apply -f aws-auth-cm.yaml
```

4. Guarda lo stato dei nodi e attendi che raggiungano lo stato `Ready`.

```
kubectl get nodes --watch
```

Inserisci `Ctrl+C` per tornare a un prompt dello shell (interprete di comandi).

> **Note**
>
> Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso negato o non autorizzato (`kubectl`)" nell'argomento relativo alla risoluzione dei problemi.

Se i nodi di lavoro non riescono a unirsi al cluster, consulta the section called "Impossibile aggiungere i nodi al cluster" nel capitolo sulla risoluzione dei problemi.

5. (Solo nodi GPU) Se hai scelto un tipo di istanza GPU e l'AMI accelerata ottimizzata per Amazon EKS, devi applicare il plug-in del dispositivo NVIDIA per Kubernetes sul tuo cluster. DaemonSet Sostituiscilo *`vX.X.X`* con la versione s-device-pluginNVIDIA/K8 desiderata prima di eseguire il comando seguente.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Fase 3: Azioni aggiuntive

1. (Facoltativo) implementare un'applicazione di esempio per testare il cluster e i nodi Linux.
2. (Facoltativo) Se la policy IAM gestita da Amazoneks_CNI_Policy (se hai `IPv4` un cluster) o la (che hai creato tu stesso se hai un `IPv6` cluster) è collegata al ruolo IAM *`AmazonEKS_CNI_IPv6_Policy`* del nodo Amazon EKS, ti consigliamo invece di assegnarlo a un ruolo IAM da associare all'account del servizio Kubernetes. `aws-node` Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
3. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati delle EC2 istanze Amazon (IMDS) per altri motivi, come il recupero della regione corrente. AWS

   Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Crea nodi autogestiti Bottlerocket

> **Note**
>
> I gruppi di nodi gestiti potrebbero offrire alcuni vantaggi per il tuo caso d'uso. Per ulteriori informazioni, consulta the section called "Gruppi di nodi gestiti".

Questo argomento descrive l'avvio di gruppi Auto Scaling di nodi Bottlerocket che si registrano con il cluster Amazon EKS. Bottlerocket è un sistema operativo open source basato su Linux AWS che è possibile utilizzare per eseguire contenitori su macchine virtuali o host bare metal. Dopo che i nodi vengono aggiunti al cluster, puoi implementare applicazioni Kubernetes per gli stessi. Per ulteriori informazioni su Bottlerocket, consulta Utilizzo di un'AMI Bottlerocket con Amazon EKS attivo GitHub e supporto AMI personalizzato nella documentazione. `eksctl`

Per informazioni sugli aggiornamenti in loco, consulta Bottlerocket Update Operator su. GitHub

> **Important**
>
> - I nodi Amazon EKS sono EC2 istanze Amazon standard e vengono fatturati in base ai normali prezzi delle EC2 istanze Amazon. Per ulteriori informazioni, consulta i EC2 prezzi di Amazon.
> - Puoi avviare nodi Bottlerocket nei cluster estesi di Amazon EKS su AWS Outposts, ma non puoi avviarli in cluster locali su Outposts. AWS Per ulteriori informazioni, consulta Amazon EKS su AWS Outposts.
> - Puoi eseguire la distribuzione su EC2 istanze Amazon con i nostri `x86` processori Arm. Tuttavia, non puoi eseguire l'implementazione su istanze con chip Inferentia.
> - Bottlerocket è compatibile con. AWS CloudFormation Tuttavia, non esiste un CloudFormation modello ufficiale che possa essere copiato per distribuire nodi Bottlerocket per Amazon EKS.
> - Le immagini Bottlerocket non vengono fornite con un server SSH o una shell. Puoi utilizzare metodi di out-of-band accesso per consentire a SSH di abilitare il contenitore di amministrazione e per eseguire alcuni passaggi di configurazione di bootstrap con i dati utente. Per ulteriori informazioni, consulta le seguenti sezioni nel README.md di Bottlerocket su GitHub:
>   - Esplorazione

- Container amministratore
- Impostazioni Kubernetes

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni su come installare o aggiornare `eksctl`, consulta Installazione nella documentazione `eksctl`. Nota: questa procedura funziona solo per i cluster creati con `eksctl`.

1. Copia i seguenti contenuti sul dispositivo. Sostituisci *`my-cluster`* con il nome del cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster. Sostituisci *`ng-bottlerocket`* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura. Per implementare su istanze Arm, sostituire *`m5.large`* con un tipo di istanza Arm. Sostituisci *`my-ec2-keypair-name`* con il nome di una coppia di chiavi Amazon EC2 SSH che puoi usare per connetterti tramite SSH ai nodi dopo il loro avvio. Se non disponi già di una coppia di EC2 chiavi Amazon, puoi crearne una in Console di gestione AWS. Per ulteriori informazioni, consulta le coppie di EC2 chiavi Amazon nella Amazon EC2 User Guide. Sostituisci tutti i valori esemplificativi rimanenti con i valori in tuo possesso. Dopo aver effettuato le sostituzioni, esegui il comando modificato per creare il file `bottlerocket.yaml`.

   Se specifichi un tipo di EC2 istanza Amazon Arm, esamina le considerazioni in Arm Amazon Linux ottimizzato per Amazon EKS AMIs prima della distribuzione. Per istruzioni su come implementare utilizzando un'AMI personalizzata, consulta Building Bottlerocket on e GitHub Custom AMI support nella documentazione. `eksctl` Per implementare un gruppo di nodi gestito, implementare un'AMI personalizzata utilizzando un modello di avvio. Per ulteriori informazioni, consulta the section called "Modelli di avvio".

> **Important**
>
> Per distribuire un gruppo di nodi nelle sottoreti AWS Outposts, AWS Wavelength AWS o Local Zone, non passare sottoreti AWS Outposts, AWS Wavelength o Local Zone quando crei il cluster. AWS È necessario specificare le sottoreti nell'esempio seguente. Per ulteriori informazioni, vedere Creare un gruppo di nodi da un file di configurazione e lo Schema del file config nella documentazione su `eksctl`. *`region-code`*Sostituiscilo con la regione in cui si trova il cluster. AWS

```
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. Implementare i nodi mediante il comando seguente.

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

Di seguito viene riportato un output di esempio:

Durante la creazione dei nodi vengono generate diverse righe. Una delle ultime righe di output è simile alla seguente riga di esempio.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Facoltativo) Creare un volume persistente Kubernetes su un nodo Bottlerocket usando il plug-in CSI di Amazon EBS. Il driver Amazon EBS di default si basa su strumenti di file system che non sono inclusi in Bottlerocket. Per ulteriori informazioni sulla creazione di una classe di archiviazione utilizzando il driver, consultare the section called "Amazon EBS".
4. (Facoltativo) Per impostazione predefinita `kube-proxy` imposta il parametro kernel `nf_conntrack_max` su un valore predefinito che può differire da quello che Bottlerocket imposta originariamente all'avvio. Per mantenere l'impostazione di default di Bottlerocket, modifica la configurazione `kube-proxy` con il comando seguente.

   ```
   kubectl edit -n kube-system daemonset kube-proxy
   ```

   Aggiungi `--conntrack-max-per-core` e `--conntrack-min` agli argomenti `kube-proxy` nell'esempio seguente. Un'impostazione di `0` non implica alcun cambiamento.

   ```
         containers:
         - command:
           - kube-proxy
           - --v=2
           - --config=/var/lib/kube-proxy-config/config
           - --conntrack-max-per-core=0
           - --conntrack-min=0
   ```
5. (Facoltativo) implementare un'applicazione di esempio per testare i nodi Bottlerocket.
6. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati delle EC2 istanze Amazon (IMDS) per altri motivi, come il recupero della regione corrente. AWS

Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Crea nodi Microsoft Windows autogestiti

Questo argomento illustra l'avvio di gruppi con scalabilità automatica di nodi Windows che si registrano con il cluster Amazon EKS. Dopo che i nodi vengono aggiunti al cluster, puoi implementare applicazioni Kubernetes per gli stessi.

> **Important**
>
> - I nodi di lavoro Amazon EKS sono istanze Amazon EC2 standard e la loro fatturazione è basata sui normali prezzi dell'istanza Amazon EC2. Per ulteriori informazioni, consulta Prezzi di Amazon EC2.
> - Puoi avviare nodi Windows nei cluster estesi di Amazon EKS su AWS Outposts, ma non puoi avviarli in cluster AWS locali su Outposts. Per ulteriori informazioni, consulta Amazon EKS su AWS Outposts.

Abilitare il supporto Windows per il cluster. Si consiglia di rivedere le considerazioni importanti prima di avviare un gruppo di nodi Windows. Per ulteriori informazioni, consulta the section called "Abilitazione del supporto Windows".

Puoi avviare nodi Windows autogestiti con uno dei seguenti modi:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

`eksctl`

Avvio dei nodi Windows autogestiti tramite **`eksctl`**

Questa procedura presuppone che `eksctl` sia installato e che la versione `eksctl` sia almeno `0.215.0`. È possibile verificare la tua versione con il seguente comando.

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

> **Note**
>
> Questa procedura funziona solo per i cluster creati con `eksctl`.

1. (Facoltativo) Se la policy IAM gestita da Amazoneks_CNI_Policy (se hai `IPv4` un cluster) o la (che hai creato tu stesso se hai un `IPv6` cluster) è collegata al ruolo IAM *`AmazonEKS_CNI_IPv6_Policy`* del nodo Amazon EKS, ti consigliamo invece di assegnarlo a un ruolo IAM da associare all'account del servizio Kubernetes. `aws-node` Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
2. In questa procedura si presuppone la disponibilità di un cluster. Se non disponi ancora di un cluster Amazon EKS e di un gruppo di nodi Amazon Linux a cui aggiungere un gruppo di nodi Windows, ti consigliamo di seguire the section called "Crea cluster (eksctl)". Questa guida fornisce una spiegazione dettagliata completa per la creazione di un cluster Amazon EKS con nodi Amazon Linux.

   Crea il tuo gruppo di nodi con il comando seguente. *`region-code`*Sostituiscilo con la regione in cui si trova il cluster. AWS Sostituisci *`my-cluster`* con il nome del cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster. Sostituisci *`ng-windows`* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura. Puoi sostituirlo *`2019`* con `2022` per usare Windows Server 2022 o `2025` usare Windows Server 2025. Sostituisci il resto dei valori di esempio con quelli in tuo possesso.

   > **Important**
   >
   > Per distribuire un gruppo di nodi nelle sottoreti AWS Outposts, AWS Wavelength AWS o Local Zone, non passare le sottoreti AWS Outposts, Wavelength o Local Zone quando crei il cluster. Crea il gruppo di nodi con un file di configurazione, specificando le sottoreti AWS Outposts, Wavelength o Local Zone. Per ulteriori informazioni, vedere

Creazione di un gruppo di nodi da un file di configurazione e lo Schema del file config nella documentazione su `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 i nodi non riescono a unirsi al cluster, consultare the section called "Impossibile aggiungere i nodi al cluster" nella Guida alla risoluzione dei problemi.
- Per vedere le opzioni disponibili per i comandi `eksctl`, inserire il comando seguente.

  ```
  eksctl command -help
  ```

Di seguito viene riportato un output di esempio: Durante la creazione dei nodi vengono generate diverse righe. Una delle ultime righe di output è simile alla seguente riga di esempio.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Opzionale) implementare un'applicazione di esempio per testare il cluster e i nodi Windows.
4. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Console di gestione AWS

Prerequisiti

- Creazione di un cluster Amazon EKS e di un gruppo di nodi Linux. Se non disponi di tali risorse, ti consigliamo di crearle utilizzando una delle nostre guide in Nozioni di base. Queste guide descrivono come creare un cluster Amazon EKS con nodi Linux.
- Creazione di un VPC e di un gruppo di sicurezza esistenti che soddisfano i requisiti per un cluster Amazon EKS. Per ulteriori informazioni, consulta the section called "Requisiti del VPC e delle sottoreti" e the section called "Requisiti relativi al gruppo di sicurezza". Le guide in Nozioni di base consentono di creare un VPC che soddisfa i requisiti. In alternativa, è possibile anche seguire Create an Amazon VPC for your Amazon EKS cluster crearne uno nuovo manualmente.
- Un cluster Amazon EKS esistente che utilizza un VPC e un gruppo di sicurezza in grado di soddisfare i requisiti di un cluster Amazon EKS. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ". Se sono presenti sottoreti nella AWS regione in cui sono abilitati AWS Outposts, AWS Wavelength o AWS Local Zones, tali sottoreti non devono essere state passate al momento della creazione del cluster.

Passaggio 1: avvia i nodi Windows autogestiti utilizzando la Console di gestione AWS

1. Attendi che lo stato del cluster risulti `ACTIVE`. Se i nodi vengono avviati prima che il cluster sia attivo, la registrazione al cluster non riesce e sarà necessario riavviarli.
2. Apri la console AWS CloudFormation
3. Seleziona Crea stack.
4. Per Specifica modello, selezionare URL Amazon S3.
5. Copia il seguente URL e incollalo nell'URL di Amazon S3.

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. Seleziona due volte Next (Avanti).

7. Nella pagina Quick create stack (Creazione rapida pila), inserire i seguenti parametri di conseguenza:
   - Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, è possibile chiamarlo `my-cluster-nodes`.
   - ClusterName: inserisci il nome che hai usato per creare il cluster Amazon EKS.

     > **Important**
     >
     > Questo nome deve corrispondere esattamente a quello presente in Step 1: Create your Amazon EKS cluster. In caso contrario, i nodi non possono aggiungersi al cluster.

   - ClusterControlPlaneSecurityGroup: scegli il gruppo di sicurezza dall' AWS CloudFormation output che hai generato quando hai creato il tuo VPC. Nella procedura seguente viene illustrato un metodo per recuperare il gruppo applicabile.
     a. Aprire la Console Amazon EKS.
     b. Scegli il nome del cluster.
     c. Scegli la scheda Reti.
     d. Utilizza il valore dei gruppi di sicurezza aggiuntivi come riferimento quando selezioni dall'ClusterControlPlaneSecurityGroupelenco a discesa.
   - NodeGroupName: inserisci un nome per il tuo gruppo di nodi. Questo nome può essere utilizzato in seguito per identificare il gruppo di nodi con dimensionamento automatico creato per i tuoi nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura.
   - NodeAutoScalingGroupMinSize: Inserisci il numero minimo di nodi su cui il gruppo Auto Scaling del nodo può scalare.
   - NodeAutoScalingGroupDesiredCapacity: Inserisci il numero di nodi desiderato su cui scalare quando viene creato lo stack.
   - NodeAutoScalingGroupMaxSize: Inserisci il numero massimo di nodi su cui il gruppo Auto Scaling del nodo può scalare orizzontalmente.
   - NodeInstanceType: scegli un tipo di istanza per i tuoi nodi. Per ulteriori informazioni, consulta the section called “Tipi di istanza Amazon EC2”.

> **Note**
>
> I tipi di istanza supportati per l'ultima versione del plug-in Amazon VPC CNI per Kubernetes sono elencati in vpc_ip_resource_limit.go on. GitHub Potrebbe essere necessario aggiornare la versione CNI per utilizzare i tipi di istanze supportati più recenti. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC".

- NodeImageIdSSMParam: precompilato con il parametro Amazon EC2 Systems Manager dell'attuale ID AMI Windows Core ottimizzato per Amazon EKS. Per utilizzare la versione completa di Windows, sostituire `Core` con `Full`.
- NodeImageId: (Facoltativo) Se utilizzi un'AMI personalizzata (anziché un'AMI ottimizzata per Amazon EKS), inserisci un ID AMI del nodo per la tua AWS regione. Se specifichi un valore per questo campo, questo sostituisce tutti i valori presenti nel NodeImageIdSSMParamcampo.
- NodeVolumeSize: Specificate la dimensione del volume root per i nodi, in GiB.
- KeyName: inserisci il nome di una coppia di chiavi SSH Amazon EC2 che puoi usare per connetterti tramite SSH ai tuoi nodi dopo il loro avvio. Se non hai già una coppia di chiavi Amazon EC2, puoi crearla nella Console di gestione AWS. Per ulteriori informazioni, consulta la sezione relativa alle coppie di chiavi Amazon EC2 nella Guida per l'utente di Amazon EC2.

> **Note**
>
> Se non fornisci una key pair qui, lo AWS CloudFormation stack non viene creato.

- BootstrapArguments: Specificate eventuali argomenti opzionali da passare allo script di bootstrap del nodo, come ad esempio l'utilizzo di `kubelet` argomenti aggiuntivi. `-KubeletExtraArgs`
- Disabilita IMDSv1: per impostazione predefinita, ogni nodo supporta Instance Metadata Service versione 1 (IMDSv1) e. IMDSv2 È possibile disabilitare IMDSv1. Per impedire l'utilizzo di nodi e Pod futuri nel gruppo di nodi MDSv1, imposta Disable su IMDSv1 true. Per ulteriori informazioni su IMDS, consulta Configurazione del servizio di metadati dell'istanza.
- VpcId: Seleziona l'ID per il VPC che hai creato.
- NodeSecurityGroups: Seleziona il gruppo di sicurezza che è stato creato per il tuo gruppo di nodi Linux quando hai creato il tuo VPC. Se ai nodi Linux è collegato più di un gruppo di sicurezza, specificarli tutti. Questo, ad esempio, se il gruppo di nodi Linux è stato creato con`eksctl`.

- Sottoreti: scegliere le sottoreti create. Se hai creato il VPC utilizzando i passaggi descritti in Create an Amazon VPC for your Amazon EKS cluster, specifica solo le sottoreti private all'interno del VPC per avviare i nodi.

> **Important**
>
> - Se una qualsiasi delle sottoreti è pubblica, devi abilitare l'impostazione di assegnazione automatica degli indirizzi IP pubblici. Se l'impostazione non è abilitata per la sottorete pubblica, a tutti i nodi distribuiti in quella sottorete pubblica non verrà assegnato un indirizzo IP pubblico e non saranno in grado di comunicare con il cluster o altri servizi. AWS Se la sottorete è stata distribuita prima del 26 marzo 2020 utilizzando uno dei modelli AWS CloudFormation VPC di Amazon EKS o utilizzando`eksctl`, l'assegnazione automatica degli indirizzi IP pubblici è disabilitata per le sottoreti pubbliche. Per informazioni su come abilitare l'assegnazione di indirizzi IP pubblici per una sottorete, consulta Modifica dell'attributo di indirizzamento pubblico per la sottorete. IPv4 Se il nodo viene distribuito in una sottorete privata, è in grado di comunicare con il cluster e altri AWS servizi tramite un gateway NAT.
> - Se le sottoreti non hanno accesso a Internet, leggi con attenzione le considerazioni e i passaggi aggiuntivi descritti in Deploy private clusters with limited internet access.
> - Se si AWS selezionano le sottoreti Outposts, Wavelength o Local Zone, le sottoreti non devono essere state passate al momento della creazione del cluster.

8. Confermare che la pila sia in grado di creare risorse IAM, quindi scegliere Crea pila.
9. Al termine della creazione della pila, selezionala nella console e scegli Outputs (Uscite).
10. Registra il per il gruppo di nodi che è stato NodeInstanceRolecreato. Questo servirà durante la configurazione dei nodi Windows Amazon EKS.

Passaggio 2: abilita l'aggiunta di nodi al cluster

1. Verifica se disponi già di una `ConfigMap` per `aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se ti viene mostrata una `ConfigMap` per `aws-auth`, aggiornala se necessario.
   a. Apri `ConfigMap` per la modifica.

```
kubectl edit -n kube-system configmap/aws-auth
```

b. Aggiungi nuove voci `mapRoles`, se necessario. Imposta i `rolearn` valori NodeInstanceRolesui valori che hai registrato nelle procedure precedenti.

```
[...]
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. Salva il file ed esci dall'editor di testo.

3. Se hai ricevuto un messaggio di errore che indica "`Error from server (NotFound): configmaps "aws-auth" not found`", applica lo `ConfigMap` di stock.

a. Scarica la mappa di configurazione.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm-windows.yaml
```

b. Nel `aws-auth-cm-windows.yaml` file, impostate i `rolearn` valori NodeInstanceRolesui valori applicabili registrati nelle procedure precedenti. Per eseguire questa operazione, utilizza un editor di testo o sostituisci i valori di esempio tramite il comando seguente:

```
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**
>
> - Non modificare altre righe in questo file.
> - Non utilizzare lo stesso ruolo IAM sia per i nodi Windows che per quelli Linux.

c. Applica la configurazione. L'esecuzione di questo comando potrebbe richiedere alcuni minuti.

```
kubectl apply -f aws-auth-cm-windows.yaml
```

4. Guarda lo stato dei nodi e attendi che raggiungano lo stato `Ready`.

```
kubectl get nodes --watch
```

Inserisci `Ctrl+C` per tornare a un prompt dello shell (interprete di comandi).

> **Note**
>
> Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso negato o non autorizzato (`kubectl`)" nell'argomento relativo alla risoluzione dei problemi.

Se i nodi di lavoro non riescono a unirsi al cluster, consulta the section called "Impossibile aggiungere i nodi al cluster" nel capitolo sulla risoluzione dei problemi.

Fase 3: Azioni aggiuntive

1. (Opzionale) implementare un'applicazione di esempio per testare il cluster e i nodi Windows.
2. (Facoltativo) Se la policy IAM gestita da Amazoneks_CNI_Policy (se hai `IPv4` un cluster) o la (che hai creato tu stesso se hai un `IPv6` cluster) è collegata al ruolo IAM *`AmazonEKS_CNI_IPv6_Policy`* del nodo Amazon EKS, ti consigliamo invece di assegnarlo a un ruolo IAM da associare all'account del servizio Kubernetes. `aws-node` Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
3. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.

- Nessun pod nel cluster richiede l'accesso al servizio di metadati dell'istanza Amazon EC2 (IMDS) per altri motivi, come il recupero della regione corrente. AWS

Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Creazione di nodi Ubuntu Linux autogestiti

> **Note**
>
> I gruppi di nodi gestiti potrebbero offrire alcuni vantaggi per il tuo caso d'uso. Per ulteriori informazioni, consulta the section called "Gruppi di nodi gestiti".

Questo argomento illustra l'avvio di gruppi Auto Scaling di Ubuntu su nodi Amazon Elastic Kubernetes Service (EKS) o Ubuntu Pro su nodi Amazon Elastic Kubernetes Service (EKS) che si registrano con il cluster Amazon EKS. Ubuntu e Ubuntu Pro for EKS si basano sull'Ubuntu Minimal LTS ufficiale, includono il AWS kernel personalizzato sviluppato congiuntamente e sono stati creati appositamente per EKS. AWS Ubuntu Pro aggiunge una copertura di sicurezza aggiuntiva supportando i periodi di supporto estesi di EKS, kernel livepatch, la conformità FIPS e la capacità di eseguire un numero illimitato di container Pro.

Dopo che i nodi sono aggiunti al cluster, è possibile implementare le applicazioni containerizzate per gli stessi. Per ulteriori informazioni, consulta la documentazione per Ubuntu su AWS e il supporto per le AMI personalizzate nella documentazione `eksctl`.

> **Important**
>
> - I nodi Amazon EKS sono EC2 istanze Amazon standard e vengono fatturati in base ai normali prezzi delle EC2 istanze Amazon. Per ulteriori informazioni, consulta i EC2 prezzi di Amazon.
> - Puoi avviare nodi Ubuntu nei cluster estesi di Amazon EKS su AWS Outposts, ma non puoi avviarli in cluster AWS locali su Outposts. Per ulteriori informazioni, consulta Amazon EKS su AWS Outposts.

- Puoi eseguire la distribuzione su EC2 istanze Amazon con i nostri x86 processori Arm. Tuttavia, le istanze che dispongono di chip Inferentia potrebbero dover installare prima l'SDK Neuron.

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni su come installare o aggiornare `eksctl`, consulta Installazione nella documentazione `eksctl`. Nota: questa procedura funziona solo per i cluster creati con `eksctl`.

1. Copia i seguenti contenuti sul dispositivo. Sostituisci `my-cluster` con il nome del cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfabetico e non può avere una lunghezza superiore a 100 caratteri. Sostituisci `ng-ubuntu` con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura. Per implementare su istanze Arm, sostituire `m5.large` con un tipo di istanza Arm. Sostituisci `my-ec2-keypair-name` con il nome di una coppia di chiavi Amazon EC2 SSH che puoi usare per connetterti tramite SSH ai nodi dopo il loro avvio. Se non disponi già di una coppia di EC2 chiavi Amazon, puoi crearne una in Console di gestione AWS. Per ulteriori informazioni, consulta le coppie di EC2 chiavi Amazon nella Amazon EC2 User Guide. Sostituisci tutti i valori esemplificativi rimanenti con i valori in tuo possesso. Dopo aver effettuato le sostituzioni, esegui il comando modificato per creare il file `ubuntu.yaml`.

   > **Important**
   >
   > Per distribuire un gruppo di nodi nelle sottoreti AWS Outposts, AWS Wavelength AWS o Local Zone, non passare sottoreti AWS Outposts, AWS Wavelength o Local Zone quando crei il cluster. AWS È necessario specificare le sottoreti nell'esempio seguente. Per ulteriori informazioni, vedere Creare un gruppo di nodi da un file di configurazione e lo Schema del file config nella documentazione su `eksctl`. *`region-code`*Sostituiscilo con la regione in cui si trova il cluster. AWS

```
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
```

Per creare un gruppo di nodi Ubuntu Pro, basta cambiare il valore `amiFamily` in `UbuntuPro2204`.

2. Implementare i nodi mediante il comando seguente.

```
eksctl create nodegroup --config-file=ubuntu.yaml
```

Di seguito viene riportato un output di esempio:

Durante la creazione dei nodi vengono generate diverse righe. Una delle ultime righe di output è simile alla seguente riga di esempio.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Facoltativo) Implementa un'applicazione esemplificativa per testare i nodi Ubuntu.
4. Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:
   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati delle EC2 istanze Amazon (IMDS) per altri motivi, come il recupero della regione corrente. AWS

   Per ulteriori informazioni, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).

Aggiornamento dei nodi autogestiti per il tuo cluster

Quando viene rilasciata una nuova AMI ottimizzata per Amazon EKS, considerare la sostituzione dei nodi nel gruppo di nodi autogestiti con la nuova AMI. Analogamente, se hai aggiornato la versione Kubernetes per il tuo cluster Amazon EKS, aggiornare i nodi per l'utilizzo di nodi con la stessa versione Kubernetes.

> **Important**
>
> In questo argomento vengono descritti gli aggiornamenti dei nodi per i gruppi di nodi autogestiti. Se utilizzi gruppi di nodi gestiti, consulta the section called "Aggiornamento".

Esistono due modi di base per aggiornare i gruppi di nodi autogestiti nei cluster per utilizzare una nuova AMI:

Migrate applications to a new node group

: crea un nuovo gruppo di nodi ed esegui la migrazione dei pod verso questo gruppo. La migrazione a un nuovo gruppo di nodi è più aggraziata del semplice aggiornamento dell'ID AMI in una pila AWS CloudFormation esistente. Questo perché il processo di migrazione esclude il vecchio gruppo di nodi come `NoSchedule` e svuota i nodi dopo che una nuova pila è pronta ad accettare il carico di lavoro del pod esistente.

Aggiornamento di una pila di nodi AWS CloudFormation

Aggiorna la pila AWS CloudFormation per un gruppo di nodi esistente in modo che utilizzi la nuova AMI. Questo metodo non è supportato per i gruppi di nodi creati con `eksctl`.

Esegui la migrazione delle applicazioni a un nuovo gruppo di nodi

Questo argomento consente di creare un nuovo gruppo di nodi, di migrare correttamente le applicazioni esistenti al nuovo gruppo e di rimuovere il precedente gruppo di nodi dal cluster. È possibile eseguire la migrazione a un nuovo gruppo di nodi utilizzando `eksctl` o la Console di gestione AWS.

- the section called "`eksctl`"
- the section called "Console di gestione AWS e AWS CLI"

`eksctl`

Esegui la migrazione delle applicazioni a un nuovo gruppo di nodi con **`eksctl`**

Per ulteriori informazioni sull'utilizzo di eksctl per la migrazione, consulta Unmanaged nodegroups nella documentazione di `eksctl`.

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

> **Note**
>
> Questa procedura funziona solo per i cluster e per i gruppi di nodi creati con `eksctl`.

1. Recuperare il nome dei gruppi di nodi esistenti, sostituendo *`my-cluster`* con il nome del cluster.

   ```
   eksctl get nodegroups --cluster=my-cluster
   ```

   Di seguito viene riportato un output di esempio:

   ```
   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. Avvia un nuovo gruppo di nodi con `eksctl` con il comando seguente. Nel comando, sostituire ogni *`example value`* con i propri valori. Il numero di versione non può essere successivo alla versione Kubernetes per il piano di controllo. Inoltre, non può essere più di due versioni secondarie precedenti rispetto a quella Kubernetes per il piano di controllo. Si consiglia di utilizzare la stessa versione del piano di controllo.

   Consigliamo di bloccare l'accesso dei pod a IMDS se si verificano le seguenti condizioni:

   - Prevedi di assegnare ruoli IAM a tutti gli account del servizio Kubernetes in modo che i pod dispongano solo delle autorizzazioni minime necessarie.
   - Nessun pod nel cluster richiede l'accesso al servizio di metadati delle EC2 istanze Amazon (IMDS) per altri motivi, come il recupero della regione corrente. AWS

     Per ulteriori informazioni, consulta Limitazione dell'accesso al profilo dell'istanza assegnato al nodo worker.

     Per bloccare l'accesso del pod a IMDS, aggiungere l'opzione `--disable-pod-imds` al seguente comando.

     > **Note**
     >
     > Per ulteriori flag disponibili e relative descrizioni, vedere 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 il comando precedente viene completato, verificare che tutti i nodi abbiano raggiunto lo stato `Ready` con il comando seguente:

```
kubectl get nodes
```

4. Eliminare il gruppo di nodi originale con il comando seguente. Nel comando, sostituire ogni *example value* con i nomi dei cluster e dei gruppi di nodi:

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes-old
```

Console di gestione AWS e AWS CLI

Migra le tue applicazioni in un nuovo gruppo di nodi con Console di gestione AWS e AWS CLI

1. Avvia un nuovo gruppo di nodi seguendo i passaggi descritti in Create self-managed Amazon Linux nodes.
2. Al termine della creazione dello stack, selezionalo nella console e scegli Output.
3. Registra il NodeInstanceRoleper il gruppo di nodi che è stato creato. Ciò è necessario per aggiungere i nuovi nodi Amazon EKS al cluster.

> **Note**
>
> Se si dispone di policy IAM aggiuntive associate al vecchio ruolo IAM del gruppo di nodi, associa le stesse policy al nuovo ruolo IAM del gruppo di nodi per mantenere tale funzionalità nel nuovo gruppo. Questo vale se hai aggiunto le autorizzazioni per il Cluster Autoscaler Kubernetes, ad esempio.

4. Aggiorna i gruppi di sicurezza per entrambi i gruppi di nodi in modo che possano comunicare tra loro. Per ulteriori informazioni, consulta the section called "Requisiti relativi al gruppo di sicurezza".
   a. Registra il gruppo di sicurezza IDs per entrambi i gruppi di nodi. Questo viene mostrato come NodeSecurityGroupvalore negli output dello AWS CloudFormation stack.

      È possibile utilizzare i seguenti comandi AWS CLI per ottenere il gruppo di sicurezza IDs dai nomi dello stack. In questi comandi, `oldNodes` c'è il nome AWS CloudFormation dello stack del vecchio stack di nodi ed `newNodes` è il nome dello stack verso cui state migrando. Sostituisci ogni *example value* con i valori in tuo possesso.

```
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. Aggiungi regole in ingresso per ciascun gruppo di sicurezza del nodo in modo da accettare il traffico tra loro.

   I seguenti comandi AWS CLI aggiungono regole in entrata a ciascun gruppo di sicurezza che consentono tutto il traffico su tutti i protocolli dell'altro gruppo di sicurezza. Questo consente ai pod in ogni gruppo di nodi di comunicare tra loro durante la migrazione del carico di lavoro verso il nuovo gruppo.

```
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. Modifica la configmap `aws-auth` per mappare il nuovo ruolo dell'istanza del nodo in RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Aggiungi una nuova voce `mapRoles` per il nuovo gruppo di nodi.

```
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
```

Sostituisci lo *ARN of instance role (not instance profile)* snippet con il NodeInstanceRolevalore registrato nel passaggio precedente. Quindi, salva e chiudi il file per applicare la configmap aggiornata.

6. Guarda lo stato dei nodi e attendi fino a quando i nuovi nodi si uniscono al cluster e raggiungono lo stato `Ready`.

```
kubectl get nodes --watch
```

7. (Facoltativo) Se stai utilizzando il Cluster Autoscaler Kubernetes, dimensiona l'implementazione fino a zero (0) repliche per evitare azioni di dimensionamento conflittuali.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. Utilizzare il comando seguente per il taint di ciascuno dei nodi che si desidera rimuovere con `NoSchedule`. In questo modo i nuovi pod non vengono pianificati o riprogrammati sui nodi che stai sostituendo. Per ulteriori informazioni, consulta Taints and Tolerations nella documentazione di Kubernetes.

```
kubectl taint nodes node_name key=value:NoSchedule
```

Se aggiorni i nodi a una nuova versione di Kubernetes, puoi identificare ed eseguire il taint di tutti i nodi di una determinata versione di Kubernetes (in questo caso, `1.31`) con il seguente frammento di codice. Il numero di versione non può essere successivo alla versione Kubernetes del piano di controllo. Inoltre, non può essere più di due versioni secondarie precedenti rispetto alla versione Kubernetes del piano di controllo. Si consiglia di utilizzare la stessa versione del piano di controllo.

```
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. Determina il provider DNS del cluster.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

Di seguito viene riportato un output di esempio. Questo cluster utilizza CoreDNS per la risoluzione DNS, ma il cluster può invece restituire `kube-dns`):

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

10. Se l'implementazione corrente è in esecuzione per un numero di volte inferiore a 2 repliche, scalare la distribuzione a 2 repliche. Sostituire *`coredns`* con `kubedns` se l'output del comando precedente ha avuto tale risultato.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11. Svuotare ciascuno dei nodi che si desidera rimuovere dal cluster con il comando seguente:

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

Se stai aggiornando i tuoi nodi a una nuova versione di Kubernetes, identifica e svuota tutti i nodi di una particolare versione di Kubernetes (in questo caso) con il seguente frammento di codice. *`1.31`*

```
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. Una volta terminata tale operazione, revocare le regole in ingresso del gruppo di sicurezza autorizzate in precedenza. Quindi, elimina lo stack per terminare le istanze. AWS CloudFormation

> **Note**
>
> Se hai collegato politiche IAM aggiuntive al tuo vecchio ruolo IAM del gruppo di nodi, ad esempio aggiungendo autorizzazioni per Kubernetes Cluster Autoscaler, scollega quelle politiche aggiuntive dal ruolo prima di poter eliminare lo stack. AWS CloudFormation

a. Revocare le regole in ingresso create in precedenza per i gruppi di sicurezza dei nodi. In questi comandi, `oldNodes` c'è il nome dello AWS CloudFormation stack del tuo vecchio stack di nodi ed `newNodes` è il nome dello stack verso cui stai 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. Apri la AWS CloudFormation console.

c. Selezionare la pila del nodo precedente.

d. Scegli Elimina.

e. Nella finestra di dialogo di conferma Delete stack (Elimina stack) scegliere Delete stack.(Elimina stack).

13 Modifica il configmap `aws-auth` per rimuovere il precedente ruolo dell'istanza del nodo da RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Eliminare la voce `mapRoles` per il precedente gruppo di nodo.

```
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>
```

Salvare e chiudere il file per applicare la configmap aggiornata.

14. (Facoltativo) Se si sta utilizzando il Cluster Autoscaler Kubernetes, dimensionare l'implementazione a 1 replica.

> **Note**
>
> È inoltre necessario applicare i tag al nuovo gruppo Auto Scaling in modo appropriato (ad esempio, `k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/my-cluster`) e aggiornare il comando di implementazione di Cluster Autoscaler per puntare al nuovo gruppo Auto Scaling contrassegnato. Per ulteriori informazioni, consulta Cluster Autoscaler on. AWS

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15. (Facoltativo) Verifica se stai utilizzando la versione più recente del plug-in CNI di Amazon VPC per Kubernetes. Potrebbe essere necessario aggiornare la versione CNI per utilizzare i tipi di istanze supportati più recenti. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC".

16. Se il cluster utilizza `kube-dns` per la risoluzione DNS (vedi [migrate-determine-dns-step]), riduci orizzontalmente l'implementazione `kube-dns` a 1 replica.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

Aggiornare uno AWS CloudFormation stack di nodi

Questo argomento descrive come aggiornare uno stack di nodi AWS CloudFormation autogestito esistente con una nuova AMI. È possibile utilizzare questa procedura per aggiornare i nodi a una nuova versione di Kubernetes in seguito all'aggiornamento di un cluster. In caso contrario, è

possibile eseguire l'aggiornamento all'AMI ottimizzata per Amazon EKS più recente per una versione Kubernetes esistente.

> **Important**
>
> In questo argomento vengono descritti gli aggiornamenti dei nodi per i gruppi di nodi autogestiti. Per informazioni sull'utilizzo di Simplify node lifecycle con gruppi di nodi gestiti, consulta the section called "Aggiornamento".

L'ultimo AWS CloudFormation modello di nodo Amazon EKS predefinito è configurato per avviare un'istanza con la nuova AMI nel cluster prima di rimuoverne una vecchia, una alla volta. Questa configurazione garantisce sempre il conteggio desiderato del gruppo Auto Scaling delle istanze attive nel cluster durante l'aggiornamento in sequenza.

> **Note**
>
> Questo metodo non è supportato per i gruppi di nodi creati con `eksctl`. Se è stato creato il cluster o il gruppo di nodi con `eksctl`, consultare the section called "Migrazione".

1. Determinare il provider DNS del cluster.

   ```
   kubectl get deployments -l k8s-app=kube-dns -n kube-system
   ```

   Di seguito viene riportato un output di esempio. Questo cluster utilizza CoreDNS per la risoluzione DNS, ma il cluster può invece restituire `kube-dns`. L'output potrebbe avere un aspetto diverso a seconda della versione di `kubectl` che stai utilizzando.

   ```
   NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
   coredns   1         1         1            1           31m
   ```

2. Se l'implementazione corrente è in esecuzione per un numero di volte inferiore a 2 repliche, scalare la distribuzione a 2 repliche. Sostituire *`coredns`* con `kube-dns` se l'output del comando precedente ha avuto tale risultato.

   ```
   kubectl scale deployments/coredns --replicas=2 -n kube-system
   ```

3. (Facoltativo) Se si sta utilizzando il Cluster Autoscaler Kubernetes, dimensiona l'implementazione fino a zero (0) repliche per evitare azioni di dimensionamento conflittuali.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

4. Determina il tipo di istanza e il numero di istanze desiderato del gruppo di nodi corrente. Questi valori vengono inseriti in un secondo momento, quando si aggiorna il AWS CloudFormation modello per il gruppo.
   a. Apri la EC2 console Amazon all'indirizzo https://console.aws.amazon.com/ec2/.
   b. Nel pannello di navigazione a sinistra, scegli Launch Configurations (Configurazioni di avvio) e prendi nota del tipo di istanza per la configurazione di avvio del nodo esistente.
   c. Nel pannello di navigazione a sinistra, scegli Auto Scaling Groups (Gruppi Auto Scaling) e prendi nota del conteggio delle istanze Desired (Desiderato) per il gruppo Auto Scaling del nodo.
5. Apri la AWS CloudFormation console.
6. Selezionare la pila del gruppo di nodi, quindi scegliere Aggiorna.
7. Selezionare Replace current template (Sostituisci modello corrente) e scegliere Amazon S3 URL (URL Amazon S3).
8. Per l'URL di Amazon S3, incolla il seguente URL nell'area di testo per assicurarti di utilizzare la versione più recente del modello di nodo AWS CloudFormation . Quindi scegliere Next (Successivo):

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

9. Nella pagina Specify stack details (Specifica dettagli pila), compilare i parametri seguenti e scegliere Next (Successivo):
   - NodeAutoScalingGroupDesiredCapacity— Inserisci il numero di istanze desiderato che hai registrato in un passaggio precedente. In alternativa, inserire il nuovo numero desiderato di nodi come riferimento del dimensionamento quando viene aggiornata la pila.
   - NodeAutoScalingGroupMaxSize— Inserisci il numero massimo di nodi a cui il gruppo Auto Scaling del nodo può scalare orizzontalmente. Questo valore deve essere almeno un nodo in più rispetto alla capacità desiderata. Ciò consente di eseguire un aggiornamento in sequenza dei nodi senza ridurre il numero di nodi durante l'aggiornamento.
   - NodeInstanceType— Scegli il tipo di istanza che hai registrato nel passaggio precedente. In alternativa, scegliere un tipo di istanza diverso per i nodi. Prima di scegliere un tipo di istanza

diverso, consulta Scegli un tipo di istanza Amazon EC2 node ottimale. Ogni tipo di EC2 istanza Amazon supporta un numero massimo di interfacce di rete elastiche (interfaccia di rete) e ogni interfaccia di rete supporta un numero massimo di indirizzi IP. Poiché a ogni nodo di lavoro e Pod viene assegnato il proprio indirizzo IP, è importante scegliere un tipo di istanza che supporti il numero massimo di Pod che desideri eseguire su ciascun EC2 nodo Amazon. Per un elenco del numero di interfacce di rete e di indirizzi IP supportati dai tipi di istanza, consulta Indirizzi IP per interfaccia di rete e per tipo di istanza. Ad esempio, il tipo di istanza `m5.large` supporta un massimo di 30 indirizzi IP per il nodo (worker) e per i pod.

> **Note**
>
> I tipi di istanza supportati per la versione più recente del plug-in CNI di Amazon VPC per Kubernetes sono riportati in vpc_ip_resource_limit.go su GitHub. Potrebbe essere necessario aggiornare la versione del plugin CNI di Amazon VPC per Kubernetes per utilizzare i tipi di istanze supportati più recenti. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC".

> **Important**
>
> Alcuni tipi di istanze potrebbero non essere disponibili in tutte le AWS regioni.

- NodeImageIdSSMParam— Il parametro Amazon EC2 Systems Manager dell'ID AMI a cui desideri eseguire l'aggiornamento. Il valore seguente utilizza l'AMI ottimizzata per Amazon EKS più recente per Kubernetes versione `1.33`.

```
/aws/service/eks/optimized-ami/1.33/amazon-linux-2/recommended/image_id
```

Puoi sostituirlo *`1.33`* con una versione della piattaforma che sia la stessa. In alternativa, può essere fino a una versione precedente alla versione di Kubernetes in esecuzione sul piano di controllo. Si consiglia di mantenere i nodi alla stessa versione del piano di controllo. Puoi anche sostituirlo *`amazon-linux-2`* con un altro tipo di AMI. Per ulteriori informazioni, consulta the section called "Ottieni le ultime novità IDs".

> **Note**
>
> L'utilizzo del parametro Amazon EC2 Systems Manager consente di aggiornare i nodi in futuro senza dover cercare e specificare un ID AMI. Se lo AWS CloudFormation stack utilizza questo valore, qualsiasi aggiornamento dello stack avvia sempre l'ultima AMI ottimizzata Amazon EKS consigliata per la versione di Kubernetes specificata. Questo è il caso anche se non si modificano valori nel modello.

- NodeImageId— Per utilizzare la tua AMI personalizzata, inserisci l'ID dell'AMI da utilizzare.

> **Important**
>
> Questo valore sostituisce qualsiasi valore specificato per. NodeImageIdSSMParam Se desideri utilizzare il NodeImageIdSSMParamvalore, assicurati che il valore per NodeImageIdsia vuoto.

- Disabilita IMDSv1: per impostazione predefinita, ogni nodo supporta Instance Metadata Service versione 1 (IMDSv1) e IMDSv2. Tuttavia, è possibile disabilitare IMDSv1. Seleziona true se non desideri utilizzare IMDSv1 alcun nodo o pod programmato nel gruppo di nodi. Per ulteriori informazioni su IMDS, consulta Configurazione del servizio di metadati dell'istanza. Se hai implementato i ruoli IAM per gli account di servizio, assegna le autorizzazioni necessarie direttamente a tutti i Pod che richiedono l'accesso ai servizi. AWS In questo modo, nessun Pod del cluster richiede l'accesso a IMDS per altri motivi, come il recupero della regione corrente. AWS Quindi, puoi anche disabilitare l'accesso ai Pod che non utilizzano IMDSv2 la rete host. Per ulteriori informazioni, consulta Limita l'accesso al profilo di istanza assegnato al nodo (worker).

10. (Facoltativo) Nella pagina Options (Opzioni), contrassegna con dei tag le risorse della pila. Scegli Next (Successivo).
11. Nella pagina Verifica, esaminare le informazioni, confermare che la pila è in grado di creare risorse IAM, quindi scegliere Aggiorna pila.

> **Note**
>
> L'aggiornamento di ogni nodo nel cluster richiede diversi minuti. Attendi il completamento dell'aggiornamento di tutti i nodi prima di eseguire la procedura successiva.

12. Se il provider DNS del cluster è `kube-dns`, riduci orizzontalmente l'implementazione di `kube-dns` a una replica.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13. (Facoltativo) Se si sta utilizzando il Cluster Autoscaler Kubernetes, dimensionare l'implementazione al numero di repliche desiderato.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

14. (Facoltativo) Verifica che si sta utilizzando la versione più recente del plugin CNI di Amazon VPC per Kubernetes. Potrebbe essere necessario aggiornare la versione del plugin CNI di Amazon VPC per Kubernetes per utilizzare i tipi di istanze supportati più recenti. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC".

Semplifica la gestione dell'elaborazione con AWS Fargate

In questo argomento viene illustrato l'utilizzo di Amazon EKS per eseguire i pod Kubernetes su AWS Fargate. Fargate è una tecnologia che fornisce capacità di calcolo on demand e di dimensioni adeguate per i container. Con Fargate, non devi effettuare personalmente il provisioning, la configurazione o il dimensionamento dei gruppi di macchine virtuali per eseguire i container. Non devi inoltre scegliere i tipi di server, decidere quando dimensionare i gruppi di nodi oppure ottimizzare l'impacchettamento dei cluster.

Puoi controllare quali pod sono avviati su Fargate e come vengono eseguiti con profili Fargate. I profili Fargate sono definiti come parte del cluster Amazon EKS. Amazon EKS integra Kubernetes con Fargate tramite l'utilizzo di controller creati da AWS utilizzando il modello upstream ed estensibile fornito da Kubernetes. Questi controller vengono eseguiti come parte del piano di controllo di Kubernetes gestito da Amazon EKS e sono responsabili della pianificazione dei pod Kubernetes nativo su Fargate. I controller Fargate includono un nuovo pianificazione che viene eseguito insieme al pianificatore di default di Kubernetes, oltre a diversi controller di ammissione mutanti e convalidanti. Quando avvii un pod che soddisfa i criteri per l'esecuzione su Fargate, i controller Fargate in esecuzione nel cluster riconoscono, aggiornano e pianificano il pod su Fargate.

Questo argomento descrive i diversi componenti di pod in esecuzione su Fargate, e illustra considerazioni speciali per l'utilizzo di Fargate con Amazon EKS.

Considerazioni su AWS Fargate

Di seguito sono elencati alcuni punti da considerare sull'uso di Fargate in Amazon EKS.

- Ogni pod in esecuzione su Fargate ha un proprio limite di calcolo. Non condividono il kernel sottostante, le risorse CPU, le risorse di memoria o l'interfaccia di rete elastica con un altro pod.
- I Network Load Balancer e gli Application Load Balancer (ALB) possono essere utilizzati solo con Fargate con destinazioni IP. Per ulteriori informazioni, consulta the section called "Creazione di un Network Load Balancer" e the section called "Bilanciamento del carico di applicazione".
- I servizi esposti Fargate vengono eseguiti solo in modalità IP di tipo destinazione e non in modalità IP del nodo. Il modo consigliato per verificare la connettività da un servizio in esecuzione su un nodo gestito e da un servizio in esecuzione su Fargate è quello di connettersi tramite il nome del servizio.
- Nel momento in cui sono programmati per l'esecuzione su Fargate, i pod devono corrispondere a un profilo Fargate. I pod che non corrispondono a un profilo Fargate potrebbero rimanere bloccati nello stato `Pending`. Se esiste un profilo Fargate corrispondente, puoi eliminare i pod in sospeso creati per riprogrammarli su Fargate.
- I Daemonset non sono supportati su Fargate. Se l'applicazione richiede un daemon, riconfigura quel daemon per essere eseguito nei pod come container sidecar.
- I container privilegiati non sono supportati su Fargate.
- I pod in esecuzione su Fargate non possono specificare `HostPort` o `HostNetwork` nel manifesto del pod.
- Per i pod Fargate, il valore di default per il limite soft `nofile` e `nproc` è 1.024, mentre per il limite hard è 65.535.
- Le GPU non sono attualmente disponibili su Fargate.
- I pod in esecuzione su Fargate sono supportati solo su sottoreti private (con un accesso NAT gateway ai servizi AWS, ma senza routing diretto a un Gateway Internet), pertanto il VPC del cluster deve disporre di sottoreti private disponibili. Per i cluster senza accesso Internet in uscita, consulta the section called "Cluster privati".
- Puoi utilizzare Adjust pod resources with Vertical Pod Autoscaler per impostare la dimensione corretta iniziale della CPU e la memoria per i Pod Fargate, quindi utilizza Scale pod deployments with Horizontal Pod Autoscaler per scalare quei Pod. Se desideri che il Vertical Pod Autoscaler ridistribuisca automaticamente i pod su Fargate con combinazioni di CPU e memoria più grandi, imposta la modalità per il Vertical Pod Autoscaler su `Auto` o `Recreate` r per garantire la corretta

funzionalità. Per ulteriori informazioni, consulta la documentazione Vertical Pod Autoscaler su GitHub.

- La risoluzione DNS e i nomi host DNS devono essere abilitati per il VPC. Per ulteriori informazioni, consulta Visualizzazione e aggiornamento del supporto DNS per il VPC.
- Amazon EKS Fargate aggiunge una difesa approfondita per le applicazioni Kubernetes isolando ogni pod all'interno di una macchina virtuale (VM). Questo limite VM impedisce l'accesso alle risorse basate su host utilizzate da altri pod in caso di evasione da un container, un metodo comune per attaccare le applicazioni containerizzate e ottenere l'accesso a risorse esterne al container.

  L'utilizzo di Amazon EKS non modifica le tue responsabilità ai sensi del modello di responsabilità condivisa. È necessario considerare attentamente la configurazione dei controlli di sicurezza e gestione del cluster. Il modo più sicuro per isolare un'applicazione è sempre quello di eseguirla in un cluster separato.
- I profili Fargate supportano la specificazione di sottoreti da blocchi CIDR secondari del VPC. È possibile specificare un blocco CIDR secondario. Ciò è necessario perché in una sottorete è disponibile un numero limitato di indirizzi IP. Di conseguenza, nel cluster puoi creare un numero limitato di pod. L'uso di sottoreti diverse per i pod ti consente di aumentare il numero di indirizzi IP disponibili. Per ulteriori informazioni, consulta Aggiunta di blocchi CIDR IPv4 a un VPC.
- Il servizio di metadati di istanza (IMDS) Amazon EC2 non è disponibile per i pod distribuiti ai nodi Fargate. Se disponi di pod distribuiti in Fargate che necessitano di credenziali IAM, assegnali ai pod utilizzando i ruoli IAM per gli account di servizio. Se i pod hanno bisogno di accedere ad altre informazioni disponibili tramite IMDS, devi eseguire la codifica fissa di queste informazioni nelle specifiche del pod. Ciò include la Regione AWS o la zona di disponibilità in cui viene implementato un pod.
- Non puoi implementare i pod Fargate in AWS Outposts, AWS Wavelength o AWS nelle zone locali.
- Periodicamente, Amazon EKS deve applicare patch ai Pod Fargate per garantire che siano sicuri. Gli aggiornamenti vengono effettuati in modo da avere il minor impatto possibile. Tuttavia, se i pod non vengono espulsi con successo, a volte è necessario eliminarli. Di seguito sono riportate le azioni che è possibile intraprendere per ridurre al minimo le interruzioni. Per ulteriori informazioni, consulta the section called "Eventi di applicazione di patch del sistema operativo".
- Il plug-in CNI di Amazon VPC per Amazon EKS è installato sui nodi Fargate. Non puoi utilizzare Alternate CNI plugins for Amazon EKS clusters con i nodi Fargate.

- Un pod in esecuzione su Fargate monta automaticamente un file system Amazon EFS senza bisogno della procedura di installazione manuale del driver. Non puoi utilizzare il provisioning dinamico dei volumi persistenti con nodi Fargate, ma puoi usare quello statico.
- Amazon EKS non supporta Fargate Spot.
- Non puoi montare volumi Amazon EBS sui pod Fargate.
- Puoi eseguire il controller Amazon EBS CSI sui nodi Fargate, ma sul nodo Amazon EBS CSI DaemonSet può essere eseguito solo su istanze Amazon EC2.
- Dopo che un Kubernetes Job è stato contrassegnato `Completed` o `Failed`, i pod creati normalmente continuano a esistere. Questo comportamento ti consente di visualizzare i tuoi log e risultati, ma con Fargate dovrai sostenere dei costi se non ripulisci il processo in seguito.

  Per eliminare automaticamente i pod correlati dopo il completamento o un errore di un processo, puoi specificare un periodo di tempo utilizzando il controller time-to-live (TTL). L'esempio seguente mostra la specifica di `.spec.ttlSecondsAfterFinished` nel manifesto del processo.

```
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
```

Tabella di confronto Fargate

Criteri	AWS Fargate
Si può implementare in AWS Outposts	No
Si può implementare in una Zona locale AWS	No

Criteri	AWS Fargate
Può eseguire container che richiedono Windows	No
Può eseguire container che richiedono Linux	Sì
Può eseguire carichi di lavoro che richiedono il chip Inferentia	No
Può eseguire carichi di lavoro che richiedono una GPU	No
Può eseguire carichi di lavoro che richiedono processori Arm	No
Può eseguire AWS Bottlerocket	No
I pod condividono un ambiente di runtime del kernel con altri pod	No: ogni pod ha un kernel dedicato
I pod condividono risorse di CPU, memoria, archiviazione e rete con altri pod.	No: ogni pod dispone di risorse dedicate e può essere ridimensionato in modo indipendente per massimizzare l'utilizzo delle risorse.
I pod possono utilizzare più hardware e memoria rispetto alle specifiche dei pod	No: il pod può essere comunque reimpleme ntato utilizzando una vCPU e una configura zione di memoria più grandi.
È necessario implementare e gestire le istanze Amazon EC2	No
È necessario proteggere, mantenere e applicare patch al sistema operativo delle istanze Amazon EC2	No
Può fornire argomenti di bootstrap all'imple mentazione di un nodo, come argomenti kubelet aggiuntivi.	No

Criteri	AWS Fargate
Può assegnare indirizzi IP ai pod da un intervallo CIDR diverso rispetto all'indirizzo IP assegnato al nodo.	No
Puoi eseguire SSH nel nodo	No: non esiste un sistema operativo host del nodo su cui eseguire il SSH.
Puoi implementare un'AMI personalizzata nei nodi	No
Può implementare una CNI personalizzata nei nodi	No
É necessario aggiornare l'AMI del nodo per conto proprio	No
È necessario aggiornare la versione del nodo Kubernetes per conto proprio	No: non sei tu a gestire i nodi.
Può utilizzare lo spazio di archiviazione Amazon EBS con i pod	No
Può utilizzare lo spazio di archiviazione di Amazon EFS con i pod	Sì
Può utilizzare lo spazio di archiviazione Amazon FSx per Lustre con i pod	No
Può utilizzare Network Load Balancer per i servizi	Sì, quando si utilizza Create a network load balancer
I pod possono essere eseguiti in una sottorete pubblica	No
Può assegnare diversi gruppi di sicurezza VPC a singoli pod	Sì
Può eseguire Kubernetes DaemonSets	No

Criteri	AWS Fargate
Supporto `HostPort` e `HostNetwork` nel manifesto pod	No
Disponibilità nelle regioni AWS	Alcune regioni supportate da Amazon EKS
Può eseguire container su host dedicati Amazon EC2	No
Prezzi	Costo di una singola configurazione di memoria Fargate e CPU. Ogni pod ha il rispettivo costo. Per ulteriori informazioni, consulta Prezzi di Fargate AWS.

Inizia a usare AWS Fargate per il tuo cluster

Questo argomento descrive come iniziare a eseguire Pods su AWS Fargate con il tuo cluster Amazon EKS.

Se limiti l'accesso all'endpoint pubblico del cluster utilizzando blocchi CIDR, consigliamo di abilitare anche l'accesso agli endpoint privati. In questo modo i pod Fargate possono comunicare con il cluster. Senza l'endpoint privato abilitato, i blocchi CIDR specificati per l'accesso pubblico devono includere le origini di uscita dal VPC. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

Prerequisito

Un cluster esistente. Se si dispone già di un cluster Amazon EKS, consultare la pagina Nozioni di base.

Fase 1: assicurarsi che i nodi esistenti possano comunicare con i pod Fargate

Se si utilizza un nuovo cluster senza nodi o un cluster con solo gruppi di nodi gestiti (consultare the section called "Gruppi di nodi gestiti"), è possibile passare a the section called "Fase 2: creare un ruolo di esecuzione del pod Fargate".

Supponiamo di lavorare con un cluster esistente che dispone già di nodi associati. Assicurarsi che i pod su questi nodi possano comunicare liberamente con i pod in esecuzione su Fargate. I pod in

esecuzione su Fargate vengono configurati automaticamente per utilizzare il gruppo di sicurezza del cluster per il cluster a cui sono associati. È necessario assicurarsi che eventuali i nodi esistenti nel cluster possano inviare e ricevere traffico da e verso il gruppo di sicurezza del cluster. I gruppi di nodi gestiti vengono configurati automaticamente per utilizzare anche il gruppo di sicurezza del cluster, quindi non è necessario modificarli o verificare questa compatibilità (consultare the section called "Gruppi di nodi gestiti").

Per i gruppi di nodi esistenti che sono stati creati con `eksctl` o i AWS CloudFormation modelli gestiti di Amazon EKS, puoi aggiungere manualmente il gruppo di sicurezza del cluster ai nodi. In alternativa, è possibile modificare il modello di avvio del gruppo Auto Scaling per il gruppo di nodi per collegare il gruppo di sicurezza cluster alle istanze. Per ulteriori informazioni, consultare Changing an instance's security groups nella Guida per l'utente di Amazon VPC.

Puoi verificare la presenza di un gruppo di sicurezza per il Console di gestione AWS tuo cluster nella sezione Rete relativa al cluster. In alternativa, è possibile eseguire questa operazione utilizzando il seguente AWS comando CLI. Se utilizzi questo comando, sostituisci `<my-cluster>` con il nome del cluster.

```
aws eks describe-cluster --name <my-cluster> --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Fase 2: creare un ruolo di esecuzione del pod Fargate

Quando il cluster crea Pod su AWS Fargate, i componenti che girano sull'infrastruttura Fargate devono effettuare chiamate AWS APIs per conto dell'utente. Il ruolo di esecuzione del pod Amazon EKS fornisce le autorizzazioni IAM per eseguire questa operazione. Per creare un ruolo di esecuzione di AWS Fargate Pod, vedere. the section called "Ruolo IAM per l'esecuzione del pod"

> **Note**
>
> Se il cluster è stato creato con `eksctl` utilizzando l'opzione `--fargate`, dispone già di un ruolo di esecuzione del pod disponibile nella console IAM con il modello `eksctl-my-cluster-FargatePodExecutionRole-ABCDEFGHIJKL`. Allo stesso modo, se si utilizza `eksctl` per creare i profili Fargate, `eksctl` crea il ruolo di esecuzione del pod se non ne esiste già uno.

Fase 3: creare un profilo Fargate per il cluster

Prima di poter pianificare i pod in esecuzione su Fargate nel cluster, sarà necessario definire un profilo Fargate che specifichi i pod che utilizzano Fargate al momento dell'avvio. Per ulteriori informazioni, consulta the section called "Definisci i profili".

> **Note**
>
> Se il cluster è stato creato con `eksctl` utilizzando l'opzione `--fargate`, allora un profilo Fargate è già stato creato per il cluster con selettori per tutti i pod nei namespace `kube-system` e `default`. Utilizza la procedura seguente per creare profili Fargate per gli altri namespace con cui desideri utilizzare Fargate.

È possibile creare un profilo Fargate utilizzando uno di questi due strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

`eksctl`

Questa procedura richiede `eksctl` versione `0.215.0` o successiva. È possibile verificare la versione con il comando seguente:

```
eksctl version
```

Per istruzioni sull'installazione o sull'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

Per creare un profilo Fargate con **`eksctl`**

Crea il tuo profilo Fargate con il seguente comando `eksctl`, sostituendo ogni `<example value>` con i valori in tuo possesso. È necessario specificare un namespace. Tuttavia, l'opzione `--labels` non è obbligatoria.

```
eksctl create fargateprofile \
    --cluster <my-cluster> \
    --name <my-fargate-profile> \
```

```
    --namespace <my-kubernetes-namespace> \
    --labels <key=value>
```

È possibile utilizzare determinati caratteri jolly per `<my-kubernetes-namespace>` e etichette `<key=value>`. Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

Console di gestione AWS

Creazione di un profilo Fargate con Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il cluster per cui creare un profilo Fargate.
3. Scegli la scheda Calcolo.
4. Nella sezione Profili Fargate, scegli Aggiungi profilo Fargate.
5. Nella pagina Configura il profilo Fargate, procedi come segue:
   a. In Nome, inserisci un nome per il profilo Fargate. Il nome deve essere univoco.
   b. Come Ruolo di esecuzione pod, scegliere il ruolo di esecuzione del pod da utilizzare con il profilo Fargate. Vengono visualizzati solo i ruoli IAM con il principale del servizio `eks-fargate-pods.amazonaws.com`. Se non vedi alcun ruolo nell'elenco, è necessario crearne uno. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod".
   c. Modifica le sottoreti selezionate in base alle esigenze.

      > **Note**
      >
      > Solo le sottoreti private sono supportate per i pod in esecuzione su Fargate.

   d. In Tag, puoi facoltativamente aggiungere tag al tuo profilo Fargate. Questi tag non si propagano ad altre risorse associate al profilo come i pod.
   e. Scegli Next (Successivo).
6. Nella pagina Configura selezione pod, procedi come segue:
   a. In Namespace, inserisci un namespace che corrisponda ai pod.
      - È possibile utilizzare namespace specifici da abbinare, ad esempio `kube-system` o `default`.
      - È possibile utilizzare determinati caratteri jolly (ad esempio, `prod-*`) per abbinare più namespace (ad esempio, `prod-deployment` e `prod-test`). Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

b. (Facoltativo) Aggiungere etichette Kubernetes al selettore. In particolare, aggiungerle al selettore con cui i pod nel namespace specificato devono corrispondere.

- È possibile aggiungere l'etichetta `infrastructure: fargate` al selettore in modo che solo i pod nel namespace specificato che hanno anche l'etichetta Kubernetes `infrastructure: fargate` corrispondano al selettore.
- È possibile utilizzare determinati caratteri jolly (ad esempio, `key?: value?`) per abbinare più namespace (ad esempio, `keya: valuea` e `keyb: valueb`). Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

c. Scegli Next (Successivo).

7. Nella pagina Rivedi e crea, controlla le informazioni relative al profilo Fargate e scegli Crea.

Fase 4: aggiornare CoreDNS

Per impostazione predefinita, CoredNS è configurato per l'esecuzione sull' EC2 infrastruttura Amazon su cluster Amazon EKS. Se si desidera solo eseguire i pod su Fargate nel cluster, seguire la procedura riportata di seguito.

> **Note**
>
> Se è stato creato il cluster `eksctl` utilizzando l'opzione `--fargate` è possibile passare a the section called "Fasi successive".

1. Creare un profilo Fargate per CoreDNS con il seguente comando. Sostituiscilo `<my-cluster>` con il nome del cluster, `<111122223333>` con l'ID dell'account, `<AmazonEKSFargatePodExecutionRole>` con il nome del ruolo di esecuzione del Pod e `<000000000000000a>` `<000000000000000c>` con le sottoreti IDs private. `<000000000000000b>` Se non si dispone di un ruolo di esecuzione del pod, è necessario prima crearne uno (consultare the section called "Fase 2: creare un ruolo di esecuzione del pod Fargate").

   > **Important**
   >
   > L'ARN del ruolo non può includere un percorso diverso da `/`. Ad esempio, se il nome del ruolo è `development/apps/AmazonEKSFargatePodExecutionRole`, è necessario modificarlo in `AmazonEKSFargatePodExecutionRole` quando si

specifica l'ARN per tale ruolo. Il formato dell'ARN del ruolo deve essere `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. Attivare un rollout dell'implementazione `coredns`.

```
kubectl rollout restart -n kube-system deployment coredns
```

Fasi successive

- Puoi avviare la migrazione delle applicazioni esistenti per l'esecuzione su Fargate con il seguente flusso di lavoro.
  a. the section called "Creazione di un profilo Fargate" che corrisponda al namespace Kubernetes e alle etichette Kubernetes dell'applicazione.
  b. Eliminare e creare nuovamente tutti i pod esistenti in modo che siano programmati su Fargate. Modificare `<namespace>` e `<deployment-type>` per aggiornare i pod specifici.

  ```
  kubectl rollout restart -n <namespace> deployment <deployment-type>
  ```

- Implementare the section called "Bilanciamento del carico di applicazione" per permettere l'esecuzione di oggetti Ingress per i pod in esecuzione su Fargate.
- È possibile utilizzare the section called "Vertical Pod Autoscaler" per impostare la dimensione corretta iniziale della CPU e la memoria per i pod Fargate, quindi utilizzare il comando the section called "Horizontal Pod Autoscaler" per scalare tali pod. Se si desidera che il Vertical Pod Autoscaler implementi di nuovo automaticamente i pod su Fargate con combinazioni di CPU e memoria più grandi, impostare la modalità del Vertical Pod Autoscaler su `Auto` o `Recreate`. Ciò garantisce una corretta funzionalità. Per ulteriori informazioni, consulta la documentazione di Vertical Pod Autoscaler su GitHub.

- Ora puoi impostare il collector AWS Distro per OpenTelemetry (ADOT) per il monitoraggio delle applicazioni seguendo queste istruzioni.

Definisci quali pod utilizzano AWS Fargate durante l'avvio

Prima di programmare i pod su Fargate nel cluster, devi definire almeno un profilo Fargate che specifichi i pod utilizzati da Fargate al momento dell'avvio.

Come amministratore, puoi utilizzare il profilo Fargate per dichiarare quali pod vengono eseguiti su Fargate. Puoi farlo attraverso i selettori del profilo. Puoi aggiungere fino a cinque selettori a ogni profilo. Ogni selettore deve contenere uno spazio dei nomi. Il selettore può includere anche etichette. Il campo etichetta è costituito da più coppie chiave-valore facoltative. I pod che corrispondono ai selettori sono programmati su Fargate. I pod vengono abbinati utilizzando uno spazio dei nomi e le etichette specificate nel selettore. Se un selettore di namespace è definito senza etichette, Amazon EKS tenterà di pianificare tutti i pod che vengono eseguiti in tale namespace su Fargate utilizzando il profilo. Se un pod da programmare corrisponde a uno dei selettori nel profilo Fargate, allora quel pod viene programmato su Fargate.

Se un pod corrisponde a più profili Fargate, puoi specificare quale profilo viene utilizzato dal pod aggiungendo la seguente etichetta Kubernetes alla specifica del pod: `eks.amazonaws.com/fargate-profile: my-fargate-profile`. Il pod deve corrispondere a un selettore in quel profilo per essere pianificato su Fargate. Le regole di affinità/anti-affinità Kubernetes non si applicano e non sono necessarie con i pod Fargate di Amazon EKS.

Quando crei un profilo Fargate, devi specificare il ruolo di esecuzione di un pod. Questo ruolo di esecuzione è per i componenti Amazon EKS che vengono eseguiti su infrastruttura Fargate utilizzando il profilo. Viene aggiunto al controllo di accesso basato sul ruolo (RBAC) del cluster di Kubernetes per l'autorizzazione. Ciò consente a `kubelet` in esecuzione sull'infrastruttura Fargate di registrarsi con il cluster Amazon EKS in modo che possa essere visualizzato nel cluster come nodo. Il ruolo di esecuzione del pod fornisce anche le autorizzazioni IAM per l'infrastruttura Fargate per consentire l'accesso in lettura ai repository di immagini di Amazon ECR. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod".

I profili Fargate non possono essere modificati. Tuttavia, è possibile creare un nuovo profilo aggiornato per sostituire un profilo esistente e quindi eliminare l'originale.

> **Note**
>
> Qualsiasi pod in esecuzione utilizzando un profilo Fargate viene arrestato e messo in sospeso quando il profilo viene eliminato.

Se tutti i profili in un cluster sono nello stato `DELETING`, è necessario attendere che tale profilo Fargate finisca l'eliminazione prima di poter creare altri profili in tale cluster.

> **Note**
>
> Attualmente, Fargate non supporta topologySpreadConstraints di Kubernetes.

Amazon EKS e Fargate cercano di implementare i pod su ciascuna delle sottoreti definite nel profilo Fargate. Tuttavia, potresti ritrovarti con una diffusione irregolare. Se devi avere una diffusione uniforme, usa due profili Fargate. Anche la diffusione è importante in scenari in cui desideri implementare due repliche e non vuoi tempi di inattività. È consigliabile che ogni profilo abbia una sola sottorete.

Componenti del profilo Fargate

Un profilo Fargate contiene i componenti elencati di seguito.

Ruolo di esecuzione del pod

Quando il cluster crea pod su AWS Fargate, il `kubelet` in esecuzione sull'infrastruttura di Fargate deve effettuare chiamate ad API AWS per tuo conto. Ad esempio, deve effettuare chiamate per estrarre le immagini del container da Amazon ECR. Il ruolo di esecuzione del pod Amazon EKS fornisce le autorizzazioni IAM per eseguire questa operazione.

Quando crei un profilo Fargate, devi specificare il ruolo di esecuzione di un pod da utilizzare col tuo pod. Questo ruolo viene aggiunto al controllo di accesso basato sul ruolo (RBAC) del cluster di Kubernetes per l'autorizzazione. Ciò consente a `kubelet` in esecuzione sull'infrastruttura Fargate di registrarsi con il cluster Amazon EKS in modo che possa essere visualizzato nel cluster come nodo. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod".

Sottoreti

Gli ID delle sottoreti per avviare i pod che utilizzano questo profilo. A questo momento, ai pod in esecuzione su Fargate non vengono assegnati indirizzi IP pubblici. Di conseguenza, in virtù di questo parametro, sono accettate solo le sottoreti private senza routing diretto a un gateway Internet.

Selettori

I selettori da abbinare ai pod per utilizzare questo profilo Fargate. È possibile specificare fino a cinque selettori in un profilo Fargate. I selettori hanno le seguenti componenti:

- Spazio dei nomi: specifica uno spazio dei nomi per un selettore. Il selettore corrisponde solo ai pod che vengono creati in questo namespace. Tuttavia, è possibile creare più selettori per rivolgersi a più spazi dei nomi.
- Etichette: è possibile specificare facoltativamente le etichette Kubernetes che corrispondono al selettore. Il selettore corrisponde solo ai pod che hanno tutte le etichette specificate nel selettore.

Wildcard del profilo di Fargate

Oltre ai caratteri consentiti da Kubernetes, puoi utilizzare `*` e `?` nei criteri di selezione per namespace, chiavi di etichette e valori di etichette:

- `*` rappresenta nessuno, uno o più caratteri. Ad esempio, `prod*` può rappresentare `prod` e `prod-metrics`.
- `?` rappresenta un singolo carattere (ad esempio, `value?` può rappresentare `valuea`). Tuttavia, non può rappresentare `value` e `value-a`, perché `?` può rappresentare solo un carattere.

Questi caratteri jolly possono essere usati in qualsiasi posizione e in combinazione (ad esempio, `prod*`, `*dev`, e `frontend*?`). Altri caratteri jolly e forme di corrispondenza del modello, come le espressioni regolari, non sono supportati.

Se sono presenti più profili corrispondenti per il namespace e le etichette nelle specifiche del pod, Fargate seleziona il profilo in base all'ordinamento alfanumerico e al nome del profilo. Ad esempio, se entrambi i profili A (con il nome `beta-workload`) e il profilo B (con il nome `prod-workload`) hanno dei selettori corrispondenti per i pod da avviare, Fargate sceglie il profilo A (`beta-workload`) per i pod. I pod hanno etichette con il profilo A sui pod (ad esempio, `eks.amazonaws.com/fargate-profile=beta-workload`).

Se desideri eseguire la migrazione dei pod Fargate esistenti su nuovi profili che utilizzano caratteri jolly, puoi farlo in due modi:

- Crea un nuovo profilo con i selettori corrispondenti, quindi elimina i vecchi profili. I pod etichettati con vecchi profili vengono riprogrammati con nuovi profili corrispondenti.
- Se desideri eseguire la migrazione dei carichi di lavoro ma non hai la certezza di quali siano le etichette Fargate su ogni pod Fargate, puoi utilizzare il seguente metodo. Crea un nuovo profilo con un nome che ordina in ordine alfanumerico per primo tra i profili dello stesso cluster. Quindi, ricicla i pod Fargate che devono essere migrati a nuovi profili.

Creazione di un profilo Fargate

Questa sezione descrive come creare un profilo Fargate. Devi inoltre aver creato un ruolo di esecuzione del pod da utilizzare per il profilo Fargate. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod". I pod in esecuzione su Fargate sono supportati solo su sottoreti private con un accesso NAT gateway ai servizi AWS, ma senza una route diretta a un gateway Internet. In questo modo il VPC del cluster deve avere sottoreti private disponibili.

Puoi creare un profilo con ciò che segue:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

`eksctl`

Creazione di un profilo Fargate con **`eksctl`**

Crea il tuo profilo Fargate con il seguente comando `eksctl`, sostituendo ogni valore di esempio con i valori in tuo possesso. Devi specificare un namespace. Tuttavia, l'opzione `--labels` non è obbligatoria.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

È possibile utilizzare determinati caratteri jolly per `my-kubernetes-namespace` e etichette `key=value`. Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

Console di gestione AWS

Creazione di un profilo Fargate con Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli il cluster per cui creare un profilo Fargate.
3. Scegli la scheda Calcolo.
4. Nella sezione Profili Fargate, scegli Aggiungi profilo Fargate.
5. Nella pagina Configure Fargate profile (Configura profilo Fargate), procedere come segue:
   a. In Nome, inserisci un nome univoco per il profilo Fargate, ad esempio `my-profile`.
   b. Per il ruolo di esecuzione del pod, scegli il ruolo di esecuzione del pod da utilizzare con il profilo Fargate. Vengono visualizzati solo i ruoli IAM con il principale del servizio `eks-fargate-pods.amazonaws.com`. Se non vedi alcun ruolo nell'elenco, è necessario crearne uno. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod".
   c. Modifica le sottoreti selezionate in base alle esigenze.

      > **Note**
      >
      > Solo le sottoreti private sono supportate per i pod in esecuzione su Fargate.

   d. In Tag, puoi facoltativamente aggiungere tag al tuo profilo Fargate. Questi tag non si propagano ad altre risorse associate al profilo, ad esempio i suoi pod.
   e. Scegli Next (Successivo).
6. Nella pagina Configura selezione pod, procedi come segue:
   a. In Namespace, inserisci uno spazio dei nomi che corrisponda ai pod.
      - È possibile utilizzare spazi dei nomi specifici da abbinare, ad esempio `kube-system` o `default`.
      - È possibile utilizzare determinati caratteri jolly (ad esempio, `prod-*`) per abbinare più spazi dei nomi (ad esempio, `prod-deployment` e `prod-test`). Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".
   b. (Facoltativo) Aggiungere etichette Kubernetes al selettore. In particolare, aggiungile al selettore a cui devono corrispondere i pod nel namespace specificato.

- Puoi aggiungere l'etichetta `infrastructure: fargate` al selettore in modo che solo i pod nel namespace specificato che hanno anche l'etichetta Kubernetes `infrastructure: fargate` corrispondano al selettore.
- È possibile utilizzare determinati caratteri jolly (ad esempio, `key?: value?`) per abbinare più spazi dei nomi (ad esempio, `keya: valuea` e `keyb: valueb`). Per ulteriori informazioni, consulta the section called "Wildcard del profilo di Fargate".

c. Scegli Next (Successivo).

7. Nella pagina Rivedi e crea, controlla le informazioni relative al profilo Fargate e scegli Crea.

Eliminare un profilo Fargate

Questo argomento descrive come eliminare un profilo Fargate. Quando si elimina un profilo Fargate, eventuali pod pianificati su Fargate con tale profilo vengono eliminati. Se tali pod corrispondono a un altro profilo Fargate, vengono pianificati su Fargate con quel profilo. Se non corrispondono più a nessun profilo Fargate, non saranno pianificati su Fargate e potrebbero rimanere in sospeso.

Solo un profilo Fargate in un cluster alla volta può essere nello stato `DELETING`. Devi attendere il completamento dell'eliminazione di un profilo Fargate prima di poter eliminare altri profili nel cluster.

possibile eliminare un profilo con uno qualsiasi dei seguenti strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"
- the section called "CLI AWS"

`eksctl`

Eliminare un profilo Fargate con **`eksctl`**

Utilizza il comando seguente per eliminare un profilo da un cluster. Sostituire ogni *`valore di esempio`* con i propri valori.

```
eksctl delete fargateprofile  --name my-profile --cluster my-cluster
```

Console di gestione AWS

Eliminare un profilo Fargate con Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster. Nell'elenco dei cluster, scegli il cluster da cui desideri eliminare il profilo Fargate.
3. Scegli la scheda Calcolo.
4. Scegli il profilo Fargate da eliminare e quindi Elimina.
5. Nella pagina Elimina profilo Fargate, digita il nome del profilo e scegli Elimina.

CLI AWS

Eliminare un profilo Fargate con AWS CLI

Utilizza il comando seguente per eliminare un profilo da un cluster. Sostituire ogni *valore di esempio* con i propri valori.

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-cluster
```

Comprendere i dettagli di configurazione dei pod di Fargate

Questa sezione descrive alcuni dei dettagli di configurazione dei pod unici per l'esecuzione dei pod Kubernetes su AWS Fargate.

Memoria e CPU del pod

Kubernetes ti consente di definire le richieste, una quantità minima di risorse vCPU e di memoria allocate a ciascun container in un pod. I pod sono programmati da Kubernetes per garantire che almeno le risorse richieste per ciascun pod siano disponibili nella risorsa di calcolo. Per ulteriori informazioni, consulta Gestione risorse di calcolo per container nella documentazione su Kubernetes.

> **Note**
>
> Poiché Amazon EKS Fargate esegue solo un pod per nodo, lo scenario di espulsione dei pod in caso di meno risorse non si verifica. Tutti i pod Amazon EKS Fargate funzionano con priorità garantita, quindi la CPU e la memoria richieste devono essere uguali al limite per tutti i container. Per ulteriori informazioni, consulta Gestire la qualità di servizio per i pod nella documentazione di Kubernetes.

Quando i pod sono programmati su Fargate, le prenotazioni di vCPU e memoria all'interno della specifica del pod determinano la quantità di CPU e memoria per effettuare il provisioning per il pod.

- La richiesta massima di qualsiasi container Init viene utilizzata per determinare i requisiti di memoria e vCPU della richiesta Init.
- Le richieste per tutti i container con esecuzione prolungata vengono aggiunte per determinare i requisiti di memoria e vCPU della richiesta con esecuzione prolungata.
- Il valore maggiore tra i due valori sopra indicati viene scelto per la richiesta di vCPU e memoria da utilizzare per il pod.
- Fargate aggiunge 256 MB alla prenotazione di memoria di ogni pod per i componenti Kubernetes richiesti (`kubelet`, `kube-proxy` e `containerd`).

Fargate arrotonda alla configurazione di calcolo mostrata di seguito che corrisponde più strettamente alla somma delle richieste di vCPU e di memoria, al fine di garantire che i pod abbiano sempre le risorse necessarie per la loro esecuzione.

Se non specifichi una combinazione di vCPU e memoria, viene utilizzata la combinazione più piccola disponibile (.25 vCPU e 0,5 GB di memoria).

La tabella seguente mostra le combinazioni di vCPU e memoria disponibili per i pod in esecuzione su Fargate.

Valore vCPU	Valore memoria
.25 vCPU	0,5 GB, 1 GB, 2 GB
.5 vCPU	1 GB, 2 GB, 3 GB, 4 GB
1 vCPU	2 GB, 3 GB, 4 GB, 5 GB, 6 GB, 7 GB, 8 GB
2 vCPU	Tra 4 GB e 16 GB in incrementi di 1 GB
4 vCPU	Tra 8 GB e 30 GB in incrementi di 1 GB
8 vCPU	Tra 16 GB e 60 GB in incrementi di 4 GB
16 vCPU	Tra 32 GB e 120 GB in incrementi di 8 GB

La memoria aggiuntiva riservata ai componenti Kubernetes può causare un'attività Fargate con più vCPUs di quelle richieste per il provisioning. Ad esempio, una richiesta per 1 vCPU e 8 GB di memoria avrà 256 MB aggiunti alla richiesta di memoria e fornirà un'attività Fargate di 2 vCPU e 9 GB di memoria, poiché non è disponibile alcuna attività con 1 vCPU e 9 GB di memoria.

Non vi è alcuna correlazione tra la dimensione del pod in esecuzione su Fargate e quella del nodo riportata da Kubernetes con `kubectl get nodes`. La dimensione del nodo riportata è spesso maggiore della capacità del pod. Puoi verificare la capacità del pod utilizzando il comando seguente. Sostituisci *`default`* con il namespace del pod e *`pod-name`* con il nome dello stesso.

```
kubectl describe pod --namespace default pod-name
```

Di seguito viene riportato un output di esempio:

```
[...]
annotations:
    CapacityProvisioned: 0.25vCPU 0.5GB
[...]
```

L'annotazione `CapacityProvisioned` rappresenta la capacità del pod applicata e determina il costo del pod in esecuzione su Fargate. Per informazioni sui prezzi di queste configurazioni di calcolo, vedi Prezzi Fargate AWS.

Archiviazione in Fargate

Un pod in esecuzione su Fargate monta automaticamente un file system Amazon EFS senza bisogno della procedura di installazione manuale del driver. Non puoi utilizzare il provisioning dinamico dei volumi persistenti con nodi Fargate, ma puoi usare quello statico. Per ulteriori informazioni, consulta Driver CSI per Amazon EFS su GitHub.

Una volta fornito, ciascun pod in esecuzione su Fargate riceve uno storage temporaneo predefinito di 20 GiB. Questo tipo di archiviazione viene eliminato dopo che un pod si ferma. Per i nuovi pod avviati su Fargate, la crittografia del volume di archiviazione temporanea è abilitata per impostazione predefinita. L'archiviazione del pod effimero viene crittografata con un algoritmo di crittografia AES-256 utilizzando chiavi gestite da Fargate AWS.

> **Note**
>
> Lo storage utilizzabile predefinito per i pod Amazon EKS che funzionano su Fargate sono inferiori a 20 GiB. Questo perché parte dello spazio viene utilizzato da `kubelet` e altri moduli Kubernetes che vengono caricati all'interno del pod.

Puoi aumentare la quantità totale di storage temporaneo fino a un massimo di 175 GiB. Per configurare la dimensione con Kubernetes, specifica le richieste della risorsa `ephemeral-storage` per ogni container in un pod. Quando Kubernetes pianifica i pod, garantisce che la somma delle richieste di risorse per ciascun pod sia inferiore alla capacità dell'attività Fargate. Per ulteriori informazioni, consulta Gestione delle risorse per pod e container nella documentazione di Kubernetes.

Amazon EKS Fargate fornisce un maggiore spazio di archiviazione temporaneo di quello richiesto ai fini dell'utilizzo del sistema. Ad esempio, una richiesta di 100 GiB fornirà un'attività Fargate con 115 GiB di storage temporaneo.

Imposta azioni per gli eventi relativi all'applicazione di patch del sistema operativo AWS Fargate

Amazon EKS applica periodicamente delle patch al sistema operativo per i nodi AWS Fargate al fine di garantirne la sicurezza. Come parte del processo di applicazione delle patch, ricicliamo i nodi per installare le patch del sistema operativo. Gli aggiornamenti vengono effettuati in modo da avere il minore impatto possibile sui servizi. Tuttavia, se i pod non vengono espulsi con successo, a volte è necessario eliminarli. Di seguito sono riportate le azioni che è possibile intraprendere per ridurre al minimo le potenziali interruzioni:

- Imposta i budget di interruzione dei pod (PDB) appropriati per limitare il numero di pod inattivi.
- Crea regole di Amazon EventBridge per gestire le espulsioni non riuscite prima che i pod vengano eliminati.
- Riavvia manualmente i pod interessati prima della data di espulsione indicata nella notifica ricevuta.
- Crea una configurazione di notifica in Notifiche utente di AWS.

Amazon EKS lavora a stretto contatto con la community di Kubernetes per rendere disponibili le correzioni dei bug e le patch di sicurezza il più rapidamente possibile. L'avvio di tutti i pod Fargate avviene con la versione più recente della patch di Kubernetes, disponibile da Amazon EKS per la

versione di Kubernetes del cluster. Se disponi di un pod con una patch di una versione precedente, Amazon EKS potrebbe riciclarlo per aggiornarlo alla versione più recente. Ciò consente di applicare ai pod gli aggiornamenti per la sicurezza più recenti, in modo da ridurre i rischi nel caso in cui si verifichi una problematica di tipo Common Vulnerabilities and Exposures (CVE).

Quando il sistema operativo AWS Fargate viene aggiornato, Amazon EKS ti invia una notifica che include le risorse interessate e la data delle prossime espulsioni dei pod. Se la data di espulsione fornita è scomoda, hai la possibilità di riavviare manualmente i pod interessati prima della data di espulsione indicata nella notifica. Tutti i pod creati prima della data di ricezione della notifica sono soggetti a espulsione. Per ulteriori istruzioni su come riavviare manualmente i pod, consulta Kubernetes Documentation.

Per limitare il numero di pod inattivi durante il riciclo degli stessi, puoi impostare i budget di interruzione dei pod (PDB). Con questa opzione è possibile specificare la disponibilità minima in base ai requisiti di ciascuna applicazione e consentire, allo stesso tempo, l'esecuzione di aggiornamenti. La disponibilità minima del PDB deve essere inferiore al 100%. Per ulteriori informazioni, consulta Specifying a Disruption Budget for your Application nella documentazione di Kubernetes.

Amazon EKS utilizza l'API di espulsione per interrompere in modo sicuro il pod rispettando i budget di interruzione dei pod impostati per l'applicazione. I pod vengono espulsi dalla zona di disponibilità per ridurre al minimo l'impatto. Se l'espulsione ha esito positivo, il nuovo pod riceve la patch più recente e non sono richieste ulteriori operazioni.

Se l'espulsione di un pod ha esito negativo, Amazon EKS invia al tuo account un evento con dettagli relativi al pod in cui si è verificato l'errore. È possibile intraprendere alcune azioni in merito al messaggio prima del periodo di interruzione programmato, che varia in base all'urgenza della patch. Quando è il momento, Amazon EKS proverà a espellere nuovamente i pod. Questa volta, tuttavia, se l'espulsione ha esito negativo non viene inviato un nuovo evento Se l'espulsione continua a non riuscire, i pod esistenti verranno eliminati periodicamente in modo che i nuovi pod possano disporre della patch più recente.

L'esempio seguente mostra un evento ricevuto in caso di espulsione del pod che non riesce. Il messaggio contiene dettagli sul cluster, il nome del pod, il namespace del pod, il profilo Fargate e il periodo di interruzione pianificato.

```
{
    "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]"
    }
}
```

L'associazione di molteplici PDB a un singolo pod può causare un evento di errore di espulsione. Questo evento restituisce il messaggio di errore seguente.

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction
 subresource does not support",
```

È possibile creare un'azione desiderata in base a questo evento. Ad esempio, puoi modificare il budget di interruzione dei pod (PDB) per controllare il modo in cui i pod vengono espulsi. Più specificamente, supponi di iniziare con un PDB che specifica la percentuale di destinazione dei pod disponibili. Prima che i pod vengano terminati in modo forzato durante un aggiornamento, puoi regolare il PDB su una percentuale di pod differente. Per ricevere questo evento, devi creare una regola di Amazon EventBridge nell'account AWS e nella Regione AWS a cui appartiene il cluster. La regola deve utilizzare il modello personalizzato seguente. Per ulteriori informazioni, consulta la sezione Creazione di regole Amazon EventBridge che reagiscono agli eventi nella Guida per l'utente di Amazon EventBridge.

```
{
  "source": ["aws.eks"],
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

L'evento può essere configurato per acquisire un obiettivo adeguato impostato dall'utente. Per un elenco completo delle destinazioni disponibili, consulta Destinazioni di Amazon EventBridge

nella Guida per l'utente di Amazon EventBridge. Inoltre puoi creare una configurazione di notifica in Notifiche utente AWS. Quando usi Console di gestione AWS per creare la notifica, sotto la voce Regole degli eventi scegli Elastic Kubernetes Service (EKS) per il Nome di servizio AWS e Arresto programmato di EKS Fargate pod per il Tipo di evento. Per ulteriori informazioni, consulta Guida introduttiva alle Notifiche utente AWS nella Guida per l'utente delle Notifiche utente AWS.

Consulta Domande frequenti: Avviso di espulsione di un Fargate Pod in AWS re:Post per le domande frequenti riguardo alle espulsioni dei pod EKS.

Raccogli le AWS metriche dell'app Fargate e dell'utilizzo

Puoi raccogliere metriche di sistema e metriche di CloudWatch utilizzo per AWS Fargate.

Parametri di applicazione

Per le applicazioni in esecuzione su Amazon EKS e AWS Fargate, puoi utilizzare AWS Distro for OpenTelemetry (ADOT). ADOT ti consente di raccogliere i parametri di sistema e inviarli ai dashboard di Container Insights. CloudWatch Per iniziare a usare ADOT per le applicazioni in esecuzione su Fargate, consultate CloudWatch Using Container Insights AWS with Distro OpenTelemetry for nella documentazione ADOT.

Parametri di utilizzo

Puoi utilizzare le metriche di CloudWatch utilizzo per fornire visibilità sull'utilizzo delle risorse da parte del tuo account. Utilizza queste metriche per visualizzare l'utilizzo corrente del servizio su CloudWatch grafici e dashboard.

AWS Le metriche di utilizzo di Fargate corrispondono alle AWS quote di servizio. È possibile configurare gli allarmi che avvisano quando l'uso si avvicina a una quota di servizio. Per ulteriori informazioni sulle quote di servizio per Fargate, consulta the section called "Quote di servizio".

AWS Fargate pubblica le seguenti metriche nel namespace. `AWS/Usage`

Metrica	Description
`ResourceCount`	Il numero totale delle risorse specificate in esecuzione nell'account. La risorsa è definita dalle dimensioni associate attraverso il parametro.

Le seguenti dimensioni vengono utilizzate per perfezionare le metriche di utilizzo pubblicate da AWS Fargate.

Dimensione	Description
`Service`	Il nome del AWS servizio che contiene la risorsa. Per le metriche di utilizzo di AWS Fargate, il valore per questa dimensione è. `Fargate`
`Type`	Il tipo di entità che viene segnalato. Attualmente, l'unico valore valido per le metriche di utilizzo di AWS Fargate è. `Resource`
`Resource`	Il tipo di risorsa in esecuzione. Attualmente, AWS Fargate restituisce informazioni sull'utilizzo di Fargate On-Demand. Il valore della risorsa per l'utilizzo on demand di Fargate è `OnDemand`. [NOTA] ==== L'utilizzo di Fargate On-Demand combina i pod Amazon EKS utilizzando Fargate, le attività Amazon ECS che utilizzano il tipo di avvio di Fargate e le attività Amazon ECS che usano il provider di capacità `FARGATE`. ====
`Class`	La classe della risorsa monitorata. Attualmente, AWS Fargate non utilizza la dimensione classe.

Creazione di un CloudWatch allarme per monitorare le metriche di utilizzo delle risorse di Fargate

AWS Fargate fornisce metriche di CloudWatch utilizzo che corrispondono alle quote di AWS servizio per l'utilizzo delle risorse Fargate On-Demand. Nella console Service Quotas (Quote di Servizio) è possibile visualizzare l'utilizzo in un grafico. È inoltre possibile configurare gli allarmi che avvisano

quando l'uso si avvicina a una quota di servizio. Per ulteriori informazioni, consulta the section called "Raccolta dei parametri".

Utilizzate i seguenti passaggi per creare un CloudWatch allarme basato sulle metriche di utilizzo delle risorse di Fargate.

1. Apri la console Service Quotas all'indirizzo. https://console.aws.amazon.com/servicequotas/
2. Nel riquadro di navigazione a sinistra, scegli AWS servizi.
3. Dall'elenco dei AWS servizi, cerca e seleziona AWS Fargate.
4. Nell'elenco Service Quotas, seleziona la quota di utilizzo Fargate per cui desideri creare un allarme.
5. Nella sezione CloudWatch Allarmi Amazon, scegli Crea.
6. Per Soglia di allarme, scegli la percentuale del valore della quota applicata che desideri impostare come valore per l'allarme.
7. Per Nome allarme, immetti un nome per l'allarme e quindi scegli Crea.

Avviare la registrazione di log AWS Fargate per il cluster

Amazon EKS su Fargate offre un router di registro integrato basato su Fluent Bit. Ciò significa che non viene eseguito esplicitamente un container Fluent Bit come sidecar, ma è Amazon a eseguirlo. Tutto quello che devi fare è configurare il router di log. La configurazione avviene attraverso un `ConfigMap` dedicato che deve soddisfare i seguenti criteri:

- Deve essere denominato `aws-logging`
- Creato in uno spazio dei nomi dedicato denominato `aws-observability`
- Non può contenere più di 5.300 caratteri.

Una volta creato `ConfigMap`, Amazon EKS su Fargate lo rileva in automatico e lo utilizza per configurare il router di log. Fargate utilizza una versione di AWS per Fluent Bit, una distribuzione upstream conforme di Fluent Bit gestita da AWS. Per ulteriori informazioni, consultare AWS per Fluent Bit su GitHub.

Il router di log consente di utilizzare la vasta gamma di servizi AWS per l'analisi e per l'archiviazione dei log. È possibile trasmettere i registri da Fargate direttamente ad Amazon CloudWatch, un servizio OpenSearch di Amazon. È possibile trasmettere i log anche a destinazioni come Amazon S3 e flusso di dati Amazon Kinesis, così come a strumenti partner tramite Amazon Data Firehose.

- Un profilo Fargate esistente che specifichi un namespace Kubernetes esistente in cui si implementano i pod Fargate. Per ulteriori informazioni, consulta the section called "Fase 3: creare un profilo Fargate per il cluster".
- Un ruolo di esecuzione del pod Fargate esistente. Per ulteriori informazioni, consulta the section called "Fase 2: creare un ruolo di esecuzione del pod Fargate".

Configurazione del router di log

> **Important**
>
> Affinché i log vengano pubblicati correttamente, deve essere possibile accedere alla rete dal VPC in cui si trova il cluster alla destinazione dei log. Ciò riguarda principalmente gli utenti che personalizzano le regole di uscita per il VPC. Per ulteriori informazioni sull'utilizzo di CloudWatch, consultare la pagina Using CloudWatch Logs with interface VPC endpoints della Guida per l'utente di Amazon CloudWatch Logs.

Nelle fasi seguenti, sostituire ogni *valore di esempio* con i propri valori.

1. Creare uno spazio dei nomi Kubernetes dedicato denominato `aws-observability`.
   a. Salva nel tuo computer i seguenti contenuti in un file denominato `aws-observability-namespace.yaml`. Il valore per `name` deve essere `aws-observability` e l'etichetta `aws-observability: enabled` è obbligatoria.

```
kind: Namespace
apiVersion: v1
metadata:
  name: aws-observability
  labels:
    aws-observability: enabled
```

   b. Crea lo spazio dei nomi.

```
kubectl apply -f aws-observability-namespace.yaml
```

2. Crea una `ConfigMap` con valore dei dati `Fluent Conf` per spedire i log del container verso una destinazione. Fluent Conf è Fluent Bit, un linguaggio di configurazione del processore di log veloce e leggero che viene utilizzato per instradare i log del container verso una destinazione di

log selezionata. Per ulteriori informazioni, consultare File di configurazione nella documentazione di Fluent Bit.

> **Important**
>
> In una `Fluent Conf` tipica, le sezioni principali incluse sono `Service`, `Input`, `Filter` e `Output`. Tuttavia, il router di log di Fargate accetta solo:
>
> - Le sezioni `Filter` e `Output`.
> - Una sezione `Parser`.
>
> Se fornisci altre sezioni, queste verranno rifiutate.

Il router di log Fargate gestisce le sezioni `Service` e `Input`. Ha la sezione `Input` seguente, che non può essere modificata e non è necessaria in `ConfigMap`. Tuttavia, è possibile ricavarne informazioni, ad esempio il limite del buffer di memoria e il tag applicato per i log.

```
[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.*
```

Quando crei la `ConfigMap`, ricorda le seguenti regole utilizzate da Fargate per convalidare i campi:

- `[FILTER]`, `[OUTPUT]`, e `[PARSER]` dovrebbero essere specificati sotto ogni chiave corrispondente. Ad esempio: `[FILTER]` deve essere in `filters.conf`. È possibile avere uno o più `[FILTER]` in `filters.conf`. Anche le sezioni `[OUTPUT]` e `[PARSER]` dovrebbero essere specificate sotto le chiavi corrispondenti. Specificando più sezioni `[OUTPUT]`, è possibile instradare i registritha a destinazioni diverse contemporaneamente.
- Fargate convalida le chiavi richieste per ogni sezione. `Name` e `match` sono necessari per ogni `[FILTER]` e `[OUTPUT]`. `Name` e `format` sono necessari per ogni `[PARSER]`. Le chiavi non fanno distinzione tra maiuscole e minuscole.

- Variabili di ambiente come `${ENV_VAR}` non sono ammesse in `ConfigMap`.
- La rientranza deve essere la stessa sia per la direttiva che per la coppia chiave-valore all'interno di ogni `filters.conf`, `output.conf` e `parsers.conf`. Le coppie chiave-valore devono essere rientranti piuttosto che direttive.
- Fargate effettua la convalida in base ai seguenti filtri supportati: `grep`, `parser`, `record_modifier`, `rewrite_tag`, `throttle`, `nest`, `modify` e `kubernetes`.
- Fargate effettua la convalida in base al seguente output supportato: `es`, `firehose`, `kinesis_firehose`, `cloudwatch`, `cloudwatch_logs`, e `kinesis`.
- In `ConfigMap` deve essere fornito almeno un plugin `Output` supportato per abilitare la registrazione di log. `Filter` e `Parser` non sono necessari per abilitarla.

  È possibile anche eseguire Fluent Bit su Amazon EC2 utilizzando la configurazione desiderata per risolvere eventuali problemi derivanti dalla convalida. Crea il `ConfigMap` utilizzando uno degli esempi seguenti.

  > **Important**
  >
  > La registrazione di log Fargate di Amazon EKS non supporta la configurazione dinamica di `ConfigMap`. Eventuali modifiche a `ConfigMap` vengono applicate solo ai nuovi pod. Le modifiche non vengono applicate ai pod esistenti.

  Crea un `ConfigMap` utilizzando l'esempio per la destinazione di log desiderata.

  > **Note**
  >
  > Puoi anche utilizzare Flusso di dati Amazon Kinesis come destinazione dei log. Se utilizzi Flusso di dati Kinesis, assicurati che al ruolo di esecuzione del pod sia stata concessa l'autorizzazione `kinesis:PutRecords`. Per ulteriori informazioni, consulta la sezione Permissions relativa a flusso di dati Amazon Kinesis in Fluent Bit: Official Manual.

Example

CloudWatch

Quando utilizzi CloudWatch, hai a disposizione due opzioni di output:

- Un plug-in di output scritto in C
- Un plug-in di output scritto in Golang

Nell'esempio seguente viene illustrato come utilizzare il plug-in `cloudwatch_logs` per inviare i log a CloudWatch.

a. Salva i contenuti seguenti in un file denominato `aws-logging-cloudwatch-configmap.yaml`. Sostituire *`region-code`* con la Regione AWS in cui si trova il cluster. I parametri in `[OUTPUT]` sono obbligatori.

```
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. Applica il manifesto al cluster.

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

Amazon OpenSearch Service

Per inviare log al Servizio OpenSearch di Amazon, è possibile utilizzare l'output es, ovvero un plugin scritto in C. L'esempio seguente spiega come utilizzare il plugin per inviare log a OpenSearch.

a. Salva i contenuti seguenti in un file denominato `aws-logging-opensearch-configmap.yaml`. Sostituire ogni *`valore di esempio`* con i propri valori.

```
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. Applica il manifesto al cluster.

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

Firehose

Sono disponibili due opzioni di output per l'invio di log a Firehose:

- kinesis_firehose: un plugin di output scritto in C.
- firehose: un plugin di output scritto in Golang.

Nell'esempio seguente viene illustrato come utilizzare il plugin `kinesis_firehose` per inviare log a Firehose.

a. Salva i contenuti seguenti in un file denominato `aws-logging-firehose-configmap.yaml`. Sostituire *`region-code`* con la Regione AWS in cui si trova il cluster.

```
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. Applica il manifesto al cluster.

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. Configurare le autorizzazioni per il ruolo di esecuzione del pod Fargate per inviare log alla destinazione.

a. Eseguire il download della policy IAM per la destinazione sul computer.

Example

CloudWatch

Esegui il download della policy IAM CloudWatch sul computer. Puoi anche consultare la policy su GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/cloudwatchlogs/permissions.json
```

Amazon OpenSearch Service

Esegui il download della policy IAM OpenSearch per il computer. Il documento di policy può essere visualizzato su GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-
fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/
permissions.json
```

Assicurati che il controllo di accesso di OpenSearch Dashboard sia configurato correttamente. `all_access role` in OpenSearch Dashboards deve avere il ruolo di esecuzione del pod Fargate e il ruolo IAM mappati. La stessa mappatura deve essere eseguita per il ruolo `security_manager`. È possibile aggiungere i mapping precedenti selezionando `Menu`, `Security`, `Roles`, quindi selezionare i rispettivi ruoli. Per ulteriori informazioni, consulta Come posso risolvere i problemi relativi ai File di log CloudWatch in modo che vengano trasmessi al mio dominio Amazon ES?.

Firehose

Eseguire il download della policy IAM Firehose sul computer Puoi anche consultare la policy su GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/kinesis-firehose/permissions.json
```

b. Creare una policy IAM utilizzando il file della policy scaricato nella fase precedente.

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document
 file://permissions.json
```

c. Collega la policy IAM al ruolo di esecuzione del pod specificato per il profilo Fargate con il seguente comando. Sostituire *111122223333* con l'ID account. Sostituire *AmazonEKSFargatePodExecutionRole* con il ruolo di esecuzione del pod (per ulteriori informazioni, consultare the section called "Fase 2: creare un ruolo di esecuzione del pod Fargate").

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Supporto filtri Kubernetes

Il filtro Fluent Bit Kubernetes consente di aggiungere metadati Kubernetes ai file di log. Per ulteriori informazioni sul filtro, consultare Kubernetes nella documentazione di Fluent Bit. È possibile applicare un filtro utilizzando l'endpoint del server 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` sono parametri di configurazione di proprietà del servizio e non devono essere specificati. Amazon EKS Fargate popola questi valori.
- `Kube_Meta_Cache_TTL` è il momento in cui Fluent Bit attende il momento in cui comunica al server API i metadati più recenti. Se `Kube_Meta_Cache_TTL` non è specificato, Amazon EKS Fargate aggiunge un valore predefinito di 30 minuti per ridurre il carico sul server API.

Per inviare log di processo Fluent Bit all'account

È possibile inviare i log di processo Fluent Bit ad Amazon CloudWatch utilizzando la seguente `ConfigMap`. La spedizione dei log di processo Fluent Bit a CloudWatch richiede costi aggiuntivi di acquisizione e archiviazione dei log. Sostituire *`region-code`* con la Regione AWS in cui si trova il cluster.

```
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
```

I log si trovano nella stessa Regione AWS in cui si trova il cluster. Il nome del gruppo di log è *my-cluster*`-fluent-bit-logs` e il nome del flusso di log Fluent Bit è `fluent-bit-`*podname*`-`*pod-namespace* .

> **Note**
>
> - I log dei processi vengono spediti solo quando il processo Fluent Bit viene avviato correttamente. Se si verifica un errore durante l'avvio di Fluent Bit, i log di processo vengono mancati. È possibile spedire i log di processo solo a CloudWatch.
> - Per eseguire il debug della spedizione dei log di processo sull'account, è possibile applicare la `ConfigMap` precedente per ottenere i log di processo. Il mancato avvio di Fluent Bit solitamente è dovuto al fatto che la `ConfigMap` non viene analizzata o accettata da Fluent Bit durante l'avvio.

Per interrompere l'invio dei log di processo Fluent Bit

La spedizione dei log di processo Fluent Bit a CloudWatch richiede costi aggiuntivi di acquisizione e archiviazione dei log. Per escludere i registri dei processii in una configurazione `ConfigMap` esistente, effettua le seguenti operazioni.

1. Individuare il gruppo di log CloudWatch creato automaticamente per i log di processo Fluent Bit del cluster Amazon EKS dopo aver abilitato la registrazione di log Fargate. Segue il formato *my-cluster*-fluent-bit-logs.
2. Eliminare i flussi di log CloudWatch esistenti creati per i log di processo di ogni pod nel gruppo di log di CloudWatch.
3. Modifica la ConfigMap e imposta flb_log_cw: "false".
4. Riavviare tutti i pod esistenti nel cluster.

Applicazione di prova

1. Implementare un pod di esempio.
   a. Salva nel tuo computer i seguenti contenuti in un file denominato 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. Applica il file manifesto al cluster.

```
kubectl apply -f sample-app.yaml
```

2. Visualizza i log NGINX utilizzando la/le destinazione/i configurata/e nel ConfigMap.

Considerazioni sulle dimensioni

Si consiglia di pianificare fino a 50 MB di memoria per il router di log. Se prevedi che l'applicazione generi registri a velocità di trasmissione effettiva molto elevata, dovresti pianificarla fino a 100 MB.

Risoluzione dei problemi

Per confermare se la funzionalità di registrazione di log è abilitata o disabilitata per qualche motivo, ad esempio per una `ConfigMap` non valida, e perché questa non è valida, controllare gli eventi pod con `kubectl describe pod` *`pod-name`* . L'output potrebbe includere eventi pod che chiariscono se la registrazione di log è abilitata o meno, ad esempio l'output di esempio seguente.

```
[...]
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
```

Gli eventi di pod sono effimeri, con un periodo di tempo che dipende dalle impostazioni. È inoltre possibile visualizzare le annotazioni di un pod tramite `kubectl describe pod` *`pod-name`* . Nell'annotazione del pod, è possibile sapere se, e per quale motivo, la funzionalità di registrazione di log è abilitata o disabilitata.

Scelta di una tipologia di istanza di nodo Amazon EC2 ottimale

Amazon EC2 offre un'ampia gamma di tipi di istanze per i nodi worker. Ogni tipo di istanza mette a disposizione diverse capacità di calcolo, memoria, archiviazione e rete ed è raggruppata in una famiglia di istanze in base a tali capacità. Per un elenco, consultare Available instance types nella Guida per l'utente di Amazon EC2 per le istanze Linux. Amazon EKS rilascia diverse varianti di Amazon EC2 AMIs per abilitare il supporto. Per assicurarti che il tipo di istanza selezionato sia compatibile con Amazon EKS, prendi in considerazione gli elementi seguenti:

- Al momento AMIs non tutti gli Amazon EKS supportano la `mac` famiglia.

- Amazon EKS Arm e non accelerato AMIs non supportano le famiglieg3, g4inf, ep.
- Amazon EKS accelerato AMIs non supporta le t famiglie ac,hpc,m, e.
- Per le istanze basate su ARM, Amazon Linux 2023 (AL2023) supporta solo i tipi di istanza che utilizzano processori Graviton2 o versioni successive. AL2023 non supporta istanze. A1

Quando scegli i tipi di istanza supportati da Amazon EKS, considera le caratteristiche seguenti per ogni tipo.

Numero di istanze in un gruppo di nodi

In generale, istanze in numero ridotto e di grandi dimensioni sono più convenienti, specialmente disponi di molti DaemonSet. Ogni istanza richiede chiamate API al server API, per cui maggiore è il numero di istanze di cui si dispone, maggiore è il carico sul server API.

Sistema operativo

Esamina i tipi di istanza supportati per Linux, Windows e Bottlerocket. Prima di creare istanze Windows, consultare Deploy Windows nodes on EKS clusters.

Architettura hardware

Hai bisogno di x86 o ARM? Prima di distribuire istanze Arm, esamina Arm Amazon Linux ottimizzato per Amazon EKS. AMIs Hai bisogno di istanze costruite sul sistema Nitro (Linux o Windows) o che hanno una elaborazione Accelerata? Se hai bisogno di funzionalità accelerate, puoi usare Linux solo con Amazon EKS.

Numero massimo di pod

Dal momento che a ogni Pod viene assegnato il proprio indirizzo IP, il numero di indirizzi IP supportati da un tipo di istanza è un fattore importante per determinare il numero di pod che possono essere eseguiti sull'istanza. Per determinare manualmente il numero di Pod supportati da un tipo di istanza, consultare the section called "Per ogni tipo di istanza Amazon EC2, Amazon EKS consiglia un numero massimo di pod".

> **Note**
>
> Se stai usando un'AMI Amazon Linux 2 versione v20220406 o più recente ottimizzata per Amazon EKS, puoi utilizzare un nuovo tipo di istanza senza eseguire l'aggiornamento all'AMI più recente. Per questi AMIs casi, l'AMI calcola automaticamente il max-pods valore necessario se non è elencato nel eni-max-podsfile.txt. Per impostazione

predefinita, i tipi di istanza attualmente in anteprima potrebbero non essere supportati da Amazon EKS. I valori `max-pods` per tali tipi di istanza devono ancora essere aggiunti nel file `eni-max-pods.txt` dell'AMI.

AWS I tipi di istanze Nitro System supportano opzionalmente un numero significativamente maggiore di indirizzi IP rispetto ai tipi di istanze non Nitro System. Tuttavia, non tutti gli indirizzi IP assegnati ad un'istanza sono disponibili per i Pod. Per assegnare un numero significativamente maggiore di indirizzi IP alle istanze, è necessario che la versione `1.9.0` o successiva del componente aggiuntivo CNI di Amazon VPC sia installata nel cluster e che sia configurata in modo appropriato. Per ulteriori informazioni, consulta the section called "Aumentare gli indirizzi IP". Per assegnare il maggior numero di indirizzi IP alle istanze, è necessario che la versione `1.10.1` o successiva del componente aggiuntivo CNI di Amazon VPC sia installata nel cluster e che si implementi il cluster con la famiglia `IPv6`.

Famiglia IP

Per la creazione di un cluster tramite la famiglia `IPv4`, puoi scegliere qualsiasi tipo di istanza supportata, il che consente al cluster di assegnare indirizzi `IPv4` privati ai Pod e servizi. Se tuttavia desideri impiegare la famiglia `IPv6` per il cluster, utilizza i tipi di istanza AWS Nitro System o bare metal. Per le istanze Windows è supportato solo `IPv4`. Il cluster deve eseguire la versione `1.10.1` o successiva del componente aggiuntivo CNI di Amazon VPC. Per ulteriori informazioni sull'utilizzo di `IPv6`, consultare the section called "IPv6".

Versione del componente aggiuntivo CNI di Amazon VPC in uso

La versione più recente del plug-in CNI di Amazon VPC per Kubernetes supporta i seguenti tipi di istanza. Per sfruttare i tipi più recenti di istanza supportati, potrebbe essere necessario aggiornare la versione del componente aggiuntivo CNI di Amazon VPC. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC". L'ultima versione supporta le funzionalità più recenti per l'utilizzo con Amazon EKS. Le versioni precedenti non supportano tutte le funzionalità. Puoi visualizzare le funzionalità supportate dalle diverse versioni in Changelog su GitHub.

La Regione AWS in cui stai creando i nodi

Non tutti i tipi di istanze sono disponibili in tutte le AWS regioni.

Se stai usando gruppi di sicurezza per i Pod

Se stai usando gruppi di sicurezza per i Pod, sono supportati solo tipi specifici di istanza. Per ulteriori informazioni, consulta the section called "Gruppi di sicurezza per pod".

Per ogni tipo di istanza Amazon EC2, Amazon EKS consiglia un numero massimo di pod

Dal momento che a ogni Pod viene assegnato il proprio indirizzo IP, il numero di indirizzi IP supportati da un tipo di istanza è un fattore importante per determinare il numero di Pod che possono essere eseguiti sull'istanza. Amazon EKS fornisce uno script da scaricare ed eseguire per determinare il numero massimo di pod consigliato da Amazon EKS per l'esecuzione su ciascun tipo di istanza. Lo script utilizza gli attributi hardware di ogni istanza e le opzioni di configurazione per determinare il numero massimo di pod. Puoi utilizzare il numero restituito in queste fasi per abilitare funzionalità quali l'assegnazione di indirizzi IP ai pod da una sottorete diversa da quella dell'istanza e l'aumento significativo del numero di indirizzi IP dell'istanza. Se si utilizza un gruppo di nodi gestiti con più tipi di istanza, scegliere un valore adatto a tutti i tipi di istanza.

1. Eseguire il download dello script, utilizzabile per calcolare il numero massimo di pod per ogni tipo di istanza.

   ```
   curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/templates/al2/runtime/max-pods-calculator.sh
   ```

2. Contrassegnare lo script come eseguibile sul computer.

   ```
   chmod +x max-pods-calculator.sh
   ```

3. Esegui lo script, sostituendo *`m5.large`* con il tipo di istanza che desideri implementare e *`1.9.0-eksbuild.1`* con la tua versione del componente aggiuntivo CNI di Amazon VPC. Per determinare la versione del componente aggiuntivo, consulta le procedure di aggiornamento in Assign IPs to Pods with Amazon VPC CNI.

   ```
   ./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
   ```

   Di seguito viene riportato un output di esempio.

   ```
   29
   ```

   È possibile aggiungere le seguenti opzioni allo script per visualizzare il numero massimo di pod supportati quando si utilizzano funzionalità opzionali.

   - `--cni-custom-networking-enabled`: utilizza questa opzione quando desideri assegnare indirizzi IP da una sottorete diversa da quella della propria istanza. Per ulteriori informazioni,

consulta the section called "Rete personalizzata". L'aggiunta di questa opzione allo script precedente, con gli stessi valori di esempio, produce `20`.

- `--cni-prefix-delegation-enabled`: utilizzare questa opzione quando si desidera assegnare un numero significativo di indirizzi IP a ciascuna interfaccia di rete elastica. Questa funzionalità richiede un'istanza Amazon Linux eseguita su Nitro System e la versione `1.9.0` o successiva del componente aggiuntivo CNI di Amazon VPC. Per ulteriori informazioni, consulta the section called "Aumentare gli indirizzi IP". L'aggiunta di questa opzione allo script precedente, con gli stessi valori di esempio, produce `110`.

Puoi inoltre eseguire lo script con l'opzione `--help` per visualizzare tutte le opzioni disponibili.

> **Note**
>
> Lo script del calcolatore di Pod massimi limita il valore restituito a `110` in base alle soglie di scalabilità di Kubernetes e alle impostazioni consigliate. Se il tipo di istanza è superiore a 30 vCPUs, questo limite passa a`250`, un numero basato sui test interni del team di scalabilità di Amazon EKS. Per ulteriori informazioni, consulta il post del blog Amazon VPC CNI plugin increases pods per node limits.

Come viene determinato MaxPods

Il `maxPods` valore finale applicato a un nodo dipende da diversi componenti che interagiscono secondo uno specifico ordine di precedenza. La comprensione di questo ordine consente di evitare comportamenti imprevisti durante la personalizzazione`maxPods`.

Ordine di precedenza (dal più alto al più basso):

1. Applicazione di gruppi di nodi gestiti: quando utilizzi un gruppo di nodi gestiti senza un'AMI personalizzata, Amazon EKS impone `maxPods` un limite ai dati utente del nodo. Per i casi con meno di 30 vCPUs, il limite è. `110` Per i casi con più di 30 vCPUs, il limite è. `250` Questo valore ha la precedenza su qualsiasi altra `maxPods` configurazione, tra cui. `maxPodsExpression`
2. **`maxPods`**configurazione kubelet — Se si imposta `maxPods` direttamente nella configurazione kubelet (ad esempio, tramite un modello di avvio con un'AMI personalizzata), questo valore ha la precedenza su. `maxPodsExpression`

3. nodeadm **`maxPodsExpression`** — Se usi `maxPodsExpression`nel tuo`NodeConfig`, nodeadm valuta l'espressione da calcolare. `maxPods` Questo è efficace solo quando il valore non è già impostato da una fonte con precedenza più alta.
4. Calcolo predefinito basato sull'ENI: se non è impostato nessun altro valore, l'AMI calcola in `maxPods` base al numero di interfacce di rete elastiche e indirizzi IP supportati dal tipo di istanza. È equivalente alla formula. `(number of ENIs × (IPs per ENI − 1)) + 2` Gli `+ 2` account per Amazon VPC CNI sono in `kube-proxy` esecuzione su ogni nodo, che non utilizzano un indirizzo IP Pod.

> **Important**
>
> Se utilizzi un gruppo di nodi gestiti e lo imposti `maxPodsExpression` nel tuo`NodeConfig`, l'applicazione del gruppo di nodi gestiti ha la precedenza sulla tua espressione. Per utilizzare un `maxPods` valore personalizzato con i gruppi di nodi gestiti, devi specificare un'AMI personalizzata nel modello di avvio e impostarla `maxPods` direttamente. Per ulteriori informazioni, consulta the section called “Modelli di avvio”.

Gruppi di nodi gestiti e nodi autogestiti

Con i gruppi di nodi gestiti (senza un'AMI personalizzata), Amazon EKS inserisce il `maxPods` valore nei dati utente bootstrap del nodo. Ciò significa che:

- Il `maxPods` valore è sempre limitato `110` o `250` dipende dalla dimensione dell'istanza.
- Tutto `maxPodsExpression` ciò che configuri viene sovrascritto da questo valore iniettato.
- Per utilizzare un `maxPods` valore diverso, specifica un AMI personalizzato nel modello di avvio e `--use-max-pods false --kubelet-extra-args '--max-pods=`*`my-value`*`'` passalo allo `bootstrap.sh` script. Per alcuni esempi, consulta the section called “Modelli di avvio”.

Con i nodi autogestiti, hai il pieno controllo del processo di bootstrap. Puoi utilizzarlo `maxPodsExpression` nel tuo `NodeConfig` o passare `--max-pods` direttamente a. `bootstrap.sh`

Considerazioni per la modalità automatica di EKS

modalità automatica di EKS limita il numero di pod sui nodi al valore inferiore tra:

- Tappo rigido da 110 pod
- Il risultato del calcolo del numero massimo di pod descritto sopra.

Creazione di nodi con immagini ottimizzate preconfigurate

Puoi implementare i nodi con versioni preconfigurate delle Amazon Machine Image (AMI) ottimizzate per Amazon EKS o con AMI personalizzate quando utilizzi i gruppi di nodi gestiti o i nodi autogestiti. Se stai eseguendo nodi ibridi, consulta the section called "Preparazione del sistema operativo". Per ulteriori informazioni sui vari tipi di AMI ottimizzate per Amazon EKS, consulta il relativo argomento di seguito. Per istruzioni su come creare un'AMI personalizzata, consulta the section called "Crea un'AMI Amazon Linux personalizzata ottimizzata per EKS".

Con Amazon EKS Auto Mode, EKS gestisce l'istanza EC2, inclusa la selezione e l'aggiornamento dell'AMI.

Argomenti

- Guida a EKS AL2 e funzionalità AL2 di transizione accelerata AMIs
- Crea nodi con Amazon Linux ottimizzato AMIs
- Crea nodi con Bottlerocket ottimizzato AMIs
- Creare nodi con AMI ottimizzate per Ubuntu Linux
- Crea nodi con Windows ottimizzato AMIs

Guida a EKS AL2 e funzionalità AL2 di transizione accelerata AMIs

> **Warning**
>
> Amazon EKS ha smesso di pubblicare Amazon Linux 2 (AL2) ottimizzato per EKS il 26 AMIs novembre 2025. AL2023 e Bottlerocket based for AMIs Amazon EKS sono disponibili per tutte le versioni di Kubernetes supportate, inclusa la 1.33 e successive.

AWS terminerà il supporto per EKS AL2 ottimizzato e AL2 accelerato a partire dal 26 novembre 2025. AMIs Sebbene sia possibile continuare a utilizzare EKS AL2 AMIs dopo la data end-of-support (EOS) (26 novembre 2025), EKS non rilascerà più nuove versioni o aggiornamenti di Kubernetes AL2 AMIs, incluse versioni minori, patch e correzioni di bug dopo tale data. Ti consigliamo di eseguire l'aggiornamento ad Amazon Linux 2023 (AL2023) o Bottlerocket: AMIs

- AL2023 consente un secure-by-default approccio con politiche di sicurezza preconfigurate, SELinux in modalità permissiva, modalità IMDSv2 solo abilitata per impostazione predefinita, tempi di avvio ottimizzati e una migliore gestione dei pacchetti per una maggiore sicurezza e prestazioni, ideale per infrastrutture che richiedono personalizzazioni significative come l'accesso diretto a livello di sistema operativo o modifiche estese dei nodi. Per ulteriori informazioni, consulta AL2 023 o visualizza le nostre linee guida dettagliate sulla migrazione all'indirizzo. FAQs the section called "Esegui l'aggiornamento a AL2023"
- Bottlerocket offre una sicurezza avanzata, tempi di avvio più rapidi e una superficie di attacco minore per una maggiore efficienza grazie al design progettato appositamente e ottimizzato per i container, adatto ad approcci nativi per i container con personalizzazioni minime dei nodi. Per saperne di più, consulta Bottlerocket FAQs o visualizza la nostra guida dettagliata alla migrazione all'indirizzo. the section called "Bottlerocket"

In alternativa, puoi farlo the section called "Crea un'AMI Amazon Linux personalizzata ottimizzata per EKS" fino alla data EOS (26 novembre 2025). Inoltre, puoi creare un'AMI personalizzata con un'istanza di base Amazon Linux 2 fino alla data EOS di Amazon Linux 2 (30 giugno 2026).

Migrazione e supporto FAQs

Come posso migrare dalla mia AL2 AMI AL2 a una 023?

Ti consigliamo di creare e implementare un piano di migrazione che includa test approfonditi del carico di lavoro delle applicazioni e procedure di rollback documentate, quindi segui le step-by-step istruzioni contenute nella documentazione ufficiale di EKS per l'aggiornamento da Amazon Linux 2 ad Amazon Linux 2023.

Posso creare un' AL2 AMI personalizzata dopo la data EKS end-of-support (EOS) per EKS optimized AL2 AMIs?

Sebbene consigliamo di passare a EKS ottimizzati AMIs per AL2 023 o Bottlerocket supportati e pubblicati ufficialmente, puoi creare EKS personalizzati AL2 ottimizzati e AL2 accelerati AMIs fino alla data AL2 AMI EOS (26 novembre 2025). In alternativa, è possibile creare un'AMI personalizzata con un'istanza base di Amazon Linux 2 fino alla data di fine del supporto di Amazon Linux 2 (30 giugno 2026). Per step-by-step istruzioni su come creare un'AMI personalizzata AL2 ottimizzata e AL2 accelerata per EKS, consulta Creare un'AMI Amazon Linux personalizzata nella documentazione ufficiale di EKS.

La policy di supporto delle versioni Kubernetes di EKS si applica alle distribuzioni Amazon Linux?

No. La data EOS per l' AL2ottimizzazione e l' AL2accelerazione di EKS AMIs è indipendente dalle tempistiche di supporto standard ed estese per le versioni Kubernetes di EKS. È necessario migrare a AL2 023 o Bottlerocket anche se si utilizza il supporto esteso EKS.

In che modo il passaggio da cgroupv1 a cgroupv2 influisce sulla migrazione?

La community di Kubernetes ha spostato il `cgroupv1` supporto (utilizzato da AL2) in modalità di manutenzione, il che significa che non verranno aggiunte nuove funzionalità e verranno fornite solo le correzioni di sicurezza e i principali bug. Per adottare `cgroupv2` in Kubernetes, è necessario garantire la compatibilità tra sistema operativo, kernel, runtime dei container e componenti Kubernetes. Ciò richiede una distribuzione Linux abilitata `cgroupv2` per impostazione predefinita, come AL2 023, Bottlerocket, Red Hat Enterprise Linux (RHEL) 9+, Ubuntu 22.04+ o Debian 11+. Tali distribuzioni vengono fornite con versioni del kernel pari o superiori alla 5.8, che è il requisito minimo per il supporto di `cgroupv2` in Kubernetes. Per ulteriori informazioni, consultare la pagina About cgroup v2.

Cosa devo fare se ho bisogno di Neuron nella mia AL2 AMI personalizzata?

Non è possibile eseguire le applicazioni complete basate su Neuron in modo nativo su una base. AL2 AMIs Per utilizzare AWS Neuron su un' AL2 AMI, devi containerizzare le tue applicazioni utilizzando un contenitore supportato da Neuron con una distribuzione non AL2 Linux (ad esempio, Ubuntu 22.04, Amazon Linux 2023, ecc.) e quindi distribuire quei contenitori su un'AMI AL2 basata su cui è installato il Neuron Driver (). `aws-neuronx-dkms`

Devo passare a un'istanza base di Amazon Linux 2 nuda dopo la data EKS AL2 AMI EOS (26 novembre 2025)?

Il passaggio a un'istanza base di Amazon Linux 2 non include le ottimizzazioni specifiche, le configurazioni di runtime dei container e le personalizzazioni fornite dalla versione ufficiale di EKS AL2, ottimizzata e accelerata. AL2 AMIs Invece, se devi continuare a utilizzare una soluzione AL2 basata, ti consigliamo di creare un'AMI personalizzata utilizzando le ricette AMI EKS su the section called “Crea un'AMI Amazon Linux personalizzata ottimizzata per EKS” o Amazon EKS AMI Build Specification. Ciò garantisce la compatibilità con i carichi di lavoro esistenti e include aggiornamenti del AL2 kernel fino alla data EOS di Amazon Linux 2 (30 giugno 2026).

Quando si crea un' AL2 AMI personalizzata utilizzando l' GitHub archivio EKS AMI dopo la data EKS AL2 AMI EOS (26 novembre 2025), quale supporto è disponibile per i pacchetti di repository come amzn2-core e amzn2extra-docker?

La ricetta di AMI EKS disponibile alla pagina Amazon EKS AMI Build Specification estrae pacchetti tramite YUM da software Amazon Linux 2 standard, come amzn2-core e amzn2extra-docker. Dopo la data EKS AL2 AMI EOS (26 novembre 2025), questo software continuerà a essere supportato fino alla data più ampia di Amazon Linux 2 EOS (30 giugno 2026). Tenere presente che durante questo periodo il supporto è limitato agli aggiornamenti del kernel, il che significa che sarà necessario gestire e applicare manualmente altri aggiornamenti dei pacchetti, patch di sicurezza e qualsiasi dipendenza non relativa al kernel per mantenere la sicurezza e la compatibilità.

Perché le applicazioni Java che utilizzano versioni precedenti di JDK8 Amazon EKS con AL2 023 potrebbero presentare eccezioni di esaurimento della memoria (OOM) e riavvii dei pod, e come è possibile risolvere questo problema?

Quando vengono eseguite su nodi Amazon EKS con AL2 023, le applicazioni Java che si basavano su versioni precedenti di JDK 8 `jdk8u372` possono causare eccezioni OOM e riavvii dei pod perché la JVM non è compatibile con. `cgroupv2` Questo problema deriva in particolare dall'incapacità della JVM di rilevare i limiti di memoria dei container utilizzando `cgroupv2`, l'impostazione predefinita in Amazon Linux 2023. Di conseguenza, basa l'allocazione della memoria heap sulla memoria totale del nodo anziché sul limite definito dal pod. La causa del problema è la modifica della posizione di archiviazione dei dati relativi al limite di memoria in `cgroupv2`, che causa la lettura errata della memoria disponibile e l'utilizzo di risorse a livello di nodo da parte delle versioni precedenti di Java. Ecco alcune possibili soluzioni:

- Aggiornare la versione del JDK: l'aggiornamento alla versione `jdk8u372` o successive o a una versione del JDK più recente con supporto completo di `cgroupv2` può risolvere questo problema. Per un elenco delle versioni Java compatibili con supporto completo di `cgroupv2`, consultare la pagina About cgroup v2.
- Crea un'AMI personalizzata: se devi continuare a utilizzare una soluzione AL2 basata su AMI, puoi creare un'AMI personalizzata AL2 (fino al 26 novembre 2025) utilizzando the section called "Crea un'AMI Amazon Linux personalizzata ottimizzata per EKS" Amazon EKS AMI Build Specification. Ad esempio, puoi creare un'AMI v1.33 AL2 basata (fino al 26 novembre 2025). Amazon EKS fornirà un servizio AL2 basato AMIs fino alla data AL2 EKS EOS (26 novembre 2025). Dopo la data di fine del supporto (26 novembre 2025), sarà necessario creare la propria AMI.
- Abilita cgroupv1: se devi continuare a utilizzare`cgroupv1`, puoi abilitarlo `cgroupv1` su un'AMI EKS AL2 023. Per abilitare, eseguire `sudo grubby --update-kernel=ALL --`

`args="systemd.unified_cgroup_hierarchy=0"` e riavviare il sistema (ad esempio, EC2 istanza o nodo che esegue Amazon Linux 2023). Ciò modificherà i parametri di avvio del sistema (ad esempio, aggiungendo il parametro di kernel 'systemd.unified_cgroup_hierarchy=0', che indica a systemd di utilizzare la gerarchia legacy `cgroupv1`, alla configurazione GRUB) e abiliterà `cgroupv1`. Tenere presente che, con l'esecuzione di questo comando grubby, si procede alla configurazione del kernel in modo che venga avviato con `cgroupv1` abilitato e `cgroupv2` disabilitato. Per la gestione attiva delle risorse su un nodo viene utilizzata solo una di queste versioni di cgroup. Ciò non equivale all'esecuzione di `cgroupv2` con compatibilità retroattiva per l'API `cgroupv1`.

> **Warning**
>
> Sconsigliamo l'uso continuato di `cgroupv1`. Consigliamo invece di eseguire la migrazione a `cgroupv2`. La community di Kubernetes ha spostato `cgroupv1` il supporto (utilizzato da AL2) in modalità di manutenzione, il che significa che non verranno aggiunte nuove funzionalità o aggiornamenti e verranno fornite solo le correzioni di sicurezza e i principali bug. La rimozione completa del supporto di `cgroupv1` è prevista in una versione futura, anche se non è stata ancora annunciata una data specifica. Se riscontri problemi con`cgroupv1`, non AWS sarà in grado di fornire supporto e ti consigliamo di eseguire l'aggiornamento a. `cgroupv2`

Compatibilità e versioni

Versioni Kubernetes supportate per AL2 AMIs

La versione 1.32 di Kubernetes è l'ultima versione per la quale Amazon EKS verrà rilasciato (AL2 Amazon Linux 2). AMIs Per le versioni di Kubernetes supportate fino alla 1.32, EKS continuerà a rilasciare AL2 AMIs (AL2_ARM_64, _x86_64) e -accelerated (AL2_x86_64_GPU) fino al 26 novembre 2025. AL2 AMIs AL2 Dopo questa data, AL2 EKS AL2 smetterà di AMIs rilasciare versioni ottimizzate e accelerate per tutte le versioni di Kubernetes. Tieni presente che la data EOS per le versioni AL2 ottimizzate e AL2 accelerate per EKS AMIs è indipendente dalle tempistiche di supporto standard ed estese per le versioni Kubernetes di EKS.

Confronto dei driver supportati e delle versioni del kernel Linux per, 023 e Bottlerocket AL2 AL2 AMIs

Componente	AL2 AMI EKS	EKS AL2 023 AMI	EKS Bottlerocket AMI
Compatibilità con il sistema operativo di base	RHEL7/CentOS 7	Fedora/CentOS 9	N/D
Driver in modalità utente CUDA	12.x	12.x, 13.x	12x, 13x
Driver GPU NVIDIA	R570	R580	R570, R580
AWS Driver neuronale	2,20 +	2,20+	2,20+
Kernel Linux	5,10	6.1, 6.12	6.1, 6.12

Per ulteriori informazioni sui driver NVIDIA e sulla compatibilità con CUDA, consulta la documentazione NVIDIA.

AWS Compatibilità di Neuron con AL2 AMIs

A partire dalla versione 2.20 di AWS Neuron, Neuron Runtime (`aws-neuronx-runtime-lib`) utilizzato da EKS AL-based AMIs non supporta più Amazon Linux 2 (). AL2 Il Neuron Driver (`aws-neuronx-dkms`) è ora l'unico pacchetto AWS Neuron che supporta Amazon Linux 2. Ciò significa che non è possibile eseguire le applicazioni basate su Neuron in modo nativo su un'AMI basata AL2. Per configurare Neuron su AL2 023 AMIs, consulta la guida all'installazione di Neuron.AWS

Compatibilità con Kubernetes con AL2 AMIs

La community Kubernetes ha spostato il `cgroupv1` supporto (utilizzato da) in modalità manutenzione. AL2 Ciò significa che non verranno aggiunte nuove funzionalità e verranno forniti solo gli aggiornamenti di sicurezza critici e le correzioni di bug più importanti. Tutte le funzionalità di Kubernetes che si basano su cgroupv2, come MemoryQo S e Enhanced Resource Isolation, non sono disponibili su. AL2 Inoltre, la versione 1.32 di Amazon EKS Kubernetes è stata l'ultima versione supportata. AL2 AMIs Per mantenere la compatibilità con le versioni più recenti di Kubernetes, consigliamo di migrare a AL2 023 o Bottlerocket, che sono abilitati per impostazione predefinita. `cgroupv2`

Compatibilità della versione Linux con AL2 AMIs

Amazon Linux 2 (AL2) è supportato AWS fino alla data end-of-support (EOS) del 30 giugno 2026. Tuttavia, con il AL2 passare del tempo, il supporto da parte della più ampia comunità Linux per nuove applicazioni e funzionalità è diventato più limitato. AL2 AMIs sono basati sul kernel Linux 5.10, mentre AL2 023 utilizza il kernel Linux 6.1. A differenza di AL2 023, AL2 ha un supporto limitato da parte della più ampia comunità Linux. Ciò significa che molti pacchetti e strumenti Linux upstream devono essere sottoposti a backport per funzionare con la vecchia versione AL2 del kernel, alcune funzionalità e miglioramenti della sicurezza di Linux moderni non sono disponibili a causa del kernel più vecchio, molti progetti open source hanno un supporto deprecato o limitato per le versioni precedenti del kernel come la 5.10.

AL2Pacchetti obsoleti non inclusi in 023

Alcuni dei pacchetti più comuni che non sono inclusi o che sono stati modificati nella AL2 versione 023, includono:

- Alcuni pacchetti binari di origine in Amazon Linux 2 non sono più disponibili in Amazon Linux 2023
- Modifiche nel modo in cui Amazon Linux supporta diverse versioni di pacchetti (ad esempio, amazon-linux-extras sistema) in AL2 023
- I pacchetti aggiuntivi per Enterprise Linux (EPEL) non sono supportati nella versione 023 AL2
- Le applicazioni a 32 bit non sono supportate in 023 AL2

Per ulteriori informazioni, vedere Confronto AL2 e AL2 023.

Confronto della convalida FIPS tra AL2, AL2 023 e Bottlerocket

Amazon Linux 2 (AL2), Amazon Linux 2023 (AL2023) e Bottlerocket forniscono supporto per la conformità agli standard federali di elaborazione delle informazioni (FIPS).

- AL2 è certificato secondo FIPS 140-2 e AL2 023 è certificato secondo FIPS 140-3. Per abilitare la modalità FIPS su AL2 023, installa i pacchetti necessari sulla tua EC2 istanza Amazon e segui i passaggi di configurazione utilizzando le istruzioni in Abilita la modalità FIPS su 023. AL2 Per ulteriori informazioni, consulta 023. AL2 FAQs
- Bottlerocket offre varianti dedicate specificamente per FIPS che limitano i componenti del kernel e dello spazio utente all'uso di moduli crittografici che sono stati inviati al programma di convalida FIPS 140-3 Cryptographic Module Validation Program.

Changelog di driver e versioni delle AMI EKS

Per un elenco completo di tutti i componenti AMI EKS e delle relative versioni, consulta le note di rilascio dell'AMI di Amazon EKS su GitHub.

Crea nodi con Amazon Linux ottimizzato AMIs

Amazon Elastic Kubernetes Service (Amazon EKS) fornisce Amazon Machine AMIs Images () specializzate ottimizzate per l'esecuzione di nodi di lavoro Kubernetes. Questi Amazon Linux (AL) ottimizzati per EKS AMIs sono preconfigurati con componenti essenziali, come AWS `kubelet` IAM Authenticator e, per garantire un'integrazione e `containerd` una sicurezza senza interruzioni all'interno dei cluster. Questa guida descrive in dettaglio le versioni AMI disponibili e delinea le opzioni specializzate per l'elaborazione accelerata e le architetture basate su ARM.

Considerazioni

- Tieni traccia degli eventi di sicurezza o di privacy per Amazon Linux in Amazon Linux security center scegliendo la scheda per la versione desiderata. Puoi anche abbonarti al feed RSS applicabile. Gli eventi di sicurezza e privacy includono una panoramica del problema, quali sono i pacchetti interessati e come aggiornare le istanze per risolvere il problema.
- Prima di distribuire un'AMI accelerata o Arm, esamina le informazioni in Amazon Linux accelerato ottimizzato per Amazon EKS e. AMIs the section called “Arm ottimizzato per Amazon EKS Amazon Linux AMIs”
- EC2 `P2`Le istanze Amazon non sono supportate su Amazon EKS perché richiedono la versione 470 o precedente del `NVIDIA` driver.
- Tutti i gruppi di nodi gestiti appena creati nei cluster a partire dalla versione `1.30` più recente utilizzeranno automaticamente AL2 023 come sistema operativo del nodo.

Amazon Linux accelerato ottimizzato per Amazon EKS AMIs

Gli Amazon Linux (AL AMIs) accelerati ottimizzati per Amazon EKS si basano sullo standard Amazon Linux ottimizzato per EKS. AMIs Sono configurate per fungere da immagini facoltative per i nodi Amazon EKS per supportare i carichi di lavoro basati su GPU, Inferentia e Trainium.

Per ulteriori informazioni, consulta the section called “Usa la GPU EKS Linux AMIs”.

Arm ottimizzato per Amazon EKS Amazon Linux AMIs

Le istanze Arm offrono significativi risparmi sui costi per applicazioni aumentabili orizzontalmente e basate su Arm come server Web, microservizi containerizzati, flotte di memorizzazione nella cache e archivi di dati distribuiti. Quando si aggiungono nodi Arm al cluster, esaminare le considerazioni riportate di seguito.

- Se il cluster è stato implementato prima del 17 agosto 2020, è necessario eseguire un aggiornamento una tantum dei manifesti critici dei componenti aggiuntivi del cluster. In questo modo Kubernetes può estrarre l'immagine corretta per ogni architettura hardware utilizzata nel cluster. Per ulteriori informazioni sull'aggiornamento dei componenti aggiuntivi del cluster, consultare the section called "Passaggio 1: preparazione per l'aggiornamento". Se il cluster è stato implementato a partire dal 17 agosto 2020, il CoreDNS, `kube-proxy` e i plug-in della CNI di Amazon VPC per i componenti aggiuntivi di Kubernetes sono già predisposti alla multi-architettura.
- Le applicazioni distribuite nei nodi Arm devono essere compilate per Arm.
- Se li hai DaemonSets implementati in un cluster esistente o desideri distribuirli in un nuovo cluster in cui desideri distribuire anche nodi Arm, verifica che sia DaemonSet possibile eseguirli su tutte le architetture hardware del cluster.
- È possibile eseguire gruppi di nodi Arm e gruppi di nodi x86 nello stesso cluster. In tal caso, prendi in considerazione di implementare immagini del container multi-architettura in un repository del container come Amazon Elastic Container Registry, quindi aggiungi selettori di nodi ai manifesti in modo che Kubernetes sappia in quale architettura hardware può essere implementato un pod. Per ulteriori informazioni, consultare Inviare un'immagine multi-architettura nella Guida per l'utente di Amazon ECR ed il blog post Presentazione di immagini container multi-architettura per Amazon ECR.

Ulteriori informazioni

Per ulteriori informazioni sull'uso di Amazon Linux ottimizzato per Amazon EKS AMIs, consulta le seguenti sezioni:

- Per utilizzare Amazon Linux con gruppi di nodi gestiti, consulta la sezione the section called "Gruppi di nodi gestiti".
- Per avviare nodi Amazon Linux autogestiti, consulta la sezione the section called "Ottieni le ultime novità IDs".
- Per informazioni sulla versione, consulta the section called "Ottenere informazioni sulla versione".

- Per recuperare la versione più recente IDs di Amazon AMIs Linux ottimizzata per Amazon EKS, consulta. the section called "Ottieni le ultime novità IDs"
- Per gli script open source utilizzati per creare applicazioni ottimizzate per Amazon EKS AMIs, consulta. the section called "Crea un'AMI Amazon Linux personalizzata ottimizzata per EKS"

Aggiornamento da Amazon Linux 2 ad Amazon Linux 2023

> **Warning**
>
> Amazon EKS ha smesso di pubblicare Amazon Linux 2 (AL2) ottimizzato per EKS il 26 AMIs novembre 2025. AL2023 e Bottlerocket based for AMIs Amazon EKS sono disponibili per tutte le versioni di Kubernetes supportate, inclusa la 1.33 e successive.

AL2023 è un sistema operativo basato su Linux progettato per fornire un ambiente sicuro, stabile e ad alte prestazioni per le tue applicazioni cloud. È la nuova generazione di Amazon Linux offerta da Amazon Web Services ed è disponibile in tutte le versioni di Amazon EKS supportate.

AL2023 offre diversi miglioramenti rispetto a. AL2 Per un confronto completo, consulta AL2 Comparating and Amazon Linux 2023 nella Amazon Linux 2023 User Guide. Sono stati aggiunti, aggiornati e rimossi diversi pacchetti da AL2. Si consiglia vivamente di testare le applicazioni AL2023 prima dell'aggiornamento. Per un elenco di tutte le modifiche ai pacchetti in AL2023, consulta Modifiche ai pacchetti in Amazon Linux 2023 nelle Note di rilascio di Amazon Linux 2023.

Oltre a queste modifiche, devi tenere presente quanto riportato di seguito:

- AL2023 introduce un nuovo processo di inizializzazione dei nodi `nodeadm` che utilizza uno schema di configurazione YAML. Se utilizzi gruppi di nodi autogestiti o un'AMI con un modello di avvio, ora dovrai fornire esplicitamente metadati del cluster aggiuntivi quando si crea un nuovo gruppo di nodi. Un esempio dei parametri minimi richiesti è come segue, dove ora `apiServerEndpoint`, `certificateAuthority` e il servizio `cidr` sono richiesti:

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
```

```
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

Nel AL2, i metadati di questi parametri sono stati scoperti dalla chiamata `DescribeCluster` API di Amazon EKS. Nel frattempo AL2023, questo comportamento è cambiato poiché la chiamata API aggiuntiva rischia di rallentare durante l'up-up di nodi su larga scala. Questa modifica non ha effetto su di te se utilizzi gruppi di nodi gestiti senza un modello di avvio o se utilizzi Karpenter. Per ulteriori informazioni su `certificateAuthority` e sul servizio `cidr`, consulta `DescribeCluster` nel Riferimento di API Amazon EKS.

- Infatti AL2023, modifica `nodeadm` anche il formato in cui applicare i parametri a `kubelet` per ogni nodo utilizzato. `NodeConfigSpec` Nel AL2, ciò è stato fatto con il `--kubelet-extra-args` parametro. Questo è usato comunemente per aggiungere etichette e taint ai nodi. L'esempio seguente mostra l'applicazione di `maxPods` e `--node-labels` al nodo.

```
---
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
```

- È richiesta la versione CNI di Amazon VPC `1.16.2` o superiore. AL2023
- AL2023 richiede per impostazione `IMDSv2` predefinita. `IMDSv2`presenta diversi vantaggi che aiutano a migliorare il livello di sicurezza. Usa un metodo di autenticazione orientato alla sessione che prevede la creazione di un token segreto in una semplice richiesta HTTP PUT per avviare la sessione. Il token di una sessione può avere una validità compresa tra 1 secondo e 6 ore. Per ulteriori informazioni su come passare da `IMDSv1` a`IMDSv2`, consulta Transizione all'utilizzo di Instance Metadata Service versione 2 e Ottieni IMDSv1 tutti i vantaggi IMDSv2 e disabilita l'intera infrastruttura AWS. Se desideri usare `IMDSv1`, puoi farlo comunque sovrascrivendo manualmente le impostazioni usando le proprietà di avvio dell'opzione dei metadati di istanza.

> **Note**
>
> Ad `IMDSv2` esempio AL2023, il numero di hop predefinito per i gruppi di nodi gestiti può variare:
>
> - Quando non si utilizza un modello di avvio, il valore predefinito è impostato su `1`. Ciò significa che i container non avranno accesso alle credenziali del nodo che usano IMDS. Se necessiti dell'accesso del container alle credenziali del nodo, puoi comunque farlo utilizzando un modello di lancio Amazon EC2 personalizzato.
> - Quando si utilizza un'AMI personalizzata in un modello di avvio, il valore predefinito `HttpPutResponseHopLimit` è impostato su `2`. Puoi sostituire manualmente `HttpPutResponseHopLimit` nel modello di avvio.
>
> In alternativa, puoi utilizzare Amazon EKS Pod Identity per fornire credenziali anziché `IMDSv2`.

- AL2023 presenta la nuova generazione di gerarchia unificata dei gruppi di controllo ()`cgroupv2`. `cgroupv2`viene utilizzato per implementare un runtime del contenitore e da. `systemd` Sebbene includa AL2023 ancora codice che può far funzionare il sistema`cgroupv1`, questa non è una configurazione consigliata o supportata. Questa configurazione verrà completamente rimossa in una delle future release principali di Amazon Linux.
- `eksctl`per il supporto è richiesta `eksctl` una versione `0.176.0` o superiore AL2023.

Per i gruppi di nodi gestiti precedentemente esistenti, puoi eseguire un aggiornamento sul posto o un blue/green aggiornamento a seconda di come stai utilizzando un modello di lancio:

- Se utilizzi un'AMI personalizzata con un gruppo di nodi gestito, puoi eseguire un aggiornamento sul posto scambiando l'ID AMI nel modello di avvio. È necessario assicurarsi che le applicazioni e tutti i dati utente vengano trasferiti a AL2023 First prima di eseguire questa strategia di aggiornamento.
- Se utilizzi gruppi di nodi gestiti con il modello di avvio standard o con un modello di avvio personalizzato che non specifica l'ID AMI, devi eseguire l'aggiornamento utilizzando una blue/green strategia. Un blue/green aggiornamento è in genere più complesso e comporta la creazione di un gruppo di nodi completamente nuovo in cui specificare AL2023 il tipo di AMI. Il nuovo gruppo di nodi dovrà quindi essere configurato con cura per garantire che tutti i dati personalizzati del gruppo di AL2 nodi siano compatibili con il nuovo sistema operativo. Una volta che il nuovo gruppo di nodi è stato testato e convalidato con le applicazioni, i Pod possono essere migrati dal vecchio gruppo

di nodi al nuovo gruppo di nodi. Una volta completata la migrazione, puoi eliminare il vecchio gruppo di nodi.

Se utilizzi Karpenter e desideri utilizzarlo AL2023, dovrai modificare il `EC2NodeClass` `amiFamily` campo con. AL2023 Per impostazione predefinita, Drift è abilitato in Karpenter. Ciò significa che una volta modificato il campo `amiFamily`, Karpenter aggiornerà automaticamente i nodi worker all'AMI più recente, quando disponibile.

Informazioni aggiuntive su nodeadm

Quando utilizzi un'AMI Amazon Linux 2023 ottimizzata per EKS o crei un'AMI Amazon Linux 2023 EKS personalizzata tramite gli script Packer forniti nel repository amazon-eks-ami GitHub ufficiale, dovresti evitare di eseguire esplicitamente nodeadm init all'interno dei dati utente EC2 o come parte della tua AMI personalizzata.

Se desideri generare dati dinamici NodeConfig nei tuoi dati utente, puoi scrivere quella configurazione in un file yaml o json drop-in. `/etc/eks/nodeadm.d` Questi file di configurazione verranno uniti e applicati al nodo quando nodeadm init verrà avviato automaticamente più avanti nel processo di avvio. Esempio:

```
cat > /etc/eks/nodeadm.d/additional-node-labels.yaml << EOF
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  kubelet:
    flags:
      - --node-labels=foo=bar
EOF
```

Amazon Linux 2023 ottimizzato per EKS esegue AMIs automaticamente nodeadm init in due fasi tramite servizi systemd separati: nodeadm-config viene eseguito prima dell'esecuzione dei dati utente, mentre nodeadm-run viene eseguito dopo. Il servizio nodeadm-config stabilisce le configurazioni di base per containerd e kubelet prima dell'esecuzione dei dati utente. Il servizio nodeadm-run esegue determinati demoni di sistema e completa tutte le configurazioni finali dopo l'esecuzione dei dati dell'utente. Se il comando nodeadm init viene eseguito un'altra volta, tramite dati utente o un'AMI personalizzata, potrebbe violare le ipotesi sull'ordine di esecuzione, portando a risultati imprevisti, tra cui configurazioni errate. ENIs

Recupero delle informazioni sulla versione dell'AMI Amazon Linux

Le versioni delle AMI Amazon Linux ottimizzate per Amazon EKS si basano sulla versione di Kubernetes e la data di rilascio dell'AMI è nel formato seguente:

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

Ogni rilascio dell'AMI include varie versioni di kubelet, del kernel Linux e di containerd. L'AMI accelerata include anche varie versioni del driver NVIDIA. Puoi trovare queste informazioni sulle versioni in Changelog su GitHub.

Recupera le AMI Amazon Linux consigliate IDs

Durante la distribuzione dei nodi, è possibile specificare un ID per un'Amazon Machine Image (AMI) preconfigurata e ottimizzata per Amazon EKS. Per recuperare un ID AMI adatto alla configurazione desiderata, interroga l'API AWS Systems Manager Parameter Store. L'utilizzo di questa API elimina la necessità di cercare manualmente le AMI ottimizzate per Amazon EKS IDs. Per ulteriori informazioni, consulta GetParameter. Il principale IAM che usi deve disporre dell'autorizzazione IAM `ssm:GetParameter` per recuperare i metadati dell'AMI ottimizzata per Amazon EKS.

È possibile recuperare l'ID immagine dell'ultima versione consigliata dell'AMI Amazon Linux ottimizzata per Amazon EKS con il comando seguente, che utilizza il parametro secondario `image_id`. Apportare le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:

- Sostituisci <kubernetes-version> con una versione supportata da Amazon EKS.
- Sostituisci *ami-type* con una delle seguenti opzioni. Per informazioni sui tipi di EC2 istanze Amazon, consulta Tipi di EC2 istanze Amazon.
  - Utilizzalo *amazon-linux-2023/x86_64/standard* per istanze x86 basate su Amazon Linux 2023 (AL2023).
  - Utilizzalo *amazon-linux-2023/arm64/standard* per AL2 023 istanze ARM, come AWS le istanze basate su Graviton.
  - Da utilizzare *amazon-linux-2023/x86_64/nvidia* per le ultime AL2 023 istanze basate su NVIDIA approvate. x86
  - Da utilizzare *amazon-linux-2023/arm64/nvidia* per le ultime istanze AL2 023 basate su NVIDIA approvate. arm64

  - Da utilizzare *amazon-linux-2023/x86_64/neuron* per le ultime istanze AL2 AWS 023 Neuron.
- Sostituisci `<region-code>` con una AWS regione supportata da Amazon EKS per la quale desideri l'ID 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
```

Ecco un comando esemplificativo dopo la sostituzione dei segnaposto.

```
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
```

Di seguito viene riportato un output di esempio.

```
ami-1234567890abcdef0
```

Crea un'AMI Amazon Linux personalizzata ottimizzata per EKS

> **Warning**
>
> Amazon EKS ha smesso di pubblicare Amazon Linux 2 (AL2) ottimizzato per EKS il 26 AMIs novembre 2025. AL2023 e Bottlerocket based for AMIs Amazon EKS sono disponibili per tutte le versioni di Kubernetes supportate, inclusa la 1.33 e successive.

Amazon EKS fornisce script di build open source nel repository Amazon EKS AMI Build Specification che puoi usare per visualizzare le configurazioni`kubelet`, il runtime, AWS IAM Authenticator per Kubernetes e creare da zero la tua AMI basata su AL.

Questo repository contiene lo script di bootstrap specializzato e lo strumento nodeadm per 023 che viene eseguito all'avvio. AL2 AL2 Questi script configurano i dati del certificato dell'istanza, l'endpoint del piano di controllo (control-plane), il nome del cluster e altro ancora. Gli script sono considerati la fonte di verità per le build AMI ottimizzate per Amazon EKS, quindi puoi seguire GitHub il repository per monitorare le modifiche al nostro. AMIs

Quando si crea una creazione personalizzata AMIs con Eks Optimized AMIs come base, non è consigliato né supportato l'esecuzione di un aggiornamento del sistema operativo (ad es. `dnf upgrade`) o aggiorna uno qualsiasi dei pacchetti Kubernetes o GPU inclusi in EKS-Optimized, poiché ciò rischia di compromettere la compatibilità AMIs dei componenti. Se si esegue l'aggiornamento del sistema operativo o dei pacchetti inclusi nei pacchetti ottimizzati per EKS AMIs, si consiglia di eseguire test approfonditi in un ambiente di sviluppo o di gestione temporanea prima di passare alla produzione.

Quando si creano istanze GPU personalizzate AMIs , si consiglia di creare istanze personalizzate separate AMIs per ogni tipo di istanza, generazione e famiglia di istanze che verrà eseguita. La versione accelerata ottimizzata per EKS installa AMIs in modo selettivo driver e pacchetti in fase di esecuzione in base alla generazione e alla famiglia di istanze sottostanti. Per ulteriori informazioni, consulta gli script EKS AMI per l'installazione e il runtime.

Prerequisiti

- Installa la AWS CLI
- Installa HashiCorp Packer v1.9.4+
- Installare GNU Make

Guida introduttiva

Questa guida rapida mostra i comandi per creare un'AMI personalizzata nel tuo AWS account. Per ulteriori informazioni sulle configurazioni disponibili per personalizzare l'AMI, consultare le variabili del modello alla pagina Amazon Linux 2023.

Prerequisiti

Installare il plugin Amazon richiesto. Esempio:

```
packer plugins install github.com/hashicorp/amazon
```

Passaggio 1. Configurare l'ambiente

Clonare o creare un fork del repository Amazon EKS AMI ufficiale. Esempio:

```
git clone https://github.com/awslabs/amazon-eks-ami.git
cd amazon-eks-ami
```

Verificare che Packer sia installato:

```
packer --version
```

Passaggio 2. Creazione di un'AMI personalizzata

Di seguito sono riportati alcuni comandi di esempio per varie AMIs personalizzazioni.

AL2 AMI NVIDIA di base:

```
make k8s=1.31 os_distro=al2 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=560 \
  enable_efa=true
```

AL2AMI NVIDIA 023 di base:

```
make k8s=1.31 os_distro=al2023 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=560 \
  enable_efa=true
```

AMI Neuron 023 conforme a STIG: AL2

```
make k8s=1.31 os_distro=al2023 \
  enable_accelerator=neuron \
  enable_fips=true \
  source_ami_id=ami-0abcd1234efgh5678 \
  kms_key_id=alias/aws-stig
```

Dopo aver eseguito questi comandi, Packer eseguirà le seguenti operazioni: * Avvia un' EC2 istanza Amazon temporanea. * Installazione di componenti, driver e configurazioni Kubernetes. * Crea l'AMI nel tuo AWS account.

L'output previsto dovrebbe essere simile al seguente:

```
==> 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
```

Fase 3. Visualizzare i valori predefiniti

Eseguire il seguente comando per visualizzare i valori predefiniti e le opzioni aggiuntive:

```
make help
```

Crea nodi con Bottlerocket ottimizzato AMIs

Bottlerocket è una distribuzione Linux open source sponsorizzata e supportata da. AWS Bottlerocket è progettato appositamente per ospitare carichi di lavoro in container. Con Bottlerocket, puoi migliorare la disponibilità delle implementazioni containerizzate e ridurre i costi operativi automatizzando gli aggiornamenti dell'infrastruttura dei container. Bottlerocket include solo il software essenziale per eseguire i container, che migliora l'utilizzo delle risorse, riduce le minacce alla sicurezza e riduce i costi di gestione. L'AMI Bottlerocket include e AWS IAM `containerd` `kubelet` Authenticator. Oltre che dai gruppi di nodi gestiti e di nodi autogestiti, Bottlerocket è supportato anche da Karpenter.

Vantaggi

L'utilizzo di Bottlerocket con il cluster Amazon EKS presenta i seguenti vantaggi:

- Tempi di attività maggiori con costi operativi inferiori e minore complessità di gestione: Bottlerocket necessita di un minore utilizzo di risorse, tempi di avvio più brevi ed è meno vulnerabile alle minacce alla sicurezza rispetto ad altre distribuzioni Linux. L'ingombro ridotto di Bottlerocket aiuta a ridurre i costi utilizzando meno risorse di archiviazione, elaborazione e rete.
- Maggiore sicurezza grazie agli aggiornamenti automatici del sistema operativo – gli aggiornamenti a Bottlerocket vengono applicati come una singola unità che può essere ripristinata, se necessario. Ciò elimina il rischio di aggiornamenti danneggiati o non riusciti che possono lasciare il sistema in uno stato inutilizzabile. Con Bottlerocket, gli aggiornamenti di sicurezza possono essere applicati automaticamente non appena sono disponibili con un impatto minimo e possono essere ripristinati in caso di guasti.
- Supporto premium: le AWS versioni fornite di Bottlerocket su Amazon EC2 sono coperte dagli stessi piani di AWS supporto che coprono anche AWS servizi come Amazon, Amazon EKS e EC2 Amazon ECR.

Considerazioni

Quando utilizzi Bottlerocket per il tuo tipo di AMI, considera quanto segue:

- Bottlerocket supporta EC2 istanze Amazon con `x86_64` processori. `arm64`
- Bottlerocket supporta EC2 istanze Amazon con. GPUs Per ulteriori informazioni, consulta the section called "Usa la GPU EKS Linux AMIs".
- Le immagini Bottlerocket non includono un server SSH o una shell. Puoi utilizzare metodi di out-of-band accesso per consentire SSH. Questi approcci consentono di utilizzare il container dell'amministratore e saltare alcune fasi di configurazione di bootstrap con i dati utente. Per ulteriori informazioni, consulta le seguenti sezioni in Bottlerocket OS su: GitHub
  - Esplorazione
  - Container amministratore
  - Impostazioni Kubernetes
- Bottlerocket utilizza diversi tipi di container:
  - Per impostazione predefinita, è abilitato un container di controllo. Questo contenitore esegue l'agente AWS Systems Manager che puoi utilizzare per eseguire comandi o avviare sessioni di shell su istanze Amazon EC2 Bottlerocket. Per ulteriori informazioni, vedere Configurazione di Session Manager nella Guida per l'utente di AWS Systems Manager.
  - Se viene fornita una chiave SSH durante la creazione del gruppo di nodi, viene abilitato un container dell'amministratore. Consigliamo di utilizzare il container amministratore solo per scenari di sviluppo e test. Non è consigliabile utilizzarlo in ambienti di produzione. Per ulteriori informazioni, consulta Container amministratore su GitHub.

Ulteriori informazioni

Per ulteriori informazioni sull'uso di Bottlerocket ottimizzato per Amazon EKS AMIs, consulta le seguenti sezioni:

- Per i dettagli su Bottlerocket, consulta la documentazione di Bottlerocket.
- Per le risorse informative sulla versione, consulta the section called "Ottenere informazioni sulla versione".
- Per utilizzare Bottlerocket con gruppi di nodi gestiti, consulta the section called "Gruppi di nodi gestiti".
- Per avviare i nodi autogestiti di Bottlerocket, consulta the section called "Portabottiglie".

- Per recuperare la versione più recente IDs del Bottlerocket AMIs ottimizzato per Amazon EKS, consulta. the section called "Ottenimento degli ID della versione più recente"
- Per informazioni dettagliate sul supporto per la conformità, consulta la sezione the section called "Supporto alla conformità".

Recupero delle informazioni sulla versione delle AMI Bottlerocket

Ogni versione delle AMI Bottlerocket include varie versioni di kubelet, del kernel Bottlerocket e di containerd. Le versioni dell'AMI accelerata includono anche varie versioni del driver NVIDIA. È possibile trovare queste informazioni sulla versione nell'argomento relativo al sistema operativo OS della documentazione di Bottlerocket. Da questa pagina, andare all'argomento secondario Version Information applicabile.

A volte la documentazione di Bottlerocket può essere in ritardo rispetto alle versioni disponibili su GitHub. È possibile trovare un elenco di modifiche relative alle versioni più recenti alla pagina releases su GitHub.

Recupero degli ID AMI Bottlerocket consigliati

Durante la distribuzione dei nodi, è possibile specificare un ID per un'Amazon Machine Image (AMI) preconfigurata e ottimizzata per Amazon EKS. Per recuperare un ID AMI adatto alla configurazione desiderata, interroga l'API AWS Systems Manager Parameter Store. L'utilizzo di tale parametro consente di evitare la ricerca manuale degli ID AMI ottimizzata per Amazon EKS. Per ulteriori informazioni, consulta Ottenere parametro. Il principale IAM che usi deve disporre dell'autorizzazione IAM `ssm:GetParameter` per recuperare i metadati dell'AMI ottimizzata per Amazon EKS.

È possibile recuperare l'ID immagine della versione più recente dell'AMI Bottlerocket ottimizzata per Amazon EKS consigliata con il seguente comando AWS CLI, che utilizza il parametro secondario `image_id`. Apportare le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:

- Sostituisci la *`versione Kubernetes`* con una versione della piattaforma supportata.
- Sostituisci *`-flavor`* con una delle seguenti opzioni.
  - Rimuovi *`-flavor`* per le varianti senza GPU.
  - Utilizza *`-nvidia`* per le varianti abilitate per la GPU.
  - Utilizza *`-fips`* per le varianti compatibili con FIPS.
- Sostituisci *`architettura`* con una delle seguenti opzioni.

  - Utilizza *x86_64* per le istanze basate su x86.
  - Utilizza *arm64* per le istanze ARM.
- Sostituisci *codice regione* con una regione AWS supportata da Amazon EKS per la quale si desidera l'ID 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
```

Ecco un comando esemplificativo dopo la sostituzione dei segnaposto.

```
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
```

Di seguito viene riportato un output di esempio:

```
ami-1234567890abcdef0
```

Soddisfare i requisiti di conformità con Bottlerocket

Bottlerocket è conforme alle raccomandazioni definite da varie organizzazioni:

- È stato definito un Benchmark CIS per Bottlerocket. In una configurazione predefinita, l'immagine Bottlerocket presenta la maggior parte dei controlli richiesti dal profilo di configurazione CIS Level 1. Puoi implementare i controlli necessari per un profilo di configurazione CIS di livello 2. Per ulteriori informazioni, consulta Validating Amazon EKS optimized Bottlerocket AMI against the CIS Benchmark sul blog AWS.
- Il set di funzionalità ottimizzato e la superficie di attacco ridotta fanno sì che le istanze Bottlerocket richiedano meno configurazioni per soddisfare i requisiti PCI DSS. Il Benchmark CIS per Bottlerocket è una risorsa eccellente per le linee guida in materia di hardening e supporta i requisiti per gli standard di configurazione sicuri ai sensi del requisito PCI DSS 2.2. Puoi sfruttare Fluent Bit per supportare i requisiti di registrazione degli audit a livello di sistema operativo ai sensi del requisito PCI DSS 10.2. AWS pubblica periodicamente nuove istanze Bottlerocket (con patch) per soddisfare i requisiti PCI DSS 6.2 (per la versione 3.2.1) e il requisito 6.3.3 (per la versione 4.0).

- Bottlerocket è una funzionalità idonea alla normativa HIPAA autorizzata per l'uso con carichi di lavoro regolamentati sia per Amazon EC2 che per Amazon EKS. Per ulteriori informazioni, consulta HIPAA Eligible Services Reference.
- Sono disponibili AMI Bottlerocket preconfigurate per utilizzare moduli crittografici convalidati FIPS 140-3. Sono inclusi il modulo crittografico Amazon Linux 2023 Kernel Crypto API e il modulo crittografico AWS-LC. Per ulteriori informazioni, consulta the section called "Bottlerocket FIPS AMIs".

Prepara i tuoi nodi di lavoro con FIPS con Bottlerocket FIPS AMIs

La pubblicazione 140-3 del Federal Information Processing Standard (FIPS) è uno standard di sicurezza dei governi degli Stati Uniti e del Canada che specifica i requisiti di sicurezza previsti per i moduli crittografici che proteggono informazioni sensibili. Bottlerocket semplifica l'adesione a FIPS offrendo un kernel FIPS. AMIs

Questi AMIs sono preconfigurati per utilizzare moduli crittografici convalidati FIPS 140-3. Ciò include il modulo crittografico Amazon Linux 2023 Kernel Crypto API e il modulo crittografico AWS-LC.

L'uso di Bottlerocket FIPS AMIs rende i nodi di lavoro «pronti per FIPS» ma non automaticamente «conformi a FIPS». Per ulteriori informazioni, vedere Federal Information Processing Standard (FIPS) 140-3.

Considerazioni

- Se il cluster usa sottoreti isolate, l'endpoint FIPS di Amazon ECR potrebbe non essere accessibile. Questo può causare il fallimento del bootstrap del nodo. Assicurati che la configurazione di rete consenta l'accesso agli endpoint FIPS necessari. Per ulteriori informazioni, consulta Accedere a una risorsa tramite un endpoint VPC di risorse nella Guida. AWS PrivateLink
- Se il cluster utilizza una sottorete con PrivateLink, il recupero delle immagini avrà esito negativo perché gli endpoint FIPS di Amazon ECR non sono disponibili tramite. PrivateLink

Creare un gruppo di nodi gestiti con un'AMI Bottlerocket FIPS

L'AMI FIPS Bottlerocket è disponibile in quattro varianti per supportare i tuoi carichi di lavoro:

- `BOTTLEROCKET_x86_64_FIPS`
- `BOTTLEROCKET_ARM_64_FIPS`

- `BOTTLEROCKET_x86_64_NVIDIA_FIPS`
- `BOTTLEROCKET_ARM_64_NVIDIA_FIPS`

Per creare un gruppo di nodi gestiti con un'AMI FIPS Bottlerocket, seleziona il tipo di AMI applicabile durante il processo di creazione. Per ulteriori informazioni, consulta the section called "Crea".

Per ulteriori informazioni sulla scelta delle varianti compatibili con FIPS, consulta the section called "Ottenimento degli ID della versione più recente".

Disabilita l'endpoint FIPS per le regioni non supportate AWS

I FIPS Bottlerocket AMIs sono supportati direttamente negli Stati Uniti d'America, comprese le regioni AWS GovCloud (Stati Uniti). Per AWS le regioni in cui AMIs sono disponibili ma non supportate direttamente, puoi comunque utilizzarle creando un gruppo di nodi gestito con un modello di avvio. AMIs

L'AMI FIPS Bottlerocket si basa sull'endpoint FIPS Amazon ECR durante il bootstrap, che generalmente non è disponibile al di fuori degli Stati Uniti. Per utilizzare l'AMI per il suo kernel FIPS in AWS regioni in cui non è disponibile l'endpoint FIPS Amazon ECR, procedi nel seguente modo per disabilitare l'endpoint FIPS:

1. Crea un nuovo file di configurazione con il contenuto seguente o incorpora il contenuto nel file di configurazione esistente.

```
[default]
use_fips_endpoint=false
```

1. Codifica il contenuto del file nel formato Base64.
2. Nel campo `UserData` del modello di avvio, aggiungi la seguente stringa codificata usando il formato TOML:

```
[settings.aws]
config = "<your-base64-encoded-string>"
```

Per altre impostazioni, consulta la descrizione delle impostazioni su di Bottlerocket. GitHub

Ecco un esempio di `UserData` in un modello di avvio:

```
[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>
```

Per ulteriori informazioni sulla creazione di un modello di avvio con i dati utente, consulta the section called "Modelli di avvio".

Creare nodi con AMI ottimizzate per Ubuntu Linux

Canonical ha collaborato con Amazon EKS per creare AMI del nodo utilizzabili nei cluster.

Canonical offre un'immagine del sistema operativo del nodo Kubernetes integrata per lo scopo. Questa immagine Ubuntu ridotta è ottimizzata per Amazon EKS e include il kernel AWS personalizzato, sviluppato in collaborazione con AWS. Per ulteriori informazioni, consultare la pagina Ubuntu on Amazon Elastic Kubernetes Service (EKS) e the section called "Ubuntu Linux". Per informazioni sul supporto, consultare la sezione Third-party software delle domande frequenti del Supporto AWS Premium.

Crea nodi con Windows ottimizzato AMIs

Windows Amazon EKS AMIs optimized si basa su Windows Server 2019, Windows Server 2022 e Windows Server 2025. Sono configurate per fungere da immagine di base per i nodi Amazon EKS. Per impostazione predefinita, AMIs includono i seguenti componenti:

- kubelet
- kube-proxy
- AWS IAM Authenticator per Kubernetes
- csi-proxy
- containerd

> **Note**
>
> É possibile monitorare gli eventi di sicurezza o privacy per Windows Server con la Microsoft Security Update Guide.

Offerte AMIs Amazon EKS ottimizzate per contenitori Windows nelle seguenti varianti:

- AMI Windows Server, versione 2019 (Core) ottimizzata per Amazon EKS
- AMI Windows Server, versione 2019 (Full) ottimizzata per Amazon EKS
- AMI Core per Windows Server 2022 ottimizzata per Amazon EKS
- AMI completa per Windows Server 2022 ottimizzata per Amazon EKS
- AMI Core per Windows Server 2025 ottimizzata per Amazon EKS
- AMI completa per Windows Server 2025 ottimizzata per Amazon EKS

> **Important**
>
> - L'AMI Windows Server 20H2 (Core) ottimizzata per Amazon EKS è obsoleta. Non verranno rilasciate nuove versioni di questa AMI.
> - Per garantire di disporre degli ultimi aggiornamenti di sicurezza AMIs per impostazione predefinita, Amazon EKS mantiene Windows ottimizzato negli ultimi 4 mesi. Ogni nuova AMI sarà disponibile per 4 mesi dal momento della pubblicazione iniziale. Dopo questo periodo, AMIs le versioni più vecchie vengono rese private e non sono più accessibili. Consigliamo di utilizzare le versioni più recenti AMIs per evitare vulnerabilità di sicurezza e perdere l'accesso alle versioni più vecchie AMIs che hanno raggiunto la fine del periodo di validità del supporto. Sebbene non possiamo garantire di poter fornire l'accesso a AMIs ciò che è stato reso privato, puoi richiedere l'accesso inviando un ticket a AWS Support.

Calendario dei rilasci

Nella tabella seguente sono elencate le date di rilascio e di fine supporto delle versioni Windows per Amazon EKS. Se una data di fine del supporto è vuota, la versione è ancora supportata.

Versione Windows	Rilascio Amazon EKS	Fine del supporto Amazon EKS
Windows Server 2025 Core	27/01/2026	
Windows Server 2025 completo	27/01/2026	
Windows Server 2022 Core	17/10/2022	
Windows Server 2022 Full	17/10/2022	
Windows Server 20H2 Core	12/08/2021	09/08/2022
Windows Server 2004 Core	19/08/2020	14/12/2021
Windows Server 2019 Core	07/10/2019	
Windows Server, versione 2019 (Full)	07/10/2019	
Windows Server, versione 1909 (Core)	07/10/2019	08/12/2020

Parametri di configurazione dello script di bootstrap

Quando crei un nodo Windows, sul nodo è presente uno script che consente la configurazione di diversi parametri. A seconda della configurazione, questo script può essere presente sul nodo in una posizione simile a: `C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1`. È possibile specificare valori di parametri personalizzati specificandoli come argomenti dello script di bootstrap. Ad esempio, puoi aggiornare i dati utente nel modello di avvio. Per ulteriori informazioni, consulta the section called "Dati utente Amazon EC2".

Lo script include i seguenti parametri della riga di comando:

- `-EKSClusterName`: specifica il nome del cluster Amazon EKS a cui il nodo worker deve essere aggiunto.
- `-KubeletExtraArgs`: specifica argomenti aggiuntivi per `kubelet` (facoltativo).
- `-KubeProxyExtraArgs`: specifica argomenti aggiuntivi per `kube-proxy` (facoltativo).

- `-APIServerEndpoint`: specifica l'endpoint del server API del cluster Amazon EKS (facoltativo). Valido solo se usato con `-Base64ClusterCA`. Bypassa la chiamata a `Get-EKSCluster`.
- `-Base64ClusterCA`: specifica il contenuto CA del cluster codificato base64 (facoltativo). Valido solo se usato con `-APIServerEndpoint`. Bypassa la chiamata a `Get-EKSCluster`.
- `-DNSClusterIP`: sostituisce l'indirizzo IP da utilizzare per le query DNS all'interno del cluster (facoltativo). L'impostazione predefinita è `10.100.0.10` o `172.20.0.10` in base all'indirizzo IP dell'interfaccia primaria.
- `-ServiceCIDR`: sostituisce l'intervallo di indirizzi IP del servizio Kubernetes da cui vengono indirizzati i servizi del cluster. L'impostazione predefinita è `172.20.0.0/16` o `10.100.0.0/16` in base all'indirizzo IP dell'interfaccia primaria.
- `-ExcludedSnatCIDRs`— Un elenco di cose da escludere `IPv4` CIDRs da Source Network Address Translation (SNAT). Ciò significa che l'IP privato del pod indirizzabile nel VPC non verrebbe tradotto nell'indirizzo IP dell'indirizzo `IPv4` primario dell'interfaccia di rete elastica (ENI) dell'istanza per il traffico in uscita. Per impostazione predefinita, viene aggiunto il CIDR `IPv4` del VPC per il nodo Windows di Amazon EKS. La specificazione CIDRs di questo parametro esclude inoltre quanto specificato. CIDRs Per ulteriori informazioni, consulta the section called "Traffico in uscita".

Oltre ai parametri della riga di comando, è anche possibile specificare alcuni parametri delle variabili di ambiente. Quando si specifica un parametro della riga di comando, questo ha la precedenza sulla rispettiva variabile di ambiente. Le variabili di ambiente devono essere definite come ambito di computer (o sistema) perché lo script bootstrap leggerà solo le variabili con ambito di computer.

Lo script tiene conto delle seguenti variabili di ambiente:

- `SERVICE_IPV4_CIDR`: per la definizione, fare riferimento al parametro della riga di comando `ServiceCIDR`.
- `EXCLUDED_SNAT_CIDRS`: dovrebbe essere una stringa separata da virgole. Fare riferimento al parametro della riga di comando `ExcludedSnatCIDRs` per la definizione.

Supporto dell'autenticazione gMSA

I pod Windows di Amazon EKS consentono diversi tipi di autenticazione dell'account del servizio gestito del gruppo (gMSA).

- Amazon EKS supporta le identità di dominio di Active Directory per l'autenticazione. Per ulteriori informazioni sulla gMSA aggiunta a un dominio, consulta la sezione Autenticazione di Windows su Amazon EKS Windowspods sul blog. AWS
- Amazon EKS offre un plug-in che consente ai nodi non-domain-joined Windows di recuperare le credenziali gMSA con un'identità utente portatile. Per ulteriori informazioni su gMSA senza dominio, consulta Domainless Windows Authentication for Amazon EKS Windowspods sul blog. AWS

Immagini di container memorizzate nella cache

Amazon EKS Windows Optimized AMIs dispone di alcune immagini di container memorizzate nella cache per il `containerd` runtime. Le immagini dei container vengono memorizzate nella cache durante la creazione personalizzata AMIs utilizzando componenti di compilazione gestiti da Amazon. Per ulteriori informazioni, consulta the section called "Utilizzo del componente di compilazione gestito da Amazon".

Le seguenti immagini di container memorizzate nella cache sono per il runtime `containerd`:

- `amazonaws.com/eks/pause-windows`
- `mcr.microsoft.com/windows/nanoserver`
- `mcr.microsoft.com/windows/servercore`

Ulteriori informazioni

Per ulteriori informazioni sull'uso di Windows ottimizzato per Amazon EKS AMIs, consulta le seguenti sezioni:

- Per informazioni dettagliate sull'esecuzione di carichi di lavoro su Windows accelerato ottimizzato per Amazon EKS AMIs, consulta. the section called "Configura le AMI GPU Windows"
- Per utilizzare Windows con gruppi di nodi gestiti, consulta the section called "Gruppi di nodi gestiti".
- Per avviare nodi Windows autogestiti, consulta the section called "Windows".
- Per informazioni sulla versione, consulta the section called "Ottenere informazioni sulla versione".
- Per recuperare la versione più recente IDs di Windows ottimizzata per Amazon EKS AMIs, consultathe section called "Scarica le ultime IDs".
- Per utilizzare Amazon EC2 Image Builder per creare finestre personalizzate ottimizzate AMIs per Amazon EKS, consulta. the section called "Compilazioni personalizzate"

- Per le best practice, consulta Amazon EKS optimized Windows AMI management nella Guida alle best practice di EKS.

Creazione di nodi Windows Server 2022 autogestiti con **`eksctl`**

È possibile utilizzare il seguente `test-windows-2022.yaml` come riferimento per creare nodi Windows Server 2022 autogestiti. Sostituisci ogni *`valore esemplificativo`* con i tuoi valori.

> **Note**
>
> Per eseguire nodi Windows Server 2022 autogestiti, è necessario utilizzare versione `eksctl` 0.116.0 o una versione successiva.

```
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
```

È quindi possibile creare i gruppi di nodi utilizzando il comando seguente.

```
eksctl create cluster -f test-windows-2022.yaml
```

Recupero delle informazioni sulla versione delle AMI Windows

Questo argomento elenca le versioni di Windows ottimizzato per Amazon EKS AMIs e le versioni corrispondenti di kubelet, containerd e csi-proxy.

I metadati di un'AMI ottimizzata per Amazon EKS, incluso l'ID AMI per ogni variante, possono essere recuperati a livello di programmazione. Per ulteriori informazioni, consulta the section called "Scarica le ultime IDs".

AMIs sono aggiornati in base alla versione di Kubernetes e alla data di rilascio dell'AMI nel seguente formato:

```
k8s_major_version.k8s_minor_version-release_date
```

> **Note**
>
> I gruppi di nodi gestiti di Amazon EKS supportano le versioni di novembre 2022 e successive di Windows AMIs.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, puoi iscriverti al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/nodes/eks-ami-versions-windows.adoc.atom
```

AMI Core per Windows Server 2025 ottimizzata per Amazon EKS

Le tabelle seguenti elencano le versioni correnti e precedenti dell'AMI Core Windows Server 2025 ottimizzata per Amazon EKS.

Example

Kubernetes versione 1.35

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.35-2026-01-22	1.35.0	2.1.6	1.2.1	

AMI completa per Windows Server 2025 ottimizzata per Amazon EKS

Le tabelle seguenti elencano le versioni attuali e precedenti dell'AMI completa Windows Server 2025 ottimizzata per Amazon EKS.

Example

Kubernetes versione 1.35

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
`1.35-2026-01-22`	`1.35.0`	`2.1.6`	`1.2.1`	

AMI Windows Server 2022 Core ottimizzata per Amazon EKS

Le tabelle seguenti riportano le versioni correnti e precedenti dell'AMI Windows Server 2022 Core ottimizzata per Amazon EKS.

Example

Kubernetes versione 1.35

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
`1.35-2026-01-22`	`1.35.0`	`2.1.6`	`1.2.1`	

Kubernetes versione 1.34

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
`1.34-2026.01.22`	`1.34.2`	`2.1.6`	`1.2.1`	`containerd` è stato

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
				aggiornato a `2.1.6`.
`1.34-2025.12.15`	`1.34.2`	`2.1.4`	`1.2.1`	
`1.34-2025.11.14`	`1.34.1`	`2.1.4`	`1.2.1`	
`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`	

Kubernetes versione 1.33

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
`1.33-2026.01.22`	`1.33.5`	`1.7.30`	`1.2.1`	`containerd` è stato aggiornato a `1.7.30`.
`1.33-2025.12.15`	`1.33.5`	`1.7.28`	`1.2.1`	
`1.33-2025.11.14`	`1.33.5`	`1.7.28`	`1.2.1`	
`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`	Modifica dei log del plugin

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
				GMSA in Eventi di 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	

Kubernetes versione 1.32

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.32-2026.01.22	1.32.9	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.32-2025.12.15	1.32.9	1.7.28	1.2.1	
1.32-2025.11.14	1.32.9	1.7.28	1.2.1	
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	Modifica dei log del plugin

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
				GMSA in Eventi di 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	containerd è stato aggiornato a 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	1.32.0	1.7.20	1.1.3	

Kubernetes versione 1.31

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.31-2026.01.22	1.31.13	1.7.30	1.2.1	containerd è stato

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
				aggiornato a `1.7.30`.
`1.31-2025.12.15`	`1.31.13`	`1.7.28`	`1.2.1`	
`1.31-2025.11.14`	`1.31.13`	`1.7.28`	`1.2.1`	
`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`	Modifica dei log del plugin GMSA in Eventi di 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`	`containerd` è stato aggiornato a `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`	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per 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	

Kubernetes versione 1.30

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.30-2026.01.22	1.30.14	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.30-2025.12.15	1.30.14	1.7.28	1.2.1	
1.30-2025.11.21	1.30.14	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	containerd è stato aggiornato a 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	containerd è stato aggiornato a 1.7.20.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per 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	Include patch per CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	containerd è stato aggiornato a 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes versione 1.29

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2026.01.22	1.29.15	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.29-2025.12.15	1.29.15	1.7.28	1.2.1	
1.29-2025.11.14	1.29.15	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	containerd è stato aggiornato a 1.7.27.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	containerd è stato aggiornato a 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	Include patch per 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	Include patch per CVE-2024-5321 .

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	containerd è stato aggiornato a 1.7.11. kubelet è stato aggiornato a 1.29.3.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd è stato aggiornato a 1.6.28. CNI ricompilato e csi-proxy con uso di 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
`1.29-2024.02.06`	`1.29.0`	`1.6.25`	`1.1.2`	Risolto un bug a causa del quale l'immagine in pausa veniva eliminata erroneamente dal processo `kubelet` di rimozione di oggetti inutili (garbage collection).

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2024.01.11	1.29.0	1.6.18	1.1.2	Windows Update autonomo escluso KB5034439su Windows Server 2022 Core. AMIs La KB si applica solo alle installazioni Windows con una partizione e WinRE separata, che non sono incluse in nessuno dei nostri Windows ottimizzati per Amazon EKS. AMIs

Kubernetes versione 1.28

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2025.11.14	1.28.15	1.7.28	1.2.1	
1.28-2025.10.18	1.28.15	1.7.28	1.2.1	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2025.09.13	1.28.15	1.7.28	1.2.1	Modifica dei log del plugin GMSA in Eventi di 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	containerd è stato aggiornato a 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	containerd è stato aggiornato a 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	Include patch per CVE-2024-9042 .

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	containerd è stato aggiornato a 1.7.11.
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	containerd è stato aggiornato a 1.6.28. kubelet è stato aggiornato a 1.28.8.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd è stato aggiornato a 1.6.25. CNI ricompilato e csi-proxy con uso di 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2024.01.11	1.28.5	1.6.18	1.1.2	Windows Update autonomo escluso KB5034439su Windows Server 2022 Core. AMIs La KB si applica solo alle installazioni Windows con una partizione e WinRE separata, che non sono incluse in nessuno dei nostri Windows ottimizzati per Amazon EKS. AMIs
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	Include patch per CVE-2023-5528 .

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd è stato aggiornato a 1.6.18. Aggiunte nuove variabili di ambiente dello script di bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Risolto un avviso di sicurezza in kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

AMI Windows Server 2022 Full ottimizzata per Amazon EKS

Le tabelle seguenti riportano le versioni correnti e precedenti dell'AMI Windows Server 2022 Full ottimizzata per Amazon EKS.

Example

Kubernetes versione 1.35

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.35-2026-01-22	1.35.0	2.1.6	1.2.1	

Kubernetes versione 1.34

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.34-2026.01.22	1.34.2	2.1.6	1.2.1	containerd è stato aggiornato a 2.1.6.
1.34-2025.12.15	1.34.2	2.1.4	1.2.1	
1.34-2025.11.14	1.34.1	2.1.4	1.2.1	
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	

Kubernetes versione 1.33

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.33-2026.01.22	1.33.5	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.33-2025.12.15	1.33.5	1.7.28	1.2.1	
1.33-2025.11.14	1.33.5	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	

Kubernetes versione 1.32

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.32-2026.01.22	1.32.9	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.32-2025.12.15	1.32.9	1.7.28	1.2.1	
1.32-2025.11.14	1.32.9	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	Aggiornamento di containrd a 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	

Kubernetes versione 1.31

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.31-2026.01.22	1.31.13	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.31-2025.12.15	1.31.13	1.7.28	1.2.1	
1.31-2025.11.14	1.31.13	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di Windows
1.31-2025.08.18	1.31.11	1.7.27	1.2.1	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	containerd è stato aggiornato a 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	Include patch per 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	

Kubernetes versione 1.30

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.30-2026.01.22	1.30.14	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.30-2025.12.15	1.30.14	1.7.28	1.2.1	
1.30-2025.11.21	1.30.14	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	containerd è stato aggiornato a 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	contianerd è stato aggiornato a 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	Include patch per 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	containerd è stato aggiornato a 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes versione 1.29

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2026.01.22	1.29.15	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.29-2025.12.15	1.29.15	1.7.28	1.2.1	
1.29-2025.11.14	1.29.15	1.7.28	1.2.1	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Modifica dei log del plugin GMSA in Eventi di 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	containerd è stato aggiornato a 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	containerd è stato aggiornato a 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Include patch per 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	Include patch per 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	containerd è stato aggiornato a 1.7.11. kubelet è stato aggiornato a 1.29.3.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd è stato aggiornato a 1.6.28. CNI ricompilato e csi-proxy con uso di 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	Risolto un bug a causa del quale l'immagine in pausa veniva eliminata erroneamente dal processo kubelet di rimozione di oggetti inutili (garbage collection).
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes versione 1.28

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2025.11.14	1.28.15	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	containerd è stato aggiornato a 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	containerd è stato aggiornato a 1.7.20.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per 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	Include patch per CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	containerd è stato aggiornato a 1.7.11.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	containerd è stato aggiornato a 1.6.28. kubelet è stato aggiornato a 1.28.8.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd è stato aggiornato a 1.6.25. CNI ricompilato e csi-proxy con uso di 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	
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	Include patch per CVE-2023-5528 .

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd è stato aggiornato a 1.6.18. Aggiunte nuove variabili di ambiente dello script di bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Risolto un avviso di sicurezza in kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

AMI Windows Server, versione 2019 (Core) ottimizzata per Amazon EKS

Le tabelle seguenti riportano le versioni correnti e precedenti dell'AMI Windows Server 2019 Core ottimizzata per Amazon EKS.

Example

Kubernetes versione 1.35

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.35-2026-01-22	1.35.0	2.1.6	1.2.1	

Kubernetes versione 1.34

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.34-2026.01.22	1.34.2	2.1.6	1.2.1	containerd è stato aggiornato a 2.1.6.
1.34-2025.12.15	1.34.2	2.1.4	1.2.1	
1.34-2025.11.14	1.34.1	2.1.4	1.2.1	
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	

Kubernetes versione 1.33

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.33-2026.01.22	1.33.5	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.33-2025.12.15	1.33.5	1.7.28	1.2.1	
1.33-2025.11.14	1.33.5	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	

Kubernetes versione 1.32

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.32-2026.01.22	1.32.9	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.32-2025.12.15	1.32.9	1.7.28	1.2.1	
1.32-2025.11.14	1.32.9	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	containerd è stato aggiornato a 1.7.27.
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	

Kubernetes versione 1.31

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.31-2026.01.22	1.31.13	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.31-2025.12.15	1.31.13	1.7.28	1.2.1	
1.31-2025.11.14	1.31.13	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di Windows

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	containerd è stato aggiornato a 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	Include patch per 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	

Kubernetes versione 1.30

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.30-2026.01.22	1.30.14	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.30-2025.12.15	1.30.14	1.7.28	1.2.1	
1.30-2025.11.21	1.30.14	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di Windows
1.30-2025.08.18	1.30.14	1.7.27	1.2.1	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	containerd è stato aggiornato a 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	containerd è stato aggiornato a 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	Include patch per 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	containerd è stato aggiornato a 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes versione 1.29

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2026.01.22	1.29.15	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.29-2025.12.15	1.29.15	1.7.28	1.2.1	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2025.11.14	1.29.15	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	containerd è stato aggiornato a 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	containerd è stato aggiornato a 1.7.20.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per 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	Include patch per 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	containerd è stato aggiornato a 1.7.11. kubelet è stato aggiornato a 1.29.3.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd è stato aggiornato a 1.6.28. CNI ricompilato e csi-proxy con uso di 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	Risolto un bug a causa del quale l'immagine in pausa veniva eliminata erroneamente dal processo kubelet di rimozione di oggetti inutili (garbage collection).
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes versione 1.28

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2025.11.14	1.28.15	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	1.7.20	1.2.1	containerd è stato aggiornato a 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	containerd è stato aggiornato a 1.7.20.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per 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	Include patch per CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	containerd è stato aggiornato a 1.7.11.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	containerd è stato aggiornato a 1.6.28. kubelet è stato aggiornato a 1.28.8.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd è stato aggiornato a 1.6.25. CNI ricompilato e csi-proxy con uso di 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	Include patch per CVE-2023-5528 .

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd è stato aggiornato a 1.6.18. Aggiunte nuove variabili di ambiente dello script di bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Risolto un avviso di sicurezza in kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

AMI Windows Server, versione 2019 (Full) ottimizzata per Amazon EKS

Le tabelle seguenti riportano le versioni correnti e precedenti dell'AMI Windows Server 2019 Full ottimizzata per Amazon EKS.

Example

Kubernetes versione 1.35

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.35-2026-01-22	1.35.0	2.1.6	1.2.1	

Kubernetes versione 1.34

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.34-2026.01.22	1.34.2	2.1.6	1.2.1	containerd è stato aggiornato a 2.1.6.
1.34-2025.12.15	1.34.2	2.1.4	1.2.1	
1.34-2025.11.14	1.34.1	2.1.4	1.2.1	
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	

Kubernetes versione 1.33

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.33-2026.01.22	1.33.5	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.33-2025.12.15	1.33.5	1.7.28	1.2.1	
1.33-2025.11.14	1.33.5	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	

Kubernetes versione 1.32

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.32-2026.01.22	1.32.9	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.32-2025.12.15	1.32.9	1.7.28	1.2.1	
1.32-2025.11.14	1.32.9	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	containerd è stato aggiornato a 1.7.27.
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	

Kubernetes versione 1.31

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.31-2026.01.22	1.31.13	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.31-2025.12.15	1.31.13	1.7.28	1.2.1	
1.31-2025.11.14	1.31.13	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di Windows

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	containerd è stato aggiornato a 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	Include patch per 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	

Kubernetes versione 1.30

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.30-2026.01.22	1.30.14	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.30-2025.12.15	1.30.14	1.7.28	1.2.1	
1.30-2025.11.21	1.30.14	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di Windows
1.30-2025.08.18	1.30.14	1.7.27	1.2.1	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	containerd è stato aggiornato a 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	containerd è stato aggiornato a 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	Include patch per 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	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	containerd è stato aggiornato a 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Kubernetes versione 1.29

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2026.01.22	1.29.15	1.7.30	1.2.1	containerd è stato aggiornato a 1.7.30.
1.29-2025.12.15	1.29.15	1.7.28	1.2.1	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2025.11.14	1.29.15	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	containerd è stato aggiornato a 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	containerd è stato aggiornato a 1.7.20.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per 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	Include patch per 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	containerd è stato aggiornato a 1.7.11. kubelet è stato aggiornato a 1.29.3.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd è stato aggiornato a 1.6.28. CNI ricompilato e csi-proxy con uso di 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	Risolto un bug a causa del quale l'immagine in pausa veniva eliminata erroneamente dal processo kubelet di rimozione di oggetti inutili (garbage collection).
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes versione 1.28

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2025.11.14	1.28.15	1.7.28	1.2.1	
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	Modifica dei log del plugin GMSA in Eventi di 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	containerd è stato aggiornato a 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	containerd è stato aggiornato a 1.7.20.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
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	Include patch per 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	Include patch per CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	containerd è stato aggiornato a 1.7.11.

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	containerd è stato aggiornato a 1.6.28. kubelet è stato aggiornato a 1.28.8.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd è stato aggiornato a 1.6.25. CNI ricompilato e csi-proxy con uso di 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	Include patch per CVE-2023-5528 .

Versione AMI	Versione kubelet	Versione containerd	Versione csi-proxy	Note di rilascio
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd è stato aggiornato a 1.6.18. Aggiunte nuove variabili di ambiente dello script di bootstrap (SERVICE_IPV4_CIDR e EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Risolto un avviso di sicurezza in kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Recupera l'AMI Microsoft Windows consigliata IDs

Durante la distribuzione dei nodi, è possibile specificare un ID per un'Amazon Machine Image (AMI) preconfigurata e ottimizzata per Amazon EKS. Per recuperare un ID AMI adatto alla configurazione desiderata, interroga l'API AWS Systems Manager Parameter Store. L'utilizzo di questa API elimina la necessità di cercare manualmente le AMI ottimizzate per Amazon EKS IDs. Per ulteriori informazioni, consulta GetParameter. Il principale IAM che usi deve disporre dell'autorizzazione IAM `ssm:GetParameter` per recuperare i metadati dell'AMI ottimizzata per Amazon EKS.

È possibile recuperare l'ID immagine della versione consigliata dell'AMI Windows ottimizzata per Amazon EKS più recente con il comando seguente che utilizza il parametro secondario `image_id`. Apportare le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:

- Sostituisci *`release`* con una delle seguenti opzioni.
  - Utilizzare *`2025`* per Windows Server 2025.
  - Utilizzare *`2022`* per Windows Server 2022.
  - Utilizzare *`2019`* per Windows Server 2019.
- Sostituisci *`installation-option`* con una delle seguenti opzioni. Per ulteriori informazioni, consulta Cos'è l'opzione di installazione Server Core in Windows Server.
  - Utilizzare *`Core`* per un'installazione minima con una superficie di attacco più piccola.
  - *`Full`*Da utilizzare per includere l'esperienza desktop di Windows.
- Sostituisci *`kubernetes-version`* con una versione della piattaforma supportata.
- Sostituisci *`region-code`* con una AWS regione supportata da Amazon EKS per la quale desideri l'ID 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
```

Ecco un comando esemplificativo dopo la sostituzione dei segnaposto.

```
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
```

Di seguito viene riportato un output di esempio.

```
ami-1234567890abcdef0
```

Creazione di un'AMI Windows personalizzata con Image Builder

Puoi usare EC2 Image Builder per creare Windows personalizzati ottimizzati per Amazon EKS AMIs con una delle seguenti opzioni:

- Utilizzo di un'AMI Windows ottimizzata per Amazon EKS come base
- Utilizzo del componente di compilazione gestito da Amazon

In entrambi i casi, è necessario creare la propria ricetta di Image Builder. Per ulteriori informazioni, consulta la sezione Creazione di una nuova versione di una ricetta di immagine nella Guida per l'utente di Image Builder.

> **Important**
>
> I seguenti componenti gestiti da Amazon per `eks` includono patch per `CVE-2024-5321`.
>
> - `1.28.2` e versioni successive
> - `1.29.2` e versioni successive
> - `1.30.1` e versioni successive
> - Tutte le versioni per Kubernetes 1.31 e versioni successive

Utilizzo di un'AMI Windows ottimizzata per Amazon EKS come base

Questa opzione è il metodo consigliato per creare Windows AMIs personalizzati. Le finestre ottimizzate per Amazon EKS AMIs che forniamo vengono aggiornate più frequentemente rispetto al componente di build gestito da Amazon.

1. Inizia a creare la tua ricetta di Image Builder.
   a. Aprire la console EC2 Image Builder in /imagebuilder. https://console.aws.amazon.com
   b. Nel riquadro di navigazione di sinistra, scegli Ricette di immagini.
   c. Scegli Crea ricetta di immagine.
2. Nella sezione Dettagli della ricetta, inserisci un nome e una versione.
3. Specifica l'ID dell'AMI Windows ottimizzata per Amazon EKS nella sezione Immagine di base.
   a. Scegli Inserisci un ID dell'AMI personalizzata.
   b. Recupera l'ID dell'AMI per la versione del sistema operativo Windows richiesta. Per ulteriori informazioni, consulta the section called "Scarica le ultime IDs".
   c. Inserisci l'ID dell'AMI personalizzata. Se l'ID AMI non viene trovato, assicurati che la AWS regione per l'ID AMI corrisponda alla AWS regione mostrata nella parte superiore destra della console.

4. (Facoltativo) Per ottenere gli ultimi aggiornamenti di sicurezza, aggiungi il componente `update-windows` nella sezione Componenti di compilazione.
   a. Dall'elenco a discesa a destra della casella di ricerca Trova componenti per nome scegli Gestito da Amazon.
   b. Nella sezione Trova componenti per nome, inserisci `update-windows`.
   c. Seleziona la casella di controllo del risultato della ricerca **`update-windows`**. Il componente include le patch di Windows più recenti per il sistema operativo.
5. Completa gli input rimanenti della ricetta di immagine con le configurazioni richieste. Per ulteriori informazioni, consulta la sezione Creazione di una nuova versione di una ricetta di immagine (console) nella Guida per l'utente di Image Builder.
6. Scegli Crea ricetta.
7. Utilizza la nuova ricetta di immagine in una pipeline di immagini nuova o esistente. Una volta che la pipeline di immagini viene eseguita correttamente, l'AMI personalizzata verrà elencata come immagine di output e sarà pronta per l'uso. Per ulteriori informazioni, vedere Creare una pipeline di immagini utilizzando la procedura guidata della console EC2 Image Builder.

Utilizzo del componente di compilazione gestito da Amazon

Quando non è possibile utilizzare un'AMI Windows ottimizzata per Amazon EKS come base, puoi utilizzare il componente di compilazione gestito da Amazon. Questa opzione potrebbe essere in ritardo rispetto alle versioni di Kubernetes supportate più recenti.

1. Inizia a creare la tua ricetta di Image Builder.
   a. Aprire la console EC2 Image Builder in /imagebuilder. https://console.aws.amazon.com
   b. Nel riquadro di navigazione di sinistra, scegli Ricette di immagini.
   c. Scegli Crea ricetta di immagine.
2. Nella sezione Dettagli della ricetta, inserisci un nome e una versione.
3. Determina quale opzione utilizzerai per creare la tua AMI personalizzata nella sezione Immagine di base:
   - Seleziona immagini gestite: scegli Windows come Sistema operativo (OS) per le immagini. Quindi scegli una delle seguenti opzioni per Origine dell'immagine.
     - Quick start (gestito da Amazon): nell'elenco a discesa Nome immagine, seleziona una versione Windows Server supportata da Amazon EKS. Per ulteriori informazioni, consulta the section called "Windows".

- Immagini di mia proprietà: per Nome immagine, seleziona l'ARN della tua immagine con la tua licenza. L'immagine fornita non deve avere componenti Amazon EKS già installati.

- Inserisci l'ID dell'AMI personalizzata: per l'ID dell'AMI, inserisci l'ID della tua AMI con la tua licenza. L'immagine fornita non deve avere componenti Amazon EKS già installati.

4. Nella sezione Componenti di compilazione - Windows, esegui le seguenti operazioni:
   a. Dall'elenco a discesa a destra della casella di ricerca Trova componenti per nome scegli Gestito da Amazon.
   b. Nella casella di ricerca Trova componenti per nome, inserisci `eks`.
   c. Seleziona la casella di controllo accanto al risultato della ricerca **`eks-optimized-ami-windows`**, anche se il risultato restituito potrebbe non essere la versione desiderata.
   d. Nella casella di ricerca Trova componenti per nome, inserisci `update-windows`.
   e. Seleziona la casella di controllo accanto al risultato della ricerca update-windows. Il componente include le patch di Windows più recenti per il sistema operativo.
5. Nella sezione Componenti selezionati, esegui le seguenti operazioni:
   a. Scegli Opzioni di controllo delle versioni per **`eks-optimized-ami-windows`**.
   b. Scegli Specifica la versione del componente.
   c. Nel campo Versione del componente, inserisci, sostituendolo ***`version`*** con una versione di *`version.x`* Kubernetes supportata. L'immissione *`x`* di una parte del numero di versione indica di utilizzare la versione più recente del componente che è anche in linea con la parte della versione definita in modo esplicito. Presta attenzione all'output della console, in quanto ti avviserà se la versione desiderata è disponibile come componente gestito. Tieni presente che le versioni di Kubernetes più recenti potrebbero non essere disponibili per il componente di compilazione. Per ulteriori informazioni sulle versioni disponibili, consulta the section called "Recupero di informazioni sulle versioni dei componenti di `eks-optimized-ami-windows`".
6. Completa gli input rimanenti della ricetta di immagine con le configurazioni richieste. Per ulteriori informazioni, consulta la sezione Creazione di una nuova versione di una ricetta di immagine (console) nella Guida per l'utente di Image Builder.
7. Scegli Crea ricetta.
8. Utilizza la nuova ricetta di immagine in una pipeline di immagini nuova o esistente. Una volta che la pipeline di immagini viene eseguita correttamente, l'AMI personalizzata verrà elencata come immagine di output e sarà pronta per l'uso. Per ulteriori informazioni, vedere Creare una pipeline di immagini utilizzando la procedura guidata della console EC2 Image Builder.

Recupero di informazioni sulle versioni dei componenti di **`eks-optimized-ami-windows`**

È possibile recuperare informazioni specifiche relative a ciò che viene installato con ciascun componente. Ad esempio, puoi verificare quale versione di `kubelet` è installata. I componenti sono sottoposti a test funzionali sulle versioni del sistema operativo Windows supportate da Amazon EKS. Per ulteriori informazioni, consulta the section called "Calendario dei rilasci". Altre eventuali versioni del sistema operativo Windows non elencate come supportate o che hanno raggiunto la fine del supporto potrebbero non essere compatibili con il componente.

1. Aprire la console EC2 Image Builder in /imagebuilder. https://console.aws.amazon.com
2. Nel riquadro di navigazione a sinistra, scegli Componenti.
3. Dall'elenco a discesa a destra della casella di ricerca Trova componenti per nome, modifica la selezione Di mia proprietà con Quick start (gestito da Amazon).
4. Nella sezione Trova componenti per nome, inserire `eks`.
5. (Facoltativo) Se stai utilizzando una versione recente, ordina la colonna Versione in ordine decrescente selezionandola due volte.
6. Scegli il collegamento **`eks-optimized-ami-windows`** con la versione desiderata.

La Descrizione nella pagina risultante mostra le informazioni specifiche.

Enable node auto repair and investigate node health issues

L'integrità del nodo si riferisce allo stato operativo e alla capacità di un nodo di eseguire efficacemente i carichi di lavoro. Un nodo integro mantiene la connettività prevista, dispone di risorse sufficienti e può eseguire correttamente i pod senza interruzioni. Per informazioni su come ottenere dettagli sui nodi, consulta the section called "Visualizzazione dello stato dei nodi" e the section called "Ottenere i log dei nodi".

Per aiutare a mantenere i nodi integri, Amazon EKS offre l'agente di monitoraggio dei nodi e la riparazione automatica dei nodi.

> **Important**
>
> L'agente di monitoraggio dei nodi e la riparazione automatica dei nodi sono disponibili solo su Linux. Queste funzionalità non sono disponibili su Windows.

Agente di monitoraggio del nodo

L'agente di monitoraggio dei nodi legge automaticamente i log dei nodi per rilevare determinati problemi di integrità. Analizza i log dei nodi per rilevare i guasti e mostra varie informazioni sullo stato dei nodi worker. Sui nodi worker è applicato un `NodeCondition` dedicato per ciascuna categoria di problemi rilevati, come problemi di archiviazione e di rete. Le descrizioni dei problemi di salute rilevati sono disponibili nella dashboard di osservabilità. Per ulteriori informazioni, consulta the section called "Problemi di integrità dei nodi".

L'agente di monitoraggio dei nodi è incluso come funzionalità per tutti i cluster in modalità automatica Amazon EKS. Per altri tipi di cluster, è possibile aggiungere l'agente di monitoraggio come componente aggiuntivo di Amazon EKS. Per ulteriori informazioni, consulta the section called "Creare un componente aggiuntivo".

Riparazione automatica del nodo

La riparazione automatica dei nodi è una funzionalità aggiuntiva che monitora continuamente lo stato dei nodi, reagendo automaticamente ai problemi rilevati e sostituendo i nodi quando possibile. Ciò aiuta la disponibilità complessiva del cluster con un intervento manuale minimo. Se un controllo dell'integrità ha esito negativo, il nodo è automaticamente isolato in modo che non siano pianificati nuovi pod sul nodo.

Di per sé, la riparazione automatica dei nodi può reagire alle condizioni `Ready` di `kubelet` e di tutti gli oggetti del nodo che sono eliminati manualmente. Se abbinato all'agente di monitoraggio dei nodi, la riparazione automatica dei nodi può reagire a più condizioni che altrimenti non sarebbero rilevate. Queste condizioni aggiuntive includono `KernelReady`, `NetworkingReady` e `StorageReady`.

Questo recupero automatico dei nodi risolve automaticamente i problemi intermittenti dei nodi, come la mancata adesione al cluster, i kubelet che non rispondono e l'aumento degli errori dell'acceleratore (dispositivo). La maggiore affidabilità aiuta a ridurre i tempi di inattività delle applicazioni e a migliorare le operazioni del cluster. Per impostazione predefinita, la riparazione automatica dei nodi non ripara automaticamente i nodi per determinate condizioni come`DiskPressure`, `MemoryPressurePIDPressure`, e gli errori dello strumento di diagnostica o monitoraggio DCGM (NVIDIA Data Center GPU Manager). Queste condizioni spesso indicano problemi relativi al comportamento dell'applicazione, alla configurazione del carico di lavoro o ai limiti delle risorse piuttosto che errori a livello di nodo, il che rende difficile determinare un'azione di riparazione predefinita appropriata. Tuttavia, è possibile personalizzare questo comportamento abilitando azioni `nodeRepairConfigOverrides` di riparazione automatiche per queste condizioni in base

al caso d'uso. Amazon EKS attende 10 minuti prima di agire su `AcceleratedHardwareReady` `NodeConditions` e 30 minuti per tutte le altre condizioni.

Inoltre, i gruppi di nodi gestiti disabiliteranno automaticamente le riparazioni dei nodi per motivi di sicurezza in due scenari. Tutte le operazioni di riparazione precedentemente in corso continueranno per entrambe le situazioni.

- Se è stato attivato uno spostamento zonale per il cluster tramite Application Recovery Controller (ARC), tutte le operazioni di riparazione successive sono interrotte.
- Se il gruppo di nodi ha più di cinque nodi e più del 20% dei nodi del gruppo di nodi non sono integri, le operazioni di riparazione sono interrotte.

È possibile abilitare la riparazione automatica dei nodi durante la creazione o la modifica di un gruppo di nodi gestito.

- Quando si utilizza la console Amazon EKS, attivare la casella di controllo Abilita riparazione automatica dei nodi per il gruppo di nodi gestito. Per ulteriori informazioni, consulta the section called "Crea".
- Quando si utilizza la AWS CLI, `--node-repair-config enabled=true` aggiungere il comando `eks create nodegroup` o `eks update-nodegroup-config`.
- Per un esempio `eksctl` `ClusterConfig` che utilizza un gruppo di nodi gestito con riparazione automatica del nodo, vedi 44-node-repair.yaml on. GitHub

Amazon EKS offre un controllo più granulare sul comportamento di riparazione automatica del nodo attraverso quanto segue:

- `maxUnhealthyNodeThresholdCount` e `maxUnhealthyNodeThresholdPercentage`
  - Questi campi consentono di specificare una soglia di conteggio o percentuale di nodi non integri, oltre la quale le azioni di riparazione automatica dei nodi saranno interrotte. Ciò fornisce un maggiore controllo sul "raggio di impatto" delle riparazioni automatiche dei nodi.
  - È possibile impostare il conteggio assoluto o la percentuale assoluta, ma non entrambi.
- `maxParallelNodesRepairedCount` e `maxParallelNodesRepairedPercentage`
  - Tali campi consentono di specificare il numero massimo di nodi che possono essere riparati contemporaneamente o in parallelo, espresso come conteggio o percentuale di tutti i nodi non integri. In questo modo è possibile controllare in modo più preciso il ritmo delle sostituzioni dei nodi.

  - Come per la soglia non integra, è possibile impostare il conteggio assoluto o la percentuale assoluta, ma non entrambi.
- `nodeRepairConfigOverrides`
  - Si tratta di una struttura complessa che consente di impostare sostituzioni granulari per azioni di riparazione specifiche. Tali sostituzioni controllano l'azione di riparazione e il tempo di ritardo della riparazione prima che un nodo sia considerato idoneo per la riparazione.
  - I campi specifici di tale struttura sono:
    - `nodeMonitoringCondition`: la condizione non integra segnalata dall'agente di monitoraggio dei nodi.
    - `nodeUnhealthyReason`: il motivo per cui l'agente di monitoraggio di nodi ha identificato il nodo come non integro.
    - `minRepairWaitTimeMins`: il tempo minimo (in minuti) in cui la condizione di riparazione e il motivo non integro devono persistere prima che il nodo sia idoneo alla riparazione.
    - `repairAction`: l'azione che il sistema di riparazione deve intraprendere quando sono soddisfatte le condizioni di cui sopra.
  - Se si utilizza questo campo, è necessario specificare tutti i campi della struttura. È anche possibile fornire un elenco di tali sostituzioni.
  - `nodeMonitoringCondition` e `nodeUnhealthyReason` sono input di testo manuali che si impostano per indicare che si desidera deviare dal comportamento predefinito del sistema.
  - I campi `minRepairWaitTimeMins` e `repairAction` consentono di specificare le deviazioni dal comportamento predefinito del sistema.
  - L'esempio seguente mostra come sostituire il tempo di attesa a 20 minuti prima che Amazon EKS riavvii un nodo che presenta condizioni. `NvidiaXID13Error` Per impostazione predefinita, Amazon EKS attende 10 minuti prima di intraprendere azioni di riparazione a `AcceleratedHardwareReady` determinate condizioni.

```
aws eks update-nodegroup-config \
  --cluster-name my-cluster \
  --nodegroup-name my-nodegroup \
  --node-repair-config
 'enabled=true,nodeRepairConfigOverrides=[{nodeMonitoringCondition=AcceleratedHardwareReady
```

Problemi di integrità dei nodi

Le tabelle seguenti descrivono i problemi di integrità dei nodi che possono essere rilevati dall'agente di monitoraggio dei nodi. Esistono due tipi di problemi:

- Condizione: un problema del terminale che richiede un'azione di riparazione come la sostituzione o il riavvio dell'istanza. Quando la riparazione automatica è abilitata, Amazon EKS eseguirà un'azione di riparazione, sostituendo o riavviando il nodo. Per ulteriori informazioni, consulta the section called "Condizioni dei nodi".
- Evento: un problema temporaneo o una configurazione non ottimale del nodo. Non sarà effettuata alcuna operazione di riparazione automatica. Per ulteriori informazioni, consulta the section called "Eventi dei nodi".

AcceleratedHardware problemi di salute dei nodi

La condizione di monitoraggio riguarda `AcceleratedHardwareReady` per i problemi riportati nella tabella seguente con una gravità pari a "Condizione".

Se la riparazione automatica è abilitata, le azioni di riparazione elencate iniziano 10 minuti dopo il rilevamento del problema. Per ulteriori informazioni sugli errori XID, consulta Errori Xid nella Documentazione di implementazione e gestione delle GPU NVIDIA. Per ulteriori informazioni sui singoli messaggi XID, consulta Comprensione dei messaggi Xid nella Documentazione di implementazione e gestione delle GPU NVIDIA.

Name	Gravità	Description	Azione di riparazione
DCGMDiagnosticFallimento	Condizione	Un caso di test della suite di test della diagnostica attiva DCGM non è riuscito.	Nessuno
DCGMError	Condizione	La connessione al processo host DCGM è stata interrotta o non è stato possibile stabilirla.	Nessuno

Name	Gravità	Description	Azione di riparazione
DCGMFieldErrore [Codice]	Event	DCGM ha rilevato un deterioramento della GPU tramite un identificatore di campo.	Nessuno
DCGMHealthCodice [Codice]	Event	Un controllo sanitario DCGM non è riuscito in modo non fatale.	Nessuno
DCGMHealthCodice [Codice]	Condizione	Un controllo sanitario DCGM non è riuscito in modo fatale.	Nessuno
Un neurone DMAError	Condizione	Un motore DMA ha rilevato un errore irreversibile.	Replace (Sostituisci)
Errore HBMUncorr ectable neuronale	Condizione	Un HBM ha riscontra to un errore non correggibile e ha prodotto risultati errati.	Replace (Sostituisci)
Errore NCUncorre ctable neuronale	Condizione	È stato rilevato un errore di memoria non correggibile di Neuron Core.	Replace (Sostituisci)
Errore SRAMUncor rectable neuronale	Condizione	Una SRAM su chip ha rilevato un errore di parità e ha prodotto risultati errati.	Replace (Sostituisci)

Name	Gravità	Description	Azione di riparazione
NvidiaDeviceCountMismatch	Event	Il numero di dispositivi GPUs visibili tramite NVML non è coerente con il numero di dispositivi NVIDIA sul file system.	Nessuno
NvidiaDoubleBitError	Condizione	Un errore a doppio bit è stato prodotto dal driver della GPU.	Replace (Sostituisci)
Nvidia NCCLError	Event	Si è verificato un errore di sicurezza nella libreria NVIDIA Collective Communications (). `libnccl`	Nessuno
Errore Nvidia NVLink	Condizione	NVLink gli errori sono stati segnalati dal driver della GPU.	Replace (Sostituisci)
Errore Nvidia PCIe	Event	PCIe i replay sono stati attivati per ripristinare gli errori di trasmissione.	Nessuno
NvidiaPageRetirement	Event	Il driver della GPU ha contrassegnato una pagina di memoria come ritirata. Ciò può verificarsi se si riscontra un singolo errore a doppio bit o due errori a singolo bit nello stesso indirizzo.	Nessuno

Name	Gravità	Description	Azione di riparazione
NvidiaPowerError	Event	L'utilizzo dell'energia ha GPUs superato le soglie consentite.	Nessuno
NvidiaThermalError	Event	Stato termico del superamento delle soglie consentite GPUs .	Nessuno
Errore nvidiaXID [Code]	Condizione	Si è verificato un errore critico della GPU.	Sostituisci o riavvia
NvidiaXID[Code]Warning	Event	Si è verificato un errore non critico della GPU.	Nessuno

ContainerRuntime problemi di integrità del nodo

La condizione di monitoraggio riguarda `ContainerRuntimeReady` per i problemi riportati nella tabella seguente con una gravità pari a "Condizione".

Name	Gravità	Description	Azione di riparazione
ContainerRuntimeFailed	Event	Il runtime del container non è riuscito a creare un container, probabilmente correlato a eventuali problemi segnalati se si verificano ripetutamente.	Nessuno

Name	Gravità	Description	Azione di riparazione
DeprecatedContainerdConfiguration	Event	Un'immagine del contenitore che utilizza il manifesto dell'immagine obsoleta versione 2, schema 1, è stata recentemente inserita nel nodo. `containerd`	Nessuno
KubeletFailed	Event	Il kubelet è entrato in uno stato di errore.	Nessuno
LivenessProbeFailures	Event	È stato rilevato un errore della sonda liveness, che potrebbe indicare problemi nel codice dell'applicazione o nei valori di timeout insufficienti se si verificano ripetutamente.	Nessuno
PodStuckTerminating	Condizione	Un pod è o è rimasto bloccato nella terminazione per un periodo di tempo eccessivo, il che può essere causato da errori CRI che impediscono la progressione dello stato del pod.	Replace (Sostituisci)

Name	Gravità	Description	Azione di riparazione
ReadinessProbeFailures	Event	È stato rilevato un errore della sonda di preparazione, che potrebbe indicare problemi nel codice dell'applicazione o nei valori di timeout insufficienti se si verificano ripetutamente.	Nessuno
[Nome] RepeatedRestart	Event	Un'unità systemd si riavvia frequentemente.	Nessuno
ServiceFailedToStart	Event	Impossibile avviare un'unità systemd.	Nessuno

Problemi di integrità dei nodi del kernel

La condizione di monitoraggio riguarda `KernelReady` per i problemi riportati nella tabella seguente con una gravità pari a "Condizione".

Name	Gravità	Description	Azione di riparazione
AppBlocked	Event	L'attività è stata bloccata per un lungo periodo di tempo dalla pianificazione, in genere a causa del blocco in ingresso o in uscita.	Nessuno
AppCrash	Event	Un'applicazione sul nodo si è bloccata.	Nessuno

Name	Gravità	Description	Azione di riparazione
ApproachingKernelPidMax	Event	Il numero di processi si sta avvicinando al numero massimo di PIDs processi disponibili per l'`kernel.pid_max` impostazione corrente, dopo di che non è possibile avviare altri processi.	Nessuno
ApproachingMaxOpenFiles	Event	Il numero di file aperti è simile al numero massimo di file aperti possibili date le impostazioni correnti del kernel, dopodiché l'apertura di nuovi file avrà esito negativo.	Nessuno
ConntrackExceededKernel	Event	Il rilevamento delle connessioni ha superato il valore massimo per il kernel e non è stato possibile stabilire nuove connessioni, il che può causare la perdita di pacchetti.	Nessuno

Name	Gravità	Description	Azione di riparazione
ExcessiveZombieProcesses	Event	I processi che non possono essere recuperati completamente si accumulano in gran numero, il che indica problemi di applicazione e può portare al raggiungimento dei limiti dei processi di sistema.	Nessuno
ForkFailedOutOfPIDs	Condizione	Una chiamata fork o exec non è riuscita a causa dell'esaurimento del processo IDs o della memoria del sistema, che può essere causato da processi zombi o dall'esaurimento della memoria fisica.	Replace (Sostituisci)
KernelBug	Event	Un bug del kernel è stato rilevato e segnalato dal kernel Linux stesso, sebbene a volte ciò può essere causato da nodi con un elevato utilizzo della CPU o della memoria, con conseguente ritardo nell'elaborazione degli eventi.	Nessuno

Name	Gravità	Description	Azione di riparazione
LargeEnvironment	Event	Il numero di variabili di ambiente per questo processo è maggiore del previsto, potenzialmente causato da molti servizi `enableServiceLinks` impostati su true, il che può causare problemi di prestazioni.	Nessuno
RapidCron	Event	Un cron job è eseguito più velocemente di ogni cinque minuti su questo nodo, il che può influire sulle prestazioni se il processo consuma risorse significative.	Nessuno
SoftLockup	Event	La CPU si è bloccata per un periodo di tempo specificato.	Nessuno

Problemi di integrità dei nodi di rete

La condizione di monitoraggio riguarda `NetworkingReady` per i problemi riportati nella tabella seguente con una gravità pari a "Condizione".

Name	Gravità	Description	Azione di riparazione
BandwidthInExceeded	Event	I pacchetti accodati o rilasciati perché la larghezza di banda aggregata in ingresso ha superato il valore massimo per l'istanza.	Nessuno
BandwidthOutExceeded	Event	I pacchetti sono stati accodati o rilasciati perché la larghezza di banda aggregata in uscita ha superato il valore massimo per l'istanza.	Nessuno
ConntrackExceeded	Event	Il rilevamento delle connessioni ha superato il valore massimo per l'istanza e non è stato possibile stabilire nuove connessioni, il che può determinare la perdita di pacchetti.	Nessuno
IPAMDInconsistentStato	Event	Lo stato del checkpoint IPAMD su disco non riflette il runtime IPs del contenitore.	Nessuno
IPAMDNoIPs	Event	IPAMD ha esaurito gli indirizzi IP.	Nessuno

Name	Gravità	Description	Azione di riparazione
IPAMDNotPronto	Condizione	IPAMD non riesce a connettersi al server API.	Replace (Sostituisci)
IPAMDNotCorrere	Condizione	Il processo CNI di Amazon VPC non è stato trovato in esecuzione.	Replace (Sostituisci)
IPAMDRepeatedlyRiavvia	Event	Si sono verificati più riavvii del servizio IPAMD.	Nessuno
InterfaceNotRunning	Condizione	Questa interfaccia sembra non essere in esecuzione o ci sono problemi di rete.	Replace (Sostituisci)
InterfaceNotUp	Condizione	Questa interfaccia sembra non essere avviata o ci sono problemi di rete.	Replace (Sostituisci)
KubeProxyNotReady	Event	Kube-proxy non è riuscito a controllare o elencare le risorse.	Nessuno
LinkLocalExceeded	Event	I pacchetti sono stati accodati o rilasciati perché il PPS del traffico verso i servizi proxy locali ha superato il valore massimo per l'interfaccia di rete.	Nessuno

Name	Gravità	Description	Azione di riparazione
MACAddressPolicyMisconfigured	Event	La configurazione del link systemd-networkd ha un valore errato. `MACAddressPolicy`	Nessuno
MissingDefaultRoutes	Event	Mancano le regole di routing predefinite.	Nessuno
Mancante IPRoutes	Event	Mancano alcune rotte per Pod IPs.	Nessuno
Mancante IPRules	Event	Mancano delle regole per Pod IPs.	Nessuno
MissingLoopbackInterface	Condizione	L'interfaccia di loopback non è presente in questa istanza, il che causa l'interruzione dei servizi a seconda della connettività locale.	Replace (Sostituisci)
NetworkSysctl	Event	Le `sysctl` impostazioni di rete di questo nodo sono potenzialmente errate.	Nessuno
PPSExceeded	Event	I pacchetti sono stati accodati o rilasciati perché il PPS bidirezionale ha superato il valore massimo per l'istanza.	Nessuno

Name	Gravità	Description	Azione di riparazione
PortConflict	Event	Se un Pod utilizza HostPort, può scrivere `iptables` regole che sovrascrivono le porte già associate dell'host , impedendo potenzialmente l'accesso al server API. `kubelet`	Nessuno
UnexpectedRejectRule	Event	È stata rilevata una `DROP` regola `REJECT` o imprevista nel traffico `iptables` previsto che potrebbe bloccare.	Nessuno

Problemi di integrità dei nodi di storage

La condizione di monitoraggio riguarda `StorageReady` per i problemi riportati nella tabella seguente con una gravità pari a "Condizione".

Name	Gravità	Description	Azione di riparazione
EBSInstanceIOPSExceeded	Event	È stato superato il numero massimo di IOPS per l'istanza.	Nessuno
EBSInstanceThroughputExceeded	Event	Il throughput massimo per l'istanza è stato superato.	Nessuno
EBSVolume IOPSExceeded	Event	È stato superato il numero massimo di IOPS per un	Nessuno

Name	Gravità	Description	Azione di riparazione
		determinato volume EBS.	
EBSVolume ThroughputExceeded	Event	È stato superato il throughput massimo per un determinato volume Amazon EBS.	Nessuno
EtcHostsMountFailed	Event	Il montaggio del kubelet generato `/etc/hosts` non è riuscito a causa del rimontaggio dei dati utente durante il funzionamento. `/var/lib/kubelet/pods kubelet-container`	Nessuno
IODelays	Event	Ritardo di ingresso o uscita rilevato in un processo, che potrebbe indicare un approvvigionamento input-output insufficiente, se eccessivo.	Nessuno

Name	Gravità	Description	Azione di riparazione
KubeletDiskUsageSlow	Event	Segnala `kubelet` un utilizzo lento del disco durante il tentativo di accesso al filesystem. Ciò potrebbe indicare problemi di input-output del disco o problemi di file system insufficienti.	Nessuno
XFSSmallAverageClusterSize	Event	La dimensione media del cluster XFS è piccola, il che indica un'eccessiva frammentazione dello spazio libero. Ciò può impedire la creazione di file nonostante gli inode disponibili o lo spazio libero.	Nessuno

Visualizzazione dello stato di integrità dei nodi

Questo argomento spiega gli strumenti e i metodi disponibili per monitorare lo stato di integrità dei nodi nei cluster Amazon EKS. Le informazioni riguardano le condizioni dei nodi, gli eventi e i casi di rilevamento che aiutano a identificare e diagnosticare problemi a livello di nodo. Utilizza i comandi e gli schemi descritti qui per ispezionare le risorse dello stato dei nodi, interpretare le condizioni dello stato e analizzare gli eventi dei nodi per la risoluzione dei problemi operativi.

Puoi ottenere alcune informazioni sullo stato dei nodi con i comandi Kubernetes per tutti i nodi. E se utilizzi l'agente di monitoraggio dei nodi tramite Amazon EKS Auto Mode o il componente aggiuntivo gestito Amazon EKS, otterrai una più ampia varietà di segnali di nodo per aiutarti a risolvere i problemi. Le descrizioni dei problemi di integrità rilevati dall'agente di monitoraggio dei nodi sono disponibili anche nella dashboard di osservabilità. Per ulteriori informazioni, consulta the section called "Integrità del nodo".

Condizioni dei nodi

Le condizioni dei nodi rappresentano problemi terminali che richiedono azioni di riparazione come la sostituzione o il riavvio dell'istanza.

Per ottenere le condizioni per tutti i nodi:

```
kubectl get nodes -o 'custom-
columns=NAME:.metadata.name,CONDITIONS:.status.conditions[*].type,STATUS:.status.conditions[*].
```

Per ottenere condizioni dettagliate per un nodo specifico

```
kubectl describe node node-name
```

Esempio di output delle condizioni di un nodo integro:

```
  - 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
```

Esempio di condizione di un nodo non integro con un problema di rete:

```
  - 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
```

Eventi dei nodi

Gli eventi dei nodi indicano problemi temporanei o configurazioni non ottimali.

Per ottenere tutti gli eventi segnalati dall'agente di monitoraggio dei nodi

Quando l'agente di monitoraggio dei nodi è disponibile, puoi eseguire il comando seguente.

```
kubectl get events --field-selector=reportingComponent=eks-node-monitoring-agent
```

Output di esempio:

```
LAST SEEN   TYPE      REASON       OBJECT
 MESSAGE
4s          Warning   SoftLockup   node/ip-192-168-71-251.us-west-2.compute.internal
 CPU stuck for 23s
```

Per ottenere eventi per tutti i nodi

```
kubectl get events --field-selector involvedObject.kind=Node
```

Per ottenere eventi per un nodo specifico

```
kubectl get events --field-selector involvedObject.kind=Node,involvedObject.name=node-name
```

Per osservare gli eventi in tempo reale

```
kubectl get events -w --field-selector involvedObject.kind=Node
```

Esempio di output di un 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
```

Comandi per la risoluzione dei problemi comuni

```
# 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
```

Recuperare i log dei nodi per un nodo gestito usando kubectl e S3

Scopri come recuperare i log dei nodi per un nodo gestito di Amazon EKS che dispone dell'agente di monitoraggio dei nodi.

Prerequisiti

Verifica di disporre di quanto segue:

- Un cluster Amazon EKS esistente con agente di monitoraggio dei nodi. Per ulteriori informazioni, consulta the section called "Integrità del nodo".
- Lo strumento di riga di comando `kubectl` installato e configurato per comunicare con il cluster.
- La AWS CLI è stata installata e ha effettuato l'accesso con autorizzazioni sufficienti per creare bucket e oggetti S3.
- Una versione recente di Python 3 installata
- L' AWS SDK per Python 3, Boto 3, è installato.

Fase 1: Creare di una destinazione bucket S3 (facoltativo)

Se non disponi già di un bucket S3 in cui archiviare i log, creane uno. Usa il seguente comando AWS CLI. Per impostazione predefinita, il bucket è l'elenco di controllo degli accessi `private`. Sostituisci *`bucket-name`* con il nome univoco del bucket scelto.

```
aws s3api create-bucket --bucket <bucket-name>
```

Fase 2: Creare un URL S3 prefirmato per HTTP Put

Amazon EKS restituisce i log dei nodi eseguendo un'operazione HTTP PUT su un URL specificato. In questo tutorial, genereremo un URL PUT HTTP S3 prefirmato.

I log verranno restituiti come gzip tarball, con l'estensione `.tar.gz`.

> **Note**
>
> È necessario utilizzare l' AWS API o un SDK per creare l'URL di caricamento S3 prefirmato per consentire a EKS di caricare il file di registro. Non puoi creare un URL di caricamento S3 prefirmato utilizzando la CLI AWS .

1. Determina in che punto del bucket desideri archiviare i log. Ad esempio, puoi usare *2024-11-12/logs1.tar.gz* come chiave.
2. Salva il seguente codice Python sul file *presign-upload.py*. Sostituisci *<bucket-name>* e *<key>*. La chiave deve terminare con `.tar.gz`.

```
import boto3; print(boto3.client('s3').generate_presigned_url(
   ClientMethod='put_object',
   Params={'Bucket': '<bucket-name>', 'Key': '<key>'},
   ExpiresIn=1000
))
```

3. Esegui lo script con

```
python presign-upload.py
```

4. Annota l'output dell'URL. Usa questo valore nella fase successiva come `http-put-destination`.

Per ulteriori informazioni, consulta Generare un URL predefinito per caricare un file nella documentazione AWS Boto3 SDK per Python.

Fase 3: Creare una risorsa NodeDiagnostic

Identifica il nome del nodo da cui si desidera raccogliere i log.

Crea un manifesto `NodeDiagnostic` che utilizzi il nome del nodo come nome della risorsa e fornire una destinazione URL PUT HTTP.

```
apiVersion: eks.amazonaws.com/v1alpha1
kind: NodeDiagnostic
metadata:
    name: <node-name>
spec:
    logCapture:
        destination: http-put-destination
```

Applica il file manifesto al cluster.

```
kubectl apply -f nodediagnostic.yaml
```

Puoi controllare lo stato della raccolta descrivendo la risorsa `NodeDiagnostic`:

- Uno stato di `Success` o `SuccessWithErrors` indica che l'attività è stata completata e i log sono stati caricati nella destinazione fornita (`SuccessWithErrors` indica che alcuni log potrebbero mancare).
- Se lo stato è Failure, conferma che l'URL di caricamento è ben formato e non è scaduto.

```
kubectl describe nodediagnostics.eks.amazonaws.com/<node-name>
```

Fase 4: Scaricare log da S3

Attendi circa un minuto prima di tentare di scaricare i log. Quindi, usa la CLI di S3 per scaricare i log.

```
# Once NodeDiagnostic shows Success status, download the logs
aws s3 cp s3://<bucket-name>/key ./<path-to-node-logs>.tar.gz
```

Fase 5: Pulizia della NodeDiagnostic risorsa

- Le risorse `NodeDiagnostic` non vengono eliminate automaticamente. Devi eliminarle in autonomia dopo aver ottenuto gli artefatti del log.

```
# Delete the NodeDiagnostic resource
kubectl delete nodediagnostics.eks.amazonaws.com/<node-name>
```

Panoramica di Amazon EKS Hybrid Nodes

Con Amazon EKS Hybrid Nodes, puoi utilizzare la tua infrastruttura on-premises ed edge come nodi nei cluster Amazon EKS. AWS gestisce il piano di controllo Kubernetes ospitato da AWS del cluster Amazon EKS e tu gestisci i nodi ibridi che vengono eseguiti nei tuoi ambienti on-premises o edge. Ciò unifica la gestione di Kubernetes in tutti gli ambienti e affida la gestione del piano di controllo di Kubernetes a AWS per le applicazioni on-premises ed edge.

Amazon EKS Hybrid Nodes funziona con qualsiasi hardware o macchina virtuale on-premises, portando l'efficienza, la scalabilità e la disponibilità di Amazon EKS ovunque sia necessario eseguire le applicazioni. Puoi utilizzare un'ampia gamma di funzionalità di Amazon EKS con Amazon EKS

Hybrid Nodes, tra cui componenti aggiuntivi Amazon EKS, Amazon EKS Pod Identity, voci di accesso al cluster, informazioni sui cluster e supporto esteso per le versioni di Kubernetes. Amazon EKS Hybrid Nodes si integra in modo nativo con i servizi AWS tra cui Systems Manager di AWS, IAM Roles Anywhere di AWS, il Servizio gestito da Amazon per Prometheus e Amazon CloudWatch per il monitoraggio centralizzato, la registrazione e la gestione delle identità.

Con Amazon EKS Hybrid Nodes non sono previsti impegni anticipati o tariffe minime e ti viene addebitato un costo orario per le risorse vCPU dei tuoi nodi ibridi quando sono collegati ai cluster Amazon EKS. Per ulteriori informazioni, consulta Prezzi di Amazon EKS.

Caratteristiche

EKS Hybrid Nodes offre le seguenti funzionalità di alto livello:

- Piano di controllo Kubernetes gestito: AWS gestisce il piano di controllo Kubernetes ospitato da AWS del cluster EKS e tu gestisci i nodi ibridi che vengono eseguiti negli ambienti on-premises o edge. Ciò unifica la gestione di Kubernetes in tutti gli ambienti e affida la gestione del piano di controllo di Kubernetes a AWS per le applicazioni on-premises ed edge. Spostando il piano di controllo Kubernetes su AWS, puoi conservare la capacità on-premises per le tue applicazioni e avere la certezza che il piano di controllo Kubernetes sia scalabile in base ai carichi di lavoro.
- Esperienza EKS coerente: la maggior parte delle funzionalità EKS è supportata da EKS Hybrid Nodes per un'esperienza EKS coerente in tutti gli ambienti on-premises e cloud, inclusi componenti aggiuntivi EKS, EKS Pod Identity, voci di accesso al cluster, approfondimenti sui cluster, supporto esteso delle versioni di Kubernetes e altro ancora. Per ulteriori informazioni sui componenti aggiuntivi EKS supportati con EKS Hybrid Nodes, consulta the section called "Configurare i componenti aggiuntivi".
- Osservabilità centralizzata e gestione delle identità: EKS Hybrid Nodes si integra in modo nativo con i servizi AWS tra cui Systems Manager di AWS, IAM Roles Anywhere di AWS, il Servizio gestito da Amazon per Prometheus e Amazon CloudWatch per il monitoraggio centralizzato, la registrazione e la gestione delle identità.
- Burst-to-cloud o aggiunta di capacità on-premises: un singolo cluster EKS può essere utilizzato per eseguire nodi e nodi ibridi in Regioni AWS, Zone locali AWS oppure Outpost AWS per burst-to-cloud o per aggiungere capacità on-premises ai cluster EKS. Per ulteriori informazioni, consulta Considerations for mixed mode clusters.
- Infrastruttura flessibile: EKS Hybrid Nodes segue un approccio basato sull'infrastruttura personalizzata ed è agnostico rispetto all'infrastruttura utilizzata per i nodi ibridi. Puoi eseguire nodi

ibridi su macchine fisiche o virtuali e architetture x86 e ARM, rendendo possibile la migrazione dei carichi di lavoro on-premises in esecuzione su nodi ibridi su diversi tipi di infrastruttura.

- Rete flessibile: con EKS Hybrid Nodes, la comunicazione tra il piano di controllo EKS e i nodi ibridi viene instradata attraverso il VPC e le sottoreti trasferite durante la creazione del cluster, il che si basa sul meccanismo esistente in EKS per il networking dal piano di controllo al nodo. Ciò offre flessibilità rispetto al tuo metodo preferito per connettere le reti on-premises a un VPC in AWS. Sono disponibili diverse opzioni documentate, tra cui VPN sito-sito di AWS, Direct Connect di AWS o la soluzione VPN, e puoi scegliere il metodo più adatto al tuo caso d'uso.

Limiti

- Per ciascun cluster sono supportati fino a 15 CIDR per reti di nodi remoti e 15 CIDR per reti di pod remoti.

Considerazioni

- EKS Hybrid Nodes può essere utilizzato con cluster EKS nuovi o esistenti.
- EKS Hybrid Nodes è disponibile in tutte le Regioni AWS, ad eccezione delle Regioni AWS GovCloud (Stati Uniti) e delle Regioni AWS in Cina.
- EKS Hybrid Nodes deve disporre di una connessione affidabile tra l'ambiente on-premises e AWS. EKS Hybrid Nodes non è adatto per ambienti disconnessi, interrotti, intermittenti o limitati (DDIL). Se utilizzi un ambiente DDIL, prendi in considerazione Amazon EKS Anywhere. Consulta Best practice for EKS Hybrid Nodes per informazioni sul comportamento dei nodi ibridi nei casi di disconnessione della rete.
- Non è supportata l'esecuzione di EKS Hybrid Nodes sull'infrastruttura cloud, tra cui Regioni AWS, Zone locali AWS, Outpost AWS o in altri cloud. Ti verrà addebitata la tariffa per i nodi ibridi se esegui nodi ibridi su istanze Amazon EC2.
- La fatturazione per i nodi ibridi inizia quando i nodi si uniscono al cluster EKS e si interrompe quando i nodi vengono rimossi dal cluster. Assicurati di rimuovere i nodi ibridi dal cluster EKS se non li utilizzi.

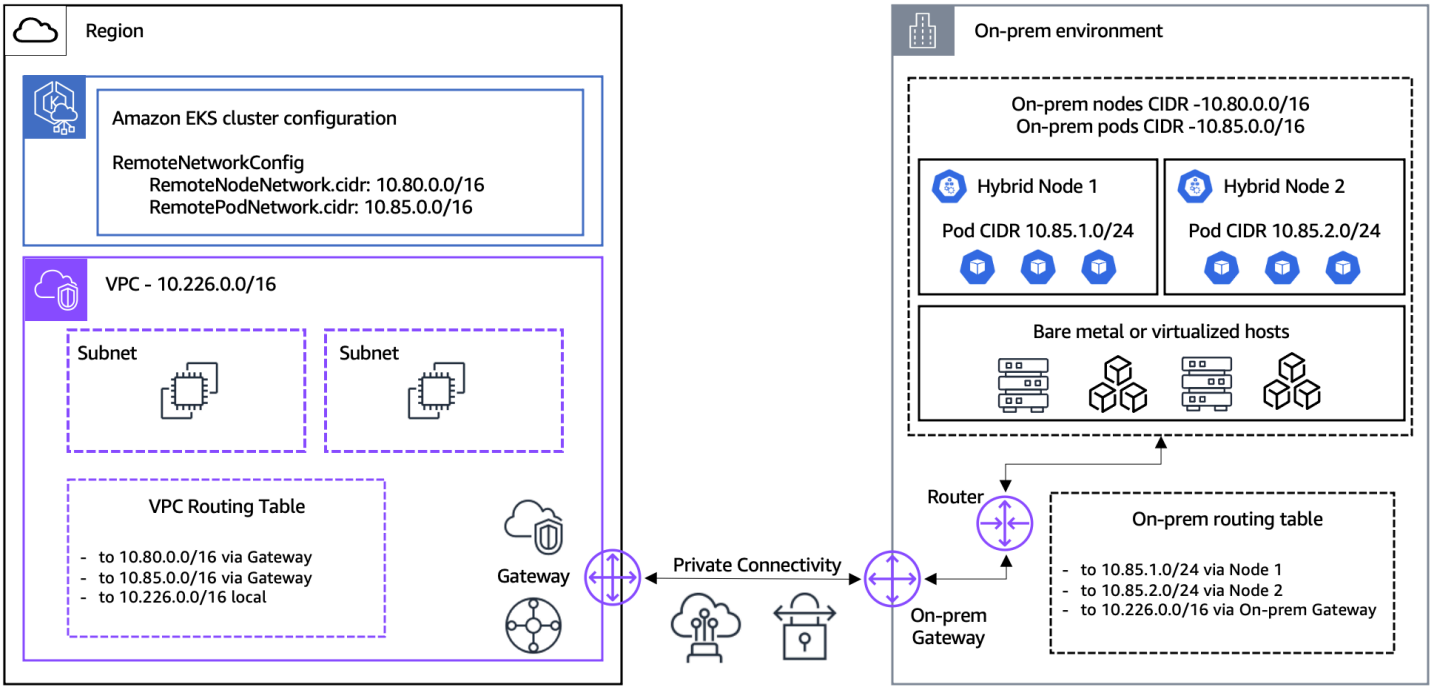
Risorse aggiuntive

- EKS Hybrid Nodes workshop: istruzioni dettagliate per l'implementazione di EKS Hybrid Nodes in un ambiente demo.
- AWS re:Invent: EKS Hybrid Nodes: sessione di AWS re:Invent che presenta il lancio di EKS Hybrid Nodes con un cliente che mostra come utilizza EKS Hybrid Nodes nel proprio ambiente.
- AWS re:Post: Cluster networking for EKS Hybrid Nodes: articolo che spiega vari metodi per configurare la rete per EKS Hybrid Nodes.
- Blog AWS: Run GenAI inference across environments with EKS Hybrid Nodes: post sul blog che mostra come eseguire l'inferenza GenAI in ambienti con EKS Hybrid Nodes.

Configurazione dei prerequisiti per i nodi ibridi

Per utilizzare Amazon EKS Hybrid Nodes devi disporre di connettività privata dal tuo ambiente on-premises da/verso AWS, server bare metal o macchine virtuali con un sistema operativo supportato e devi aver configurato attivazioni ibride di IAM Roles Anywhere di AWS o Systems Manager (SSM) di AWS. Sei responsabile della gestione di questi prerequisiti durante l'intero ciclo di vita dei nodi ibridi.

- Connettività di rete ibrida dall'ambiente on-premises da/verso AWS
- Infrastruttura sotto forma di macchine fisiche o virtuali
- Sistema operativo compatibile con i nodi ibridi
- Provider di credenziali IAM on-premises configurato



Connettività di rete ibrida

La comunicazione tra il piano di controllo di Amazon EKS e i nodi ibridi viene instradata attraverso il VPC e le sottoreti trasferite durante la creazione del cluster, il che si basa sul meccanismo esistente in Amazon EKS per il networking dal piano di controllo al nodo. Sono disponibili diverse opzioni documentate per connettere l'ambiente on-premises al VPC, tra cui VPN sito-sito di AWS, Direct Connect di AWS o la tua connessione VPN personalizzata. Consulta le guide per l'utente VPN sito-sito di AWS e Direct Connect di AWS per ulteriori informazioni su come utilizzare queste soluzioni per la connessione di rete ibrida.

Per un'esperienza ottimale, ti consigliamo di disporre di una connettività di rete affidabile di almeno 100 Mbps e una latenza massima di 200 ms di andata e ritorno per la connessione dei nodi ibridi alla Regione AWS. Questa è una guida generale adatta alla maggior parte dei casi d'uso, ma non è un requisito fondamentale. I requisiti di larghezza di banda e latenza possono variare in base al numero di nodi ibridi e alle caratteristiche del carico di lavoro, come la dimensione dell'immagine dell'applicazione, l'elasticità dell'applicazione, le configurazioni di monitoraggio e registrazione e le dipendenze delle applicazioni dall'accesso ai dati archiviati in altri servizi AWS. Ti consigliamo di eseguire test con applicazioni e ambienti prima di passare all'implementazione per verificare che la configurazione di rete soddisfi i requisiti per i carichi di lavoro.

Configurazione di reti on-premises

Devi abilitare l'accesso alla rete in entrata dal piano di controllo di Amazon EKS all'ambiente on-premises per consentire al piano di controllo di Amazon EKS di comunicare con il `kubelet` in esecuzione sui nodi ibridi e, facoltativamente, con i webhook in esecuzione sui nodi ibridi. Inoltre, devi abilitare l'accesso alla rete in uscita per i nodi ibridi e i componenti in esecuzione su di essi per comunicare con il piano di controllo di Amazon EKS. Puoi configurare questa comunicazione in modo che rimanga completamente privata con Direct Connect di AWS, VPN sito-sito di AWS o con la tua connessione VPN.

Gli intervalli di Routing interdominio senza classi (CIDR) utilizzati per le reti di nodi e pod on-premises devono utilizzare intervalli di indirizzi IPv4 RFC-1918. Il router on-premises deve essere configurato con percorsi verso i nodi locali e, facoltativamente, verso i pod. Consulta the section called "Configurazione di rete on-premises" per ulteriori informazioni sui requisiti di rete on-premises, incluso l'elenco completo delle porte e dei protocolli richiesti che devono essere abilitati nel firewall e nell'ambiente on-premises.

Configurazione del cluster EKS

Per ridurre al minimo la latenza, ti consigliamo di creare il cluster Amazon EKS nella Regione AWS più vicina all'ambiente on-premises o edge. I CIDR dei nodi e dei pod on-premises vengono trasmessi durante la creazione di cluster Amazon EKS tramite due campi API: `RemoteNodeNetwork` e `RemotePodNetwork`. Potrebbe essere necessario consultare il team di rete on-premises per individuare i CIDR dei nodi e dei pod on-premises. Il CIDR dei nodi viene allocato dalla rete on-premises e il CIDR dei pod viene allocato dalla Container Network Interface (CNI) che utilizzi se utilizzi una rete overlay per la tua CNI. Cilium e Calico utilizzano reti overlay per impostazione predefinita.

I CIDR dei nodi e dei pod on-premises che configuri tramite i campi `RemoteNodeNetwork` e `RemotePodNetwork` vengono utilizzati per configurare il piano di controllo di Amazon EKS per indirizzare il traffico attraverso il tuo VPC verso il `kubelet` e i pod in esecuzione sui tuoi nodi ibridi. I CIDR dei nodi e dei pod on-premises non possono sovrapporsi tra loro, con il CIDR del VPC che trasferisci durante la creazione del cluster o con la configurazione IPv4 del servizio per il tuo cluster Amazon EKS. Inoltre, i CIDR dei pod devono essere unici per ogni cluster EKS in modo che il router on-premises possa instradare il traffico.

Ti consigliamo di utilizzare l'accesso agli endpoint pubblico o privato per l'endpoint del server API Amazon EKS Kubernetes. Se scegli "Pubblico e privato", l'endpoint del server API Amazon EKS Kubernetes si risolverà sempre negli IP pubblici per i nodi ibridi in esecuzione all'esterno del tuo VPC,

il che può impedire ai nodi ibridi di unirsi al cluster. Quando utilizzi l'accesso pubblico agli endpoint, l'endpoint del server dell'API Kubernetes viene risolto in IP pubblici e la comunicazione dai nodi ibridi al piano di controllo di Amazon EKS verrà instradata su Internet. Quando scegli l'accesso privato agli endpoint, l'endpoint del server dell'API Kubernetes viene risolto in IP privati e la comunicazione dai nodi ibridi al piano di controllo di Amazon EKS verrà instradata tramite il tuo collegamento di connettività privato, nella maggior parte dei casi Direct Connect di AWS o VPN sito-sito di AWS.

Configurazione VPC

Devi configurare il VPC che trasferisci durante la creazione del cluster Amazon EKS con i percorsi nella tabella di routing per il nodo on-premises e, facoltativamente, le reti pod con il gateway privato virtuale (VGW) o il gateway di transito (TGW) come destinazione. Di seguito è riportato un esempio. Sostituisci `REMOTE_NODE_CIDR` e `REMOTE_POD_CIDR` con i valori per la tua rete on-premises.

Destinazione	Target	Descrizione
10.226.0.0/16	local	Percorsi del traffico locale verso il VPC all'interno del VPC
REMOTE_NODE_CIDR	tgw-abcdef123456	CIDR nodo on-premises, indirizza il traffico verso il TGW
REMODE_POD_CIDR	tgw-abcdef123456	CIDR pod on-premises, indirizza il traffico verso il TGW

Configurazione del gruppo di sicurezza

Quando crei un cluster, Amazon EKS crea un gruppo di sicurezza denominato `eks-cluster-sg-<cluster-name>-<uniqueID>`. Non puoi modificare le regole in entrata di questo gruppo di sicurezza del cluster, ma puoi limitare le regole in uscita. Devi aggiungere un gruppo di sicurezza aggiuntivo al cluster per abilitare il kubelet e, facoltativamente, i webhook in esecuzione sui nodi ibridi per contattare il piano di controllo di Amazon EKS. Le regole in entrata richieste per questo gruppo di sicurezza aggiuntivo sono riportate di seguito. Sostituisci `REMOTE_NODE_CIDR` e `REMOTE_POD_CIDR` con i valori per la tua rete on-premises.

Nome	ID regola del gruppo di sicurezza	Versione IP	Tipo	Protocollo	Intervallo porte	Origine
Nodo on-premises in entrata	sgr-abcdef123456	IPv4	HTTPS	TCP	443	REMOTE_NODE_CIDR
Pod on-premises in entrata	sgr-abcdef654321	IPv4	HTTPS	TCP	443	REMOTE_POD_CIDR

Infrastruttura

Devi disporre di server bare metal o macchine virtuali da utilizzare come nodi ibridi. I nodi ibridi sono indipendenti dall'infrastruttura sottostante e supportano le architetture x86 e ARM. Amazon EKS Hybrid Nodes segue un approccio "a infrastruttura personalizzata", in cui sei responsabile del provisioning e della gestione dei server bare metal o delle macchine virtuali che usi per i nodi ibridi. Sebbene non sia previsto un requisito minimo rigoroso di risorse, ti consigliamo di utilizzare host con almeno 1 vCPU e 1 GB di RAM per i nodi ibridi.

Sistema operativo

Bottlerocket, Amazon Linux 2023 (AL2023), Ubuntu e RHEL vengono convalidati su base continuativa per l'uso come sistema operativo di nodi per nodi ibridi. Bottlerocket è supportato da AWS solo in ambienti VMware vSphere. AL2023 non è coperto dai piani di supporto di AWS se eseguito al di fuori di Amazon EC2. AL2023 può essere utilizzato solo in ambienti virtualizzati on-premises, consulta Amazon Linux 2023 User Guide per ulteriori informazioni. AWS supporta l'integrazione dei nodi ibridi con i sistemi operativi Ubuntu e RHEL ma non fornisce supporto per il sistema operativo stesso.

Il provisioning e la gestione del sistema operativo sono una tua responsabilità. Quando testi i nodi ibridi per la prima volta, è più semplice eseguire la CLI Amazon EKS Hybrid Nodes (`nodeadm`) su un host già fornito. Per le implementazioni di produzione, ti consigliamo di includere `nodeadm` nelle immagini "gold" del tuo sistema operativo configurato per l'esecuzione come servizio systemd per unire automaticamente gli host ai cluster Amazon EKS all'avvio dell'host.

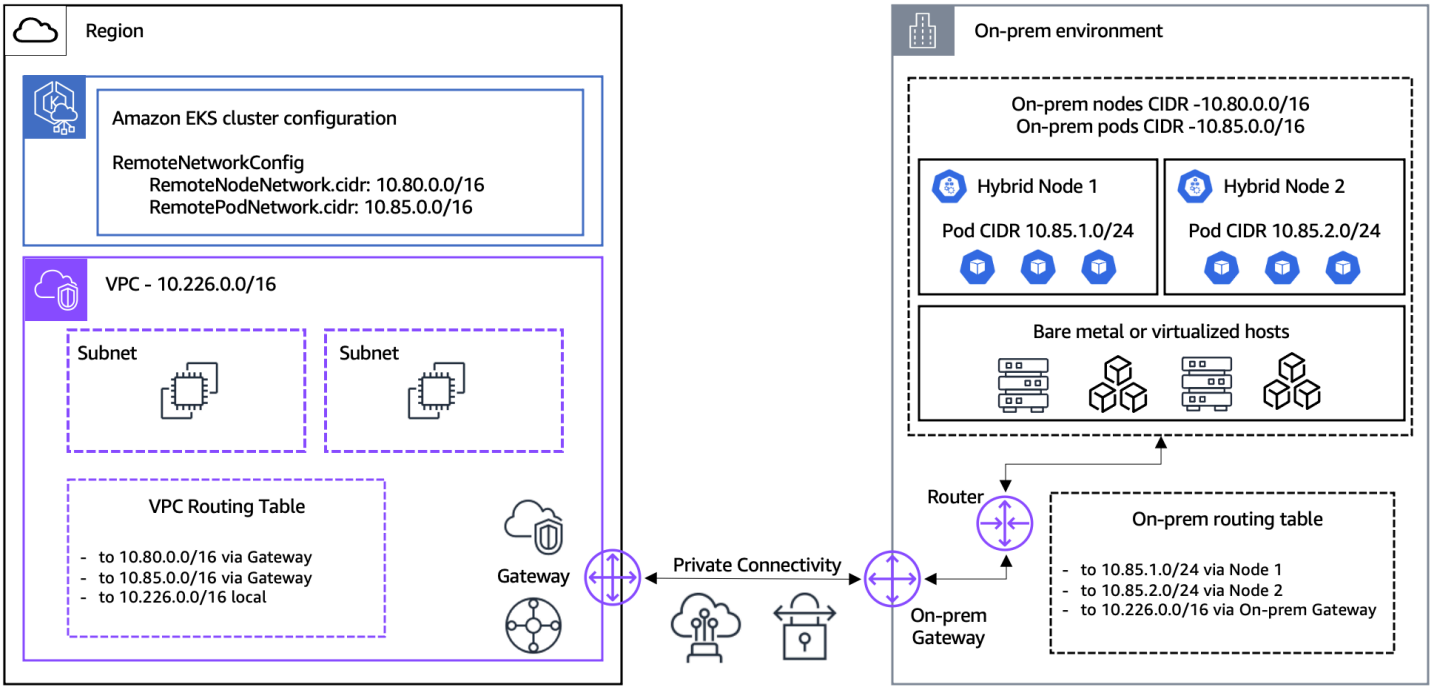
Provider di credenziali IAM on-premises

Amazon EKS Hybrid Nodes utilizza credenziali IAM temporanee fornite da attivazioni ibride SSM di AWS o IAM Roles Anywhere di AWS per l'autenticazione con il cluster Amazon EKS. Devi utilizzare attivazioni ibride SSM AWS o IAM Roles Anywhere di AWS con la CLI Amazon EKS Hybrid Nodes (`nodeadm`). Ti consigliamo di utilizzare le attivazioni ibride SSM AWS se non disponi di un'infrastruttura a chiave pubblica (PKI) esistente con un'autorità di certificazione (CA) e certificati per i tuoi ambienti on-premises. Se disponi di PKI e certificati esistenti on-premises, utilizza IAM Roles Anywhere di AWS.

Analogamente a the section called "Ruolo IAM del nodo" per i nodi in esecuzione su Amazon EC2, creerai un ruolo IAM Hybrid Nodes con le autorizzazioni necessarie per unire nodi ibridi ai cluster Amazon EKS. Se utilizzi IAM Roles Anywhere di AWS, configura una policy di fiducia che consenta a IAM Roles Anywhere di AWS di assumere il ruolo IAM di Hybrid Nodes e configura il tuo profilo IAM Roles Anywhere di AWS con il ruolo IAM di Hybrid Nodes come ruolo assumibile. Se utilizzi SSM di AWS, configura una policy di fiducia che consenta a SSM di AWS di assumere il ruolo IAM di Hybrid Nodes e creare l'attivazione ibrida con il ruolo IAM di Hybrid Nodes. Consulta the section called "Preparazione delle credenziali" su come creare il ruolo IAM di Hybrid Nodes con le autorizzazioni richieste.

Preparazione della rete per i nodi ibridi

Questo argomento fornisce una panoramica della configurazione di rete che devi adottare prima di creare un cluster Amazon EKS e collegare nodi ibridi. Questa guida presuppone che tu abbia soddisfatto i requisiti preliminari per la connettività di rete ibrida utilizzando AWS Site-to-Site VPN, AWS Direct Connect o la tua soluzione VPN.



Configurazione di rete on-premises

Requisiti minimi di rete

Per un'esperienza ottimale, ti consigliamo di disporre di una connettività di rete affidabile di almeno 100 Mbps e una latenza massima di 200 ms di andata e ritorno per la connessione dei nodi ibridi alla Regione. AWS Questa è una guida generale adatta alla maggior parte dei casi d'uso, ma non è un requisito fondamentale. I requisiti di larghezza di banda e latenza possono variare in base al numero di nodi ibridi e alle caratteristiche del carico di lavoro, come la dimensione dell'immagine dell'applicazione, l'elasticità dell'applicazione, le configurazioni di monitoraggio e registrazione e le dipendenze delle applicazioni dall'accesso ai dati archiviati in altri servizi. AWS Ti consigliamo di eseguire test con applicazioni e ambienti prima di passare all'implementazione per verificare che la configurazione di rete soddisfi i requisiti per i carichi di lavoro.

Nodo e pod locali CIDRs

Identifica il nodo e il pod CIDRs che utilizzerai per i tuoi nodi ibridi e i carichi di lavoro in esecuzione su di essi. Il CIDR dei nodi viene allocato dalla rete on-premises e quello dei pod viene allocato dalla Container Network Interface (CNI) se utilizzi una rete overlay per la CNI. Trasmetti il nodo CIDRs e il pod locali CIDRs come input quando crei il tuo cluster EKS con i `RemoteNodeNetwork` campi and. `RemotePodNetwork` Il nodo locale CIDRs deve essere instradabile sulla rete locale. Per ulteriori informazioni sull'instradabilità del CIDR dei pod on-premises, consulta la sezione seguente.

Gli intervalli CIDR dei nodi e dei pod on-premises devono soddisfare i seguenti requisiti:

1. Essere compresi in uno degli intervalli RFC-1918 `IPv4` seguenti: `10.0.0.0/8`, `172.16.0.0/12` o `192.168.0.0/16`.
2. Non si sovrappongano tra loro, né con il CIDR del VPC per il tuo cluster EKS né con il CIDR `IPv4` del tuo servizio Kubernetes.

Instradamento della rete pod on-premises

Quando si utilizzano i nodi ibridi EKS, in genere si consiglia di rendere il pod locale CIDRs instradabile sulla rete locale per consentire la comunicazione e la funzionalità complete del cluster tra ambienti cloud e locali.

Reti di pod instradabili

Se riesci a rendere la tua rete di pod instradabile su quella on-premises, segui le istruzioni riportate di seguito.

1. Configura il campo `RemotePodNetwork` per il cluster EKS con il CIDR del pod on-premises, le tabelle di routing del VPC con il CIDR del pod on-premises e il gruppo di sicurezza del cluster EKS con il CIDR del pod on-premises.
2. Esistono diverse tecniche che puoi utilizzare per rendere il CIDR del pod on-premises instradabile sulla tua rete on-premises, tra cui Border Gateway Protocol (BGP), route statiche o altre soluzioni di instradamento personalizzate. BGP è la soluzione consigliata in quanto è più scalabile e facile da gestire rispetto alle soluzioni alternative che richiedono una configurazione del percorso personalizzata o manuale. AWS supporta le funzionalità BGP di Cilium e Calico per il pod CIDRs pubblicitario, vedi e per ulteriori informazioni. the section called "Configurazione della CNI" the section called "CIDR dei pod remoti instradabili"
3. I webhook possono essere eseguiti su nodi ibridi poiché il piano di controllo EKS è in grado di comunicare con gli indirizzi IP dei pod assegnati ai webhook.
4. I carichi di lavoro in esecuzione su nodi cloud sono in grado di comunicare direttamente con i carichi di lavoro in esecuzione su nodi ibridi nello stesso cluster EKS.
5. Altri AWS servizi, come AWS Application Load Balancers e Amazon Managed Service for Prometheus, sono in grado di comunicare con i carichi di lavoro in esecuzione su nodi ibridi per bilanciare il traffico di rete e le metriche dei pod.

Reti di pod non instradabili

Se non riesci a rendere instradabili le reti di pod su quella on-premises, segui le istruzioni riportate di seguito.

1. I webhook non possono essere eseguiti su nodi ibridi perché richiedono la connettività dal piano di controllo EKS agli indirizzi IP dei pod assegnati ai webhook. In questo caso, ti consigliamo di eseguire webhook su nodi cloud nello stesso cluster EKS dei nodi ibridi, consulta the section called "Configurazione di webhook" per ulteriori informazioni.
2. I carichi di lavoro in esecuzione sui nodi cloud non sono in grado di comunicare direttamente con i carichi di lavoro in esecuzione su nodi ibridi quando si utilizza la CNI del VPC per i nodi cloud e Cilium o Calico per i nodi ibridi.
3. Utilizza la Distribuzione del traffico del servizio per mantenere il traffico locale per la zona da cui proviene. Per ulteriori informazioni sulla Distribuzione del traffico del servizio, consulta the section called "Configurazione della distribuzione del traffico del servizio".
4. Configura la CNI per utilizzare la mascheratura in uscita o la Network Address Translation (NAT) per il traffico dei pod in uscita dagli host on-premises. Ciò è abilitato per impostazione predefinita in Cilium. Calico necessita che `natOutgoing` sia impostato su `true`.
5. Altri AWS servizi, come AWS Application Load Balancers e Amazon Managed Service for Prometheus, non sono in grado di comunicare con i carichi di lavoro in esecuzione su nodi ibridi.

Accesso richiesto durante l'installazione e l'aggiornamento del nodo ibrido

Devi avere accesso ai seguenti domini durante il processo di installazione in cui installi le dipendenze dei nodi ibridi sui tuoi host. Questa procedura può essere eseguita una sola volta durante la creazione delle immagini del sistema operativo oppure può essere eseguita su ciascun host in Passaggio di runtime. Ciò include l'installazione iniziale e l'aggiornamento della versione Kubernetes dei nodi ibridi.

Alcuni pacchetti vengono installati utilizzando il gestore di pacchetti predefinito del sistema operativo. Per AL2 023 e RHEL, il `yum` comando viene utilizzato per installare, e. `containerd` `ca-certificates` `iptables` `amazon-ssm-agent` Per Ubuntu viene utilizzato `apt` per installare `containerd`, `ca-certificates` e `iptables`, e viene utilizzato `snap` per installare `amazon-ssm-agent`.

Componente	URL	Protocollo	Porta
Artefatti del nodo EKS (S3)	https://hybrid-assets.eks.amazonaws.com	HTTPS	443
Endpoint del servizio EKS	https://eks.*region*.amazonaws.com	HTTPS	443
Endpoint del servizio ECR	https://api.ecr.*region*.amazonaws.com	HTTPS	443
Endpoint EKS ECR	Consulta the section called "Visualizza i registri delle immagini di Amazon" per gli endpoint regionali.	HTTPS	443
Endpoint binario SSM [1]	https://amazon-ssm - .s3. *region* *region*.amazonaws.com	HTTPS	443
Endpoint del servizio SSM [1]	https://ssm.*region*.amazonaws.com	HTTPS	443
Endpoint binario IAM Anywhere [2]	https://rolesanywhere.amazonaws.com	HTTPS	443
Endpoint del servizio IAM Anywhere [2]	https://rolesanywhere.*region*.amazonaws.com	HTTPS	443
Endpoint del gestore di pacchetti del sistema operativo	Gli endpoint dei repository di pacchetti sono specifici del	HTTPS	443

Componente	URL	Protocollo	Porta
	sistema operativo e possono variare in base all'area geografica.		

> **Note**
>
> [1] L'accesso agli endpoint AWS SSM è richiesto solo se utilizzi attivazioni ibride AWS SSM per il tuo provider di credenziali IAM locale.
> [2] L'accesso agli endpoint AWS IAM è richiesto solo se utilizzi IAM Roles Anywhere per il tuo provider di credenziali AWS IAM locale.

Accesso richiesto per le operazioni in corso del cluster

Per le operazioni in corso del cluster è necessario il seguente accesso alla rete per il firewall on-premises.

> **Important**
>
> A seconda della CNI scelta, devi configurare ulteriori regole di accesso alla rete per le porte della CNI. Consulta Cilium documentation e Calico documentation per i dettagli.

Tipo	Protocollo	Direzione	Porta	Origine	Destinazione	Utilizzo
HTTPS	TCP	In uscita	443	CIDR del nodo remoto	Cluster EKS 1 IPs	Da kubelet al server API Kubernetes
HTTPS	TCP	In uscita	443	CIDR del pod remoto	Gruppo EKS IPs [1]	Da pod al server API Kubernetes

Tipo	Protocollo	Direzione	Porta	Origine	Destinazione	Utilizzo
HTTPS	TCP	In uscita	443	CIDR del nodo remoto	Endpoint del servizio SSM	Aggiornamento delle credenziali delle attivazioni ibride SSM e heartbeat SSM ogni 5 minuti
HTTPS	TCP	In uscita	443	CIDR del nodo remoto	Endpoint del servizio IAM Anywhere	Aggiornamento delle credenziali di IAM Roles Anywhere
HTTPS	TCP	In uscita	443	CIDR del pod remoto	Endpoint regionali STS	Da pod a endpoint STS, richiesto solo per IRSA

Tipo	Protocollo	Direzione	Porta	Origine	Destinazione	Utilizzo
HTTPS	TCP	In uscita	443	CIDR del nodo remoto	Endpoint del servizio di autenticazione Amazon EKS	Da nodo a endpoint di autenticazione Amazon EKS, richiesto solo per Amazon EKS Pod Identity
HTTPS	TCP	In entrata	10250	Gruppo EKS IPs [1]	CIDR del nodo remoto	Da server API Kubernetes a kubelet
HTTPS	TCP	In entrata	Porte del webhook	Gruppo EKS IPs [1]	CIDR del pod remoto	Da server API Kubernetes a webhook

Tipo	Protocollo	Direzione	Porta	Origine	Destinazione	Utilizzo
HTTPS	TCP, UDP	Inbound, Outbound	53	CIDR del pod remoto	CIDR del pod remoto	Da pod a CoreDNS. Se esegui almeno una replica di CoreDNS nel cloud, devi consentire il traffico DNS al VPC su cui è in esecuzione CoreDNS.
Definito dall'utente	Definito dall'utente	Inbound, Outbound	Porte dell'app	CIDR del pod remoto	CIDR del pod remoto	Da pod a pod

Note

[1] Il IPs cluster EKS. Consulta la sezione seguente sulle interfacce di rete elastiche di Amazon EKS.

Interfacce di rete elastiche di Amazon EKS

Amazon EKS collega le interfacce di rete alle sottoreti del VPC che passi durante la creazione del cluster per abilitare la comunicazione tra il piano di controllo EKS e il tuo VPC. Le interfacce di rete create da Amazon EKS possono essere trovate dopo la creazione del cluster nella EC2 console Amazon o con la AWS CLI. Le interfacce di rete originali vengono eliminate e vengono create di nuove quando vengono applicate modifiche al cluster EKS, ad esempio gli aggiornamenti delle versioni di Kubernetes. Puoi limitare l'intervallo IP per le interfacce di rete Amazon EKS utilizzando

dimensioni di sottorete vincolate per le sottoreti che passate durante la creazione del cluster, il che semplifica la configurazione del firewall locale per consentire la inbound/outbound connettività a questo insieme noto e vincolato di. IPs Per verificare in quali sottoreti vengono create le interfacce di rete, puoi limitare il numero di sottoreti specificate a due quando crei un cluster o puoi aggiornare le sottoreti dopo aver creato il cluster.

Le interfacce di rete fornite da Amazon EKS hanno una descrizione del formato `Amazon EKS` *`your-cluster-name`* . Guarda l'esempio seguente per un comando AWS CLI che puoi usare per trovare gli indirizzi IP delle interfacce di rete fornite da Amazon EKS. Sostituisci `VPC_ID` con l'ID del VPC che passi durante la creazione del cluster.

```
aws ec2 describe-network-interfaces \
--query 'NetworkInterfaces[?(VpcId == VPC_ID && contains(Description,Amazon
 EKS))].PrivateIpAddress'
```

AWS Configurazione di VPC e sottorete

Ai cluster con nodi ibridi si applicano gli attuali Requisiti per VPC e sottoreti per Amazon EKS. Inoltre, il tuo VPC CIDR non può sovrapporsi al nodo e al pod locali. CIDRs È necessario configurare i percorsi nella tabella di routing VPC per il nodo locale e, facoltativamente, il pod. CIDRs Questi percorsi devono essere configurati per indirizzare il traffico verso il gateway utilizzato per la connettività di rete ibrida, che in genere è un gateway privato virtuale (VGW) o un gateway di transito (TGW). Se utilizzi TGW o VGW per connettere il tuo VPC all'ambiente on-premises, devi creare un allegato TGW o VGW per il tuo VPC. Il VPC deve supportare il nome host DNS e la risoluzione DNS.

I passaggi seguenti utilizzano la AWS CLI. Puoi anche creare queste risorse in Console di gestione AWS o con altre interfacce come AWS CDK o AWS CloudFormation Terraform.

Passaggio 1: creazione di un VPC

1. Per creare un nuovo VPC, esegui il comando seguente. Sostituisci `VPC_CIDR` con un intervallo CIDR `IPv4` RFC-1918 (privato) o non RFC-1918 (pubblico) (ad esempio `10.0.0.0/16`). Nota: la risoluzione DNS, che è un requisito EKS, è abilitata per impostazione predefinita per il VPC.

   ```
   aws ec2 create-vpc --cidr-block VPC_CIDR
   ```

2. Abilita i nomi degli host DNS per il tuo VPC. Ricorda, la risoluzione DNS è abilitata per impostazione predefinita per il VPC. Sostituisci `VPC_ID` con l'ID del VPC creato nel passaggio precedente.

```
aws ec2 modify-vpc-attribute --vpc-id VPC_ID --enable-dns-hostnames
```

Passaggio 2: creazione di sottoreti

Crea almeno due sottoreti. Amazon EKS utilizza queste sottoreti per le interfacce di rete del cluster. Per maggiori informazioni, consulta Subnets requirements and considerations.

1. È possibile trovare le zone di disponibilità per una AWS regione con il seguente comando. Sostituisci `us-west-2` con la tua regione.

```
aws ec2 describe-availability-zones \
     --query 'AvailabilityZones[?(RegionName == us-west-2)].ZoneName'
```

2. Crea una sottorete. Sostituisci `VPC_ID` con l'ID del VPC. Sostituisci `SUBNET_CIDR` con l'intervallo CIDR per la sottorete (ad esempio, 10.0.1.0/24). Sostituisci `AZ` con la zona di disponibilità in cui verrà creata la sottorete (ad esempio us-west-2a). Le sottoreti che crei devono trovarsi in almeno due zone di disponibilità differenti.

```
aws ec2 create-subnet \
    --vpc-id VPC_ID \
    --cidr-block SUBNET_CIDR \
    --availability-zone AZ
```

(Facoltativo) Fase 3: collegare il VPC con Amazon VPC Transit Gateway (TGW) o il gateway privato virtuale AWS Direct Connect (VGW)

Se utilizzi un TGW o un VGW, collega il tuo VPC al TGW o al VGW. Per ulteriori informazioni, consulta Amazon VPC attachments in Amazon VPC Transit Gateways o AWS Direct Connect virtual private gateway associations.

Gateway di transito

Esegui il comando seguente per collegare un gateway di transito. Sostituisci `VPC_ID` con l'ID del VPC. Sostituisci `SUBNET_ID1` e `SUBNET_ID2` con le IDs sottoreti create nel passaggio precedente. Sostituisci `TGW_ID` con l'ID del TGW.

```
aws ec2 create-transit-gateway-vpc-attachment \
```

```
    --vpc-id VPC_ID \
    --subnet-ids SUBNET_ID1 SUBNET_ID2 \
    --transit-gateway-id TGW_ID
```

Gateway virtuale privato

Esegui il comando seguente per collegare un Gateway di transito. Sostituisci `VPN_ID` con l'ID del VGW. Sostituisci `VPC_ID` con l'ID del VPC.

```
aws ec2 attach-vpn-gateway \
    --vpn-gateway-id VPN_ID \
    --vpc-id VPC_ID
```

(Facoltativo) Passaggio 4: creazione di una tabella di routing

Puoi modificare la tabella di routing principale per il VPC o creare una tabella di routing personalizzata. I passaggi seguenti creano una tabella di routing personalizzata con le rotte verso il nodo e il pod locali. CIDRs Per ulteriori informazioni, consulta le Tabelle di routing delle sottoreti. Sostituisci `VPC_ID` con l'ID del VPC.

```
aws ec2 create-route-table --vpc-id VPC_ID
```

Passaggio 5: creazione di percorsi per nodi e pod on-premises

Crea percorsi nella tabella di routing per ciascuno dei tuoi nodi remoti on-premises. Puoi modificare la tabella di routing principale per il VPC o utilizzare la tabella di routing personalizzata creata nel passaggio precedente.

Gli esempi seguenti mostrano come creare percorsi per il nodo e il pod locali. CIDRs Negli esempi viene utilizzato un gateway di transito (TGW) per connettere il VPC all'ambiente on-premises. Se disponi di più nodi e pod locali CIDRs, ripeti i passaggi per ogni CIDR.

- Se utilizzi un gateway Internet o un gateway privato virtuale (VGW), sostituisci `--transit-gateway-id` con `--gateway-id`.
- Sostituisci `RT_ID` con l'ID della tabella di routing creata nel passaggio precedente.
- Sostituisci `REMOTE_NODE_CIDR` con l'intervallo CIDR che utilizzerai per i tuoi nodi ibridi.

- Sostituisci `REMOTE_POD_CIDR` con l'intervallo CIDR che utilizzerai per i pod in esecuzione su nodi ibridi. L'intervallo CIDR dei pod corrisponde alla configurazione Container Networking Interface (CNI), che comunemente utilizza una rete overlay on-premises. Per ulteriori informazioni, consulta the section called "Configurazione della CNI".
- Sostituisci `TGW_ID` con l'ID del TGW.

Rete di nodi remoti

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_NODE_CIDR \
    --transit-gateway-id TGW_ID
```

Rete di pod remoti

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_POD_CIDR \
    --transit-gateway-id TGW_ID
```

(Facoltativo) Passaggio 6: associazione delle sottoreti alla tabella di routing

Se hai creato una tabella di routing personalizzata nel passaggio precedente, associa ciascuna delle sottoreti che hai creato nel passaggio precedente alla tua tabella di routing personalizzata. Se stai modificando la tabella di routing principale del VPC, le sottoreti vengono automaticamente associate alla tabella di routing principale del VPC e puoi saltare questo passaggio.

Esegui questo comando per ciascuna sottorete creata nei passaggi precedenti. Sostituisci `RT_ID` con la tabella di routing creata nel passaggio precedente. Sostituisci `SUBNET_ID` con l'ID di una sottorete.

```
aws ec2 associate-route-table --route-table-id RT_ID --subnet-id SUBNET_ID
```

Configurazione del gruppo di sicurezza del cluster

Il seguente accesso per il gruppo di sicurezza del cluster EKS è necessario per le operazioni in corso del cluster. Amazon EKS crea automaticamente le regole dei gruppi di sicurezza in entrata

richieste per i nodi ibridi quando crei o aggiorni il cluster con reti di nodi e pod remoti configurate. Poiché i gruppi di sicurezza consentono tutto il traffico in uscita per impostazione predefinita, Amazon EKS non modifica automaticamente le regole in uscita del gruppo di sicurezza del cluster per i nodi ibridi. Se desideri personalizzare il gruppo di sicurezza del cluster, puoi limitare il traffico alle regole riportate nella tabella seguente.

Tipo	Protocollo	Direzione	Porta	Origine	Destinazione	Utilizzo
HTTPS	TCP	In entrata	443	CIDR del nodo remoto	N/D	Da Kubelet al server API Kubernetes
HTTPS	TCP	In entrata	443	CIDR del pod remoto	N/D	Pod che richiedono l'accesso al server API K8s quando il CNI non utilizza NAT per il traffico dei pod.
HTTPS	TCP	In uscita	10250	N/D	CIDR del nodo remoto	Da server API Kubernetes a Kubelet
HTTPS	TCP	In uscita	Porte del webhook	N/D	CIDR del pod remoto	Da server API Kubernetes a webhook (se esegui webhook

Tipo	Protocollo	Direzione	Porta	Origine	Destinazione	Utilizzo
						su nodi ibridi)

> **Important**
>
> Limiti delle regole dei gruppi di sicurezza: i gruppi EC2 di sicurezza di Amazon hanno un massimo di 60 regole in entrata per impostazione predefinita. Le regole in entrata del gruppo di sicurezza potrebbero non essere applicabili se il gruppo di sicurezza del cluster si avvicina a questo limite. In questo caso, potrebbe essere necessario aggiungere manualmente le regole in entrata mancanti.
> Responsabilità di pulizia CIDR: se rimuovi reti di nodi o pod remoti dai cluster EKS, quest'ultimo non rimuove automaticamente le regole del gruppo di sicurezza corrispondenti. È responsabilità dell'utente rimuovere manualmente le reti di nodi o pod remoti non utilizzate dalle regole del gruppo di sicurezza.

Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza".

(Facoltativo) Configurazione manuale del gruppo di sicurezza

Se devi creare gruppi di sicurezza aggiuntivi o modificare le regole create automaticamente, puoi utilizzare i seguenti comandi come riferimento. Per impostazione predefinita, il comando seguente crea un gruppo di sicurezza che consente tutti gli accessi in uscita. Puoi limitare l'accesso in uscita in modo da includere solo le regole precedenti. Se stai pianificando una limitazione delle regole, ti consigliamo di testare accuratamente la connettività delle applicazioni e dei pod prima di applicare le regole modificate a un cluster di produzione.

- Nel primo comando, sostituisci `SG_NAME` con un nome per il tuo gruppo di sicurezza
- Nel primo comando, sostituisci `VPC_ID` con l'ID del VPC creato nel passaggio precedente
- Nel secondo comando, sostituisci `SG_ID` con l'ID del gruppo di sicurezza creato nel primo comando
- Nel secondo comando, sostituisci `REMOTE_NODE_CIDR` e `REMOTE_POD_CIDR` con i valori per i nodi ibridi e la rete 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"}]}]'
```

Preparazione del sistema operativo per i nodi ibridi

Bottlerocket, Amazon Linux 2023 (AL2023), Ubuntu e RHEL vengono convalidati su base continuativa per l'uso come sistema operativo di nodi per nodi ibridi. Bottlerocket è supportato solo da AWS ambienti VMware vSphere. AL2023 non è coperto dai piani di AWS supporto se eseguito al di fuori di Amazon EC2. AL2023 può essere utilizzato solo in ambienti virtualizzati locali, consulta la Guida per l'utente di Amazon Linux 2023 per ulteriori informazioni. AWS supporta l'integrazione dei nodi ibridi con i sistemi operativi Ubuntu e RHEL ma non fornisce supporto per il sistema operativo stesso.

Il provisioning e la gestione del sistema operativo sono una tua responsabilità. Quando testi i nodi ibridi per la prima volta, è più semplice eseguire la CLI Amazon EKS Hybrid Nodes (`nodeadm`) su un host già fornito. Per le implementazioni di produzione, ti consigliamo di includere `nodeadm` nelle immagini del tuo sistema operativo configurate per l'esecuzione come servizio systemd per unire automaticamente gli host ai cluster Amazon EKS all'avvio dell'host. Se utilizzi Bottlerocket come sistema operativo del nodo su vSphere, non è necessario utilizzare `nodeadm` poiché Bottlerocket contiene già le dipendenze necessarie per i nodi ibridi e si connetterà automaticamente al cluster configurato all'avvio dell'host.

Compatibilità delle versioni

La tabella seguente rappresenta le versioni del sistema operativo compatibili e convalidate per l'uso come sistema operativo dei nodi per i nodi ibridi. Se si utilizzano altre varianti o versioni del sistema operativo non incluse in questa tabella, la compatibilità dei nodi ibridi con la variante o la versione del sistema operativo non è coperta da AWS Support. I nodi ibridi sono indipendenti dall'infrastruttura sottostante e supportano le architetture x86 e ARM.

Sistema operativo	Versioni
Amazon Linux	Amazon Linux 2023 (AL2023)
Portabottiglie	v1.37.0 e versioni successive che eseguono Kubernetes v1.28 e VMware versioni successive
Ubuntu	Ubuntu 20.04, Ubuntu 22.04, Ubuntu 24.04
Red Hat Enterprise Linux	RHEL 8, RHEL 9

Considerazioni sul sistema operativo

Ambito generale

- La CLI Amazon EKS Hybrid Nodes (`nodeadm`) può essere utilizzata per semplificare l'installazione e la configurazione dei componenti e delle dipendenze dei nodi ibridi. Puoi eseguire il processo `nodeadm install` durante le pipeline di creazione dell'immagine del sistema operativo o in Passaggio di runtime su ogni host on-premises. Per ulteriori informazioni sui componenti che `nodeadm` installa, consulta the section called "nodeadm dei nodi ibridi".
- Se utilizzi un proxy nell'ambiente on-premises per accedere a Internet, è necessaria una configurazione aggiuntiva del sistema operativo per i processi di installazione e aggiornamento per configurare il gestore di pacchetti per l'utilizzo del proxy. Per istruzioni, consulta the section called "Configurazione del proxy".

Portabottiglie

- I passaggi e gli strumenti per connettere un nodo Bottlerocket sono diversi da quelli per altri sistemi operativi e sono descritti separatamente in the section called "Connessione dei nodi ibridi con Bottlerocket", anziché nelle istruzioni di the section called "Connessione di nodi ibridi".
- I passaggi per Bottlerocket non utilizzano lo strumento CLI dei nodi ibridi, `nodeadm`.
- Solo VMware le varianti della versione Bottlerocket v1.37.0 e successive sono supportate con EKS Hybrid Nodes. VMware le varianti di Bottlerocket sono disponibili per le versioni Kubernetes v1.28 e successive. Altre varianti di Bottlerocket non sono supportate come sistema operativo dei nodi ibridi. NOTA: le VMware varianti di Bottlerocket sono disponibili solo per l'architettura x86_64.

Containerd

- Containerd è il runtime del container Kubernetes standard ed è una dipendenza per i nodi ibridi, nonché per tutti i tipi di calcolo dei nodi Amazon EKS. La CLI di Amazon EKS Hybrid Nodes (`nodeadm`) tenta di installare containerd durante il processo `nodeadm install`. È possibile configurare l'installazione containerd nella Passaggio di runtime di `nodeadm install` con l'opzione `--containerd-source` della riga di comando. Le opzioni valide sono `none`, `distro` e `docker`. Se stai usando RHEL, `distro` non è un'opzione valida e puoi configurare `nodeadm` per installare la build containerd dai repository di Docker oppure puoi installare containerd manualmente. Quando si usa AL2 023 o Ubuntu, per `nodeadm` impostazione predefinita installa containerd dalla distribuzione del sistema operativo. Se non vuoi che nodeadm installi containerd, usa l'opzione `--containerd-source none`.

Ubuntu

- Se usi Ubuntu 24.04, potresti dover aggiornare la tua versione di containerd o modificare AppArmor la configurazione per adottare una correzione che consenta ai pod di terminare correttamente, vedi Ubuntu #2065423. È necessario un riavvio per applicare le modifiche al profilo. AppArmor L'ultima versione di Ubuntu 24.04 ha una versione containerd aggiornata nel suo gestore di pacchetti con la correzione (versione containerd 1.7.19 o successive).

ARM

- Se si utilizza hardware ARM, è necessario un processore compatibile con ARMv8 .2 con l'estensione di crittografia (ARMv8.2+crypto) per eseguire la versione 1.31 e successive del componente aggiuntivo EKS kube-proxy. Tutti i sistemi Raspberry Pi precedenti al Raspberry Pi 5, così come i processori basati su Cortex-A72, non soddisfano questo requisito. Come soluzione alternativa, puoi continuare a utilizzare la versione 1.30 del componente aggiuntivo EKS kube-proxy fino alla fine del supporto esteso a luglio del 2026, consultare il Calendario delle versioni di Kubernetes o utilizzare un'immagine kube-proxy personalizzata dall'upstream.
- Il seguente messaggio di errore nel registro kube-proxy indica questa incompatibilità:

```
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.
```

Creazione di immagini del sistema operativo

Amazon EKS fornisce modelli di Packer di esempio che puoi utilizzare per creare immagini del sistema operativo che includono `nodeadm` e lo configurano per l'esecuzione all'avvio dell'host. Questa procedura è consigliata per evitare di trasferire le dipendenze dei nodi ibridi singolarmente su ciascun host e per automatizzare il processo di bootstrap dei nodi ibridi. Puoi utilizzare i modelli Packer di esempio con un'immagine ISO di Ubuntu 22.04, Ubuntu 24.04, RHEL 8 o RHEL 9 e produrre immagini con questi formati: OVA, Qcow2 o raw.

Prerequisiti

Prima di utilizzare i modelli Packer di esempio, è necessario che sul computer da cui esegui Packer sia installato quanto segue.

- Packer versione 1.11.0 o successiva. Per istruzioni sull'installazione di Packer, consulta Install Packer nella documentazione di Packer.
- In fase di compilazione OVAs, VMware vSphere plugin 1.4.0 o versione successiva
- Se stai creando immagini `Qcow2` o grezze, plug-in QEMU versione 1.x

Impostazione delle variabili di ambiente

Prima di eseguire la build di Packer, imposta le seguenti variabili di ambiente sul computer da cui stai eseguendo Packer.

Ambito generale

Le seguenti variabili di ambiente devono essere impostate per creare immagini con tutti i sistemi operativi e i formati di output.

Variabile di ambiente	Tipo	Description
PKR_SSH_PASSWORD	Stringa	Packer utilizza le variabili `ssh_username` e `ssh_password` di SSH nella macchina creata durante il provisioning. Ciò deve corrispondere alle password utilizzate per creare l'utente iniziale all'interno dei file

Variabile di ambiente	Tipo	Description
		kickstart o di dati utente del rispettivo sistema operativo. L'impostazione predefinita è "builder" o "ubuntu" a seconda del sistema operativo. Quando imposti la password, assicurati di cambiarla all'interno del file `ks.cfg` o `user-data` corrispondente.
ISO_URL	Stringa	URL dell'ISO da utilizzare. Può essere un collegamento web da scaricare da un server o un percorso assoluto verso un file locale
ISO_CHECKSUM	Stringa	Checksum associato per l'ISO fornito.
CREDENTIAL_PROVIDER	Stringa	Provider di credenziali per nodi ibridi. I valori validi sono `ssm` (impostazione predefinita) per le attivazioni ibride SSM e `iam` per IAM Roles Anywhere
K8S_VERSION	Stringa	Versione di Kubernetes per nodi ibridi (ad esempio `1.31`). Per le versioni di Kubernetes supportate, consulta la pagina Amazon EKS supported versions.
NODEADM_ARCH	Stringa	Architettura per `nodeadm install`. Seleziona `amd` oppure `arm`.

RHEL

Se utilizzi RHEL, è necessario impostare le seguenti variabili di ambiente.

Variabile di ambiente	Tipo	Description
RH_USERNAME	Stringa	nome utente del gestore di sottoscrizioni RHEL
RH_PASSWORD	Stringa	password del gestore delle sottoscrizioni RHEL
RHEL_VERSION	Stringa	Versione iso di Rhel in uso. I valori validi sono 8 e 9.

Ubuntu

Non sono richieste variabili di ambiente specifiche per Ubuntu.

vSphere

Se si sta creando un VMware vSphere OVA, è necessario impostare le seguenti variabili di ambiente.

Variabile di ambiente	Tipo	Description
VSPHERE_SERVER	Stringa	Indirizzo del server vSphere
VSPHERE_USER	Stringa	Nome utente vSphere
VSPHERE_PASSWORD	Stringa	Password vSphere
VSPHERE_DATACENTER	Stringa	Nome del data center vSphere
VSPHERE_CLUSTER	Stringa	Nome del cluster vSphere
VSPHERE_DATASTORE	Stringa	Nome del datastore vSphere
VSPHERE_NETWORK	Stringa	Nome della rete vSphere

Variabile di ambiente	Tipo	Description
VSPHERE_OUTPUT_FOLDER	Stringa	Cartella di output vSphere per i modelli

QEMU

Variabile di ambiente	Tipo	Description
PACKER_OUTPUT_FORMAT	Stringa	Formato di output per il generatore QEMU. I valori validi sono `qcow2` e `raw`.

Convalida del modello

Prima di eseguire la build, convalida il modello con il seguente comando dopo aver impostato le variabili di ambiente. Sostituisci `template.pkr.hcl` se stai usando un nome diverso per il tuo modello.

```
packer validate template.pkr.hcl
```

Creazione di immagini

Crea le tue immagini con i seguenti comandi e usa il flag `-only` per specificare la destinazione e il sistema operativo delle tue immagini. Sostituisci `template.pkr.hcl` se stai usando un nome diverso per il tuo modello.

vSphere OVAs

> **Note**
>
> Se utilizzi RHEL con vSphere, devi convertire i file kickstart in un'immagine OEMDRV e passarla come ISO da cui avviare il sistema. Per ulteriori informazioni, consulta il file Readme di Packer nell'archivio EKS Hybrid Nodes. GitHub

OVA Ubuntu 22.04

```
packer build -only=general-build.vsphere-iso.ubuntu22 template.pkr.hcl
```

OVA Ubuntu 24.04

```
packer build -only=general-build.vsphere-iso.ubuntu24 template.pkr.hcl
```

OVA RHEL 8

```
packer build -only=general-build.vsphere-iso.rhel8 template.pkr.hcl
```

OVA RHEL 9

```
packer build -only=general-build.vsphere-iso.rhel9 template.pkr.hcl
```

QEMU

> **Note**
>
> Se stai creando un'immagine per una CPU host specifica che non corrisponde all'host del tuo generatore, consulta la documentazione QEMU per cercare il nome che corrisponde alla tua CPU host e usa il flag `-cpu` con il nome della CPU host quando esegui i seguenti comandi.

Qcow2/Raw Ubuntu 22.04

```
packer build -only=general-build.qemu.ubuntu22 template.pkr.hcl
```

Qcow2/Raw Ubuntu 24.04

```
packer build -only=general-build.qemu.ubuntu24 template.pkr.hcl
```

Qcow2/Raw RHEL 8

```
packer build -only=general-build.qemu.rhel8 template.pkr.hcl
```

Qcow2/Raw RHEL 9

```
packer build -only=general-build.qemu.rhel9 template.pkr.hcl
```

Trasferimento della configurazione di nodeadm tramite i dati utente

Puoi trasferire la configurazione per `nodeadm` ai tuoi dati utente tramite cloud-init per configurare e connettere automaticamente i nodi ibridi al cluster EKS all'avvio dell'host. Di seguito è riportato un esempio di come eseguire questa operazione quando si utilizza VMware vSphere come infrastruttura per i nodi ibridi.

1. Installa la `govc` CLI seguendo le istruzioni nel readme di govc su. GitHub
2. Dopo aver eseguito la build di Packer nella sezione precedente e aver effettuato il provisioning del modello, puoi clonare il modello per creare più nodi diversi utilizzando quanto segue. Devi clonare il modello per ogni nuova macchina virtuale che stai creando e che verrà utilizzata per i nodi ibridi. Sostituisci le variabili nel comando seguente con i valori per il tuo ambiente. Il `VM_NAME` comando seguente viene usato come tuo `NODE_NAME` quando inserisci i nomi per te tramite il tuo file. VMs `metadata.yaml`

   ```
   govc vm.clone -vm "/PATH/TO/TEMPLATE" -ds="YOUR_DATASTORE" \
       -on=false -template=false -folder=/FOLDER/TO/SAVE/VM "VM_NAME"
   ```

3. Dopo aver clonato il modello per ognuno dei tuoi nuovi VMs, crea un `userdata.yaml` e `metadata.yaml` per il tuo. VMs VMs Puoi condividerli `userdata.yaml` `metadata.yaml` e li compilerai per ogni singola macchina virtuale nei passaggi seguenti. La configurazione di `nodeadm` viene creata e definita nella sezione `write_files` del tuo `userdata.yaml`. L'esempio seguente utilizza le attivazioni ibride AWS SSM come provider di credenziali locali per i nodi ibridi. Per ulteriori informazioni sulla configurazione di `nodeadm`, consulta the section called "nodeadm dei nodi ibridi".

   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:

Crea un `metadata.yaml` per il tuo ambiente. Mantieni il formato della variabile `"$NODE_NAME"` nel file poiché questo verrà compilato con valori in un passaggio successivo.

```
instance-id: "$NODE_NAME"
local-hostname: "$NODE_NAME"
network:
  version: 2
  ethernets:
    nics:
      match:
        name: ens*
      dhcp4: yes
```

4. Aggiungi i file `userdata.yaml` e `metadata.yaml` come stringhe `gzip+base64` con i seguenti comandi. I seguenti comandi devono essere eseguiti per ciascuno dei comandi che si stanno creando. VMs Sostituisci `VM_NAME` con il nome della macchina virtuale che stai aggiornando.

```
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. Accendi il tuo nuovo VMs, che dovrebbe connettersi automaticamente al cluster EKS che hai configurato.

```
govc vm.power -on "${NODE_NAME}"
```

Preparazione delle credenziali per i nodi ibridi

I nodi ibridi Amazon EKS utilizzano credenziali IAM temporanee fornite da attivazioni ibride AWS SSM o AWS IAM Roles Anywhere per l'autenticazione con il cluster Amazon EKS. È necessario utilizzare attivazioni ibride AWS SSM o AWS IAM Roles Anywhere con l'Amazon EKS Hybrid Nodes CLI (). `nodeadm` Non dovresti usare sia le attivazioni ibride AWS SSM che IAM Roles Anywhere. AWS Ti consigliamo di utilizzare le attivazioni ibride AWS SSM se non disponi di un'infrastruttura a chiave pubblica (PKI) esistente con un'autorità di certificazione (CA) e certificati per i tuoi ambienti locali. Se disponi di PKI e certificati esistenti in locale, usa IAM Roles Anywhere. AWS

Ruolo IAM dei nodi ibridi

Prima di poter connettere i nodi ibridi al cluster Amazon EKS, devi creare un ruolo IAM che verrà utilizzato con le attivazioni ibride AWS SSM o AWS IAM Roles Anywhere per le credenziali dei tuoi nodi ibridi. Dopo la creazione del cluster, utilizzerai questo ruolo con una voce o `aws-auth` ConfigMap una voce di accesso Amazon EKS per mappare il ruolo IAM a Kubernetes Role-Based Access Control (RBAC). Per ulteriori informazioni sull'associazione del ruolo IAM dei nodi ibridi con il RBAC di Kubernetes, consulta the section called "Preparazione dell'accesso ai cluster".

Il ruolo IAM dei nodi ibridi deve disporre delle seguenti autorizzazioni.

- Autorizzazioni per utilizzare l'`eks:DescribeCluster`azione `nodeadm` per raccogliere informazioni sul cluster a cui desideri connettere i nodi ibridi. Se non abiliti l'`eks:DescribeCluster`azione, devi passare l'endpoint dell'API Kubernetes, il cluster CA bundle e il servizio IPv4 CIDR nella configurazione del nodo che passi al comando. `nodeadm init`

- Autorizzazioni per `nodeadm` utilizzare l'`eks:ListAccessEntries`azione per elencare le voci di accesso sul cluster a cui desideri connettere i nodi ibridi. Se non abiliti l'`eks:ListAccessEntries`azione, devi passare il `--skip cluster-access-validation` flag quando esegui il `nodeadm init` comando.
- Autorizzazioni per il kubelet a utilizzare le immagini dei contenitori da Amazon Elastic Container Registry (Amazon ECR) come definito nella policy di Amazon. EC2 ContainerRegistryPullOnly
- Se si utilizza AWS SSM, le autorizzazioni per `nodeadm init` utilizzare le attivazioni ibride AWS SSM come definito nella policy. https://docs.aws.amazon.com/aws-managed-policy/latest/reference/AmazonSSMManagedInstanceCore.html
- Se si utilizza AWS SSM, le autorizzazioni per utilizzare l'`ssm:DeregisterManagedInstance`azione e `ssm:DescribeInstanceInformation` l'azione per annullare la registrazione delle istanze. `nodeadm uninstall`
- (Facoltativo) Autorizzazioni per consentire al Pod Identity Agent di Amazon EKS di utilizzare l'azione `eks-auth:AssumeRoleForPodIdentity` per recuperare le credenziali per i pod.

AWS Configura le attivazioni ibride SSM

Prima di configurare le attivazioni ibride AWS SSM, è necessario creare e configurare un ruolo IAM Hybrid Nodes. Per ulteriori informazioni, consulta the section called "Creazione del ruolo IAM dei nodi ibridi". Segui le istruzioni in Creare un'attivazione ibrida per registrare i nodi con Systems Manager nella AWS Systems Manager User Guide per creare un'attivazione ibrida AWS SSM per i tuoi nodi ibridi. Il codice di attivazione e l'ID che ricevi vengono utilizzati con `nodeadm` quando registri i tuoi host come nodi ibridi con il tuo cluster Amazon EKS. Puoi tornare a questo passaggio in un secondo momento dopo aver creato e preparato i cluster Amazon EKS per i nodi ibridi.

> **Important**
>
> Systems Manager restituisce immediatamente il codice e l'ID di attivazione alla console o finestra di comando, a seconda di come è stato creata l'attivazione. Copia queste informazioni e archiviale in un luogo sicuro. Se esci dalla console o chiudi la finestra di comando, potresti perdere queste informazioni. Se le smarrisci, devi creare una nuova attivazione.

Per impostazione predefinita, le attivazioni ibride AWS SSM sono attive per 24 ore. In alternativa, puoi specificare una `--expiration-date` quando crei l'attivazione ibrida in formato timestamp, ad

esempio `2024-08-01T00:00:00`. Quando utilizzi AWS SSM come provider di credenziali, il nome del nodo per i nodi ibridi non è configurabile e viene generato automaticamente da SSM. AWS È possibile visualizzare e gestire le istanze gestite AWS SSM nella console AWS Systems Manager in Fleet Manager. È possibile registrare fino a 1.000 nodi standard attivati in modalità ibrida per account per AWS regione senza costi aggiuntivi. Tuttavia, per registrare più di 1.000 nodi ibridi è necessario attivare il piano istanze avanzate. Viene addebitato un costo per l'utilizzo del piano istanze avanzate che non è incluso nel prezzo di Amazon EKS Hybrid Nodes. Per ulteriori informazioni, consulta Prezzi di AWS Systems Manager.

Vedi l'esempio seguente per come creare un'attivazione ibrida AWS SSM con il ruolo IAM di Hybrid Nodes. Quando utilizzi le attivazioni ibride AWS SSM per le credenziali dei tuoi nodi ibridi, i nomi dei tuoi nodi ibridi avranno il formato `mi-012345678abcdefgh` e le credenziali temporanee fornite da AWS SSM sono valide per 1 ora. Non è possibile modificare il nome del nodo o la durata delle credenziali quando si utilizza SSM come provider di credenziali. AWS Le credenziali temporanee vengono ruotate automaticamente da AWS SSM e la rotazione non influisce sullo stato dei nodi o delle applicazioni.

Ti consigliamo di utilizzare un'attivazione ibrida AWS SSM per cluster EKS per definire l'`ssm:DeregisterManagedInstance`autorizzazione AWS SSM del ruolo IAM di Hybrid Nodes per poter annullare la registrazione solo delle istanze associate all'attivazione ibrida SSM. AWS Nell'esempio in questa pagina, viene utilizzato un tag con l'ARN del cluster EKS, che può essere utilizzato per mappare l'attivazione ibrida AWS SSM al cluster EKS. In alternativa, puoi utilizzare il tag e il metodo preferiti per definire l'ambito delle autorizzazioni AWS SSM in base ai limiti e ai requisiti di autorizzazione. L'`REGISTRATION_LIMIT`opzione nel comando seguente è un numero intero utilizzato per limitare il numero di macchine che possono utilizzare l'attivazione ibrida AWS SSM (ad esempio) `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
```

Consulta le istruzioni su Creare un'attivazione ibrida per registrare i nodi con Systems Manager per ulteriori informazioni sulle impostazioni di configurazione disponibili per le attivazioni ibride AWS SSM.

Configura AWS IAM Roles Anywhere

Segui le istruzioni riportate in Getting started with IAM Roles Anywhere nella Guida per l'utente di IAM Roles Anywhere per configurare l'ancora di fiducia e il profilo che utilizzerai per le credenziali IAM temporanee per il tuo ruolo IAM dei nodi ibridi. Quando crei il tuo profilo, puoi crearlo senza aggiungere alcun ruolo. Puoi creare questo profilo, tornare a questi passaggi per creare il tuo ruolo IAM dei nodi ibridi e quindi aggiungere il tuo ruolo al profilo dopo averlo creato. In alternativa, puoi utilizzare i AWS CloudFormation passaggi riportati più avanti in questa pagina per completare la configurazione di IAM Roles Anywhere per i nodi ibridi.

Quando aggiungi il ruolo IAM Hybrid Nodes al tuo profilo, seleziona Accetta il nome della sessione di ruolo personalizzato nel pannello Nome sessione di ruolo personalizzato nella parte inferiore della pagina Modifica profilo nella console AWS IAM Roles Anywhere. Corrisponde al campo acceptRoleSessionNome dell'`CreateProfile`API. Ciò consente di fornire un nome di nodo personalizzato per i nodi ibridi nella configurazione a cui si passa durante il processo di bootstrap `nodeadm`. È necessario passare un nome di nodo personalizzato durante il processo `nodeadm init`. Puoi aggiornare il tuo profilo per accettare un nome di sessione di ruolo personalizzato dopo aver creato il tuo profilo.

Puoi configurare la durata di validità delle credenziali con AWS IAM Roles Anywhere tramite il campo DurationSeconds del AWS tuo profilo IAM Roles Anywhere. La durata predefinita è di 1 ora con un massimo di 12 ore. L'`MaxSessionDuration`impostazione sul tuo ruolo IAM Hybrid Nodes deve essere maggiore dell'`durationSeconds`impostazione sul tuo profilo AWS IAM Roles Anywhere. Per ulteriori informazioni`MaxSessionDuration`, consulta la documentazione UpdateRole dell'API.

I certificati e le chiavi per computer generati dalla tua autorità di certificazione (CA) devono essere inseriti nella directory `/etc/iam/pki` di ogni nodo ibrido con i nomi dei file `server.pem` per il certificato e `server.key` per la chiave.

Creazione del ruolo IAM dei nodi ibridi

Per eseguire i passaggi di questa sezione, il principale IAM che utilizza la AWS console o la AWS CLI deve disporre delle seguenti autorizzazioni.

- `iam:CreatePolicy`
- `iam:CreateRole`
- `iam:AttachRolePolicy`
- Se si utilizza AWS IAM Roles Anywhere
  - `rolesanywhere:CreateTrustAnchor`

- `rolesanywhere:CreateProfile`
- `iam:PassRole`

AWS CloudFormation

Installa e configura la AWS CLI, se non l'hai già fatto. Vedi Installazione o aggiornamento all'ultima versione della AWS CLI.

Passaggi per le attivazioni AWS ibride SSM

Lo CloudFormation stack crea il ruolo IAM di Hybrid Nodes con le autorizzazioni sopra descritte. Il CloudFormation modello non crea l'attivazione ibrida AWS SSM.

1. Scarica il CloudFormation modello AWS SSM per i nodi ibridi:

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ssm-cfn.yaml'
```

2. Creare un `cfn-ssm-parameters.json` con le seguenti opzioni:
   a. Sostituisci `ROLE_NAME` con il nome del ruolo IAM dei nodi ibridi. Per impostazione predefinita, il CloudFormation modello utilizza `AmazonEKSHybridNodesRole` come nome del ruolo creato se non si specifica un nome.
   b. `TAG_KEY`Sostituiscilo con la chiave del tag di risorsa AWS SSM che hai usato durante la creazione dell'attivazione ibrida AWS SSM. La combinazione della chiave del tag e del valore del tag viene utilizzata nella condizione di consentire solo `ssm:DeregisterManagedInstance` al ruolo IAM di Hybrid Nodes di annullare la registrazione delle istanze gestite AWS SSM associate all'attivazione ibrida SSM. AWS Nel modello, l'impostazione predefinita è. CloudFormation `TAG_KEY` `EKSClusterARN`
   c. Sostituisci `TAG_VALUE` con il valore del tag di risorsa AWS SSM che hai usato durante la creazione dell'attivazione ibrida AWS SSM. La combinazione della chiave del tag e del valore del tag viene utilizzata nella condizione di consentire solo `ssm:DeregisterManagedInstance` al ruolo IAM di Hybrid Nodes di annullare la registrazione delle istanze gestite AWS SSM associate all'attivazione ibrida SSM. AWS Se utilizzi l'impostazione predefinita `TAG_KEY` di `EKSClusterARN`, passa l'ARN del cluster EKS come `TAG_VALUE`. I cluster EKS hanno il formato. ARNs `arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`

```
{
```

```
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "SSMDeregisterConditionTagKey": "TAG_KEY",
    "SSMDeregisterConditionTagValue": "TAG_VALUE"
  }
}
```

3. Distribuisci lo CloudFormation stack. `STACK_NAME`Sostituiscilo con il tuo nome per lo stack 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
```

Passaggi per AWS IAM Roles Anywhere

Lo CloudFormation stack crea il trust anchor di AWS IAM Roles Anywhere con l'autorità di certificazione (CA) che configuri, crea il profilo AWS IAM Roles Anywhere e crea il ruolo IAM di Hybrid Nodes con le autorizzazioni descritte in precedenza.

1. Configurazione di un'autorità di certificazione (CA)
   a. Per utilizzare una risorsa CA AWS privata, apri la console Private AWS Certificate Authority. Segui le istruzioni contenute in AWS Private CA User Guide.
   b. Per utilizzare una CA esterna, segui le istruzioni fornite dalla CA. Fornisci l'ente di certificazione in una fase successiva.
   c. I certificati emessi da fonti pubbliche CAs non possono essere utilizzati come ancoraggi di fiducia.
2. Scarica il CloudFormation modello AWS IAM Roles Anywhere per nodi ibridi

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ira-cfn.yaml'
```

3. Creare un `cfn-iamra-parameters.json` con le seguenti opzioni:
   a. Sostituisci `ROLE_NAME` con il nome del ruolo IAM dei nodi ibridi. Per impostazione predefinita, il CloudFormation modello utilizza `AmazonEKSHybridNodesRole` come nome del ruolo che crea se non specifichi un nome.

b. Sostituisci `CERT_ATTRIBUTE` con l'attributo del certificato per computer che identifica in modo univoco l'host. L'attributo del certificato utilizzato deve corrispondere al nodeName utilizzato per la configurazione `nodeadm` quando si connettono i nodi ibridi al cluster. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi". Per impostazione predefinita, il CloudFormation modello utilizza `${aws:PrincipalTag/x509Subject/CN}` come`CERT_ATTRIBUTE`, che corrisponde al campo CN dei certificati per computer. In alternativa puoi passare `$(aws:PrincipalTag/x509SAN/Name/CN}` come tuo `CERT_ATTRIBUTE`.

c. Sostituisci `CA_CERT_BODY` con l'ente del certificato della tua CA senza interruzioni di riga. Il `CA_CERT_BODY` deve essere in formato Privacy-Enhanced Mail (PEM). Se disponi di un certificato CA in formato PEM, rimuovi le interruzioni di riga e le righe BEGIN CERTIFICATE e END CERTIFICATE prima di inserire l'ente del certificato CA nel file `cfn-iamra-parameters.json`.

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "CertAttributeTrustPolicy": "CERT_ATTRIBUTE",
    "CABundleCert": "CA_CERT_BODY"
  }
}
```

4. Implementa il modello. CloudFormation `STACK_NAME`Sostituiscilo con il tuo nome per lo CloudFormation stack.

```
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

Installa e configura la AWS CLI, se non l'hai già fatto. Vedi Installazione o aggiornamento all'ultima versione della AWS CLI.

Crea la policy del cluster per descrivere EKS

1. Crea un file denominato `eks-describe-cluster-policy.json` con i seguenti contenuti:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Creare la policy con il seguente comando:

```
aws iam create-policy \
    --policy-name EKSDescribeClusterPolicy \
    --policy-document file://eks-describe-cluster-policy.json
```

Passaggi per le attivazioni AWS ibride SSM

1. Crea un file denominato `eks-hybrid-ssm-policy.json` con i seguenti contenuti. La policy concede l'autorizzazione per le azioni `ssm:DescribeInstanceInformation` e `ssm:DeregisterManagedInstance`. La policy limita l'`ssm:DeregisterManagedInstance`autorizzazione alle istanze gestite da AWS SSM associate all'attivazione ibrida AWS SSM in base al tag di risorsa specificato nella politica di attendibilità.
   a. `AWS_REGION`Sostituiscila con la AWS regione per l'attivazione ibrida SSM AWS .
   b. `AWS_ACCOUNT_ID`Sostituiscilo con l'ID AWS del tuo account.
   c. `TAG_KEY`Sostituiscilo con la chiave del tag di risorsa AWS SSM che hai usato durante la creazione dell'attivazione ibrida AWS SSM. La combinazione della chiave del tag e del valore del tag viene utilizzata nella condizione di consentire solo `ssm:DeregisterManagedInstance` al ruolo IAM di Hybrid Nodes di annullare la registrazione delle istanze gestite AWS SSM associate all'attivazione ibrida SSM. AWS Nel modello, l'impostazione predefinita è. CloudFormation `TAG_KEY` `EKSClusterARN`
   d. Sostituisci `TAG_VALUE` con il valore del tag di risorsa AWS SSM che hai usato durante la creazione dell'attivazione ibrida AWS SSM. La combinazione della

chiave del tag e del valore del tag viene utilizzata nella condizione di consentire solo `ssm:DeregisterManagedInstance` al ruolo IAM di Hybrid Nodes di annullare la registrazione delle istanze gestite AWS SSM associate all'attivazione ibrida SSM. AWS Se utilizzi l'impostazione predefinita `TAG_KEY` di `EKSClusterARN`, passa l'ARN del cluster EKS come `TAG_VALUE`. I cluster EKS hanno il formato. ARNs `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. Creare la policy con il seguente comando

```
aws iam create-policy \
    --policy-name EKSHybridSSMPolicy \
    --policy-document file://eks-hybrid-ssm-policy.json
```

3. Crea un file denominato `eks-hybrid-ssm-trust.json`. Sostituiscilo `AWS_REGION` con la AWS regione dell'attivazione ibrida AWS SSM e `AWS_ACCOUNT_ID` con l'ID AWS del tuo account.

```
{
   "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. Crea il ruolo con il comando seguente.

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-ssm-trust.json
```

5. Collega `EKSDescribeClusterPolicy` e `EKSHybridSSMPolicy` che hai creato nei passaggi precedenti. `AWS_ACCOUNT_ID`Sostituiscilo con l'ID AWS del tuo account.

```
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. Allega le politiche `AmazonEC2ContainerRegistryPullOnly` e `AmazonSSMManagedInstanceCore` AWS gestisci.

```
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
```

Passaggi per AWS IAM Roles Anywhere

Per utilizzare AWS IAM Roles Anywhere, devi configurare il tuo trust anchor AWS IAM Roles Anywhere prima di creare il ruolo IAM di Hybrid Nodes. Per istruzioni, consulta the section called "Configura AWS IAM Roles Anywhere".

1. Crea un file denominato `eks-hybrid-iamra-trust.json`. Sostituisci `TRUST_ANCHOR ARN` con l'ARN dell'ancora di fiducia che hai creato nei passaggi the section called "Configura AWS IAM Roles Anywhere". La condizione contenuta in questa policy di fiducia limita la capacità di AWS IAM Roles Anywhere di assumere il ruolo IAM di Hybrid Nodes per lo scambio di credenziali IAM temporanee solo quando il nome della sessione di ruolo corrisponde al CN nel certificato x509 installato sui nodi ibridi. In alternativa, puoi utilizzare altri attributi del certificato per identificare in modo univoco il tuo nodo. L'attributo del certificato che usi nella policy di fiducia deve corrispondere a `nodeName` che hai impostato nella configurazione `nodeadm`. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi".

```
{
    "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. Crea il ruolo con il comando seguente.

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-iamra-trust.json
```

3. Collega la `EKSDescribeClusterPolicy` che hai creato nei passaggi precedenti. Sostituiscilo `AWS_ACCOUNT_ID` con l'ID del tuo account. AWS

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

4. Allega la politica `AmazonEC2ContainerRegistryPullOnly` AWS gestita

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

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

Console di gestione AWS

Crea la policy del cluster per descrivere EKS

1. Apri la console Amazon IAM
2. Nel pannello di navigazione a sinistra, seleziona Policy.
3. Nella pagina Policy, scegli Crea policy.
4. Nella pagina Specifica le autorizzazioni, in Seleziona un pannello di servizio scegli EKS.
   a. Filtra le azioni DescribeClustere seleziona l'azione DescribeClusterLeggi.
   b. Scegli Next (Successivo).
5. Nella pagina Verifica e crea
   a. Immetti un Nome policy per la tua policy, come `EKSDescribeClusterPolicy`.
   b. Scegli Crea policy.

Passaggi per le attivazioni ibride AWS SSM

1. Apri la console Amazon IAM
2. Nel pannello di navigazione a sinistra, seleziona Policy.
3. Nella pagina Policy, scegli Crea policy.
4. Nella pagina Specifica autorizzazioni, nella barra di navigazione in alto a destra Editor della policy, scegli JSON. Incollare il frammento seguente. `AWS_REGION`Sostituiscila con la AWS regione dell'attivazione ibrida AWS SSM e sostituiscila `AWS_ACCOUNT_ID` con l'ID del tuo AWS account. Sostituisci `TAG_KEY` e `TAG_VALUE` con la chiave del tag di risorsa AWS SSM che hai usato durante la creazione dell'attivazione ibrida 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. Scegli Next (Successivo).

5. Nella pagina Verifica e crea
   a. Immetti un nome Policy per la tua policy, come `EKSHybridSSMPolicy`.
   b. Scegli Crea policy.
6. Nel pannello di navigazione a sinistra, seleziona Ruoli.
7. Nella pagina Ruoli, seleziona Crea ruolo.
8. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   a. Nel sezione Tipo di entità attendibile, scegli Policy di attendibilità personalizzata. Incolla quanto segue nell'editor della policy di attendibilità personalizzata. `AWS_REGION`Sostituiscilo con la AWS regione dell'attivazione ibrida AWS SSM e `AWS_ACCOUNT_ID` con l'ID del tuo AWS account.

```
{
   "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. Scegli Next (Successivo).

9. Nella pagina Aggiungi autorizzazioni, collega una policy personalizzata o esegui le operazioni seguenti:

   a. Nella casella Filtra policy, inserisci `EKSDescribeClusterPolicy` o il nome della policy creata in precedenza. Seleziona la casella di controllo a sinistra il nome della policy nei risultati della ricerca.

   b. Nella casella Filtra policy, inserisci `EKSHybridSSMPolicy` o il nome della policy creata in precedenza. Seleziona la casella di controllo a sinistra il nome della policy nei risultati della ricerca.

   c. Nella casella Filtra policy, inserisci `AmazonEC2ContainerRegistryPullOnly`. Seleziona la casella di controllo a sinistra di `AmazonEC2ContainerRegistryPullOnly` nei risultati di ricerca.

   d. Nella casella Filtra policy, inserisci `AmazonSSMManagedInstanceCore`. Seleziona la casella di controllo a sinistra della `AmazonSSMManagedInstanceCore` nei risultati di ricerca.

   e. Scegli Successivo.

10. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:

    a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSHybridNodesRole`.

    b. Per Description (Descrizione), sostituisci il testo corrente con un testo descrittivo come `Amazon EKS - Hybrid Nodes role`.

    c. Scegli Crea ruolo.

Passaggi per AWS IAM Roles Anywhere

Per utilizzare AWS IAM Roles Anywhere, devi configurare il tuo trust anchor AWS IAM Roles Anywhere prima di creare il ruolo IAM di Hybrid Nodes. Per istruzioni, consulta the section called "Configura AWS IAM Roles Anywhere".

1. Apri la console Amazon IAM

2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   a. Nel sezione Tipo di entità attendibile, scegli Policy di attendibilità personalizzata. Incolla quanto segue nell'editor della policy di attendibilità personalizzata. Sostituisci `TRUST_ANCHOR ARN` con l'ARN dell'ancora di fiducia che hai creato nei passaggi the section called "Configura AWS IAM Roles Anywhere". La condizione contenuta in questa policy di fiducia limita la capacità di AWS IAM Roles Anywhere di assumere il ruolo IAM di Hybrid Nodes per lo scambio di credenziali IAM temporanee solo quando il nome della sessione di ruolo corrisponde al CN nel certificato x509 installato sui nodi ibridi. In alternativa, puoi utilizzare altri attributi del certificato per identificare in modo univoco il tuo nodo. L'attributo del certificato che usi nella policy di fiducia deve corrispondere al nodeName che hai impostato nella configurazione bideadm. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi".

```
{
    "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. Scegli Next (Successivo).

5. Nella pagina Aggiungi autorizzazioni, collega una policy personalizzata o esegui le operazioni seguenti:

   a. Nella casella Filtra policy, inserisci `EKSDescribeClusterPolicy` o il nome della policy creata in precedenza. Seleziona la casella di controllo a sinistra il nome della policy nei risultati della ricerca.

   b. Nella casella Filtra policy, inserisci `AmazonEC2ContainerRegistryPullOnly`. Seleziona la casella di controllo a sinistra della `AmazonEC2ContainerRegistryPullOnly` nei risultati di ricerca.

   c. Scegli Successivo.

6. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:

   a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSHybridNodesRole`.

   b. Per Description (Descrizione), sostituisci il testo corrente con un testo descrittivo come `Amazon EKS - Hybrid Nodes role`.

   c. Scegli Crea ruolo.

Creazione di un cluster Amazon EKS con nodi ibridi

Questo argomento offre una panoramica delle opzioni disponibili e descrive gli aspetti da considerare quando crei un cluster Amazon EKS abilitato per i nodi ibridi. EKS Hybrid Nodes ha lo stesso supporto per le versioni di Kubernetes dei cluster Amazon EKS con nodi cloud, incluso il supporto standard ed esteso.

Se non hai intenzione di utilizzare EKS Hybrid Nodes, consulta la documentazione principale di Amazon EKS create cluster all'indirizzo the section called "Creazione di un cluster ".

Prerequisiti

- Il the section called "Prerequisiti" è completato. Prima di creare un cluster abilitato ai nodi ibridi, è necessario che il nodo locale e, facoltativamente, il pod vengano identificati CIDRs , il VPC e le sottoreti creati in base ai requisiti EKS e ai requisiti dei nodi ibridi e il gruppo di sicurezza con le regole in entrata per il pod locale e facoltativamente. CIDRs Per ulteriori informazioni su questi prerequisiti, consulta the section called "Preparazione della rete".
- La versione più recente dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo. Per verificare la versione attuale, usa `aws --version`. I gestori di pacchetti come yum, apt-get o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione o aggiornamento all'ultima versione della AWS CLI e Configurazione delle impostazioni per la CLI nella Guida per l'utente dell' AWS interfaccia a riga di AWS comando.
- Un principale IAM con le autorizzazioni per creare ruoli IAM e allegare policy, nonché creare e descrivere cluster EKS

Considerazioni

- Il cluster deve utilizzare una modalità di autenticazione del cluster tra `API` e `API_AND_CONFIG_MAP`.
- Il cluster deve utilizzare IPv4 la famiglia di indirizzi.
- Il cluster deve utilizzare la connettività degli endpoint del cluster pubblico o privato. Il tuo cluster non può utilizzare la connettività endpoint del cluster «pubblica e privata», poiché l'endpoint del server API Amazon EKS Kubernetes si trasformerà in pubblico IPs per i nodi ibridi in esecuzione al di fuori del tuo VPC.
- L'autenticazione OIDC è supportata per i cluster EKS con nodi ibridi.

- È possibile aggiungere, modificare o rimuovere la configurazione dei nodi ibridi dei cluster esistenti. Per ulteriori informazioni, consulta the section called "Cluster esistente".

Passaggio 1: creazione di un ruolo IAM del cluster

Se disponi già di un ruolo IAM del cluster o intendi creare il tuo cluster con `eksctl` o AWS CloudFormation, puoi saltare questo passaggio. Per impostazione predefinita, `eksctl` il AWS CloudFormation modello crea automaticamente il ruolo IAM del cluster.

1. Per creare un file JSON della policy di attendibilità IAM, esegui il comando seguente.

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

2. Crea il ruolo IAM del cluster Amazon EKS. Se necessario, inserisci come prefazione eks-cluster-role-trust -policy.json il percorso sul computer in cui hai scritto il file nel passaggio precedente. Il comando associa il criterio di attendibilità creato nella Passaggio precedente al ruolo. Per creare un ruolo IAM, l'azione `iam:CreateRole` (autorizzazione) deve essere assegnata al principale IAM che sta creando il ruolo.

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

3. Puoi assegnare la policy gestita da Amazon EKS o creare una policy personalizzata. Per conoscere le autorizzazioni minime che devi utilizzare nella policy personalizzata, consulta the section called "Ruolo IAM del nodo". Allega al ruolo la policy gestita da Amazon EKS, denominata `AmazonEKSClusterPolicy`. Per allegare una policy IAM a un principale IAM, è necessario assegnare al principale che sta allegando la policy una delle azioni IAM (autorizzazioni) seguenti: `iam:AttachUserPolicy` o `iam:AttachRolePolicy`.

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

Passaggio 2: creazione di cluster abilitato per i nodi ibridi

Puoi creare un cluster utilizzando:

- eksctl
- AWS CloudFormation
- AWS CLI
- Console di gestione AWS

Crea un cluster abilitato per i nodi ibridi - eksctl

È necessario installare la versione più recente dello strumento della riga di comando `eksctl`. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

1. Crea `cluster-config.yaml` per definire un cluster Amazon IPv4 EKS abilitato ai nodi ibridi. Effettua le seguenti sostituzioni in `cluster-config.yaml`. Per un elenco completo delle impostazioni, consulta eksctl documentation.
   a. Sostituisci `CLUSTER_NAME` con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.
   b. Sostituiscilo `AWS_REGION` con la AWS regione in cui desideri creare il cluster.
   c. Sostituisci `K8S_VERSION` con qualsiasi versione supportata da Amazon EKS.
   d. Sostituisci `CREDS_PROVIDER` con `ssm` o `ira` in base al provider di credenziali che hai configurato nei passaggi per the section called "Preparazione delle credenziali".
   e. Sostituisci `CA_BUNDLE_CERT` se il tuo provider di credenziali è impostato su`ira`, che utilizza AWS IAM Roles Anywhere come provider di credenziali. Il CA_BUNDLE_CERT è l'ente di certificazione dell'autorità di certificazione (CA) e dipende dalla CA scelta. Il certificato deve essere in formato Privacy-Enhanced Mail (PEM).

f. Sostituisci `GATEWAY_ID` con l'ID del tuo gateway privato virtuale o gateway di transito da collegare al tuo VPC.

g. Sostituisci `REMOTE_NODE_CIDRS` con il nodo on-premises CIDR per i tuoi nodi ibridi.

h. Sostituisci `REMOTE_POD_CIDRS` con il pod CIDR on-premises per i carichi di lavoro in esecuzione su nodi ibridi o rimuovi la riga dalla configurazione se non esegui webhook su nodi ibridi. È necessario configurare `REMOTE_POD_CIDRS` se il CNI non utilizza Network Address Translation (NAT) o il mascheramento per gli indirizzi IP dei pod quando il traffico dei pod lascia gli host on-premises. È necessario eseguire la configurazione `REMOTE_POD_CIDRS` se si eseguono webhook su nodi ibridi; per ulteriori informazioni consulta the section called "Configurazione di webhook".

i. I blocchi CIDR del nodo e del pod in on-premises devono soddisfare i seguenti requisiti:

   i. Rientra in uno degli intervalli IPv4 RFC-1918:, o. `10.0.0.0/8` `172.16.0.0/12` `192.168.0.0/16`

   ii. Non si sovrappongano tra loro, né per il tuo cluster o `VPC CIDR` per il tuo servizio Kubernetes CIDR IPv4

```
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. Esegui il comando seguente:

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

Il provisioning del cluster richiede diversi minuti. Durante la creazione del cluster, vengono visualizzate diverse righe di output. L'ultima riga di output è simile alla seguente riga di esempio.

```
[#]  EKS cluster "CLUSTER_NAME" in "REGION" region is ready
```

3. Continuare con the section called "Passaggio 3: aggiornamento di kubeconfig".

Crea un cluster ibrido abilitato ai nodi - AWS CloudFormation

Lo CloudFormation stack crea il ruolo IAM del cluster EKS e un cluster EKS con `RemoteNodeNetwork` e `RemotePodNetwork` da te specificato. Modifica il CloudFormation modello Se devi personalizzare le impostazioni per il tuo cluster EKS che non sono esposte nel CloudFormation modello.

1. Scaricate il CloudFormation modello.

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-eks-cfn.yaml'
```

2. Crea `cfn-eks-parameters.json` e specifica la tua configurazione per ogni valore.
   a. `CLUSTER_NAME`: il nome del nuovo cluster EKS da creare.
   b. `CLUSTER_ROLE_NAME`: il nome del nuovo ruolo IAM del cluster EKS da creare. L'impostazione predefinita nel modello è «EKSClusterRuolo».
   c. `SUBNET1_ID`: l'ID della prima sottorete creata nei passaggi preliminari.
   d. `SUBNET2_ID`: l'ID della seconda sottorete creata nei passaggi preliminari.
   e. `SG_ID`: il gruppo di sicurezza creato nei passaggi preliminari.
   f. `REMOTE_NODE_CIDRS`: il nodo on-premises CIDR per i tuoi nodi ibridi.
   g. `REMOTE_POD_CIDRS`: il pod CIDR on-premises per i carichi di lavoro in esecuzione su nodi ibridi. È necessario configurare `REMOTE_POD_CIDRS` se il CNI non utilizza Network Address Translation (NAT) o il mascheramento per gli indirizzi IP dei pod quando il traffico dei pod lascia gli host on-premises. È necessario eseguire la configurazione `REMOTE_POD_CIDRS` se si eseguono webhook su nodi ibridi; per ulteriori informazioni consulta the section called "Configurazione di webhook".
   h. I blocchi CIDR del nodo e del pod in on-premises devono soddisfare i seguenti requisiti:
      i. Rientrare in uno degli intervalli IPv4 RFC-1918:`10.0.0.0/8`,, `172.16.0.0/12` o. `192.168.0.0/16`

ii. Non si sovrappongano tra loro, né con il tuo cluster o con VPC CIDR il tuo servizio Kubernetes CIDR. IPv4

i. CLUSTER_AUTH: la modalità di autenticazione del cluster per il tuo cluster. I valori validi sono API e API_AND_CONFIG_MAP. L'impostazione predefinita nel modello è API_AND_CONFIG_MAP.

j. CLUSTER_ENDPOINT: la connettività degli endpoint del cluster per il cluster. I valori validi sono "Public" e "Private". L'impostazione predefinita nel modello è Private, il che significa che potrai connetterti all'endpoint dell'API Kubernetes solo dall'interno del tuo VPC.

k. K8S_VERSION: la versione Kubernetes desiderata per il cluster. Consulta Versioni supportate da 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. CloudFormation Implementa lo stack. Sostituiscilo STACK_NAME con il tuo nome per CloudFormation lo stack e AWS_REGION con la AWS regione in cui verrà creato il cluster.

```
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
```

Il provisioning del cluster richiede diversi minuti. È possibile controllare lo stato dello stack con il comando seguente. Sostituiscilo STACK_NAME con il tuo nome per CloudFormation lo stack e AWS_REGION con la AWS regione in cui verrà creato il cluster.

```
aws cloudformation describe-stacks \
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --query 'Stacks[].StackStatus'
```

4. Continuare con the section called "Passaggio 3: aggiornamento di kubeconfig".

Crea un cluster abilitato ai nodi ibridi - CLI AWS

1. Esegui il comando seguente per creare un cluster EKS abilitato per i nodi ibridi. Prima di eseguire il comando, sostituisci le impostazioni seguenti con le tue. Per un elenco completo delle impostazioni, consulta la documentazione the section called "Creazione di un cluster ".
   a. `CLUSTER_NAME`: il nome del nuovo cluster EKS da creare.
   b. `AWS_REGION`: AWS Regione in cui verrà creato il cluster.
   c. `K8S_VERSION`: la versione Kubernetes desiderata per il cluster. Controlla le versioni supportate da Amazon EKS.
   d. `ROLE_ARN`: il ruolo del cluster Amazon EKS che hai configurato per il tuo cluster. Per ulteriori informazioni, consulta il ruolo IAM del cluster Amazon EKS.
   e. `SUBNET1_ID`: l'ID della prima sottorete creata nei passaggi preliminari.
   f. `SUBNET2_ID`: l'ID della seconda sottorete creata nei passaggi preliminari.
   g. `SG_ID`: il gruppo di sicurezza creato nei passaggi preliminari.
   h. È possibile utilizzare `API` e `API_AND_CONFIG_MAP` per la modalità di autenticazione dell'accesso al cluster. Nel comando seguente, la modalità di autenticazione dell'accesso al cluster è impostata su `API_AND_CONFIG_MAP`.
   i. È possibile utilizzare i parametri `endpointPublicAccess` e `endpointPrivateAccess` per abilitare o disabilitare l'accesso pubblico e privato all'endpoint del server API Kubernetes del cluster. Nel comando seguente `endpointPublicAccess` è impostato su false e `endpointPrivateAccess` è impostato su true.
   j. `REMOTE_NODE_CIDRS`: il nodo on-premises CIDR per i tuoi nodi ibridi.
   k. `REMOTE_POD_CIDRS` (facoltativo): il pod CIDR on-premises per i carichi di lavoro in esecuzione su nodi ibridi.
   l. I blocchi CIDR del nodo e del pod in on-premises devono soddisfare i seguenti requisiti:
      i. Rientra in uno degli intervalli IPv4 RFC-1918:`10.0.0.0/8`,, `172.16.0.0/12` o. `192.168.0.0/16`

ii. Non si sovrappongano tra loro, né `VPC CIDR` per il tuo cluster Amazon EKS o il tuo servizio Kubernetes CIDR. IPv4

```
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. Il provisioning del cluster richiede diversi minuti. È possibile eseguire query sullo stato del cluster con il comando seguente. Sostituiscilo `CLUSTER_NAME` con il nome del cluster che stai creando e `AWS_REGION` con la AWS regione in cui il cluster sta creando. Non passare alla fase successiva finché l'output restituito non è `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. Continuare con the section called "Passaggio 3: aggiornamento di kubeconfig".

Crea un cluster abilitato ai nodi ibridi - Console di gestione AWS

1. Apri la console Amazon EKS in Amazon EKS console.
2. Seleziona Aggiungi cluster e quindi Crea.
3. Nella pagina Configura cluster compila i campi seguenti:
   a. Nome: un nome per il cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole), trattini e trattini bassi. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.
   b. Ruolo IAM del cluster: scegli il ruolo IAM del cluster Amazon EKS che hai creato per consentire al piano di controllo Kubernetes di gestire AWS le risorse per tuo conto.

c. Versione Kubernetes – la versione di Kubernetes da utilizzare per il cluster. È preferibile selezionare la versione più recente, a meno che non occorra una versione precedente.

d. Policy di aggiornamento: scegli tra esteso o standard.

   i. Esteso: questa opzione supporta la versione Kubernetes per 26 mesi dopo la data di rilascio. Il periodo di supporto esteso ha un costo orario aggiuntivo che inizia dopo la fine del periodo di quello standard. Al termine del supporto esteso, il cluster verrà aggiornato automaticamente alla versione successiva.

   ii. Standard: questa opzione supporta la versione Kubernetes per 14 mesi dopo la data di rilascio. Non è previsto alcun costo aggiuntivo Al termine del supporto standard, il cluster verrà aggiornato automaticamente alla versione successiva.

e. Accesso al cluster: scegli di consentire o negare l'accesso all'amministratore del cluster e seleziona una modalità di autenticazione. Le seguenti modalità di autenticazione sono supportate per i cluster abilitati per i nodi ibridi.

   i. API EKS: il cluster genererà i principali IAM autenticati solo dall'accesso EKS. APIs

   ii. API EKS e ConfigMap: Il cluster fornirà i principali IAM autenticati sia dalla voce di accesso EKS che da. APIs `aws-auth` ConfigMap

f. Crittografia dei segreti (Facoltativo): scegli di abilitare la crittografia dei segreti per i segreti Kubernetes tramite una chiave KMS. Puoi abilitare questa funzionalità anche dopo la creazione del cluster. Assicurati, tuttavia, di acquisire familiarità con le informazioni contenute nella sezione the section called "Abilitare la crittografia dei segreti".

g. Spostamento zonale ARC: se abilitato, EKS registrerà il cluster con lo spostamento zonale ARC per consentirti di utilizzare quest'ultimo per spostare il traffico delle applicazioni lontano da una zona di disponibilità.

h. Tag: (facoltativo) aggiunge eventuali tag al cluster. Per ulteriori informazioni, consulta the section called "Assegnazione di tag alle risorse ".

i. Quando hai finito con questa pagina, seleziona Avanti.

4. Nella pagina Specifica reti, seleziona i valori dei campi riportati di seguito:

   a. VPC: scegli un VPC esistente che soddisfi i requisiti di the section called "Requisiti del VPC e delle sottoreti" e Amazon EKS Hybrid Nodes. Prima di scegliere un VPC, ti consigliamo di acquisire familiarità con i requisiti e le considerazioni nella sezione Visualizza i requisiti di rete di Amazon EKS per VPC, sottoreti e nodi ibridi. Dopo la creazione del cluster, non potrai più modificare il VPC da utilizzare. Se non VPCs ne è elencato nessuno, devi prima crearne uno. Per ulteriori informazioni consulta the section called "Crea un VPC" e Amazon EKS Hybrid Nodes networking requirements.

b. Sottoreti: per impostazione predefinita, tutte le sottoreti disponibili nel VPC specificato nel campo precedente sono preselezionate. È necessario selezionarne almeno due.

c. Gruppi di sicurezza: (facoltativo) specifica uno o più gruppi di sicurezza da associare alle interfacce di rete create da Amazon EKS. Almeno uno dei gruppi di sicurezza specificati deve disporre di regole in entrata per il nodo locale e, facoltativamente, per il pod. CIDRs Per ulteriori informazioni, consulta Amazon EKS Hybrid Nodes networking requirements. Indipendentemente dal fatto che tu scelga o meno un gruppo di sicurezza, Amazon EKS ne crea uno nuovo per consentire la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". È possibile modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS.

d. Scegli la famiglia di indirizzi IP del cluster: devi scegliere tra cluster abilitati ai nodi IPv4 ibridi.

e. (Facoltativo) Scegli Configura l'intervallo di indirizzi IP del servizio Kubernetes e specifica un intervallo di servizi. IPv4

f. Scegli Configura reti remote per abilitare i nodi ibridi e specifica il nodo e il pod CIDRs locali per i nodi ibridi.

g. È necessario configurare i pod CIDR remoti se il CNI non utilizza Network Address Translation (NAT) o il mascheramento per gli indirizzi IP dei pod quando il traffico dei pod lascia gli host on-premises. È necessario configurare il pod CIDR remoto se si eseguono webhook su nodi ibridi.

h. I blocchi CIDR del nodo e del pod in on-premises devono soddisfare i seguenti requisiti:

   i. Rientra in uno degli intervalli IPv4 RFC-1918:`10.0.0.0/8`,, o. `172.16.0.0/12` `192.168.0.0/16`

   ii. Non si sovrappongano tra loro, né per il tuo cluster o `VPC CIDR` per il tuo servizio Kubernetes CIDR IPv4

i. Per Cluster endpoint access (Accesso all'endpoint del cluster), seleziona un'opzione. Puoi modificare questa opzione dopo aver creato il cluster. Per i cluster abilitati per i nodi ibridi, devi scegliere Pubblico o Privato. Prima di selezionare un'opzione non predefinita, assicurati di familiarizzare con le opzioni e le relative implicazioni. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

j. Quando hai finito con questa pagina, seleziona Avanti.

5. (Facoltativo) Nella pagina Configura osservabilità, scegli le opzioni Parametri e Registrazione del piano di controllo da attivare. Per impostazione predefinita, i tipi di log sono disattivati.

a. Per ulteriori informazioni sull'opzione relativa alle metriche di Prometheus, consulta the section called "Parametri Prometheus".

b. Per ulteriori informazioni sulle opzioni Registrazione del controllo EKS, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".

c. Quando hai finito con questa pagina, seleziona Avanti.

6. Nella pagina Select add-ons (Seleziona componenti aggiuntivi), scegli i componenti aggiuntivi da aggiungere al tuo cluster.

a. Puoi scegliere tutti i componenti aggiuntivi Amazon EKS e i componenti aggiuntivi AWS Marketplace di cui hai bisogno. I componenti aggiuntivi di Amazon EKS che non sono compatibili con i nodi ibridi sono contrassegnati con la dicitura "Non compatibile con nodi ibridi" e hanno una regola di anti-affinità per impedirne l'esecuzione su nodi ibridi. Per ulteriori informazioni, consulta Configurazione dei componenti aggiuntivi per i nodi ibridi. Se i componenti aggiuntivi del AWS Marketplace che desideri installare non sono elencati, puoi cercare i componenti aggiuntivi di AWS Marketplace disponibili inserendo il testo nella casella di ricerca. Puoi eseguire una ricerca anche per categoria, fornitore o modello di prezzi e quindi scegliere i componenti aggiuntivi dai risultati della ricerca.

b. Alcuni componenti aggiuntivi, come CoreDNS e kube-proxy, sono installati per impostazione predefinita. Disabilitare uno dei componenti aggiuntivi predefiniti potrebbe influire sulla tua capacità di eseguire le applicazioni Kubernetes.

c. Quando hai finito con questa pagina, seleziona `Next`.

7. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, seleziona la versione da installare.

a. Dopo la creazione del cluster, puoi eseguire in qualsiasi momento l'aggiornamento a una versione successiva. Puoi aggiornare la configurazione di ogni componente aggiuntivo dopo la creazione del cluster. Per ulteriori informazioni sulla configurazione dei componenti aggiuntivi, consulta the section called "Aggiornamento di un componente aggiuntivo". Per le versioni dei componenti aggiuntivi compatibili con i nodi ibridi, consulta the section called "Configurare i componenti aggiuntivi".

b. Quando hai finito con questa pagina, seleziona Avanti.

8. Nella pagina Rivedi e crea, controlla le informazioni che hai inserito o selezionato nelle pagine precedenti. Se devi apportare modifiche, seleziona Edit (Modifica). Al termine della configurazione, seleziona Create (Crea). Durante il provisioning del cluster, nel campo Stato viene visualizzato il messaggio CREAZIONE. Il provisioning del cluster richiede diversi minuti.

9. Continuare con the section called "Passaggio 3: aggiornamento di kubeconfig".

Passaggio 3: aggiornamento di kubeconfig

Se hai creato il cluster utilizzando `eksctl`, puoi ignorare questo passaggio, poiché è già stato completato da `eksctl`. Abilita `kubectl` per consentire la comunicazione con il cluster aggiungendo un nuovo contesto al file di configurazione `kubectl`. Per ulteriori informazioni su come creare e aggiornare il file, consulta the section called "Accedere al cluster con kubectl".

```
aws eks update-kubeconfig --name CLUSTER_NAME --region AWS_REGION
```

Di seguito viene riportato un output di esempio:

```
Added new context arn:aws: eks:AWS_REGION:111122223333:cluster/CLUSTER_NAME to /home/
username/.kube/config
```

Conferma la comunicazione con il cluster eseguendo il comando seguente.

```
kubectl get svc
```

Di seguito viene riportato un output di esempio.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

Passaggio 4: configurazione del cluster

Come passaggio successivo, consulta the section called "Preparazione dell'accesso ai cluster" per abilitare l'accesso ai nodi ibridi al fine di entrare a far parte del cluster.

Abilitazione dei nodi ibridi su un cluster Amazon EKS esistente o modifica della configurazione

Questo argomento offre una panoramica delle opzioni disponibili e descrive gli aspetti da considerare quando aggiungi, modifichi o rimuovi la configurazione dei nodi ibridi per un cluster Amazon EKS.

Per consentire a un cluster Amazon EKS di utilizzare nodi ibridi, aggiungi gli intervalli CIDR degli indirizzi IP del nodo on-premises e, facoltativamente, la rete pod nella configurazione `RemoteNetworkConfig`. EKS utilizza questo elenco di CIDR per abilitare la connettività tra il cluster e le reti on-premises. Per un elenco completo delle opzioni per l'aggiornamento della configurazione del cluster, consulta UpdateClusterConfig nella Documentazione di riferimento API Amazon EKS.

Puoi eseguire una delle seguenti azioni sulla configurazione della rete dei nodi ibridi EKS in un cluster:

- Aggiungi una configurazione di rete remota per abilitare i nodi ibridi EKS in un cluster esistente.
- Aggiungi, modifica o rimuovi le reti di nodi remoti e quelle pod remote in un cluster esistente.
- Rimuovi tutti gli intervalli CIDR della rete di nodi remoti per disabilitare i nodi ibridi EKS in un cluster esistente.

Prerequisiti

- Prima di abilitare il cluster Amazon EKS per nodi ibridi, assicurati che l'ambiente soddisfi i requisiti descritti in the section called "Prerequisiti" e specificati nel dettaglio in the section called "Preparazione della rete", the section called "Preparazione del sistema operativo" e the section called "Preparazione delle credenziali".
- Il cluster deve utilizzare la famiglia di indirizzi IPv4.
- Il cluster deve utilizzare una modalità di autenticazione del cluster tra `API` e `API_AND_CONFIG_MAP`. Il processo di modifica della modalità di autenticazione del cluster è descritto in the section called "Modalità di autenticazione".
- Ti consigliamo di utilizzare l'accesso all'endpoint pubblico o privato per l'endpoint del server API Amazon EKS Kubernetes, ma non entrambi. Se scegli "Pubblico e privato", l'endpoint del server API Amazon EKS Kubernetes si risolverà sempre negli IP pubblici per i nodi ibridi in esecuzione all'esterno del tuo VPC, il che può impedire ai nodi ibridi di unirsi al cluster. Il processo per modificare l'accesso di rete al cluster è descritto in the section called "Accesso all'endpoint del cluster".
- La versione più recente dell'interfaccia a riga di comando AWS (AWS CLI) installata e configurata sul dispositivo. Per verificare la versione attuale, usa `aws --version`. I programmi di gestione dei pacchetti, come yum, apt-get o Homebrew per macOS, spesso sono aggiornati a versioni precedenti della CLI AWS. Per installare versione più recente, consulta Installare o aggiornare l'ultima versione di AWS CLI e Configurare le impostazioni per AWS CLI nella Guida per l'utente di AWS per l'interfaccia a riga di comando.
- Un principale IAM con l'autorizzazione per chiamare UpdateClusterConfig sul tuo cluster Amazon EKS.
- Aggiorna i componenti aggiuntivi alle versioni compatibili con i nodi ibridi. Per le versioni dei componenti aggiuntivi compatibili con i nodi ibridi, consulta the section called "Configurare i componenti aggiuntivi".

- Se esegui componenti aggiuntivi non compatibili con i nodi ibridi, assicurati che il componente aggiuntivo DaemonSet o Implementazione abbia la seguente regola di affinità per impedire la distribuzione su nodi ibridi. Aggiungi la seguente regola di affinità se non è già presente.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Considerazioni

L'oggetto `remoteNetworkConfig` JSON ha il seguente comportamento durante un aggiornamento:

- Qualsiasi parte esistente della configurazione che non specifichi rimane invariata. Se non specificate nessuno né `remoteNodeNetworks` né `remotePodNetworks`, quella parte rimarrà la stessa.
- Se modifichi l'elenco `remoteNodeNetworks` o l'elenco `remotePodNetworks` dei CIDR, è necessario specificare quello completo dei CIDR che si desidera inserire nella configurazione finale. Quando specifichi una modifica all'elenco `remoteNodeNetworks` o all'elenco `remotePodNetworks` CIDR, EKS sostituisce l'elenco originale durante l'aggiornamento.
- I blocchi CIDR del nodo e del pod in on-premises devono soddisfare i seguenti requisiti:
  1. Stai all'interno di uno degli intervalli di IPv4 RFC-1918: 10.0.0.0/8, 172.16.0.0/12 o 192.168.0.0/16.
  2. Non si sovrappongano tra loro, né con tutti i CIDR del VPC per il tuo cluster Amazon EKS né con il tuo servizio Kubernetes IPv4 CIDR.

Abilitazione dei nodi ibridi su un cluster esistente

È possibile abilitare i nodi ibridi EKS in un cluster esistente utilizzando:

- AWS CloudFormation
- AWS CLI

- Console di gestione AWS

Abilita i nodi ibridi EKS in un cluster esistente - AWS CloudFormation

1. Per abilitare i nodi ibridi EKS nel tuo cluster, aggiungi `RemoteNodeNetwork` e (facoltativo) `RemotePodNetwork` al tuo modello CloudFormation e aggiorna lo stack. Tieni presente che `RemoteNodeNetwork` è un elenco con un massimo di un elemento `Cidrs` e che `Cidrs` è un elenco di più intervalli IP CIDR.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [RemoteNodeCIDR]
  RemotePodNetworks:
    - Cidrs: [RemotePodCIDR]
```

2. Continua su the section called "Preparazione dell'accesso ai cluster".

Abilita i nodi ibridi EKS in un cluster esistente - AWS CLI

1. Esegui il comando seguente per abilitare `RemoteNetworkConfig` per i nodi ibridi EKS per il cluster EKS. Prima di eseguire il comando, sostituisci le impostazioni seguenti con le tue. Per un elenco completo delle impostazioni, consulta UpdateClusterConfig nella Documentazione di riferimento API Amazon EKS.
   a. `CLUSTER_NAME`: nome del cluster EKS da aggiornare
   b. `AWS_REGION`: Regione AWS in cui è in esecuzione il cluster EKS.
   c. `REMOTE_NODE_CIDRS`: il nodo on-premises CIDR per i tuoi nodi ibridi.
   d. `REMOTE_POD_CIDRS` (facoltativo): il pod CIDR on-premises per i carichi di lavoro in esecuzione su nodi ibridi.

```
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. L'aggiornamento del cluster richiede diversi minuti. È possibile eseguire query sullo stato del cluster con il comando seguente. Sostituisci `CLUSTER_NAME` con il nome del cluster che stai

modificando e `AWS_REGION` con la Regione AWS in cui il cluster in esecuzione. Non passare alla Passaggio successiva finché l'output restituito non è `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. Continua su the section called "Preparazione dell'accesso ai cluster".

Abilita i nodi ibridi EKS in un cluster esistente - Console di gestione AWS

1. Apri la console Amazon EKS in Amazon EKS console.
2. Scegliere il nome del cluster per visualizzare le informazioni sul cluster.
3. Scegliere la scheda Reti, quindi Gestione.
4. Nel menu a discesa, scegli Reti remote.
5. Scegli Configura reti remote per abilitare i nodi ibridi e specifica i CIDR di nodi e pod on-premises per i nodi ibridi.
6. Scegliere Salva le modifiche per terminare. Attendi che lo stato del cluster ritorni su Attivo.
7. Continua su the section called "Preparazione dell'accesso ai cluster".

Aggiornamento della configurazione dei nodi ibridi in un cluster esistente

È possibile effettuare modifiche `remoteNetworkConfig` in un cluster ibrido esistente utilizzando uno dei seguenti metodi:

- AWS CloudFormation
- AWS CLI
- Console di gestione AWS

Aggiornamento della configurazione ibrida in un cluster esistente - AWS CloudFormation

1. Aggiorna il tuo modello CloudFormation con i nuovi valori CIDR di rete.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [NEW_REMOTE_NODE_CIDRS]
```

```
  RemotePodNetworks:
    - Cidrs: [NEW_REMOTE_POD_CIDRS]
```

> **Note**
>
> Quando aggiorni gli elenchi `RemoteNodeNetworks` o `RemotePodNetworks` CIDR, includi tutti i CIDR (nuovi ed esistenti). EKS sostituisce l'intero elenco durante gli aggiornamenti. L'omissione di questi campi dalla richiesta di aggiornamento mantiene le configurazioni esistenti.

2. Aggiorna lo stack CloudFormation con il modello modificato e attendi il completamento dell'aggiornamento dello stack.

Aggiornamento della configurazione ibrida in un cluster esistente - AWS CLI

1. Per modificare i CIDR della rete remota, eseguire questo comando. Sostituisci i valori le tue impostazioni:

```
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**
>
> Quando aggiorni gli elenchi `remoteNodeNetworks` o `remotePodNetworks` CIDR, includi tutti i CIDR (nuovi ed esistenti). EKS sostituisce l'intero elenco durante gli aggiornamenti. L'omissione di questi campi dalla richiesta di aggiornamento mantiene le configurazioni esistenti.

2. Attendi che lo stato del cluster ritorni ad ATTIVO prima di procedere.

Aggiornamento della configurazione ibrida in un cluster esistente - Console di gestione AWS

1. Apri la console Amazon EKS in Amazon EKS console.
2. Scegliere il nome del cluster per visualizzare le informazioni sul cluster.

3. Scegliere la scheda Reti, quindi Gestione.
4. Nel menu a discesa, scegli Reti remote.
5. Aggiorna i CIDR sotto `Remote node networks` e `Remote pod networks - Optional` secondo necessità.
6. Scegli Salva modifiche e attendi che lo stato del cluster ritorni Attivo.

Disabilitazione dei nodi ibridi in un cluster esistente

È possibile disabilitare i nodi ibridi EKS in un cluster esistente utilizzando:

- AWS CloudFormation
- AWS CLI
- Console di gestione AWS

Disabilitazione dei nodi ibridi EKS in un cluster esistente - AWS CloudFormation

1. Per disabilitare i nodi ibridi EKS nel tuo cluster, imposta `RemoteNodeNetworks` e `RemotePodNetworks` per svuotare gli array nel tuo modello CloudFormation e aggiorna lo stack.

```
RemoteNetworkConfig:
  RemoteNodeNetworks: []
  RemotePodNetworks: []
```

Disabilita i nodi ibridi EKS in un cluster esistente - AWS CLI

1. Esegui i seguenti comandi per rimuovere `RemoteNetworkConfig` dal cluster EKS. Prima di eseguire il comando, sostituisci le impostazioni seguenti con le tue. Per un elenco completo delle impostazioni, consulta UpdateClusterConfig nella Documentazione di riferimento API Amazon EKS.
   a. `CLUSTER_NAME`: nome del cluster EKS da aggiornare
   b. `AWS_REGION`: Regione AWS in cui è in esecuzione il cluster EKS.

```
aws eks update-cluster-config \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --remote-network-config '{"remoteNodeNetworks":[],"remotePodNetworks":[]}'
```

2. L'aggiornamento del cluster richiede diversi minuti. È possibile eseguire query sullo stato del cluster con il comando seguente. Sostituisci `CLUSTER_NAME` con il nome del cluster che stai modificando e `AWS_REGION` con la Regione AWS in cui il cluster in esecuzione. Non passare alla Passaggio successiva finché l'output restituito non è `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

Disabilitazione dei nodi ibridi EKS in un cluster esistente - Console di gestione AWS

1. Apri la console Amazon EKS in Amazon EKS console.
2. Scegliere il nome del cluster per visualizzare le informazioni sul cluster.
3. Scegliere la scheda Reti, quindi Gestione.
4. Nel menu a discesa, scegli Reti remote.
5. Scegli Configura reti remote per abilitare i nodi ibridi e rimuovi tutti i CIDR sotto `Remote node networks` e `Remote pod networks - Optional`.
6. Scegliere Salva le modifiche per terminare. Attendi che lo stato del cluster ritorni su Attivo.

Preparazione dell'accesso al cluster per i nodi ibridi

Prima di connettere i nodi ibridi al cluster Amazon EKS, devi abilitare le autorizzazioni del ruolo IAM per i nodi ibridi con Kubernetes per entrare a far parte del cluster. Per ulteriori informazioni su come creare il ruolo IAM per i nodi ibridi, consulta the section called "Preparazione delle credenziali". Amazon EKS supporta due modi per associare i principali IAM a Kubernetes Role-Based Access Control (RBAC), alle voci di accesso Amazon EKS e a. `aws-auth` ConfigMap Per ulteriori informazioni sulla gestione dell'accesso di Amazon EKS, consultathe section called "Accesso API Kubernetes" .

Utilizza le procedure seguenti per associare il ruolo IAM dei nodi ibridi alle autorizzazioni Kubernetes. Per utilizzare le voci di accesso di Amazon EKS, il cluster deve essere stato creato con le modalità di autenticazione `API` o `API_AND_CONFIG_MAP`. Per utilizzare `aws-auth` ConfigMap, il cluster deve essere stato creato con la modalità di autenticazione. `API_AND_CONFIG_MAP` La modalità di autenticazione `CONFIG_MAP`-only non è supportata per i cluster Amazon EKS abilitati per i nodi ibridi.

Utilizzo delle voci di accesso di Amazon EKS per il ruolo IAM dei nodi ibridi

Esiste un tipo di accesso Amazon EKS per i nodi ibridi denominato HYBRID_LINUX che può essere utilizzato con un ruolo IAM. Con questo tipo di accesso, il nome utente viene impostato automaticamente su system:node: {{SessionName}}. Per ulteriori informazioni sulla creazione delle voci di accesso, consulta the section called "Creare voci di accesso".

AWS CLI

1. È necessario che sul dispositivo sia installata e configurata la versione più recente della AWS CLI. Per verificare la versione attuale, usa `aws --version`. I gestori di pacchetti come yum, apt-get o Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI. AWS Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell' AWS interfaccia a riga di comando.
2. Creare la tua voce di accesso con il comando seguente. Sostituisci CLUSTER_NAME con il nome del tuo cluster e HYBRID_NODES_ROLE_ARN con l'ARN del ruolo che hai creato nei passaggi per the section called "Preparazione delle credenziali".

```
aws eks create-access-entry --cluster-name CLUSTER_NAME \
    --principal-arn HYBRID_NODES_ROLE_ARN \
    --type HYBRID_LINUX
```

Console di gestione AWS

1. Apri la console Amazon EKS in Amazon EKS console.
2. Scegli il nome del cluster ibrido abilitato per i nodi.
3. Scegliere la scheda Accesso.
4. Seleziona Crea voce di accesso.
5. Per principale IAM, seleziona il ruolo IAM dei nodi ibridi che hai creato nei passaggi per the section called "Preparazione delle credenziali".
6. Per Tipo, seleziona Hybrid Linux.
7. (Facoltativo) Per Tag, assegna etichette alla voce di accesso. Ad esempio, per facilitare la ricerca di tutte le risorse con lo stesso tag.
8. Scegli Vai alla revisione e alla creazione. Non è possibile aggiungere policy alla voce di accesso Hybrid Linux o modificarne l'ambito di accesso.

9. Verifica la configurazione per la tua voce di accesso. Se noti qualcosa di errato, scegli Precedente per tornare indietro e correggere l'errore. Se la configurazione è corretta, scegli Crea.

Utilizzo di aws-auth ConfigMap per il ruolo IAM di Hybrid Nodes

Nei passaggi seguenti, creerai o aggiornerai `aws-auth` ConfigMap con l'ARN il ruolo IAM dei nodi ibridi che hai creato nei passaggi per. the section called "Preparazione delle credenziali"

1. Verifica se ne hai uno esistente `aws-auth` ConfigMap per il tuo cluster. Considera che se stai usando un file `kubeconfig` specifico, devi usare il flag `--kubeconfig`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se ti viene mostrato un `aws-auth` ConfigMap, aggiornalo secondo necessità.
   a. Apri il file ConfigMap per modificarlo.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Aggiungi una nuova voce `mapRoles`, se necessario. Sostituisci `HYBRID_NODES_ROLE_ARN` con l'ARN del ruolo IAM dei nodi ibridi. Nota, `{{SessionName}}` è il formato di modello corretto da salvare in ConfigMap. Non sostituirlo con altri valori.

```
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
```

   c. Salva il file ed esci dall'editor di testo.
3. Se non ne esiste uno `aws-auth` ConfigMap per il tuo cluster, crealo con il seguente comando. Sostituisci `HYBRID_NODES_ROLE_ARN` con l'ARN del ruolo IAM dei nodi ibridi. Nota che `{{SessionName}}` è il formato di modello corretto da salvare in ConfigMap. Non sostituirlo con altri valori.

```
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
```

Esegui carichi di lavoro on-premises su nodi ibridi

In un cluster EKS con nodi ibridi abilitati, puoi eseguire applicazioni on-premises ed edge sulla tua infrastruttura con gli stessi cluster, le stesse funzionalità e gli stessi strumenti Amazon EKS che usi nel cloud AWS.

Le seguenti sezioni contengono istruzioni dettagliate per l'utilizzo di nodi ibridi.

Argomenti

- Connessione di nodi ibridi
- Connessione dei nodi ibridi con Bottlerocket
- Aggiornamento dei nodi ibridi per il tuo cluster
- Aggiornamenti di sicurezza delle patch per i nodi ibridi
- Rimuovi i nodi ibridi

Connessione di nodi ibridi

> **Note**
>
> I seguenti passaggi si applicano ai nodi ibridi che eseguono sistemi operativi compatibili ad eccezione di Bottlerocket. Per i passaggi per connettere un nodo ibrido che esegue Bottlerocket, consultare the section called "Connessione dei nodi ibridi con Bottlerocket".

In questo argomento viene descritto come connettere nodi ibridi a un cluster Amazon EKS. Dopo che i nodi ibridi si sono uniti al cluster, verranno visualizzati con lo stato Not Ready nella console Amazon

EKS e in strumenti compatibili con Kubernetes come kubectl. Dopo aver completato i passaggi in questa pagina, procedi su the section called "Configurazione della CNI" per preparare i nodi ibridi per l'esecuzione delle applicazioni.

Prerequisiti

Prima di connettere i nodi ibridi al cluster Amazon EKS, assicurati di aver completato tutti i passaggi relativi ai prerequisiti.

- Disponi di una connettività di rete dal proprio ambiente on-premises alla Regione AWS che ospita il cluster Amazon EKS. Per ulteriori informazioni, consulta the section called "Preparazione della rete".
- Si dispone di un sistema operativo compatibile per i nodi ibridi installato sugli host on-premises. Per ulteriori informazioni, consulta the section called "Preparazione del sistema operativo".
- Hai creato il ruolo IAM Nodi ibridi e configurato il provider di credenziali on-premises (attivazioni ibride di Systems Manager di AWS o IAM Roles Anywhere di AWS). Per ulteriori informazioni, consulta the section called "Preparazione delle credenziali".
- Hai creato il cluster Amazon EKS abilitato per i nodi ibridi. Per ulteriori informazioni, consulta the section called "Crea cluster".
- Hai associato il ruolo IAM Nodi ibridi alle autorizzazioni con il controllo degli accessi basato sui ruoli (RBAC) di Kubernetes. Per ulteriori informazioni, consulta the section called "Preparazione dell'accesso ai cluster".

Passaggio 1: installazione della CLI per nodi ibridi (**`nodeadm`**) su ogni host on-premises

Se si include la CLI Amazon EKS Hybrid Nodes (`nodeadm`) nelle immagini predefinite del sistema operativo, è possibile saltare questo passaggio. Per maggiori informazioni sulla versione per nodi ibridi del `nodeadm`, consultare the section called "nodeadm dei nodi ibridi".

La versione di `nodeadm` per nodi ibridi è ospitata in Amazon S3 e gestita da Amazon CloudFront. Per installare `nodeadm` su ciascun host on-premises, puoi eseguire il seguente comando dagli host on-premises.

Per host x86_64:

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/
nodeadm'
```

Per host ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/
nodeadm'
```

Aggiungi l'autorizzazione per i file eseguibili al file binario scaricato su ciascun host.

```
chmod +x nodeadm
```

Passaggio 2: installazione delle dipendenze dei nodi ibridi con **`nodeadm`**

Se si stanno installando le dipendenze dei nodi ibridi in immagini del sistema operativo predefinite, è possibile ignorare questo passaggio. Il comando `nodeadm install` può essere utilizzato per installare tutte le dipendenze richieste per i nodi ibridi. Le dipendenze dei nodi ibridi includono containerd, kubelet, kubectl e i componenti SSM di AWS o IAM Roles Anywhere di AWS. Consultare the section called "nodeadm dei nodi ibridi" per ulteriori informazioni sui componenti e le posizioni dei file installati da `nodeadm install`. Consultare la sezione the section called "Preparazione della rete" dedicata ai nodi ibridi per ulteriori informazioni sui domini che devono essere consentiti nel firewall on-premises per il processo `nodeadm install`.

Eseguire il comando seguente per installare le dipendenze dei nodi ibridi sull'host on-premises. Il comando seguente deve essere eseguito con un utente con accesso sudo/root sull'host.

> **Important**
>
> La CLI per nodi ibridi (`nodeadm`) deve essere eseguita con un utente con accesso sudo/root sull'host.

- Sostituire `K8S_VERSION` con la versione secondaria Kubernetes del cluster Amazon EKS, ad esempio `1.31`. Consultare Amazon EKS supported versions per un elenco delle versioni Kubernetes supportate.
- Sostituire `CREDS_PROVIDER` con il provider di credenziali on-premises che si sta utilizzando. I valori validi sono `ssm` per SSM di AWS e `iam-ra` per IAM Roles Anywhere di AWS.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER
```

Passaggio 3: connessione dei nodi ibridi al cluster

Prima di connettere i nodi ibridi al cluster, assicurati di aver consentito l'accesso richiesto nel firewall on-premises e nel gruppo di sicurezza del cluster per il piano di controllo di Amazon EKS da/verso la comunicazione tra nodi ibridi. La maggior parte dei problemi in questa Passaggio riguarda la configurazione del firewall, la configurazione del gruppo di sicurezza o la configurazione del ruolo IAM dei nodi ibridi.

> **Important**
>
> La CLI per nodi ibridi (`nodeadm`) deve essere eseguita con un utente con accesso sudo/root sull'host.

1. Crea un file `nodeConfig.yaml` su ogni host con i valori per la tua implementazione. Per una descrizione completa delle impostazioni di configurazione disponibili, consulta the section called "nodeadm dei nodi ibridi". Se il ruolo IAM dei nodi ibridi non dispone dell'autorizzazione per l'azione `eks:DescribeCluster`, devi passare l'endpoint dell'API Kubernetes, il cluster CA bundle e il servizio Kubernetes IPv4 CIDR nella sezione cluster del tuo `nodeConfig.yaml`.
   a. Utilizza il `nodeConfig.yaml` di esempio seguente se utilizzi attivazioni ibride SSM di AWS per il tuo provider di credenziali on-premises.
      i. Sostituisci `CLUSTER_NAME` con il nome del cluster.
      ii. Sostituisci `AWS_REGION` con la Regione AWS in cui si trova il cluster. Ad esempio, `us-west-2`.
      iii. Sostituisci `ACTIVATION_CODE` con il codice di attivazione che hai ricevuto durante la creazione dell'attivazione ibrida SSM di AWS. Per ulteriori informazioni, consulta the section called "Preparazione delle credenziali".
      iv. Sostituisci `ACTIVATION_ID` con l'ID di attivazione che hai ricevuto durante la creazione dell'attivazione ibrida SSM di AWS. È possibile recuperare queste informazioni dalla console Systems Manager di AWS o dal comando `aws ssm describe-activations` della 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. Utilizza il `nodeConfig.yaml` di esempio seguente se utilizzi IAM Roles Anywhere di AWS per il tuo provider di credenziali on-premises.

   i. Sostituisci `CLUSTER_NAME` con il nome del cluster.

   ii. Sostituisci `AWS_REGION` con la Regione AWS in cui si trova il cluster. Ad esempio, `us-west-2`.

   iii. Sostituisci `NODE_NAME` con il nome del ruolo del nodo. Il nome del nodo deve corrispondere al CN del certificato sull'host se hai configurato la policy di attendibilità del ruolo IAM di Hybrid Nodes con la condizione della risorsa `"sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"`. Il `nodeName` che usi non deve superare i 64 caratteri.

   iv. Sostituisci `TRUST_ANCHOR_ARN` con l'ARN dell'ancora di fiducia che hai configurato nei passaggi per Preparare le credenziali per i nodi ibridi.

   v. Sostituisci `PROFILE_ARN` con l'ARN dell'ancora di fiducia che hai configurato nei passaggi per the section called "Preparazione delle credenziali".

   vi. Sostituisci `ROLE_ARN` con l'ARN del ruolo IAM dei nodi ibridi.

   vii. Sostituisci `CERTIFICATE_PATH` con il percorso su disco del certificato del tuo nodo. Se non lo specifichi, il valore predefinito è `/etc/iam/pki/server.pem`.

   viii. Sostituisci `KEY_PATH` con il percorso su disco della chiave privata del certificato. Se non lo specifichi, il valore predefinito è `/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. Esegui il comando `nodeadm init` con il tuo `nodeConfig.yaml` per connettere i nodi ibridi al cluster Amazon EKS.

```
nodeadm init -c file://nodeConfig.yaml
```

Se il comando precedente viene effettuato correttamente, il nodo ibrido si unisce al cluster Amazon EKS. Puoi verificarlo nella console Amazon EKS accedendo alla scheda Calcolo del tuo cluster (assicurati che il principale IAM disponga delle autorizzazioni per la visualizzazione) o con `kubectl get nodes`.

> **Important**
>
> I nodi avranno lo stato `Not Ready` previsto, dovuto alla mancanza di una CNI in esecuzione sui nodi ibridi. Se i tuoi nodi non si sono uniti al cluster, consulta the section called "risoluzione dei problemi".

Passaggio 4: configurazione di una CNI per i nodi ibridi

Per preparare i nodi ibridi all'esecuzione delle applicazioni, continua con i passaggi su the section called "Configurazione della CNI".

Connessione dei nodi ibridi con Bottlerocket

Questo argomento descrive come connettere i nodi ibridi che eseguono Bottlerocket a un cluster Amazon EKS. Bottlerocket è una distribuzione Linux open source sponsorizzata e supportata da. AWS Bottlerocket è progettato appositamente per ospitare carichi di lavoro in container. Con Bottlerocket, puoi migliorare la disponibilità delle implementazioni containerizzate e ridurre i costi operativi automatizzando gli aggiornamenti dell'infrastruttura dei container. Bottlerocket include solo il software essenziale per eseguire i container, che migliora l'utilizzo delle risorse, riduce le minacce alla sicurezza e riduce i costi di gestione.

Solo VMware le varianti della versione Bottlerocket v1.37.0 e successive sono supportate con EKS Hybrid Nodes. VMware le varianti di Bottlerocket sono disponibili per le versioni Kubernetes v1.28 e successive. Le immagini del sistema operativo per queste varianti includono kubelet, containerd aws-iam-authenticator e altri prerequisiti software per EKS Hybrid Nodes. È possibile configurare

questi componenti utilizzando un file di impostazioni Bottlerocket che include dati utente codificati in base64 per i contenitori bootstrap e admin di Bottlerocket. La configurazione di queste impostazioni consente a Bottlerocket di utilizzare il provider di credenziali dei nodi ibridi per autenticare i nodi ibridi nel cluster. Dopo che i nodi ibridi si sono uniti al cluster, verranno visualizzati con lo stato `Not Ready` nella console Amazon EKS e in strumenti compatibili con Kubernetes come `kubectl`. Dopo aver completato i passaggi indicati in questa pagina, procedi a preparare i nodi ibridi the section called "Configurazione della CNI" per l'esecuzione delle applicazioni.

Prerequisiti

Prima di connettere i nodi ibridi al cluster Amazon EKS, assicurati di aver completato tutti i passaggi relativi ai prerequisiti.

- Hai una connettività di rete dal tuo ambiente locale alla AWS regione che ospita il tuo cluster Amazon EKS. Per ulteriori informazioni, consulta the section called "Preparazione della rete".
- Hai creato il tuo ruolo IAM Hybrid Nodes e configurato il tuo provider di credenziali locale (AWS Systems Manager hybrid activations o AWS IAM Roles Anywhere). Per ulteriori informazioni, consulta the section called "Preparazione delle credenziali".
- Hai creato il cluster Amazon EKS abilitato per i nodi ibridi. Per ulteriori informazioni, consulta the section called "Crea cluster".
- Hai associato il ruolo IAM Nodi ibridi alle autorizzazioni con il controllo degli accessi basato sui ruoli (RBAC) di Kubernetes. Per ulteriori informazioni, consulta the section called "Preparazione dell'accesso ai cluster".

Passaggio 1: creazione del file TOML delle impostazioni Bottlerocket

Per configurare Bottlerocket per i nodi ibridi, è necessario creare un file `settings.toml` con la configurazione necessaria. Il contenuto del file TOML sarà diverso in base al provider di credenziali utilizzato (SSM o IAM Roles Anywhere). Questo file verrà passato come dati utente durante il provisioning dell'istanza Bottlerocket.

> **Note**
>
> I file TOML forniti di seguito rappresentano solo le impostazioni minime richieste per inizializzare una VMWare macchina Bottlerocket come nodo su un cluster EKS. Bottlerocket offre un'ampia gamma di impostazioni per soddisfare diversi casi d'uso, quindi per ulteriori opzioni di configurazione oltre all'inizializzazione del nodo ibrido, consulta la documentazione di Bottlerocket per l'elenco completo di tutte le impostazioni documentate per la versione

Bottlerocket che stai utilizzando (ad esempio, ecco tutte le impostazioni disponibili per Bottlerocket 1.51.x).

SSM

Se utilizzi AWS Systems Manager come provider di credenziali, crea un `settings.toml` file con il seguente contenuto:

```
[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
```

Sostituire i segnaposto con i seguenti valori:

- `<cluster-name>`: il nome del cluster Amazon EKS.
- `<api-server-endpoint>`: l'endpoint del server API del cluster.
- `<cluster-certificate-authority>`: il bundle CA con codifica base64 del cluster.
- `<region>`: La AWS regione che ospita il cluster, ad esempio «us-east-1".
- `<hostname>`: il nome host dell'istanza Bottlerocket, che verrà configurato anche come nome del nodo. Questo può essere un valore univoco a tua scelta, ma deve rispettare le convenzioni di denominazione degli oggetti Kubernetes. Inoltre, il nome host che usi non può superare i 64 caratteri. NOTA: quando si utilizza il provider SSM, il nome host e il nome del nodo verranno sostituiti dall'ID dell'istanza gestita (ad esempio, `mi-*` ID) dopo la registrazione dell'istanza con SSM.
- `<base64-encoded-admin-container-userdata>`: il contenuto con codifica base64 della configurazione del container di amministrazione Bottlerocket. L'abilitazione del container di amministrazione consente di connettersi all'istanza Bottlerocket con SSH per l'esplorazione e il debug del sistema. Sebbene questa non sia un'impostazione obbligatoria, consigliamo di abilitarla per facilitare la risoluzione dei problemi. Consulta Bottlerocket admin container documentation per ulteriori informazioni sull'autenticazione con il container di amministrazione. Il container di amministrazione accetta l'input dell'utente e della chiave SSH in formato JSON, ad esempio

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>`: il contenuto con codifica base64 della configurazione del container del bootstrap Bottlerocket. Consulta Bottlerocket bootstrap container documentation per ulteriori informazioni sulla sua configurazione. Il contenitore bootstrap è responsabile della registrazione dell'istanza come istanza gestita AWS SSM e dell'aggiunta come nodo Kubernetes sul tuo cluster Amazon EKS. I dati utente passati nel container bootstrap assumono la forma di un'invocazione di comando che accetta come input il codice di attivazione ibrido SSM e l'ID creati in precedenza:

```
eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-code=<activation-
code> --region=<region>
```

IAM Roles Anywhere

Se utilizzi AWS IAM Roles Anywhere come provider di credenziali, crea un file con il seguente contenuto: `settings.toml`

```
[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>"
```

Sostituire i segnaposto con i seguenti valori:

- `<cluster-name>`: il nome del cluster Amazon EKS.
- `<api-server-endpoint>`: l'endpoint del server API del cluster.
- `<cluster-certificate-authority>`: il bundle CA con codifica base64 del cluster.
- `<region>`: La AWS regione che ospita il cluster, ad esempio «us-east-1"
- `<hostname>`: il nome host dell'istanza Bottlerocket, che verrà configurato anche come nome del nodo. Questo può essere un valore univoco a tua scelta, ma deve rispettare le convenzioni di denominazione degli oggetti Kubernetes. Inoltre, il nome host che usi non può superare i 64 caratteri. NOTA: quando si utilizza il provider IAM-RA, il nome del nodo deve corrispondere al CN del certificato sull'host se è stata configurata la policy di attendibilità del ruolo IAM dei nodi ibridi con la condizione della risorsa `"sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"`.
- `<base64-encoded-aws-config-file>`: I contenuti codificati in base64 del file di configurazione. AWS Il contenuto del file dovrebbe essere il seguente:

```
[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>`: il contenuto con codifica base64 della configurazione del container di amministrazione Bottlerocket. L'abilitazione del container di amministrazione consente di connettersi all'istanza Bottlerocket con SSH per l'esplorazione e il debug del sistema. Sebbene questa non sia un'impostazione obbligatoria, consigliamo di abilitarla per facilitare la risoluzione dei problemi. Consulta Bottlerocket admin container documentation per ulteriori informazioni sull'autenticazione con il container di amministrazione. Il container di amministrazione accetta l'input dell'utente e della chiave SSH in formato JSON, ad esempio

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>`: il contenuto con codifica base64 della configurazione del container del bootstrap Bottlerocket. Consulta Bottlerocket bootstrap container documentation per ulteriori informazioni sulla sua configurazione. Il container bootstrap è responsabile della creazione del certificato host IAM Roles Anywhere e dei file delle chiavi private del certificato sull'istanza. Questi verranno quindi utilizzati da `aws_signing_helper` per ottenere credenziali temporanee per l'autenticazione con il cluster Amazon EKS. I dati utente passati nel container bootstrap assumono la forma di un'invocazione di comando che accetta come input i contenuti del certificato la chiave privata creati in precedenza:

```
eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key>
```

Passaggio 2: fornitura dei dati utente alla macchina virtuale Bottlerocket vSphere

Dopo aver creato il file TOML, passalo come dati utente durante la creazione della macchina virtuale vSphere. Tieni presente che i dati utente devono essere configurati prima che la macchina virtuale venga accesa per la prima volta. Pertanto, sarà necessario fornirli al momento della creazione dell'istanza o, se si desidera creare la macchina virtuale in anticipo; quest'ultima deve essere nello stato PoweredOff fino a quando non si configurano i relativi dati utente. Ad esempio, se utilizzi la `govc` CLI:

Creazione della macchina virtuale per la prima volta

```
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>
```

Aggiornamento dei dati utente per una macchina virtuale esistente

```
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>
```

Nelle sezioni precedenti, l'opzione `-e guestinfo.userdata.encoding="base64"` specifica che i dati utente sono codificati con Base64. L'opzione `-e guestinfo.userdata` passa i contenuti codificati con Base64 del file `settings.toml` come dati utente all'istanza Bottlerocket. Sostituisci i segnaposto con i tuoi valori specifici, come il modello OVA Bottlerocket e i dettagli di rete.

Passaggio 3: verifica della connessione del nodo ibrido

Dopo l'avvio, l'istanza Bottlerocket tenterà di entrare a far parte del tuo cluster Amazon EKS. Puoi verificare la connessione nella console Amazon EKS accedendo alla scheda Calcolo per il tuo cluster o eseguendo il seguente comando:

```
kubectl get nodes
```

> **Important**
>
> I nodi avranno lo stato `Not Ready` previsto, dovuto alla mancanza di una CNI in esecuzione sui nodi ibridi. Se i tuoi nodi non si sono uniti al cluster, consulta the section called "risoluzione dei problemi".

Passaggio 4: configurazione di una CNI per i nodi ibridi

Per preparare i nodi ibridi all'esecuzione delle applicazioni, continua con i passaggi su the section called "Configurazione della CNI".

Aggiornamento dei nodi ibridi per il tuo cluster

Le linee guida per l'aggiornamento dei nodi ibridi sono simili ai nodi Amazon EKS autogestiti eseguiti in Amazon EC2. Ti consigliamo di creare nuovi nodi ibridi nella versione di Kubernetes di destinazione, eseguire la migrazione con facilità delle applicazioni esistenti sui nodi ibridi sulla nuova versione di Kubernetes e rimuovere questi ultimi sulla vecchia versione di Kubernetes dal tuo cluster. Assicurati di consultare le Best practice di Amazon EKS per gli aggiornamenti prima di iniziare un aggiornamento. Amazon EKS Hybrid Nodes ha lo stesso Kubernetes version support per i cluster Amazon EKS con nodi cloud, incluso il supporto standard ed esteso.

Amazon EKS Hybrid Nodes segue la stessa policy di disallineamento delle versioni per i nodi di Kubernetes upstream. Amazon EKS Hybrid Nodes non può avere una versione più recente del piano di controllo di Amazon EKS e i nodi ibridi possono contenere fino a tre versioni secondarie di Kubernetes precedenti a quella secondaria del piano di controllo Amazon EKS.

Se non disponi di capacità di riserva per creare nuovi nodi ibridi sulla versione Kubernetes di destinazione per una strategia di conversione della migrazione, puoi in alternativa utilizzare la CLI di Amazon EKS Hybrid Nodes (`nodeadm`) per aggiornare la versione Kubernetes dei tuoi nodi ibridi sul posto.

> **Important**
>
> Se stai aggiornando i tuoi nodi ibridi sul posto con `nodeadm`, è previsto un tempo di inattività per il nodo durante la procedura, in cui la versione precedente dei componenti di Kubernetes viene chiusa e i componenti della nuova versione vengono installati e avviati.

Prerequisiti

Prima di effettuare la connessione, assicurati di avere soddisfatto tutti i prerequisiti.

- La versione Kubernetes di destinazione per l'aggiornamento dei nodi ibridi deve essere uguale o inferiore a quella del piano di controllo di Amazon EKS.
- Se stai seguendo una strategia di conversione della migrazione e dell'aggiornamento, i nuovi nodi ibridi che stai installando sulla versione Kubernetes di destinazione devono soddisfare i requisiti the section called "Prerequisiti". Ciò include la presenza di indirizzi IP all'interno del CIDR della rete di nodi remoti che hai trasferito durante la creazione del cluster Amazon EKS.
- Sia per la conversione della migrazione che per gli aggiornamenti sul posto, i nodi ibridi devono avere accesso ai domini richiesti per sfruttare le nuove versioni delle dipendenze dei nodi ibridi.
- Devi avere kubectl installato sul computer o sull'istanza locale che stai utilizzando per interagire con l'endpoint dell'API Amazon EKS Kubernetes.
- La versione del tuo CNI deve supportare la versione di Kubernetes verso cui stai effettuando l'aggiornamento. In caso contrario, aggiorna la versione del CNI prima di aggiornare i nodi ibridi. Per ulteriori informazioni, consulta the section called "Configurazione della CNI".

Aggiornamenti conversione migrazione (blu-verde)

Gli aggiornamenti della conversione della migrazione si riferiscono al processo di creazione di nuovi nodi ibridi su nuovi host con la versione di Kubernetes di destinazione, con la migrazione graduale delle applicazioni esistenti ai nuovi nodi ibridi sulla versione Kubernetes di destinazione e la rimozione dei nodi ibridi sulla vecchia versione di Kubernetes dal cluster. Questa strategia è anche chiamata migrazione blu-verde.

1. Connetti i tuoi nuovi host come nodi ibridi seguendo i passaggi di the section called "Connessione di nodi ibridi". Quando esegui il comando `nodeadm install`, utilizza la versione di Kubernetes di destinazione.

2. Abilita la comunicazione tra i nuovi nodi ibridi nella versione Kubernetes di destinazione e quelli nella vecchia versione di Kubernetes. Ciò consente ai pod di comunicare tra loro durante la migrazione del carico di lavoro verso i nodi ibridi nella versione Kubernetes di destinazione.
3. Verifica che i nodi ibridi sulla versione di Kubernetes di destinazione si uniscano correttamente al cluster e mostrino lo stato Pronto.
4. Utilizza il comando seguente per contrassegnare come non programmabile ciascuno dei nodi che desideri rimuovere. In questo modo i nuovi pod non vengono pianificati né riprogrammati sui nodi che stai sostituendo. Per ulteriori informazioni, consulta cordonazione di kubelet nella documentazione di Kubernetes. Sostituisci `NODE_NAME` con il nome dei nodi ibridi nella vecchia versione di Kubernetes.

   ```
   kubectl cordon NODE_NAME
   ```

   Puoi individuare e isolare tutti i nodi di una determinata versione di Kubernetes (in questo caso `1.28`) con il seguente frammento di codice.

   ```
   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 l'implementazione attuale è in esecuzione per un numero di volte inferiore a due repliche CoreDNS, aumenta orizzontalmente l'implementazione a due repliche. Consigliamo di eseguire almeno due repliche CoreDNS su nodi ibridi per garantire la resilienza durante le normali operazioni.

   ```
   kubectl scale deployments/coredns --replicas=2 -n kube-system
   ```

6. Svuota ciascuno dei nodi della vecchia versione di Kubernetes che desideri rimuovere dal cluster con il comando seguente. Per ulteriori informazioni su come svuotare i nodi, consulta Safely Drain a Node nella documentazione di Kubernetes. Sostituisci `NODE_NAME` con il nome dei nodi ibridi nella vecchia versione di Kubernetes.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

Puoi individuare e svuotare tutti i nodi di una determinata versione di Kubernetes (in questo caso, `1.28`) con il seguente frammento di codice.

```
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. Puoi utilizzare `nodeadm` per arrestare e rimuovere gli artefatti dei nodi ibridi dall'host. Devi eseguire `nodeadm` con un utente con privilegi root/sudo. Per impostazione predefinita, `nodeadm uninstall` non proseguirà se sul nodo rimangono dei pod. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi".

```
nodeadm uninstall
```

8. Una volta interrotti e disinstallati gli artefatti dei nodi ibridi, rimuovi la risorsa del nodo dal cluster.

```
kubectl delete node node-name
```

Puoi individuare ed eliminare tutti i nodi di una determinata versione di Kubernetes (in questo caso `1.28`) con il seguente frammento di codice.

```
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. A seconda del CNI scelto, potrebbero rimanere degli artefatti sui nodi ibridi dopo aver eseguito i passaggi precedenti. Per ulteriori informazioni, consulta the section called "Configurazione della CNI".

Aggiornamenti sul posto

La procedura di aggiornamento sul posto si riferisce all'utilizzo di `nodeadm upgrade` per l'aggiornamento della versione di Kubernetes per nodi ibridi senza utilizzare nuovi host fisici o virtuali e alla conversione della strategia di migrazione. Il processo `nodeadm upgrade` chiude i componenti Kubernetes precedenti esistenti in esecuzione sul nodo ibrido, disinstalla i componenti Kubernetes precedenti esistenti, installa i nuovi componenti Kubernetes di destinazione e avvia i nuovi componenti Kubernetes di destinazione. Si consiglia vivamente di aggiornare un nodo alla volta per ridurre al minimo l'impatto sulle applicazioni in esecuzione sui nodi ibridi. La durata di questo processo dipende dalla larghezza di banda e dalla latenza della rete.

1. Usa il comando seguente per contrassegnare come non programmabile il nodo che stai aggiornando. In questo modo i nuovi pod non vengono pianificati né riprogrammati sul nodo che stai sostituendo. Per ulteriori informazioni, consulta cordonazione di kubelet nella documentazione di Kubernetes. Sostituisci `NODE_NAME` con il nome del nodo ibrido che stai aggiornando

```
kubectl cordon NODE_NAME
```

2. Svuota il nodo da aggiornare con il comando seguente. Per ulteriori informazioni su come svuotare i nodi, consulta Safely Drain a Node nella documentazione di Kubernetes. Sostituisci `NODE_NAME` con il nome del nodo ibrido che stai aggiornando.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

3. Esegui `nodeadm upgrade` sul nodo ibrido che stai aggiornando. Devi eseguire `nodeadm` con un utente con privilegi root/sudo. Il nome del nodo viene preservato durante l'aggiornamento per i provider di credenziali SSM di AWS e IAM Roles Anywhere di AWS. Durante il processo di aggiornamento non puoi modificare i provider di credenziali. Consulta the section called "nodeadm dei nodi ibridi" per i valori di configurazione per `nodeConfig.yaml`. Sostituisci `K8S_VERSION` con la versione di Kubernetes di destinazione verso cui esegui l'aggiornamento.

```
nodeadm upgrade K8S_VERSION -c file://nodeConfig.yaml
```

4. Per consentire la pianificazione dei pod sul nodo dopo l'aggiornamento, digita quanto segue. Sostituisci `NODE_NAME` con il nome del nodo.

```
kubectl uncordon NODE_NAME
```

5. Osserva lo stato dei nodi ibridi e attendi fino a quando i nodi si chiudono e si riavviano sulla nuova versione di Kubernetes con lo stato Pronto.

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

Aggiornamenti di sicurezza delle patch per i nodi ibridi

Questo argomento descrive la procedura per eseguire l'applicazione sul posto delle patch degli aggiornamenti di sicurezza per pacchetti e dipendenze specifici in esecuzione sui nodi ibridi. Come best practice, ti consigliamo di aggiornare regolarmente i nodi ibridi per ricevere CVE e patch di sicurezza.

Per i passaggi per aggiornare la versione di Kubernetes, consulta the section called "Aggiornamento dei nodi ibridi".

Un esempio di software che potrebbe richiedere l'applicazione di patch di sicurezza è `containerd`.

`Containerd`

`containerd` è il runtime standard del container Kubernetes e la dipendenza principale per EKS Hybrid Nodes. Viene utilizzato per gestire il ciclo di vita dei container, inclusa l'estrazione delle immagini e la gestione dell'esecuzione dei container. Su un nodo ibrido, puoi installare `containerd` tramite la CLI nodeadm o manualmente. A seconda del sistema operativo del nodo, `nodeadm` installerà `containerd` dal pacchetto distribuito dal sistema operativo o dal pacchetto Docker.

Una volta pubblicato un CVE in `containerd`, hai le seguenti opzioni per eseguire l'aggiornamento alla versione con patch di `containerd` sui tuoi nodi ibridi.

Passaggio 1: controlla se la patch è stata pubblicata sui gestori di pacchetti

Puoi verificare se la patch CVE `containerd` è stata pubblicata su ogni rispettivo gestore di pacchetti del sistema operativo facendo riferimento ai corrispondenti bollettini sulla sicurezza:

- Amazon Linux 2023

- RHEL
- Ubuntu 20.04
- Ubuntu 22.04
- Ubuntu 24.04

Se utilizzi il repository Docker come fonte di `containerd`, puoi controllare gli annunci di sicurezza del Docker per identificare la disponibilità della versione con patch nel repository di Docker.

Passaggio 2: scegli il metodo per installare la patch

Esistono tre metodi per applicare le patch e installare gli aggiornamenti di sicurezza sui nodi. Il metodo che puoi utilizzare dipende dal fatto che la patch sia disponibile o meno nel sistema operativo nel gestore di pacchetti:

1. Per installare le patch con `nodeadm upgrade` pubblicate nei gestori di pacchetti, consulta Step 2 a.
2. Per installare direttamente le patch con i gestori di pacchetti, consulta Step 2 b.
3. Per installare patch personalizzate che non sono pubblicate nei gestori di pacchetti. Nota che ci sono considerazioni speciali per le patch personalizzate per `containerd`, Step 2 c.

Passaggio 2 a: applicazione delle patch con **`nodeadm upgrade`**

Dopo aver confermato che la patch CVE `containerd` è stata pubblicata nel sistema operativo o nei repository di Docker (Apt o RPM), puoi utilizzare il comando `nodeadm upgrade` per eseguire l'aggiornamento alla versione più recente di `containerd`. Poiché non si tratta di un aggiornamento della versione di Kubernetes, devi passare alla versione corrente di Kubernetes al comando `nodeadm` upgrade.

```
nodeadm upgrade K8S_VERSION --config-source file:///root/nodeConfig.yaml
```

Passaggio 2 b: applicazione di patch con i gestori di pacchetti del sistema operativo

In alternativa puoi eseguire l'aggiornamento anche tramite il rispettivo gestore di pacchetti e utilizzarlo per aggiornare il pacchetto `containerd` come 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
```

Passaggio 2 c: patch CVE **`Containerd`** non pubblicata nei gestori di pacchetti

Se la `containerd` versione con patch è disponibile solo con altri mezzi anziché nel gestore di pacchetti, ad esempio nelle versioni di GitHub, puoi installare `containerd` dal sito ufficiale di GitHub.

1. Se il computer è già entrato a far parte del cluster come nodo ibrido, devi eseguire il comando `nodeadm uninstall`.
2. Installa i binari ufficiali `containerd`. Puoi utilizzare le fasi di installazione ufficiali su GitHub.
3. Esegui il comando `nodeadm install` con l'argomento `--containerd-source` impostato su `none`, che salterà l'installazione di `containerd` fino a `nodeadm`. Puoi utilizzare il valore di `none` nella sorgente `containerd` per qualsiasi sistema operativo in esecuzione sul nodo.

   ```
   nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --containerd-source
    none
   ```

Rimuovi i nodi ibridi

In questa sezione viene descritto come eliminare i nodi ibridi dal cluster Amazon EKS. Devi eliminare i nodi ibridi utilizzando strumenti compatibili con Kubernetes di propria scelta, come kubectl. I costi per i nodi ibridi si interrompono quando l'oggetto nodo viene rimosso dal cluster Amazon EKS. Per ulteriori informazioni sui prezzi dei nodi ibridi, consulta i Prezzi di Amazon EKS.

> **Important**
>
> La rimozione dei nodi interrompe i carichi di lavoro in esecuzione sul nodo. Prima di eliminare i nodi ibridi, ti consigliamo innanzitutto di svuotare il nodo per spostare i pod su un altro nodo attivo. Per ulteriori informazioni su come svuotare i nodi, consulta Safely Drain a Node nella documentazione Kubernetes.

Esegui i seguenti passaggi di kubectl dal computer o dall'istanza locale che usi per interagire con l'endpoint API Kubernetes del cluster Amazon EKS. Se stai usando un file `kubeconfig` specifico, usa il flag `--kubeconfig`.

Passaggio 1: elenca i nodi

```
kubectl get nodes
```

Passaggio 2: svuota il nodo

Consulta kubectl drain nella documentazione di Kubernetes per maggiori informazioni sul comando `kubectl drain`.

```
kubectl drain --ignore-daemonsets <node-name>
```

Passaggio 3: arrestare e disinstallare gli artefatti dei nodi ibridi

Puoi utilizzare il CLI Amazon EKS Hybrid Nodes (`nodeadm`) per arrestare e rimuovere gli artefatti dei nodi ibridi dall'host. Devi eseguire `nodeadm` con un utente che abbia privilegi root/sudo. Per impostazione predefinita, `nodeadm uninstall` non proseguirà se sul nodo rimangono dei pod. Se utilizzi AWS Systems Manager (SSM) come provider di credenziali, il comando `nodeadm uninstall` annulla la registrazione dell'host come istanza gestita AWS SSM. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi".

```
nodeadm uninstall
```

Passaggio 4: eliminare il nodo dal cluster

Una volta interrotti e disinstallati gli artefatti dei nodi ibridi, rimuovi la risorsa del nodo dal cluster.

```
kubectl delete node <node-name>
```

Passaggio 5: verifica la presenza di artefatti rimanenti

A seconda del CNI scelto, potrebbero rimanere degli artefatti sui nodi ibridi dopo aver eseguito i passaggi precedenti. Per ulteriori informazioni, consulta the section called "Configurazione della CNI".

Configurazione di reti di applicazioni, componenti aggiuntivi e webhook per nodi ibridi

Dopo aver creato un cluster EKS per nodi ibridi, configura funzionalità aggiuntive per le reti di applicazioni (CNI, BGP, ingresso, bilanciamento del carico, policy di rete), i componenti aggiuntivi, i webhook e le impostazioni proxy. Per l'elenco completo dei componenti aggiuntivi EKS e della community compatibili con i nodi ibridi, consulta the section called "Configurare i componenti aggiuntivi".

Approfondimenti del cluster EKS EKS include il rilevamento approfondito delle configurazioni errate nella configurazione dei nodi ibridi che potrebbero compromettere la funzionalità del cluster o dei carichi di lavoro. Per ulteriori informazioni sugli approfondimenti del cluster, consulta the section called "Approfondimenti sui cluster".

Di seguito sono elencate le funzionalità e i componenti aggiuntivi comuni puoi utilizzare con i nodi ibridi:

- Container Networking Interface (CNI): AWS supporta Cilium come la CNI per i nodi ibridi. Per ulteriori informazioni, consulta the section called "Configurazione della CNI". Tieni presente che la CNI di AWS VPC non può essere utilizzata con nodi ibridi.
- CoredNS e **`kube-proxy`**: CoreDNS e `kube-proxy` vengono installati automaticamente quando i nodi ibridi si uniscono al cluster EKS. Questi componenti aggiuntivi possono essere gestiti come componenti aggiuntivi EKS dopo la creazione del cluster.
- Ingresso e bilanciamento del carico: puoi utilizzare AWS Load Balancer Controller e Application Load Balancer (ALB) o Network Load Balancer (NLB) con il tipo di destinazione `ip` per i carichi di lavoro in esecuzione su nodi ibridi. AWS supporta le funzionalità integrate di bilanciamento del

carico ingresso, gateway e del servizio Kubernetes di Cilium per i carichi di lavoro in esecuzione su nodi ibridi. Per ulteriori informazioni, consultare the section called "Configurazione dell'ingresso" e the section called "Configurazione dei servizi LoadBalancer".

- Metriche: puoi utilizzare gli scraper senza agenti del servizio gestito da Amazon per Prometheus (AMP), AWS Distro per Open Telemetry (ADOT) e l'agente di Amazon CloudWatch Observability con nodi ibridi. Per utilizzare gli scraper senza agenti AMP per le metriche dei pod sui nodi ibridi, i pod devono essere accessibili dal VPC utilizzato per il cluster EKS.
- Log: puoi abilitare la registrazione del piano di controllo (control-plane) EKS per i cluster abilitati ai nodi ibridi. Puoi utilizzare i componenti aggiuntivi ADOT e l'agente di Amazon CloudWatch Observability di EKS per la registrazione ibrida di nodi e pod.
- Pod Identity e IRSA: puoi utilizzare Pod Identity EKS e ruoli IAM per gli account del servizio (IRSA) con applicazioni in esecuzione su nodi ibridi per consentire l'accesso granulare ai pod in esecuzione su nodi ibridi con altri servizi AWS.
- Webhook: se utilizzi webhook, consulta the section called "Configurazione di webhook" per le considerazioni e i passaggi per eseguire i webhook sui nodi cloud, se non riesci a rendere instradabili le reti di pod on-premises.
- Proxy: se utilizzi un server proxy nell'ambiente on-premises per il traffico in uscita dal data center o dall'ambiente edge, puoi configurare i nodi e il cluster ibridi per usare il server proxy. Per ulteriori informazioni, consulta the section called "Configurazione del proxy".

Argomenti

- Configurazione della CNI per nodi ibridi
- Configurare componenti aggiuntivi per nodi ibridi
- Configurazione di webhook per nodi ibridi
- Configurare il proxy per i nodi ibridi
- Configurazione di Cilium BGP per nodi ibridi
- Configurazione dell'ingresso Kubernetes per nodi ibridi
- Configurazione dei servizi di tipo LoadBalancer per nodi ibridi
- Configurazione delle policy di rete Kubernetes per i nodi ibridi

Configurazione della CNI per nodi ibridi

Cilium è la Container Networking Interface (CNI) AWS supportata per Amazon EKS Hybrid Nodes. Devi installare una CNI affinché i nodi ibridi siano pronti a servire i carichi di lavoro. I nodi ibridi

vengono visualizzati con lo stato `Not Ready` fino all'esecuzione di una CNI. Puoi gestire la CNI con strumenti a tua scelta come Helm. Le istruzioni in questa pagina riguardano la gestione del ciclo di vita di Cilium (installazione, aggiornamento, eliminazione). Consulta the section called "Panoramica dell'ingresso di Cilium e del gateway di Cilium", the section called "Tipo servizio LoadBalancer" e the section called "Configurazione delle policy di rete" per scoprire come configurare Cilium per l'ingresso, il bilanciamento del carico e le policy di rete.

Cilium non è supportato da AWS quando viene eseguito su nodi in Cloud. AWS La CNI di Amazon VPC non è compatibile con i nodi ibridi e la CNI del VPC è configurata con la non-affinità per l'etichetta `eks.amazonaws.com/compute-type: hybrid`.

La documentazione di Calico precedentemente contenuta in questa pagina è stata spostata in EKS Hybrid Examples Repository.

Compatibilità delle versioni

Le versioni `v1.17.x` di Cilium `v1.18.x` sono supportate per EKS Hybrid Nodes per ogni versione di Kubernetes supportata in Amazon EKS.

> **Note**
>
> Requisiti del kernel Cilium v1.18.3: a causa dei requisiti del kernel (kernel Linux >= 5.10), Cilium v1.18.3 non è supportato su:

- Ubuntu 20.04
- Red Hat Enterprise Linux (RHEL) 8

Per i requisiti di sistema, consulta i requisiti di sistema di Cilium.

Consulta il supporto delle versioni Kubernetes per individuare quali sono le versioni supportate da Amazon EKS. EKS Hybrid Nodes ha lo stesso supporto per le versioni di Kubernetes dei cluster Amazon EKS con nodi cloud.

Funzionalità supportate

AWS mantiene versioni di Cilium for EKS Hybrid Nodes basate sul progetto open source Cilium. Per ricevere supporto da AWS Cilium, è necessario utilizzare le build Cilium AWS mantenute e le versioni di Cilium supportate.

AWS fornisce supporto tecnico per le configurazioni predefinite delle seguenti funzionalità di Cilium da utilizzare con EKS Hybrid Nodes. Se si prevede di utilizzare funzionalità che non rientrano nell'ambito del AWS supporto, si consiglia di ottenere un supporto commerciale alternativo per Cilium o di disporre dell'esperienza interna necessaria per risolvere i problemi e contribuire alle correzioni del progetto Cilium.

Funzionalità di Cilium	Supportato da AWS
Conformità della rete Kubernetes	Sì
Connettività del cluster principale	Sì
Famiglia di IP	IPv4
Gestione del ciclo di vita	Helm
Modalità di rete	Incapsulamento VXLAN
Gestione indirizzi IP (IPAM)	Ambito del cluster IPAM (Gestione degli indirizzi IP) di Cilium
Policy di rete	Policy di rete Kubernetes
Border Gateway Protocol (BGP)	Piano di controllo (control-plane) BGP di Cilium
Ingresso Kubernetes	Ingresso di Cilium, gateway di Cilium
Allocazione LoadBalancer IP del servizio	IPAM del bilanciatore del carico di Cilium
Pubblicità LoadBalancer dell'indirizzo IP del servizio	Piano di controllo (control-plane) BGP di Cilium
sostituzione kube-proxy	Sì

Considerazioni su Cilium

- Repository Helm: AWS ospita il grafico Cilium Helm nell'Amazon Elastic Container Registry Public (Amazon ECR Public) presso Amazon EKS Cilium/Cilium. Le versioni disponibili includono:

- 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`

  I comandi in questo argomento utilizzano questo repository. Tieni presente che alcuni comandi `helm repo` non sono validi per i repository Helm in Amazon ECR Public, quindi non puoi fare riferimento a questo repository da un nome di repository Helm locale. Invece, utilizza l'URI completo nella maggior parte dei comandi.

- Per impostazione predefinita, Cilium è configurato per l'esecuzione in modalità sovrapposizione/tunnel con VXLAN come metodo di incapsulamento. Questa modalità ha il minor numero di requisiti sulla rete fisica sottostante.
- Per impostazione predefinita, Cilium maschera l'indirizzo IP di origine di tutto il traffico dei pod in uscita dal cluster all'indirizzo IP del nodo. Se disabiliti il mascheramento, il tuo pod CIDRs deve essere instradabile sulla tua rete locale.
- Se esegui webhook su nodi ibridi, il pod CIDRs deve essere instradabile sulla rete locale. Se i pod non CIDRs sono instradabili sulla rete locale, si consiglia di eseguire webhook sui nodi cloud dello stesso cluster. Per ulteriori informazioni, consulta the section called "Configurazione di webhook" e the section called "Preparazione della rete".
- AWS consiglia di utilizzare la funzionalità BGP integrata di Cilium per rendere il pod instradabile sulla rete locale. CIDRs Per ulteriori informazioni su come configurare il BGP di Cilium con nodi ibridi, consulta the section called "Configurazione di BGP".
- La gestione degli indirizzi IP (IPAM) predefinita in Cilium si chiama Cluster Scope, in cui l'operatore Cilium alloca gli indirizzi IP per ogni nodo in base al pod configurato dall'utente. CIDRs

Installazione di Cilium su nodi ibridi

Procedura

1. Crea un file YAML denominato `cilium-values.yaml`. L'esempio seguente configura Cilium per l'esecuzione solo su nodi ibridi impostando l'affinità per l'etichetta `eks.amazonaws.com/compute-type: hybrid` per l'agente e l'operatore di Cilium.
   - Configura *`clusterPoolIpv4PodCIDRList`* con lo stesso pod che CIDRs hai configurato per le reti di pod remoti del tuo cluster EKS. Ad esempio, `10.100.0.0/24`. L'operatore di Cilium alloca le porzioni di indirizzi IP dall'interno dello spazio IP `clusterPoolIpv4PodCIDRList` configurato. Il CIDR del pod non deve sovrapporsi al nodo CIDR on-premises, al CIDR del VPC o del servizio Kubernetes.

- Configura `clusterPoolIpv4MaskSize` in base ai pod richiesti per nodo. Ad esempio, 25 per una dimensione dei segmenti di /25 di 128 pod per nodo.
- Non modificare `clusterPoolIpv4PodCIDRList` o `clusterPoolIpv4MaskSize` dopo aver implementato Cilium sul cluster, consulta Expanding the cluster pool per ulteriori informazioni.
- Se esegui Cilium in modalità sostitutiva kube-proxy, imposta `kubeProxyReplacement: "true"` nei valori Helm e assicurati di non avere un'implementazione kube-proxy esistente in esecuzione sugli stessi nodi di Cilium.
- L'esempio seguente disabilita il proxy Envoy Layer 7 (L7) utilizzato da Cilium per le policy di rete e l'ingresso L7. Per ulteriori informazioni, consultare the section called "Configurazione delle policy di rete" e the section called "Panoramica dell'ingresso di Cilium e del gateway di Cilium".
- L'esempio seguente configura `loadBalancer.serviceTopology`: `true` affinché la distribuzione del traffico del servizio funzioni correttamente se configurata per i tuoi servizi. Per ulteriori informazioni, consulta the section called "Configurazione della distribuzione del traffico del servizio".
- Per un elenco completo dei valori Helm per Cilium, consulta Helm reference nella documentazione di 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. Installa Cilium nel cluster.
   - Sostituisci `CILIUM_VERSION` con una versione Cilium (ad esempio `1.17.9-0` o`1.18.3-0`). Consigliamo di utilizzare la versione della patch più recente per la versione secondaria di Cilium.
   - Assicurati che i tuoi nodi soddisfino i requisiti del kernel per la versione che scegli. Cilium v1.18.3 richiede un kernel Linux >= 5.10.
   - Se stai utilizzando un file kubeconfig specifico, usa il flag `--kubeconfig` con il comando di installazione Helm.

```
helm install cilium oci://public.ecr.aws/eks/cilium/cilium \
    --version CILIUM_VERSION \
    --namespace kube-system \
    --values cilium-values.yaml
```

3. Conferma che l'installazione di Cilium sia stata eseguita correttamente con i comandi seguenti. Dovresti visualizzare l'implementazione `cilium-operator` e `cilium-agent` in esecuzione su ciascuno dei tuoi nodi ibridi. Inoltre, i nodi ibridi dovrebbero avere lo stato `Ready`. Per informazioni su come configurare Cilium BGP per pubblicizzare il pod sulla rete locale, procedi a. CIDRs the section called "Configurazione di 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
```

Aggiornamento di Cilium su nodi ibridi

Prima di aggiornare l'implementazione di Cilium, consulta attentamente la documentazione sull'aggiornamento di Cilium e le note sull'aggiornamento per comprendere le modifiche nella versione di Cilium di destinazione.

1. Assicurati di aver installato la CLI `helm` nel tuo ambiente a riga di comando. Per istruzioni sull'installazione, consulta la documentazione di Helm.
2. Esegui il controllo preflight dell'aggiornamento di Cilium. Sostituisci `CILIUM_VERSION` con la versione di Cilium di destinazione. Ti consigliamo di eseguire la versione della patch più recente per la versione secondaria di Cilium. Puoi trovare la versione della patch più recente per una determinata versione minore di Cilium nella sezione Stable Releases della documentazione di 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. Dopo aver applicato `cilium-preflight.yaml`, assicurati che il numero di pod `READY` sia lo stesso numero di pod di Cilium in esecuzione.

```
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. Una volta che il numero di pod READY è uguale, assicurati che anche l'implementazione preflight di Cilium sia contrassegnata come READY 1/1. Se viene visualizzato READY 0/1, consulta la

sezione di CNP Validation e risolvi i problemi relativi all'implementazione prima di continuare con l'aggiornamento.

```
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. Elimina il preflight

```
helm uninstall cilium-preflight --namespace kube-system
```

6. Prima di eseguire il comando `helm upgrade`, conserva i valori dell'implementazione in `existing-cilium-values.yaml` o utilizza le opzioni `--set` della riga di comando per le impostazioni quando esegui il comando di aggiornamento. L'operazione di aggiornamento sovrascrive Cilium ConfigMap, quindi è fondamentale che i valori di configurazione vengano trasmessi durante l'aggiornamento.

```
helm get values cilium --namespace kube-system -o yaml > existing-cilium-values.yaml
```

7. Durante le normali operazioni del cluster, tutti i componenti Cilium devono eseguire la stessa versione. I passaggi seguenti descrivono come aggiornare tutti i componenti da una versione stabile a una successiva. Quando esegui l'aggiornamento da una versione secondaria a un'altra, consigliamo di aggiornare prima alla versione della patch più recente per la versione secondaria esistente di Cilium. Per ridurre al minimo le interruzioni, imposta l'opzione `upgradeCompatibility` sulla versione iniziale di Cilium installata in questo 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. (Facoltativo) Se devi eseguire il rollback dell'aggiornamento a causa di problemi, esegui i comandi seguenti.

```
helm history cilium --namespace kube-system
helm rollback cilium [REVISION] --namespace kube-system
```

Eliminazione di Cilium dai nodi ibridi

1. Esegui il comando seguente per disinstallare tutti i componenti Cilium dal tuo cluster. Nota, la disinstallazione della CNI potrebbe influire sulla salute di nodi e pod e non dovrebbe essere eseguita sui cluster di produzione.

   ```
   helm uninstall cilium --namespace kube-system
   ```

   Le interfacce e le rotte configurate da Cilium non vengono rimosse per impostazione predefinita quando il CNI viene rimosso dal cluster, consulta il GitHub problema per ulteriori informazioni.
2. Per ripulire i file e le risorse di configurazione su disco, se si utilizzano le directory di configurazione standard, è possibile rimuovere i file come mostrato dallo `cni-uninstall.sh`script nel repository Cilium su. GitHub
3. Per rimuovere Cilium Custom Resource Definitions (CRDs) dal tuo cluster, puoi eseguire i seguenti comandi.

   ```
   kubectl get crds -oname | grep "cilium" | xargs kubectl delete
   ```

Configurare componenti aggiuntivi per nodi ibridi

Questa pagina descrive le considerazioni sull'esecuzione di componenti aggiuntivi e AWS componenti aggiuntivi della community su Amazon EKS Hybrid Nodes. Per ulteriori informazioni sui componenti aggiuntivi di Amazon EKS e sui processi per la creazione, l'aggiornamento e la rimozione di componenti aggiuntivi dal cluster, consulta the section called "Componenti aggiuntivi Amazon EKS". Salvo diversa indicazione in questa pagina, i processi per la creazione, l'aggiornamento e la rimozione dei componenti aggiuntivi Amazon EKS sono gli stessi per i cluster Amazon EKS con nodi ibridi e per i cluster Amazon EKS con nodi in esecuzione nel cloud. AWS Solo i componenti aggiuntivi inclusi in questa pagina sono stati convalidati per la compatibilità con Amazon EKS Hybrid Nodes.

I seguenti AWS componenti aggiuntivi sono compatibili con Amazon EKS Hybrid Nodes.

AWS componente aggiuntivo	Versioni dei componenti aggiuntivi compatibili
kube-proxy	v1.25.14-eksbuild.2 e versioni successive
CoreDNS	v1.9.3-eksbuild.7 e versioni successive

AWS componente aggiuntivo	Versioni dei componenti aggiuntivi compatibili
AWS Distro per OpenTelemetry (ADOT)	v0.102.1-eksbuild.2 e versioni successive
CloudWatch Agente di osservabilità	v2.2.1-eksbuild.1 e versioni successive
EKS Pod Identity Agent	• v1.3.3-eksbuild.1 e versioni successive, ad eccezione di Bottlerocket • v1.3.7-eksbuild.2 e versioni successive per Bottlerocket
Agente di monitoraggio del nodo	v1.2.0-eksbuild.1 e versioni successive
Controller di snapshot CSI	v8.1.0-eksbuild.1 e versioni successive
AWS Connettore CA privato per Kubernetes	v1.6.0-eksbuild.1 e versioni successive
Driver Amazon FSx CSI	v1.7.0-eksbuild.1 e versioni successive
AWS Fornitore di driver CSI Secrets Store	v2.1.1-eksbuild.1 e versioni successive

I seguenti componenti aggiuntivi della community sono compatibili con Amazon EKS Hybrid Nodes. Per ulteriori informazioni sui componenti aggiuntivi della community, consultathe section called "Componenti aggiuntivi della community".

Componente aggiuntivo	Versioni dei componenti aggiuntivi compatibili
Kubernetes Metrics Server	v0.7.2-eksbuild.1 e versioni successive
cert-manager	v1.17.2-eksbuild.1 e versioni successive
Prometheus Node Exporter	v1.9.1-eksbuild.2 e versioni successive
kube-state-metrics	v2.15.0-eksbuild.4 e versioni successive
DNS esterno	v0.19.0-eksbuild.1 e versioni successive

Oltre ai componenti aggiuntivi Amazon EKS indicati nelle tabelle precedenti, il Collettore Amazon Managed Service for Prometheus e il AWS Load Balancer Controller per l'ingresso delle applicazioni (HTTP) e il bilanciamento del carico (TCP/UDP) sono compatibili con i nodi ibridi.

Esistono AWS componenti aggiuntivi e componenti aggiuntivi della community che non sono compatibili con Amazon EKS Hybrid Nodes. Le versioni più recenti di questi componenti aggiuntivi prevedono una regola di antiaffinità per l'etichetta predefinita `eks.amazonaws.com/compute-type: hybrid` applicata ai nodi ibridi. Ciò impedisce loro di funzionare su nodi ibridi quando vengono distribuiti nei cluster. Se disponi di cluster con nodi ibridi e nodi in esecuzione nel AWS cloud, puoi distribuire questi componenti aggiuntivi nel cluster su nodi in esecuzione nel cloud. AWS Amazon VPC CNI non è compatibile con i nodi ibridi e Cilium e Calico sono supportati come Container Networking Interfaces (CNIs) per i nodi ibridi Amazon EKS. Per ulteriori informazioni, consulta the section called "Configurazione della CNI".

AWS componenti aggiuntivi

Le sezioni seguenti descrivono le differenze tra l'esecuzione di AWS componenti aggiuntivi compatibili su nodi ibridi rispetto ad altri tipi di elaborazione Amazon EKS.

kube-proxy e CoreDNS

EKS installa kube-proxy e CoredNS come componenti aggiuntivi autogestiti per impostazione predefinita quando crei un cluster EKS con l'API e, anche dalla CLI. AWS AWS SDKs AWS É possibile sovrascrivere questi componenti aggiuntivi con i componenti aggiuntivi Amazon EKS dopo la creazione del cluster. Fai riferimento alla documentazione EKS per i dettagli su the section called "kube-proxy" e the section called "CoreDNS". Se stai eseguendo un cluster in modalità mista con nodi ibridi e nodi nel AWS cloud, AWS consiglia di avere almeno una replica CoreDNS sui nodi ibridi e almeno una replica CoredNS sui nodi nel cloud. AWS Per i passaggi di configurazione, consulta la sezione the section called "Configurazione delle repliche CoreDNS".

CloudWatch Agente di osservabilità

L'operatore dell'agente CloudWatch Observability utilizza i webhook. Se esegui l'operatore su nodi ibridi, il tuo pod CIDR on-premises deve essere instradabile sulla rete on-premises e devi configurare il cluster EKS con la tua rete di pod remota. Per ulteriori informazioni, consulta Configure webhooks for hybrid nodes.

Le metriche a livello di nodo non sono disponibili per i nodi ibridi perché CloudWatch Container Insights dipende dalla disponibilità di Instance Metadata Service (IMDS) per le metriche a livello di nodo. Per i nodi ibridi sono disponibili metriche a livello di cluster, carico di lavoro, pod e container.

Dopo aver installato il componente aggiuntivo seguendo i passaggi descritti in Installa l' CloudWatch agente con Amazon CloudWatch Observability, il manifesto del componente aggiuntivo deve essere aggiornato prima che l'agente possa essere eseguito correttamente sui nodi ibridi. Modifica la risorsa `amazoncloudwatchagents` sul cluster per aggiungere la variabile di ambiente `RUN_WITH_IRSA` come mostrato di seguito.

```
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
          ...
```

Raccoglitore gestito di Amazon Managed Prometheus per nodi ibridi

Un servizio gestito da Amazon per Prometheus (AMP) è costituito da uno scraper che rileva e raccoglie i parametri da un cluster Amazon EKS. AMP gestisce lo scraper per te, eliminando la necessità di gestire personalmente istanze, agenti o scraper.

Puoi utilizzare i raccoglitori gestiti AMP senza alcuna configurazione aggiuntiva specifica per i nodi ibridi. Tuttavia, gli endpoint metrici per le applicazioni sui nodi ibridi devono essere raggiungibili dal VPC, compresi i percorsi dal VPC alla rete pod remota CIDRs e le porte aperte nel firewall locale. Inoltre, il cluster deve disporre di un accesso privato agli endpoint del cluster.

Segui i passaggi descritti in Utilizzo di un raccoglitore AWS gestito nella Guida per l'utente di Amazon Managed Service for Prometheus.

AWS Distro per (ADOT) OpenTelemetry

Puoi utilizzare il componente aggiuntivo AWS Distro for OpenTelemetry (ADOT) per raccogliere metriche, log e dati di tracciamento dalle tue applicazioni in esecuzione su nodi ibridi. ADOT utilizza i webhook di ammissione per mutare e convalidare le richieste delle risorse personalizzate del raccoglitore. Se esegui l'operatore ADOT su nodi ibridi, il tuo pod CIDR on-premises deve essere instradabile sulla rete on-premises e devi configurare il cluster EKS con la tua rete di pod remota. Per ulteriori informazioni, consulta Configure webhooks for hybrid nodes.

Segui la procedura descritta in Guida introduttiva a AWS Distro per OpenTelemetry utilizzare i componenti aggiuntivi EKS nella Distro per la documentazione. AWS OpenTelemetry

AWS Controller Load Balancer

È possibile utilizzare AWS Load Balancer Controller e Application Load Balancer (ALB) o Network Load Balancer (NLB) con il `ip` tipo di destinazione per i carichi di lavoro sui nodi ibridi. Le destinazioni IP utilizzate con ALB o NLB devono essere instradabili da. AWSIl controller AWS Load Balancer utilizza anche i webhook. Se esegui l'operatore AWS Load Balancer Controller su nodi ibridi, il tuo pod CIDR locale deve essere instradabile sulla tua rete locale e devi configurare il tuo cluster EKS con la tua rete di pod remoti. Per ulteriori informazioni, consulta Configure webhooks for hybrid nodes.

Per installare il AWS Load Balancer Controller, segui i passaggi indicati in the section called "AWS Application Load Balancer" o. the section called "Network Load Balancer di AWS"

Per l'accesso con ALB, è necessario specificare le annotazioni riportate di seguito. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di applicazione".

```
alb.ingress.kubernetes.io/target-type: ip
```

Per il bilanciamento del carico con NLB, è necessario specificare le annotazioni riportate di seguito. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di rete".

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

EKS Pod Identity Agent

> **Note**
>
> Per implementare correttamente il componente aggiuntivo EKS Pod Identity Agent su nodi ibridi che eseguono Bottlerocket, assicurati che la versione di Bottlerocket sia almeno la v1.39.0. Il Pod Identity Agent non è supportato nelle versioni precedenti di Bottlerocket in ambienti con nodi ibridi.

L'originale Amazon EKS Pod Identity Agent DaemonSet si basa sulla disponibilità di EC2 IMDS sul nodo per ottenere le credenziali richieste. AWS Poiché IMDS non è disponibile sui nodi ibridi, a partire dalla versione 1.3.3-eksbuild.1, il componente aggiuntivo Pod Identity Agent implementa facoltativamente un componente che monta le credenziali richieste. DaemonSet I nodi ibridi che eseguono Bottlerocket richiedono un metodo diverso per montare le credenziali e, a partire dalla versione 1.3.7-eksbuild.2, il componente aggiuntivo Pod Identity Agent ne implementa facoltativamente uno che si rivolge specificamente ai nodi ibridi Bottlerocket. DaemonSet Le sezioni DaemonSets seguenti descrivono il processo per abilitare l'opzione.

Ubuntu/RHEL/AL2023

1. Per utilizzare l'agente Pod Identity sui nodi ibridi Ubuntu/RHEL/Al 2023, `enableCredentialsFile: true` impostalo nella sezione ibrida di `nodeadm` config come mostrato di seguito:

   ```
   apiVersion: node.eks.aws/v1alpha1
   kind: NodeConfig
   spec:
       hybrid:
           enableCredentialsFile: true # <-- Add this
   ```

   Questo configurerà `nodeadm` per creare un file di credenziali da configurare sul nodo sotto `/eks-hybrid/.aws/credentials`, che verrà utilizzato dai pod `eks-pod-identity-agent`. Questo file di credenziali conterrà AWS credenziali temporanee che verranno aggiornate periodicamente.
2. Dopo aver aggiornato la configurazione `nodeadm` su ogni nodo, esegui il seguente comando `nodeadm init` con il tuo `nodeConfig.yaml` per unire i nodi ibridi al cluster Amazon EKS. Se i tuoi nodi si sono uniti al cluster in precedenza, esegui comunque di nuovo il comando `nodeadm init`.

```
nodeadm init -c file://nodeConfig.yaml
```

3. Installa `eks-pod-identity-agent` con il supporto per i nodi ibridi abilitato, utilizzando la AWS CLI o. Console di gestione AWS

   a. AWS CLI: dalla macchina che stai utilizzando per amministrare il cluster, esegui il seguente comando per l'installazione `eks-pod-identity-agent` con il supporto per i nodi ibridi abilitato. Sostituisci `my-cluster` con il nome del cluster.

   ```
   aws eks create-addon \
       --cluster-name my-cluster \
       --addon-name eks-pod-identity-agent \
       --configuration-values '{"daemonsets":{"hybrid":{"create": true}}}'
   ```

   b. Console di gestione AWS: Se stai installando il componente aggiuntivo Pod Identity Agent tramite la AWS console, aggiungi quanto segue alla configurazione opzionale per distribuire i DaemonSet nodi ibridi destinati.

   ```
   {"daemonsets":{"hybrid":{"create": true}}}
   ```

Bottlerocket

1. Per utilizzare l'agente Pod Identity sui nodi ibridi Bottlerocket, aggiungi il flag `--enable-credentials-file=true` al comando utilizzato per i dati utente del contenitore bootstrap Bottlerocket, come descritto in the section called "Connessione dei nodi ibridi con Bottlerocket".

   a. Se stai usando il provider di credenziali SSM, il tuo comando dovrebbe avere il seguente aspetto:

   ```
   eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-
   code=<activation-code> --region=<region> --enable-credentials-file=true
   ```

   b. Se stai usando il provider di credenziali IAM Roles Anywhere, il tuo comando dovrebbe avere il seguente aspetto:

   ```
   eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key> --enable-
   credentials-file=true
   ```

Questo configurerà lo script di bootstrap per creare un file di credenziali sul nodo sotto `/var/eks-hybrid/.aws/credentials`, che verrà utilizzato dai pod `eks-pod-identity-agent`. Questo file di credenziali conterrà AWS credenziali temporanee che verranno aggiornate periodicamente.

2. Installa `eks-pod-identity-agent` con il supporto per i nodi ibridi Bottlerocket abilitato, utilizzando la CLI AWS o. Console di gestione AWS

   a. AWS CLI: dalla macchina che stai utilizzando per amministrare il cluster, esegui il seguente comando per l'installazione `eks-pod-identity-agent` con il supporto per i nodi ibridi Bottlerocket abilitato. Sostituisci `my-cluster` con il nome del cluster.

   ```
   aws eks create-addon \
       --cluster-name my-cluster \
       --addon-name eks-pod-identity-agent \
       --configuration-values '{"daemonsets":{"hybrid-bottlerocket":{"create":
    true}}}'
   ```

   b. Console di gestione AWS: Se stai installando il componente aggiuntivo Pod Identity Agent tramite la AWS console, aggiungi quanto segue alla configurazione opzionale per distribuire i nodi ibridi Bottlerocket destinati. DaemonSet

   ```
   {"daemonsets":{"hybrid-bottlerocket":{"create": true}}}
   ```

Controller di snapshot CSI

A partire dalla versione`v8.1.0-eksbuild.2`, il componente aggiuntivo del controller snapshot CSI applica una regola di antiaffinità morbida per i nodi ibridi, preferendo che il controller `deployment` venga eseguito su EC2 nella stessa regione del piano di controllo di Amazon AWS EKS. La collocazione congiunta `deployment` nella stessa AWS regione del piano di controllo di Amazon EKS migliora la latenza.

Componenti aggiuntivi della community

Le sezioni seguenti descrivono le differenze tra l'esecuzione di componenti aggiuntivi della community compatibili su nodi ibridi rispetto ad altri tipi di elaborazione Amazon EKS.

Kubernetes Metrics Server

Il piano di controllo deve raggiungere l'IP del pod di Metrics Server (o l'IP del nodo se hostNetwork è abilitato). Pertanto, a meno che non esegui Metrics Server in modalità hostNetwork, devi configurare una rete di pod remota durante la creazione del cluster Amazon EKS e devi rendere instradabili gli indirizzi IP del pod. L'implementazione del Border Gateway Protocol (BGP) con il CNI è un modo comune per rendere instradabili gli indirizzi IP dei pod.

cert-manager

`cert-manager` utilizza i webhook. Se esegui `cert-manager` su nodi ibridi, il tuo pod CIDR on-premises deve essere instradabile sulla rete on-premises e devi configurare il cluster EKS con la tua rete di pod remota. Per ulteriori informazioni, consulta Configure webhooks for hybrid nodes.

Configurazione di webhook per nodi ibridi

Questa pagina descrive in dettaglio le considerazioni sull'esecuzione di webhook con nodi ibridi. I webhook vengono utilizzati nelle applicazioni Kubernetes e nei progetti open source, come il Controller del bilanciatore del carico di AWS e l'agente osservabilità di CloudWatch, per eseguire funzionalità di mutazione e convalida in fase di runtime.

Reti di pod instradabili

Se riesci a rendere il CIDR del tuo pod on-premises instradabile sulla tua rete on-premises, puoi eseguire webhook su nodi ibridi. Esistono diverse tecniche che puoi utilizzare per rendere il CIDR del pod on-premises instradabile sulla tua rete on-premises, tra cui Border Gateway Protocol (BGP), route statiche o altre soluzioni di instradamento personalizzate. Il BGP è la soluzione consigliata in quanto è più scalabile e facile da gestire rispetto alle soluzioni alternative che richiedono una configurazione del percorso personalizzata o manuale. AWS supporta le funzionalità BGP di Cilium e Calico per la pubblicità dei CIDR dei pod, consulta the section called "Configurazione della CNI" e the section called "CIDR dei pod remoti instradabili" per ulteriori informazioni.

Reti di pod non instradabili

Se non riesci a rendere il CIDR del tuo pod on-premises instradabile sulla tua rete on-premises e devi eseguire i webhook, ti consigliamo di eseguire tutti i webhook sui nodi cloud nello stesso cluster EKS dei nodi ibridi.

Considerazioni per i cluster in modalità mista

I cluster in modalità mista sono definiti come cluster EKS con nodi ibridi e nodi in esecuzione nel cloud AWS. Quando esegui un cluster in modalità mista, considera i seguenti consigli:

- Esegui la CNI del VPC sui nodi nel cloud AWS e Cilium o Calico sui nodi ibridi. Cilium e Calico non sono supportati da AWS quando vengono eseguiti su nodi in nel cloud AWS.
- Configura i webhook per l'esecuzione su nodi nel cloud AWS. Consulta the section called "Configurazione dei webhook per i componenti aggiuntivi" su come configurare i webhook per e i componenti aggiuntivi di AWS e della community.
- Se le tue applicazioni richiedono pod in esecuzione su nodi nel cloud AWS per comunicare direttamente con pod in esecuzione su nodi ibridi ("comunicazione est-ovest") e utilizzi la CNI del VPC sui nodi nel cloud AWS e Cilium o Calico sui nodi ibridi, il CIDR del tuo pod on-premises deve essere instradabile sulla tua rete on-premises.
- Esegui almeno una replica di CoreDNS sui nodi nel cloud AWS e almeno una replica di CoreDNS sui nodi ibridi.
- Configura la distribuzione del traffico del servizio per mantenerlo locale rispetto alla zona da cui proviene. Per ulteriori informazioni sulla Distribuzione del traffico del servizio, consulta the section called "Configurazione della distribuzione del traffico del servizio".
- Se utilizzi Application Load Balancer (ALB) o Network Load Balancer (NLB) di AWS per il traffico del carico di lavoro in esecuzione su nodi ibridi, è necessario che sia possibile instradare le destinazioni IP utilizzate con ALB o NLB da AWS.
- Il componente aggiuntivo Metrics Server richiede la connettività dal piano di controllo EKS all'indirizzo IP del pod Metrics Server. Se esegui il componente aggiuntivo Metrics Server su nodi ibridi, il CIDR del pod on-premises deve essere instradabile sulla rete on-premises.
- Per raccogliere metriche per i nodi ibridi utilizzando il Servizio gestito da Amazon per Prometheus (AMP), il CIDR del pod on-premises deve essere instradabile sulla rete on-premises. In alternativa, puoi utilizzare il collettore gestito AMP per le metriche e le risorse del piano di controllo EKS in esecuzione nel cloud AWS e il componente aggiuntivo Distro for OpenTelemetry (ADOT) di AWS per raccogliere metriche per i nodi ibridi.

Configurazione di cluster in modalità mista

Per visualizzare i webhook in fase di mutazione e convalida in esecuzione sul cluster, puoi visualizzare il tipo di risorsa Estensioni nel pannello Risorse della console EKS per il cluster oppure utilizzare i seguenti comandi. EKS riporta anche le metriche dei webhook nella dashboard

di osservabilità del cluster, consulta the section called "Dashboard di osservabilità" per ulteriori informazioni.

```
kubectl get mutatingwebhookconfigurations
```

```
kubectl get validatingwebhookconfigurations
```

Configurazione della distribuzione del traffico del servizio

Quando esegui cluster in modalità mista, ti consigliamo di utilizzare la Distribuzione del traffico del servizio per mantenere il traffico del servizio locale rispetto alla zona da cui proviene. La Distribuzione del traffico del servizio (disponibile per le versioni 1.31 e successive di Kubernetes in EKS) è la soluzione consigliata rispetto al Topology Aware Routing perché è più prevedibile. Con la Distribuzione del traffico del servizio, gli endpoint integri della zona riceveranno tutto il traffico di quella zona. Con il Topology Aware Routing, ciascun servizio deve soddisfare diverse condizioni in quella zona per applicare il routing personalizzato, altrimenti indirizza il traffico in modo uniforme verso tutti gli endpoint.

Se utilizzi Cilium come CNI, devi eseguire la CNI con il set `enable-service-topology` su `true` per abilitare la distribuzione del traffico del servizio. Puoi trasferire questa configurazione con il flag di installazione Helm `--set loadBalancer.serviceTopology=true` oppure puoi aggiornare un'installazione esistente con il comando CLI `cilium config set enable-service-topology true` di Cilium. L'agente Cilium in esecuzione su ciascun nodo deve essere riavviato dopo aver aggiornato la configurazione per un'installazione esistente.

Nella sezione seguente è riportato un esempio di come configurare la distribuzione del traffico del servizio per il servizio CoreDNS, ti consigliamo di abilitare la stessa opzione per tutti i servizi del cluster per evitare traffico indesiderato tra gli ambienti.

Configurazione delle repliche CoreDNS

Se stai eseguendo un cluster in modalità mista con nodi ibridi e nodi nel cloud AWS, ti consigliamo di avere almeno una replica CoreDNS sui nodi ibridi e almeno una sui tuoi nodi nel cloud AWS. Per evitare problemi di latenza e di rete in una configurazione di cluster in modalità mista, puoi configurare il servizio CoreDNS in modo da preferire la replica CoreDNS più vicina con la Distribuzione del traffico del servizio.

La Distribuzione del traffico del servizio (disponibile per le versioni 1.31 e successive di Kubernetes in EKS) è la soluzione consigliata rispetto al Topology Aware Routing perché è più prevedibile. Nella

Distribuzione del traffico del servizio, gli endpoint integri della zona riceveranno tutto il traffico di quella zona. Nel Topology Aware Routing, ciascun servizio deve soddisfare diverse condizioni in quella zona per applicare il routing personalizzato, altrimenti indirizza il traffico in modo uniforme verso tutti gli endpoint. I seguenti passaggi configurano la Distribuzione del traffico del servizio.

Se utilizzi Cilium come CNI, devi eseguire la CNI con il set `enable-service-topology` su `true` per abilitare la Distribuzione del traffico del servizio. Puoi trasferire questa configurazione con il flag di installazione Helm `--set loadBalancer.serviceTopology=true` oppure puoi aggiornare un'installazione esistente con il comando CLI `cilium config set enable-service-topology true` di Cilium. L'agente Cilium in esecuzione su ciascun nodo deve essere riavviato dopo aver aggiornato la configurazione per un'installazione esistente.

1. Aggiungi un'etichetta di zona topologica per ciascuno dei tuoi nodi ibridi, ad esempio `topology.kubernetes.io/zone: onprem`. In alternativa, puoi impostare l'etichetta nella fase `nodeadm init`, specificando l'etichetta nella configurazione di `nodeadm`. Consultare the section called "Configurazione del nodo per personalizzare kubelet (Facoltativo)". Attenzione, ai nodi in esecuzione nel cloud AWS viene applicata automaticamente un'etichetta di zona topologica che corrisponde alla zona di disponibilità (AZ) del nodo.

   ```
   kubectl label node hybrid-node-name topology.kubernetes.io/zone=zone
   ```

2. Aggiungi `podAntiAffinity` all'implementazione di CoreDNS con la chiave della zona di topologia. In alternativa, puoi configurare l'implementazione di CoreDNS durante l'installazione con i componenti aggiuntivi 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. Aggiungi l'impostazione `trafficDistribution: PreferClose` alla configurazione del servizio `kube-dns` per abilitare la Distribuzione del traffico del servizio.

```
kubectl patch svc kube-dns -n kube-system --type=merge -p '{
  "spec": {
    "trafficDistribution": "PreferClose"
  }
}'
```

4. Puoi confermare che la Distribuzione del traffico del servizio è abilitata visualizzando le slice degli endpoint per il servizio `kube-dns`. Le sezioni degli endpoint devono mostrare gli `hints` per le etichette delle zone topologiche, a conferma che la Distribuzione del traffico del servizio è abilitata. Se non vedi l'`hints` per ciascun indirizzo degli endpoint, la Distribuzione del traffico del servizio non è abilitato.

```
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
```

Configurazione dei webhook per i componenti aggiuntivi

I seguenti componenti aggiuntivi utilizzano i webhook e sono supportati per l'uso con nodi ibridi.

- Controller del bilanciatore del carico AWS
- Agente osservabilità di CloudWatch
- Distro for OpenTelemetry (ADOT) di AWS
- `cert-manager`

Consulta le seguenti sezioni per configurare i webhook utilizzati da questi componenti aggiuntivi per l'esecuzione su nodi nel cloud AWS.

Controller del bilanciatore del carico AWS

Per utilizzare il Controller del bilanciatore del carico di AWS in una configurazione di cluster in modalità mista, devi eseguire il controller sui nodi nel cloud AWS. A tale scopo, aggiungi quanto segue alla configurazione dei valori di Helm o specifica i valori utilizzando la configurazione aggiuntiva EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Agente osservabilità di CloudWatch

Il componente aggiuntivo Agente osservabilità di CloudWatch dispone di un operatore Kubernetes che utilizza webhook. Per eseguire l'operatore sui nodi nel cloud AWS in una configurazione di cluster in modalità mista, modifica la configurazione dell'operatore Agente osservabilità di CloudWatch. Non puoi configurare l'affinità dell'operatore durante l'installazione con i componenti aggiuntivi Helm ed EKS (consulta problema containers-roadmap #2431).

```
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
```

Distro for OpenTelemetry (ADOT) di AWS

Il componente aggiuntivo Distro for OpenTelemetry (ADOT) di AWS dispone di un operatore Kubernetes che utilizza i webhook. Per eseguire l'operatore sui nodi nel cloud AWS in una configurazione di cluster in modalità mista, aggiungi quanto segue alla configurazione dei valori di Helm o specifica i valori utilizzando la configurazione aggiuntiva EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
```

```
          - hybrid
```

Se il CIDR del pod non è instradabile sulla rete on-premises, il collettore ADOT deve essere eseguito su nodi ibridi per acquisire le metriche dai nodi ibridi e dai carichi di lavoro in esecuzione su di essi. A tale scopo, modifica la Definizione di risorse personalizzate (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
```

Puoi configurare il collettore ADOT per acquisire solo le metriche dai nodi ibridi e dalle risorse in esecuzione sui nodi ibridi aggiungendo il seguente `relabel_configs` a ciascun `scrape_configs` nella configurazione CRD del collettore ADOT.

```
relabel_configs:
  - action: keep
    regex: hybrid
    source_labels:
    - __meta_kubernetes_node_label_eks_amazonaws_com_compute_type
```

Il componente aggiuntivo ADOT presenta un requisito preliminare per installare `cert-manager` per i certificati TLS utilizzati dal webhook dell'operatore ADOT. `cert-manager` esegue anche webhook e puoi configurarlo per l'esecuzione su nodi nel cloud AWS con la seguente configurazione dei valori Helm.

```
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`

Il componente aggiuntivo `cert-manager` esegue i webhook e puoi configurarlo per l'esecuzione su nodi nel cloud AWS con la seguente configurazione dei valori Helm.

```
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
```

Configurare il proxy per i nodi ibridi

Se utilizzi un server proxy nell'ambiente on-premises per il traffico in uscita dal data center o dall'ambiente periferico, devi configurare separatamente i nodi e il cluster per usare il server proxy.

Cluster

Sul cluster, devi configurare `kube-proxy` per utilizzare il server proxy. Devi configurare `kube-proxy` dopo aver creato il cluster Amazon EKS.

Nodi

Sui tuoi nodi, devi configurare il sistema operativo, `containerd`, `kubelet` e l'agente Amazon SSM per utilizzare il tuo server proxy. Puoi apportare queste modifiche durante il processo di creazione delle immagini del sistema operativo o prima di eseguire `nodeadm init` su ciascun nodo ibrido.

Configurazione a livello di nodo

Devi applicare le seguenti configurazioni nelle immagini del sistema operativo o prima di eseguire `nodeadm init` su ciascun nodo ibrido.

Configurazione del proxy **`containerd`**

`containerd` è il runtime di gestione dei container predefinito per Kubernetes. Se utilizzi un proxy per l'accesso a Internet, devi configurare `containerd` in modo che possa estrarre le immagini dei container richieste da Kubernetes e Amazon EKS.

Crea un file su ogni nodo ibrido chiamato `http-proxy.conf` nella directory `/etc/systemd/system/containerd.service.d` con i seguenti contenuti. Sostituisci `proxy-domain` e `port` con i valori per il tuo ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configurazione di **`containerd`** a partire dai dati dell'utente

La directory `containerd.service.d` dovrà essere creata per questo file. Dovrai ricaricare systemd per recuperare il file di configurazione senza dover riavviare il sistema. In AL2023, il servizio sarà probabilmente già in esecuzione al momento dell'esecuzione dello script, quindi sarà necessario riavviarlo.

```
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
```

Configurazione del proxy **`kubelet`**

`kubelet` è l'agente del nodo Kubernetes che viene eseguito su ogni nodo Kubernetes ed è responsabile della gestione del nodo e dei pod in esecuzione su di esso. Se utilizzi un proxy nel tuo ambiente on-premises, devi configurare `kubelet` in modo che possa comunicare con gli endpoint pubblici o privati del cluster Amazon EKS.

Crea un file su ogni nodo ibrido chiamato `http-proxy.conf` nella directory `/etc/systemd/system/kubelet.service.d/` col seguente contenuto. Sostituisci `proxy-domain` e `port` con i valori per il tuo ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configurazione di **`kubelet`** a partire dai dati dell'utente

La directory `kubelet.service.d` deve essere creata per questo file. Dovrai ricaricare systemd per recuperare il file di configurazione senza dover riavviare il sistema. In AL2023, il servizio sarà probabilmente già in esecuzione al momento dell'esecuzione dello script, quindi sarà necessario riavviarlo.

```
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
```

Configurazione del proxy **`ssm`**

`ssm` è uno dei provider di credenziali che possono essere utilizzati per inizializzare un nodo ibrido. `ssm` è responsabile dell'autenticazione con AWS e della generazione di credenziali temporanee utilizzate da `kubelet`. Se utilizzi un proxy nel tuo ambiente on-premises e usi `ssm` come fornitore di credenziali sul nodo, devi configurare `ssm` in modo che possa comunicare con gli endpoint del servizio Amazon SSM.

Crea un file su ogni nodo ibrido chiamato `http-proxy.conf` nel percorso seguente, a seconda del sistema operativo

- 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`

Popola il file con i seguenti contenuti. Sostituisci `proxy-domain` e `port` con i valori per il tuo ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configurazione di **`ssm`** a partire dai dati dell'utente

La directory `ssm` del file di servizio systemd deve essere creata per questo file. Il percorso della directory dipende dal sistema operativo utilizzato nel nodo.

- 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`

Sostituisci il nome del servizio systemd nel comando riavvio riportato di seguito a seconda del sistema operativo utilizzato nel nodo

- 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
```

Configurazione del proxy del sistema operativo

Se utilizzi un proxy per l'accesso a Internet, devi configurare il sistema operativo in modo da poter estrarre le dipendenze dei nodi ibridi dal gestore di pacchetti dei sistemi operativi.

Ubuntu

1. Configura `snap` per utilizzare il proxy con i seguenti comandi:

   ```
   sudo snap set system proxy.https=http://proxy-domain:port
   sudo snap set system proxy.http=http://proxy-domain:port
   ```

2. Per abilitare il proxy per `apt`, crea un file chiamato `apt.conf` nella directory `/etc/apt/`. Sostituisci proxy-domain e port con i valori per il tuo ambiente.

   ```
   Acquire::http::Proxy "http://proxy-domain:port";
   Acquire::https::Proxy "http://proxy-domain:port";
   ```

Amazon Linux 2023

1. Configura `dnf` per l'utilizzo del proxy. Crea un file `/etc/dnf/dnf.conf` con i valori proxy-domain e port per il tuo ambiente.

   ```
   proxy=http://proxy-domain:port
   ```

Red Hat Enterprise Linux

1. Configura `yum` per l'utilizzo del proxy. Crea un file `/etc/yum.conf` con i valori proxy-domain e port per il tuo ambiente.

```
proxy=http://proxy-domain:port
```

Configurazione del proxy di IAM Roles Anywhere

Il servizio del fornitore di credenziali IAM Roles Anywhere è responsabile dell'aggiornamento delle credenziali quando si utilizza IAM Roles Anywhere con il flag `enableCredentialsFile` (consulta the section called "EKS Pod Identity Agent"). Se utilizzi un proxy nel tuo ambiente on-premises, devi configurare il servizio in modo che possa comunicare con gli endpoint di IAM Roles Anywhere.

Crea un file denominato `http-proxy.conf` nella directory `/etc/systemd/system/aws_signing_helper_update.service.d/` con i seguenti contenuti. Sostituisci `proxy-domain` e `port` con i valori per il tuo ambiente.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configurazione a livello di cluster

Le configurazioni in questa sezione devono essere applicate dopo aver creato il cluster Amazon EKS e prima di eseguire `nodeadm init` su ogni nodo ibrido.

Configurazione di un proxy kube-proxy

Amazon EKS installa automaticamente `kube-proxy` su ogni nodo ibrido come DaemonSet quando i nodi entrano a far parte del cluster. `kube-proxy` abilita il routing tra servizi supportati da pod su cluster Amazon EKS. Per configurare ogni host, `kube-proxy` necessita della risoluzione DNS per l'endpoint del cluster Amazon EKS.

1. Modifica il DaemonSet `kube-proxy` mediante il comando seguente

```
kubectl -n kube-system edit ds kube-proxy
```

Verrà aperta la definizione del DaemonSet `kube-proxy` sull'editor configurato.

2. Aggiungi le variabili di ambiente per `HTTP_PROXY` e `HTTPS_PROXY`. Nota che la variabile di ambiente `NODE_NAME` dovrebbe già esistere nella configurazione. Sostituisci `proxy-domain` e `port` con valori per il tuo 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
```

Configurazione di Cilium BGP per nodi ibridi

Questo argomento descrive come configurare il Border Gateway Protocol (BGP) di Cilium per Amazon EKS Hybrid Nodes. La funzionalità BGP di Cilium si chiama Cilium BGP Control Plane e può essere utilizzata per pubblicizzare indirizzi di pod e servizi sulla rete on-premises. Per metodi alternativi per rendere i pod CIDR instradabili sulla on-premises, consulta the section called "CIDR dei pod remoti instradabili".

Configurazione del BGP di Cilium

Prerequisiti

- Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".

Procedura

1. Per utilizzare BGP con Cilium per pubblicizzare pod o indirizzi di servizi sulla rete on-premises, è necessario installare Cilium con `bgpControlPlane.enabled: true`. Se stai abilitando BGP per un'implementazione Cilium esistente, devi riavviare l'operatore Cilium per applicare la configurazione BGP se BGP non era stato precedentemente abilitato. È possibile impostare i valori di Helm `operator.rollOutPods` per `true` al fine di riavviare l'operatore Cilium come parte del processo di installazione/aggiornamento di 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. Conferma che l'operatore e gli agenti Cilium sono stati riavviati e sono in esecuzione.

```
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. Crea un file chiamato `cilium-bgp-cluster.yaml` con una definizione `CiliumBGPClusterConfig`. Potrebbe essere necessario ottenere le seguenti informazioni dall'amministratore di rete.
   - Configura `localASN` con l'ASN per i nodi che eseguono Cilium.
   - Configura `peerASN` con l'ASN per il router on-premises.
   - Configura `peerAddress` con l'IP del router on-premises con cui verrà eseguito il peering di ogni nodo su cui è in esecuzione Cilium.

```
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. Applica la configurazione del cluster Cilium BGP al cluster.

```
kubectl apply -f cilium-bgp-cluster.yaml
```

5. Crea un file denominato `cilium-bgp-peer.yaml` con la risorsa `CiliumBGPPeerConfig` che definisce una configurazione peer BGP. Più peer possono condividere la stessa configurazione e fornire un riferimento alla risorsa comune `CiliumBGPPeerConfig`. Consulta BGP Peer configuration nella documentazione di Cilium per un elenco completo delle opzioni di configurazione.

   I valori per le seguenti impostazioni peer Cilium devono corrispondere a quelli del router on-premises con cui si esegue il peering.

   - Configura `holdTimeSeconds` che determina per quanto tempo un peer BGP attende un messaggio keepalive o update prima di dichiarare la sessione inattiva. Il valore predefinito è 90 secondi.
   - Configura `keepAliveTimeSeconds` che determina se un peer BGP è ancora raggiungibile e la sessione BGP è attiva. Il valore predefinito è 30 secondi.
   - Configura `restartTimeSeconds` che determina l'ora in cui il piano di controllo BGP di Cilium dovrebbe ristabilire la sessione BGP dopo un riavvio. Il valore predefinito è 120 secondi.

```
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. Applica la configurazione del peer Cilium BGP al cluster.

```
kubectl apply -f cilium-bgp-peer.yaml
```

7. Crea un file denominato `cilium-bgp-advertisement-pods.yaml` con una risorsa `CiliumBGPAdvertisement` per pubblicizzare i pod CIDR sulla rete on-premises.
   - La risorsa `CiliumBGPAdvertisement` viene utilizzata per definire i tipi di pubblicità e gli attributi ad essi associati. L'esempio seguente configura Cilium per pubblicizzare solo i pod CIDR. Vedi gli esempi in the section called "Tipo servizio LoadBalancer" e the section called "Bilanciamento del carico Cilium all'interno del cluster" per ulteriori informazioni sulla configurazione di Cilium per pubblicizzare gli indirizzi dei servizi.
   - Ogni nodo ibrido che esegue l'agente Cilium esegue il peering con il router upstream abilitato per BGP. Ogni nodo pubblicizza la gamma di pod CIDR che possiede quando `advertisementType` di Cilium è impostato su `PodCIDR` come nell'esempio seguente. Per ulteriori informazioni, consulta BGP Advertisements configuration nella documentazione di Cilium.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisement-pods
  labels:
    advertise: bgp
spec:
  advertisements:
```

```
    - advertisementType: "PodCIDR"
```

8. Applica la configurazione degli annunci pubblicitari BGP di Cilium al cluster.

```
kubectl apply -f cilium-bgp-advertisement-pods.yaml
```

9. È possibile confermare che il peering BGP ha funzionato con la CLI di Cilium utilizzando il comando `cilium bgp peers`. Dovresti vedere i valori corretti nell'output per il tuo ambiente e lo stato della sessione come `established`. Per ulteriori informazioni sulla risoluzione dei problemi, consulta Troubleshooting and Operations Guide nella documentazione di Cilium.

   Negli esempi seguenti, ci sono cinque nodi ibridi che eseguono l'agente Cilium e ogni nodo pubblicizza la gamma Pod CIDR di sua proprietà.

```
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
```

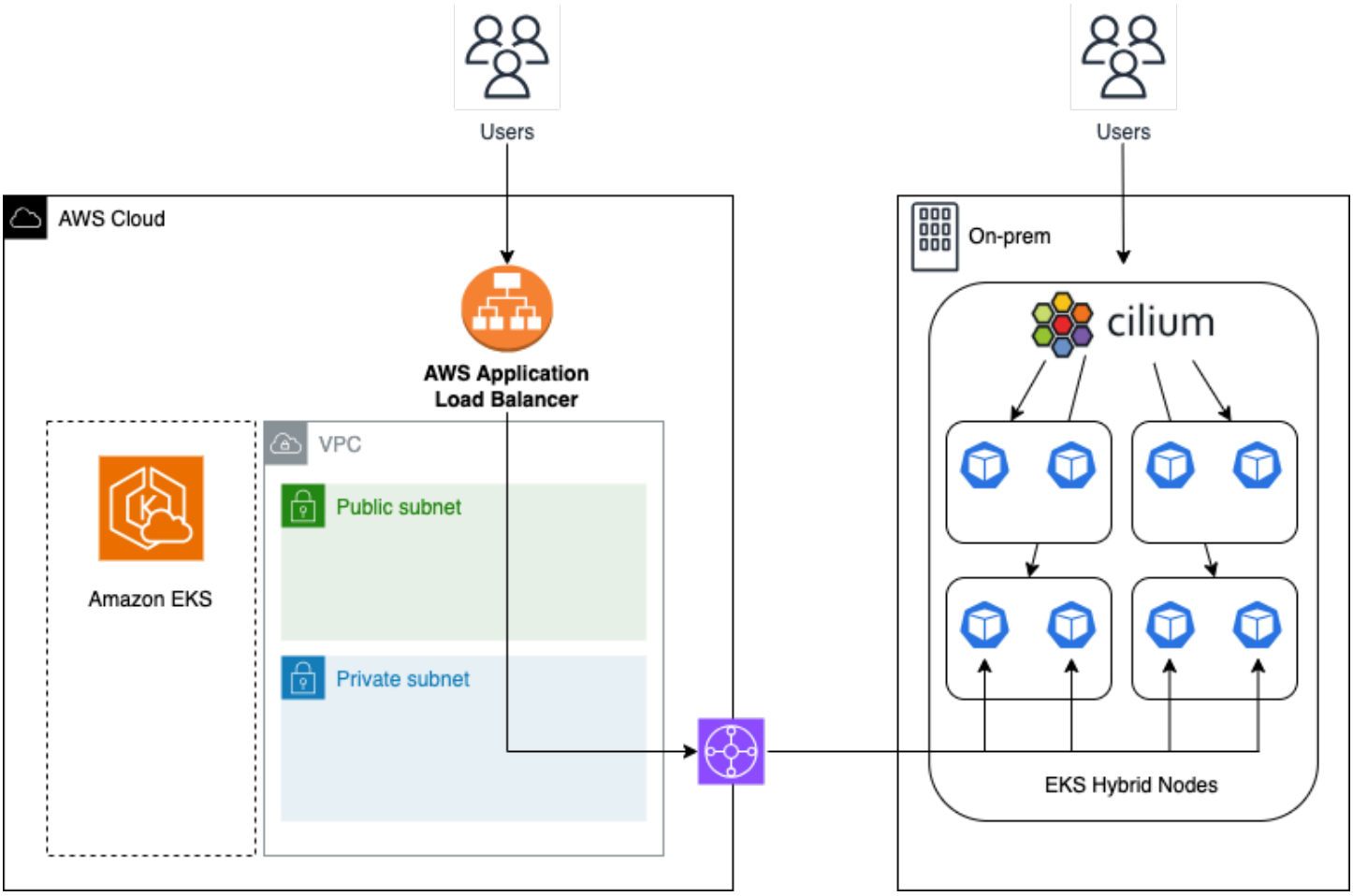
```
cilium bgp routes
```

```
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}]
```

Configurazione dell'ingresso Kubernetes per nodi ibridi

Questo argomento descrive come configurare l'ingresso Kubernetes per i carichi di lavoro in esecuzione su Amazon EKS Hybrid Nodes. L'ingresso Kubernetes espone i percorsi HTTP e HTTPS dall'esterno del cluster ai servizi all'interno del cluster. Per utilizzare le risorse dell'ingresso, è necessario un controller dell'ingresso Kubernetes per configurare l'infrastruttura di rete e i componenti che servono il traffico di rete.

AWS supporta AWS Application Load Balancer (ALB) e Cilium per l'ingresso Kubernetes per carichi di lavoro in esecuzione su nodi ibridi EKS. La decisione di utilizzare ALB o Cilium per l'ingresso si basa sulla fonte del traffico delle applicazioni. Se il traffico dell'applicazione proviene da una Regione AWS, AWS consiglia di utilizzare ALB AWS e il controller del bilanciatore del carico AWS. Se il traffico delle applicazioni proviene dall'ambiente on-premises o edge, AWS consiglia di utilizzare le funzionalità dell'ingresso integrate di Cilium, che possono essere utilizzate con o senza l'infrastruttura di bilanciatore del carico nell'ambiente.



AWS Application Load Balancer

Puoi utilizzare il Controller del bilanciatore del carico AWS e Application Load Balancer (ALB) con il tipo di destinazione `ip` per i carichi di lavoro in esecuzione su nodi ibridi. Quando si utilizza il tipo di destinazione `ip`, ALB inoltra il traffico direttamente ai pod, aggirando il percorso di rete del livello di servizio. Affinché ALB raggiunga le destinazioni IP del pod sui nodi ibridi, il CIDR del pod on-premises deve essere instradabile sulla rete locale. Inoltre, il controller del bilanciatore del carico AWS utilizza webhook e richiede una comunicazione diretta dal piano di controllo EKS. Per ulteriori informazioni, consulta the section called "Configurazione di webhook".

Considerazioni

- Per ulteriori informazioni su AWS Application Load Balancer e il controller del bilanciatore del carico AWS, consulta the section called "Bilanciamento del carico di applicazione" e the section called "Installa con Helm".
- Consulta Best Practices for Load Balancing per informazioni su come scegliere tra AWS Application Load Balancer e AWS Network Load Balancer.

- Consulta AWS Load Balancer Controller Ingress annotations per l'elenco delle annotazioni che possono essere configurate per le risorse di ingresso con AWS Application Load Balancer.

Prerequisiti

- Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".
- Il piano di controllo BGP di Cilium è abilitato seguendo le istruzioni contenute in the section called "Configurazione di BGP". Se non desideri utilizzare BGP, devi utilizzare un metodo alternativo per rendere i pod CIDR on-premises instradabili sulla rete on-premises. Se non rendi instradabili i tuoi pod CIDR on-premises, ALB non sarà in grado di registrarsi o contattare i destinatari IP del pod.
- Helm è installato nell'ambiente a riga di comando, consulta le istruzioni di configurazione di Helm per ulteriori informazioni.
- eksctl è installato nell'ambiente a riga di comando, consulta le istruzioni di installazione di eksctl per ulteriori informazioni.

Procedura

1. Scaricare una policy IAM per il Controller del load balancer AWS che consente di effettuare chiamate alle API AWS per conto dell'utente.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/refs/heads/main/docs/install/iam_policy.json
```

2. Creare una policy IAM utilizzando le policy scaricate nel passaggio precedente.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

3. Sostituisci il valore per nome del cluster (`CLUSTER_NAME`), Regione AWS (`AWS_REGION`) e ID dell'account AWS (`AWS_ACCOUNT_ID`) con le tue impostazioni ed esegui il comando seguente.

```
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. Aggiungi il repository di grafici Helm di eks-charts e aggiorna il repository Helm locale per assicurarti di avere i grafici più recenti.

```
helm repo add eks https://aws.github.io/eks-charts
```

```
helm repo update eks
```

5. Installazione del Controller del bilanciatore del carico AWS. Sostituisci il valore per nome del cluster (`CLUSTER_NAME`), Regione AWS (`AWS_REGION`), ID del VPC (`VPC_ID`) e versione del grafico Helm del controller del bilanciatore del carico AWS (`AWS_LBC_HELM_VERSION`) con le tue impostazioni ed esegui il comando seguente. Se stai eseguendo un cluster in modalità mista con nodi ibridi e nodi nel cloud AWS, puoi eseguire il controller del bilanciatore del carico AWS sui nodi cloud seguendo le istruzioni riportate all'indirizzo the section called "Controller del bilanciatore del carico AWS".
   - Puoi trovare la versione più recente del grafico Helm eseguendo `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. Verifica che il controller del bilanciatore del carico AWS sia stato installato correttamente.

```
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. Crea un'applicazione di esempio. L'esempio seguente utilizza l'applicazione di microservizi di esempio Istio Bookinfo.

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
```

8. Crea un file denominato `my-ingress-alb.yaml` con i seguenti contenuti.

```
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. Applica la configurazione dell'ingresso al cluster.

```
kubectl apply -f my-ingress-alb.yaml
```

10. Il provisioning dell'ALB per la risorsa in ingresso può richiedere alcuni minuti. Una volta effettuato il provisioning dell'ALB, alla risorsa Ingress verrà assegnato un indirizzo che corrisponde al nome DNS dell'implementazione ALB. L'indirizzo avrà il 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 Accedi al Servizio utilizzando l'indirizzo dell'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"
}
```

Panoramica dell'ingresso di Cilium e del gateway di Cilium

Le funzionalità dell'ingresso di Cilium sono integrate nell'architettura di Cilium e possono essere gestite con l'API dell'ingresso di Kubernetes o l'API Gateway. Se non disponi di risorse dell'ingresso esistenti, AWS consiglia di iniziare con l'API Gateway, poiché è un modo più significativo e flessibile per definire e gestire le risorse di rete Kubernetes. L'API del Gateway di Kubernetes mira a standardizzare il modo in cui le risorse di rete per ingresso, bilanciamento del carico e mesh di servizi vengono definite e gestite nei cluster Kubernetes.

Quando abiliti le funzionalità di ingresso o gateway di Cilium, l'operatore Cilium riconcilia gli oggetti ingresso/gateway nel cluster e i proxy Envoy su ciascun nodo elaborano il traffico di rete di livello 7 (L7). Cilium non fornisce direttamente l'infrastruttura ingresso/gateway come i bilanciatori del carico. Se prevedi di utilizzare l'ingresso/gateway di Cilium con un bilanciatore del carico, devi utilizzare gli strumenti di quest'ultimo, in genere un controller di ingresso o gateway, per distribuire e gestire l'infrastruttura del sistema del bilanciatore del carico.

Per il traffico ingresso/gateway, Cilium gestisce il traffico di rete principale e l'applicazione delle policy L3/L4, mentre i proxy Envoy integrati elaborano il traffico di rete L7. Con l'ingresso/gateway di Cilium, Envoy è responsabile dell'applicazione delle regole di routing L7, delle policy e della manipolazione

delle richieste, della gestione avanzata del traffico come la suddivisione e il mirroring del traffico e della terminazione e origine del TLS. I proxy Envoy di Cilium vengono distribuiti come DaemonSet (`cilium-envoy`) separato per impostazione predefinita, il che consente a Envoy e all'agente Cilium di essere aggiornati, scalati e gestiti separatamente.

Per ulteriori informazioni su come funzionano l'ingresso di Cilium e il gateway di Cilium, consulta le pagine Cilium Ingress e Cilium Gateway nella documentazione di Cilium.

Confronto tra ingresso e gateway di Cilium

La tabella seguente riassume le funzionalità di ingresso e gateway di Cilium, a partire dalla versione 1.17.x di Cilium.

Funzionalità	Ingress	Gateway
Tipo servizio LoadBalancer	Sì	Sì
Tipo servizio NodePort	Sì	No[1]
Rete host	Sì	Sì
Bilanciatore del carico condiviso	Sì	Sì
Bilanciatore del carico dedicato	Sì	No[2]
Policy di rete	Sì	Sì
Protocolli	Livello 7 (HTTP(S), gRPC)	Livello 7 (HTTP(S), gRPC)[3]
TLS Passthrough	Sì	Sì
Gestione del traffico	Routing del percorso e dell'host	Routing del percorso e dell'host, reindirizzamento e riscrittura degli URL, suddivisione del traffico, modifica dell'intestazione

[1] Il supporto del gateway di Cilium per i servizi NodePort è pianificato per la versione 1.18.x di Cilium (#27273)

[2] Supporto del gateway di Cilium per bilanciatori del carico dedicati (#25567)

[3] Supporto del gateway di Cilium per TCP/UDP (#21929)

Installazione del gateway di Cilium

Considerazioni

- Cilium deve essere configurato con `nodePort.enabled` impostato su `true` come mostrato negli esempi seguenti. Se utilizzi la funzionalità di sostituzione kube-proxy di Cilium, non devi impostare `nodePort.enabled` su `true`.
- Cilium deve essere configurato con `envoy.enabled` impostato su `true` come mostrato negli esempi seguenti.
- Il gateway di Cilium può essere distribuito nel bilanciatore del carico (impostazione predefinita) o in modalità rete host.
- Quando si utilizza il gateway di Cilium in modalità bilanciatore del carico, l'annotazione `service.beta.kubernetes.io/aws-load-balancer-type: "external"` deve essere impostata sulla risorsa del gateway per impedire al provider AWS cloud legacy di creare un Classic Load Balancer per il servizio di tipo LoadBalancer che Cilium crea per la risorsa gateway.
- Quando si utilizza il gateway di Cilium in modalità rete host, il servizio di tipo LoadBalancer è disabilitato. La modalità di rete host è utile per gli ambienti che non dispongono di un'infrastruttura del bilanciatore del carico; per ulteriori informazioni, consulta the section called "Rete host".

Prerequisiti

1. Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.
2. Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".

Procedura

1. Installa le definizioni di risorse personalizzate (CDR) dell'API del gateway di 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. Crea un file denominato `cilium-gateway-values.yaml` con i seguenti contenuti. L'esempio seguente configura il gateway di Cilium per utilizzare la modalità del bilanciatore del carico predefinita e per utilizzare un proxy DaemonSet for Envoy separato `cilium-envoy` configurato per l'esecuzione solo su nodi ibridi.

```
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. Applica il file di valori Helm al 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. Verifica che l'operatore, l'agente e i pod Envoy Cilium siano in esecuzione.

```
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
```

Configurazione del gateway di Cilium

Il gateway di Cilium è abilitato sugli oggetti del gateway impostando `gatewayClassName` su `cilium`. Il servizio creato da Cilium per le risorse del gateway può essere configurato con campi sull'oggetto del gateway. Le annotazioni comuni utilizzate dai controller del gateway per configurare l'infrastruttura del bilanciatore del carico possono essere configurate con il campo `infrastructure` dell'oggetto del gateway. Quando si utilizza LoadBalancer IPAM di Cilium (vedi esempio in the section called "Tipo servizio LoadBalancer"), l'indirizzo IP da utilizzare per il servizio di tipo LoadBalancer può essere configurato nel campo `addresses` dell'oggetto del Gateway. Per ulteriori informazioni sulla configurazione del gateway, consulta Kubernetes Gateway API specification.

```
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:
```

```
...
```

Cilium e le specifiche del gateway di Kubernetes supportano le risorse GatewayClass, Gateway, HTTPRoute, GRPCRroute e ReferenceGrant.

- Consulta le specifiche HTTPRoute e GRPCRoute per l'elenco dei campi disponibili.
- Vedi gli esempi nella sezione the section called "Implementazione del gateway di Cilium" seguente e gli esempi in Cilium documentation per scoprire come utilizzare e configurare queste risorse.

Implementazione del gateway di Cilium

1. Crea un'applicazione di esempio. L'esempio seguente utilizza l'applicazione di microservizi di esempio Istio Bookinfo.

   ```
   kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
   samples/bookinfo/platform/kube/bookinfo.yaml
   ```

2. Verifica che l'applicazione sia in esecuzione correttamente.

   ```
   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. Crea un file denominato `my-gateway.yaml` con i seguenti contenuti. L'esempio di seguito utilizza l'annotazione `service.beta.kubernetes.io/aws-load-balancer-type: "external"` per impedire al provider AWS cloud legacy di creare un Classic Load Balancer per il servizio di tipo LoadBalancer che Cilium crea per la risorsa 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. Applica la risorsa del gateway al cluster.

```
kubectl apply -f my-gateway.yaml
```

5. Conferma che la risorsa del gateway e il servizio corrispondente sono stati creati. In questa fase, si prevede che il campo ADDRESS della risorsa del gateway non sia popolato con un indirizzo IP o un nome host e che analogamente al servizio di tipo LoadBalancer per la risorsa del gateway non sia assegnato un indirizzo IP o un nome host.

```
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. Procedi the section called "Tipo servizio LoadBalancer" per configurare della risorsa del gateway per utilizzare un indirizzo IP allocato dal bilanciatore del carico IPAM di Cilium e the section called "Tipo servizio NodePort" o the section called "Rete host" per configurare la risorsa del gateway per utilizzare NodePort o gli indirizzi di rete host.

Installazione dell'ingresso di Cilium

Considerazioni

- Cilium deve essere configurato con `nodePort.enabled` impostato su `true` come mostrato negli esempi seguenti. Se utilizzi la funzionalità di sostituzione kube-proxy di Cilium, non devi impostare `nodePort.enabled` su `true`.
- Cilium deve essere configurato con `envoy.enabled` impostato su `true` come mostrato negli esempi seguenti.
- Con `ingressController.loadbalancerMode` impostato su `dedicated`, Cilium crea servizi dedicati per ogni risorsa dell'ingresso. Con `ingressController.loadbalancerMode` impostato su `shared`, Cilium crea un servizio condiviso di tipo LoadBalancer per tutte le risorse dell'ingresso nel cluster. Quando si utilizza la modalità del bilanciatore del carico `shared`, le impostazioni per il servizio condiviso come `labels`, `annotations`, `type` e `loadBalancerIP` sono configurate nella sezione dei valori Helm `ingressController.service`. Per ulteriori informazioni, consulta Cilium Helm values reference.
- Con `ingressController.default` impostato su `true`, Cilium è configurato come controller di ingresso predefinito per il cluster e creerà voci di ingresso anche quando `ingressClassName` non è specificato nelle risorse di ingresso.
- L'ingresso di Cilium può essere distribuito nel bilanciatore del carico (impostazione predefinita), nella porta del nodo o in modalità rete host. Quando Cilium è installato in modalità rete host, il servizio di tipo LoadBalancer e il servizio di tipo NodePort sono disabilitati. Per ulteriori informazioni, consulta the section called "Rete host".

- Imposta sempre `ingressController.service.annotations` su `service.beta.kubernetes.io/aws-load-balancer-type: "external"` nei valori Helm per impedire al provider AWS cloud legacy di creare un Classic Load Balancer per il servizio `cilium-ingress` predefinito creato dal grafico Helm di Cilium.

Prerequisiti

1. Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.
2. Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".

Procedura

1. Crea un file denominato `cilium-ingress-values.yaml` con i seguenti contenuti. L'esempio seguente configura l'ingresso di Cilium per utilizzare la modalità `dedicated` del bilanciatore del carico predefinita e per utilizzare un proxy DaemonSet for Envoy separato `cilium-envoy` configurato per l'esecuzione solo su nodi ibridi.

```
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. Applica il file di valori Helm al 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. Verifica che l'operatore, l'agente e i pod Envoy Cilium siano in esecuzione.

```
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
```

Configurazione dell'ingresso di Cilium

L'ingresso di Cilium è abilitato sugli oggetti dell'ingresso impostando `ingressClassName` su `cilium`. I servizi che Cilium crea per le risorse dell'ingresso possono essere configurati con annotazioni sugli oggetti dell'ingresso quando si utilizza la modalità del bilanciatore del carico `dedicated` e nella configurazione Cilium/Helm quando si utilizza la modalità del bilanciatore del carico `shared`. Queste annotazioni vengono comunemente utilizzate dai controller dell'ingresso per configurare l'infrastruttura del bilanciatore del carico o altri attributi del Servizio come il tipo di servizio, la modalità del bilanciatore del carico, le porte e il passthrough TLS. Le annotazioni chiave sono descritte di seguito. Per un elenco completo delle annotazioni supportate, consulta Cilium Ingress annotations nella documentazione di Cilium.

Annotazione	Descrizione
`ingress.cilium.io/loadbalancer-mode`	`dedicated` : servizio dedicato di tipo LoadBalancer per ogni risorsa dell'ingresso (impostazione predefinita). `shared`: servizio singolo di tipo LoadBalancer per tutte le risorse dell'ingresso.
`ingress.cilium.io/service-type`	`LoadBalancer` : il servizio sarà di tipo LoadBalancer (impostazione predefinita) `NodePort`: il servizio sarà di tipo NodePort.
`service.beta.kubernetes.io/aws-load-balancer-type`	`"external"` : impedisci al provider di servizi AWS cloud legacy di effettuare il provisioning di Classic Load Balancer for Services di tipo LoadBalancer.
`lbipam.cilium.io/ips`	Elenco di indirizzi IP da allocare da Cilium LoadBalancer IPAM

Cilium e le specifiche dell'ingresso di Kubernetes supportano regole di corrispondenza specifiche per Exact, Prefix e Implementation per i percorsi dell'ingresso. Cilium supporta l'espressione regolare come regola di corrispondenza specifica per l'implementazione. Per ulteriori informazioni, consulta Ingress path types and precedence e Path types examples nella documentazione di Cilium e gli esempi nella sezione the section called "Implementazione dell'ingresso di Cilium" di questa pagina.

Un esempio di oggetto dell'ingresso di Cilium è mostrato di seguito.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    service.beta.kuberentes.io/...
    service.kuberentes.io/...
spec:
  ingressClassName: cilium
  rules:
  ...
```

Implementazione dell'ingresso di Cilium

1. Crea un'applicazione di esempio. L'esempio seguente utilizza l'applicazione di microservizi di esempio Istio Bookinfo.

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
```

2. Verifica che l'applicazione sia in esecuzione correttamente.

```
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. Crea un file denominato `my-ingress.yaml` con i seguenti contenuti. L'esempio seguente utilizza l'annotazione `service.beta.kubernetes.io/aws-load-balancer-type: "external"` per impedire al provider AWS cloud legacy di creare un Classic Load Balancer per il servizio di tipo LoadBalancer creato da Cilium per la risorsa dell'ingresso.

```
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. Applica la risorsa dell'ingresso al cluster.

```
kubectl apply -f my-ingress.yaml
```

5. Conferma che la risorsa dell'ingresso e il servizio corrispondente sono stati creati. In questa fase, si prevede che il campo ADDRESS della risorsa dell'ingresso non sia popolato con un indirizzo IP o un nome host e che analogamente al servizio di tipo LoadBalancer condiviso o dedicato per la risorsa dell'ingresso non sia assegnato un indirizzo IP o un nome host.

```
kubectl get ingress my-ingress
```

```
NAME         CLASS    HOSTS   ADDRESS   PORTS   AGE
my-ingress   cilium   *                 80      8s
```

Per la modalità del bilanciatore del carico `shared`

```
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
```

Per la modalità del bilanciatore del carico `dedicated`

```
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. Procedi con the section called "Tipo servizio LoadBalancer" per configurare la risorsa dell'ingresso per utilizzare un indirizzo IP allocato dal bilanciatore del carico IPAM di Cilium e the section called

"Tipo servizio NodePort" o the section called "Rete host" per configurare la risorsa dell'ingresso per utilizzare NodePort o gli indirizzi di rete host.

Tipo servizio LoadBalancer

Infrastruttura del bilanciatore del carico esistente

Per impostazione predefinita, sia per l'ingresso che per il gateway di Cilium, quest'ultimo crea uno o più servizi Kubernetes di tipo LoadBalancer per le risorse ingresso/gateway. Gli attributi dei Servizi creati da Cilium possono essere configurati tramite le risorse dell'ingresso e del gateway. Quando crei risorse ingresso o gateway, l'indirizzo IP o i nomi host esposti esternamente per ingresso o gateway vengono allocati dall'infrastruttura del bilanciatore del carico, che in genere viene fornita da un controller di ingresso o gateway.

Molti controller dell'ingresso e del gateway utilizzano le annotazioni per rilevare e configurare l'infrastruttura del bilanciatore del carico. Le annotazioni per questi controller dell'ingresso e del gateway sono configurate sulle risorse dell'ingresso o del gateway come mostrato negli esempi precedenti. Fai riferimento alla documentazione del controller dell'ingresso o del gateway per le annotazioni che supporta e consulta Kubernetes Ingress documentation e Kubernetes Gateway documentation per un elenco dei controller più diffusi.

> **Important**
>
> L'ingresso e il gateway di Cilium non possono essere utilizzati con il controller del bilanciatore del carico AWS e i Network Load Balancer (NLB) AWS con nodi ibridi EKS. Il tentativo di utilizzarli insieme porta a destinazioni non registrate, poiché l'NLB tenta di connettersi direttamente agli IP del Pod che supportano il servizio di tipo LoadBalancer quando l'NLB `target-type` è impostato su `ip` (requisito per l'utilizzo di NLB con carichi di lavoro in esecuzione su nodi ibridi EKS).

Nessuna infrastruttura del bilanciatore del carico

Se non disponi dell'infrastruttura del bilanciatore del carico e del corrispondente controller ingresso/gateway nel proprio ambiente, le risorse ingresso/gateway e i servizi corrispondenti di tipo LoadBalancer possono essere configurati per utilizzare gli indirizzi IP allocati dalla gestione degli indirizzi IP del bilanciatore del carico (LB IPAM) di Cilium. LB IPAM di Cilium può essere configurato con intervalli di indirizzi IP noti dall'ambiente on-premises e può utilizzare il supporto Border Gateway

Protocol (BGP) integrato di Cilium o gli annunci L2 per pubblicizzare gli indirizzi IP LoadBalancer sulla rete on-premises.

L'esempio seguente mostra come configurare l'IPAM LB di Cilium con un indirizzo IP da utilizzare per le risorse ingresso/gateway e come configurare il piano di controllo di Cilium BGP per pubblicizzare l'indirizzo IP LoadBalancer con la rete on-premises. La funzionalità LB IPAM di Cilium è abilitata per impostazione predefinita, ma non viene attivata finché non viene creata una risorsa `CiliumLoadBalancerIPPool`.

Prerequisiti

- Ingresso e gateway di Cilium installati seguendo le istruzioni in the section called "Installazione dell'ingresso di Cilium" o the section called "Installazione del gateway di Cilium".
- Risorse dell'ingresso o del gateway di Cilium con applicazione di esempio distribuita seguendo le istruzioni in the section called "Implementazione dell'ingresso di Cilium" o the section called "Implementazione del gateway di Cilium".
- Il piano di controllo BGP di Cilium è abilitato seguendo le istruzioni contenute in the section called "Configurazione di BGP". Se non desideri utilizzare BGP, puoi saltare questo prerequisito, ma non sarai in grado di accedere alla tua risorsa dell'ingresso o del gateway finché l'indirizzo IP LoadBalancer allocato da Cilium LB IPAM non sarà instradabile sulla tua rete on-premises.

Procedura

1. Facoltativamente, applica una patch alla risorsa dell'ingresso o del gateway per richiedere un indirizzo IP specifico da utilizzare per il servizio di tipo LoadBalancer. Se non richiedi un indirizzo IP specifico, Cilium ne assegnerà uno dall'intervallo di indirizzi IP configurato nella risorsa `CiliumLoadBalancerIPPool` nel passaggio successivo. Nei comandi seguenti, sostituisci `LB_IP_ADDRESS` con l'indirizzo IP da richiedere per il servizio di tipo LoadBalancer.

   Gateway

   ```
   kubectl patch gateway -n default my-gateway --type=merge -p '{
     "spec": {
       "addresses": [{"type": "IPAddress", "value": "LB_IP_ADDRESS"}]
     }
   }'
   ```

   Ingresso

```
kubectl patch ingress my-ingress --type=merge -p '{
  "metadata": {"annotations": {"lbipam.cilium.io/ips": "LB_IP_ADDRESS"}}
}'
```

2. Crea un file denominato `cilium-lbip-pool-ingress.yaml` con una risorsa `CiliumLoadBalancerIPPool` per configurare l'intervallo di indirizzi IP del bilanciatore del carico per le tue risorse ingresso/gateway.
   - Se utilizzi l'ingresso di Cilium, Cilium applica automaticamente l'etichetta `cilium.io/ingress: "true"` ai Servizi che crea per le risorse dell'ingresso. Puoi utilizzare questa etichetta nel campo `serviceSelector` della definizione delle risorse `CiliumLoadBalancerIPPool` per selezionare i Servizi idonei per LB IPAM.
   - Se utilizzi il gateway di Cilium, puoi usare l'etichetta `gateway.networking.k8s.io/gateway-name` nei campi `serviceSelector` della definizione delle risorse per selezionare le risorse `CiliumLoadBalancerIPPool` gateway idonee per LB IPAM.
   - Sostituisci `LB_IP_CIDR` con l'intervallo di indirizzi IP da utilizzare per quelli del bilanciatore del carico. Per selezionare un singolo indirizzo IP, utilizza un CIDR `/32`. Per ulteriori informazioni, consulta LoadBalancer IP Address Management nella documentazione di 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. Applica la risorsa `CiliumLoadBalancerIPPool` al cluster.

```
kubectl apply -f cilium-lbip-pool-ingress.yaml
```

4. Conferma che un indirizzo IP è stato allocato da Cilium LB IPAM per la risorsa dell'ingresso/gateway.

   Gateway

   ```
   kubectl get gateway my-gateway
   ```

   ```
   NAME         CLASS    ADDRESS        PROGRAMMED   AGE
   my-gateway   cilium   LB_IP_ADDRESS    True         6m41s
   ```

   Ingresso

   ```
   kubectl get ingress my-ingress
   ```

   ```
   NAME         CLASS    HOSTS   ADDRESS        PORTS   AGE
   my-ingress   cilium   *       LB_IP_ADDRESS   80      10m
   ```

5. Crea un file denominato `cilium-bgp-advertisement-ingress.yaml` con una risorsa `CiliumBGPAdvertisement` per pubblicizzare l'intervallo di indirizzi IP del LoadBalancer per le risorse dell'ingresso/gateway. Se non utilizzi un proxy, puoi ignorare questo passaggio. L'indirizzo IP LoadBalancer utilizzato per la risorsa dell'ingresso/gateway deve essere instradabile sulla rete on-premises per poter interrogare il servizio nel passaggio successivo.

   ```
   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. Applica la risorsa `CiliumBGPAdvertisement` al cluster.

```
kubectl apply -f cilium-bgp-advertisement-ingress.yaml
```

7. Accedi al servizio utilizzando l'indirizzo IP assegnato da 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 servizio NodePort

Se non disponi dell'infrastruttura del bilanciatore del carico e del controller dell'ingresso corrispondente nel tuo ambiente, o se gestisci autonomamente la tua infrastruttura del bilanciatore del carico o utilizzi il bilanciatore del carico basato su DNS, puoi configurare l'ingresso di Cilium per creare Servizi di tipo NodePort per le risorse dell'ingresso. Quando si utilizza NodePort con l'ingresso di Cilium, il servizio di tipo NodePort è esposto su una porta su ciascun nodo nell'intervallo di porte 30000-32767. In questa modalità, quando il traffico raggiunge qualsiasi nodo del cluster su NodePort, viene quindi inoltrato a un pod che supporta il servizio, che può trovarsi sullo stesso nodo o su un nodo diverso.

> **Note**
>
> Il supporto del gateway di Cilium per i servizi NodePort è pianificato per la versione 1.18.x di Cilium (#27273)

Prerequisiti

- Ingresso di Cilium installato seguendo le istruzioni in the section called "Installazione dell'ingresso di Cilium".
- Risorse dell'ingresso di Cilium con applicazione di esempio distribuita seguendo le istruzioni in the section called "Implementazione dell'ingresso di Cilium".

Procedura

1. Applica una patch alla risorsa dell'ingresso `my-ingress` esistente per cambiarla dal tipo di servizio LoadBalancer a NodePort.

```
kubectl patch ingress my-ingress --type=merge -p '{
    "metadata": {"annotations": {"ingress.cilium.io/service-type": "NodePort"}}
}'
```

   Se non hai creato la risorsa dell'ingresso, puoi crearla applicando la seguente definizione dell'ingresso al tuo cluster. Nota, la definizione dell'ingresso riportata di seguito utilizza l'applicazione di esempio Istio Bookinfo descritta in the section called "Implementazione dell'ingresso di 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. Conferma che la risorsa del servizio per l'ingresso è stata aggiornata per utilizzare il tipo di servizio NodePort. Nota la porta per il protocollo HTTP nell'output. Nell'esempio seguente, questa è la porta HTTP `32353`, che verrà utilizzata in un passaggio successivo per interrogare il servizio. Il vantaggio dell'utilizzo dell'ingresso di Cilium con il servizio di tipo NodePort è che puoi applicare il routing basato su percorso e host, nonché policy di rete per il traffico in ingresso, cosa che non puoi fare per un servizio standard di tipo NodePort senza l'ingresso.

```
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. Ottieni l'indirizzo IP dei nodi nel 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. Accedi al servizio di tipo NodePort utilizzando gli indirizzi IP dei tuoi nodi e il NodePort catturato sopra. Nell'esempio seguente l'indirizzo IP del nodo utilizzato è `10.80.146.32` e il NodePort è `32353`. Sostituisci questi valori per il tuo 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"
}
```

Rete host

Analogamente al servizio di tipo NodePort, se non disponi di un'infrastruttura del bilanciatore del carico e di un controller dell'ingresso o del gateway, o se gestisci automaticamente il bilanciatore del carico con un bilanciatore del carico esterno, puoi configurare l'ingresso di Cilium e il gateway di Cilium per esporre le risorse dell'ingresso e del gateway direttamente sulla rete host. Quando la modalità di rete host è abilitata per una risorsa dell'ingresso o del gateway, le modalità Service di tipo LoadBalancer e NodePort vengono disabilitate automaticamente, la modalità di rete host si esclude a vicenda con queste modalità alternative per ogni risorsa dell'ingresso o del gateway. Rispetto al servizio di tipo NodePort, la modalità di rete host offre una maggiore flessibilità per la gamma di porte che possono essere utilizzate (non è limitata all'intervallo NodePort 30000-32767) ed è possibile configurare un sottoinsieme di nodi in cui i proxy Envoy vengono eseguiti sulla rete host.

Prerequisiti

- Ingresso e gateway di Cilium installati seguendo le istruzioni in the section called "Installazione dell'ingresso di Cilium" o the section called "Installazione del gateway di Cilium".

Procedura

Gateway

1. Creare un archivio denominato `cilium-gateway-host-network.yaml` con i seguenti contenuti.

```
gatewayAPI:
  enabled: true
  hostNetwork:
    enabled: true
```

```
    # uncomment to restrict nodes where Envoy proxies run on the host network
    # nodes:
    #   matchLabels:
    #     role: gateway
```

2. Applica la configurazione del gateway di Cilium della rete host al 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 non hai creato la risorsa del gateway, puoi crearla applicando la seguente definizione di gateway al tuo cluster. La definizione del gateway riportata di seguito utilizza l'applicazione di esempio Istio Bookinfo descritta in the section called "Implementazione del gateway di Cilium". Nell'esempio seguente, la risorsa del gateway è configurata per utilizzare la porta 8111 per il listener HTTP, che è la porta listener condivisa per i proxy Envoy in esecuzione sulla rete host. Se stai utilizzando una porta privilegiata (inferiore a 1023) per la risorsa del gateway, fai riferimento a Cilium documentation per le istruzioni.

```
---
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
```

Puoi osservare la configurazione Cilium Envoy applicata con il seguente comando.

```
kubectl get cec cilium-gateway-my-gateway -o yaml
```

Puoi ottenere la porta del listener Envoy per il servizio `cilium-gateway-my-gateway` con il seguente comando. In questo esempio, la porta del listener condiviso è `8111`.

```
kubectl get cec cilium-gateway-my-gateway -o jsonpath={.spec.services[0].ports[0]}
```

3. Ottieni l'indirizzo IP dei nodi nel 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. Accedi al servizio utilizzando gli indirizzi IP dei nodi e la porta del listener per la risorsa `cilium-gateway-my-gateway`. Nell'esempio seguente l'indirizzo IP del nodo utilizzato è `10.80.146.32` e la porta del listener è `8111`. Sostituisci questi valori per il tuo 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

A causa di un problema con Cilium upstream (#34028), l'ingresso di Cilium in modalità rete host richiede l'utilizzo di `loadbalancerMode: shared`, che crea un unico servizio di tipo ClusterIP per tutte le risorse dell'ingresso nel cluster. Se stai utilizzando una porta privilegiata (inferiore a 1023) per la risorsa dell'ingresso, fai riferimento a Cilium documentation per le istruzioni.

1. Crea un archivio denominato `cilium-ingress-host-network.yaml` con i seguenti contenuti.

```
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. Applica la configurazione dell'ingresso di Cilium della rete host al tuo 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 non hai creato la risorsa dell'ingresso, puoi crearla applicando la seguente definizione dell'ingresso al tuo cluster. La definizione dell'ingresso riportata di seguito utilizza l'applicazione di esempio Istio Bookinfo descritta in the section called "Implementazione dell'ingresso di 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
```

Puoi osservare la configurazione Cilium Envoy applicata con il seguente comando.

```
kubectl get cec -n kube-system cilium-ingress -o yaml
```

Puoi ottenere la porta del listener Envoy per il servizio `cilium-ingress` con il seguente comando. In questo esempio, la porta del listener condiviso è `8111`.

```
kubectl get cec -n kube-system cilium-ingress -o
 jsonpath={.spec.services[0].ports[0]}
```

3. Ottieni l'indirizzo IP dei nodi nel 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. Accedi al servizio utilizzando gli indirizzi IP dei tuoi nodi e la `sharedListenerPort` per la risorsa`cilium-ingress`. Nell'esempio seguente l'indirizzo IP del nodo utilizzato è `10.80.146.32` e la porta del listener è `8111`. Sostituisci questi valori per il tuo 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"
}
```

Configurazione dei servizi di tipo LoadBalancer per nodi ibridi

Questo argomento descrive come configurare il bilanciamento del carico di livello 4 (L4) per le applicazioni in esecuzione su Amazon EKS Hybrid Nodes. I servizi Kubernetes di tipo LoadBalancer vengono utilizzati per esporre le applicazioni Kubernetes esterne al cluster. I servizi di tipo

LoadBalancer vengono comunemente utilizzati con un'infrastruttura di bilanciamento del carico fisico in ambiente cloud oppure on-premises per servire il traffico del carico di lavoro. Questa infrastruttura di bilanciamento del carico viene in genere fornita con un controller specifico per l'ambiente.

AWS supporta AWS Network Load Balancer (NLB) e Cilium per i servizi di tipo LoadBalancer in esecuzione su nodi ibridi EKS. La decisione di utilizzare NLB o Cilium si basa sulla fonte del traffico dell'applicazione. Se il traffico dell'applicazione proviene da una Regione AWS, AWS consiglia di utilizzare NLB di AWS e il controller del bilanciatore del carico di AWS. Se il traffico delle applicazioni proviene dall'ambiente locale on-premises o edge, AWS consiglia di utilizzare le funzionalità di bilanciamento del carico integrate di Cilium, che possono essere utilizzate con o senza l'infrastruttura di bilanciamento del carico nell'ambiente.

Per il bilanciamento del carico del traffico delle applicazioni di livello 7 (L7), consulta the section called "Configurazione dell'ingresso". Per informazioni generali sul bilanciamento del carico con EKS, consulta Best Practices for Load Balancing.

Network Load Balancer di AWS

Puoi utilizzare il Controller di bilanciatore del carico di AWS e NLB con il tipo di destinazione `ip` per i carichi di lavoro in esecuzione su nodi ibridi. Quando utilizzi il tipo di destinazione `ip`, NLB inoltra il traffico direttamente ai pod, aggirando il percorso di rete del livello di servizio. Affinché NLB raggiunga le destinazioni IP dei pod sui nodi ibridi, i pod CIDR on-premises devono essere instradabili sulla rete on-premises. Inoltre, il Controller del bilanciatore del carico di AWS utilizza webhook e richiede una comunicazione diretta dal piano di controllo EKS. Per ulteriori informazioni, consulta the section called "Configurazione di webhook".

- Consulta the section called "Bilanciamento del carico di rete" per i requisiti di configurazione delle sottoreti, consulta the section called "Installa con Helm" e Best Practices for Load Balancing per ulteriori informazioni sul Network Load Balancer di AWS e il Controller del bilanciatore del carico di AWS.
- Consulta le AWS Load Balancer Controller NLB configurations per le configurazioni che possono essere applicate ai servizi di tipo `LoadBalancer` con il Network Load Balancer di AWS.

Prerequisiti

- Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".
- Il piano di controllo BGP di Cilium è abilitato seguendo le istruzioni contenute in the section called "Configurazione di BGP". Se non desideri utilizzare BGP, devi utilizzare un metodo alternativo per

rendere i pod CIDR on-premises instradabili sulla rete on-premises; consulta the section called "CIDR dei pod remoti instradabili" per ulteriori informazioni.

- Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.
- eksctl è installato nell'ambiente a riga di comando, consulta Setup eksctl instructions.

Procedura

1. Scaricare una policy IAM per il Controller del bilanciatore del carico AWS che consente di effettuare chiamate alle API AWS per conto dell'utente.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/refs/heads/main/docs/install/iam_policy.json
```

2. Creare una policy IAM utilizzando le policy scaricate nel passaggio precedente.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

3. Sostituisci i valori per il nome del cluster (`CLUSTER_NAME`), la Regione AWS (`AWS_REGION`) e l'ID dell'account AWS (`AWS_ACCOUNT_ID`) con le tue impostazioni ed esegui il comando seguente.

```
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. Aggiungi il repository di grafici Helm eks-charts. AWS conserva questo repository su GitHub.

```
helm repo add eks https://aws.github.io/eks-charts
```

5. Aggiorna il repository locale per assicurarti di avere i grafici più recenti.

```
helm repo update eks
```

6. Installazione del Controller del bilanciatore del carico AWS. Sostituisci i valori per il nome del cluster (`CLUSTER_NAME`), la Regione AWS (`AWS_REGION`), l'ID del VPC (`VPC_ID`) e la versione del Controller del bilanciatore del carico di AWS del grafico Helm (`AWS_LBC_HELM_VERSION`) con le tue impostazioni. Puoi trovare la versione più recente del grafico Helm eseguendo `helm search repo eks/aws-load-balancer-controller --versions`. Se stai eseguendo un cluster in modalità mista con nodi ibridi e nodi nel Cloud di AWS, puoi eseguire il Controller del bilanciatore del carico di AWS sui nodi cloud seguendo le istruzioni riportate in the section called "Controller del bilanciatore del carico AWS".

```
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. Verifica che il Controller del bilanciatore del carico di AWS sia stato installato correttamente.

```
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. Definisci un'applicazione di esempio in un file denominato `tcp-sample-app.yaml`. L'esempio seguente utilizza una semplice implementazione NGINX con una 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. Applica l'implementazione al cluster.

```
kubectl apply -f tcp-sample-app.yaml
```

10. Definisci un servizio di tipo LoadBalancer per l'implementazione in un file denominato `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. Applica la mappa di configurazione al cluster.

```
kubectl apply -f tcp-sample-service.yaml
```

12. Il provisioning dell'NLB per il Servizio può richiedere alcuni minuti. Una volta effettuato il provisioning dell'NLB, al Servizio verrà assegnato un indirizzo che corrisponde al nome DNS dell'implementazione dell'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 Accedi al Servizio utilizzando l'indirizzo dell'NLB.

```
curl k8s-default-tcpsampl-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.<region>.amazonaws.com
```

Di seguito viene riportato un output di esempio.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

14 Eliminare tutte le risorse create.

```
kubectl delete -f tcp-sample-service.yaml
kubectl delete -f tcp-sample-app.yaml
```

Bilanciamento del carico Cilium all'interno del cluster

Cilium può essere utilizzato come bilanciatore del carico all'interno del cluster per i carichi di lavoro in esecuzione su nodi ibridi EKS, il che può essere utile per ambienti che non dispongono di un'infrastruttura di bilanciamento del carico. Le funzionalità di bilanciamento del carico di Cilium si basano su una combinazione di funzionalità di Cilium, tra cui la sostituzione del kube-proxy, la gestione indirizzi IP (IPAM) del bilanciatore del carico e il piano di controllo BGP. Le responsabilità di queste funzionalità sono descritte di seguito:

- Sostituzione del Cilium proxy-kube: gestisce il traffico del servizio di routing verso i pod di backend.
- IPAM del bilanciatore del carico Cilium: gestisce gli indirizzi IP che possono essere assegnati a servizi di tipo `LoadBalancer`.

- Piano di controllo BGP Cilium: pubblicizza gli indirizzi IP assegnati dalla IPAM del bilanciatore del carico alla rete on-premises.

Se non utilizzi la sostituzione del kube-proxy Cilium, puoi comunque utilizzare l'IPAM del bilanciatore del carico e il Piano di controllo BGP Cilium per allocare e assegnare indirizzi IP per i servizi di tipo LoadBalancer. Se non utilizzi la sostituzione del kube-proxy Cilium, il bilanciamento del carico dei servizi sui pod di backend viene gestito per impostazione predefinita dalle regole kube-proxy e iptables in EKS.

Prerequisiti

- Cilium è stato installato seguendo le istruzioni contenute in the section called "Configurazione della CNI" con o senza la sostituzione di kube-proxy abilitata. La sostituzione del kube-proxy Cilium richiede l'esecuzione di un sistema operativo con un kernel Linux recente come v4.19.57, v5.1.16 o v5.2.0. Tutte le versioni recenti dei sistemi operativi supportati per l'uso con nodi ibridi soddisfano questi criteri, ad eccezione di Red Hat Enterprise Linux (RHEL) 8.x.
- Il piano di controllo BGP di Cilium è abilitato seguendo le istruzioni contenute in the section called "Configurazione di BGP". Se non desideri utilizzare BGP, devi utilizzare un metodo alternativo per rendere i pod CIDR on-premises instradabili sulla rete on-premises; consulta the section called "CIDR dei pod remoti instradabili" per ulteriori informazioni.
- Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.

Procedura

1. Crea un file denominato `cilium-lbip-pool-loadbalancer.yaml` con una risorsa `CiliumLoadBalancerIPPool` per configurare l'intervallo di indirizzi IP del bilanciatore del carico per i tuoi servizi di tipo LoadBalancer.
   - Sostituisci `LB_IP_CIDR` con l'intervallo di indirizzi IP da utilizzare per quelli del bilanciatore del carico. Per selezionare un singolo indirizzo IP, utilizza un CIDR `/32`. Per ulteriori informazioni, consulta LoadBalancer IP Address Management nella documentazione di Cilium.
   - Il campo `serviceSelector` è configurato in modo che corrisponda al nome del servizio che creerai in un passaggio successivo. Con questa configurazione, gli IP di questo pool verranno assegnati solo ai servizi con lo stesso 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. Applica la risorsa `CiliumLoadBalancerIPPool` al cluster.

```
kubectl apply -f cilium-lbip-pool-loadbalancer.yaml
```

3. Verifica che nel pool sia disponibile almeno un indirizzo IP.

```
kubectl get ciliumloadbalancerippools.cilium.io
```

```
NAME               DISABLED   CONFLICTING   IPS AVAILABLE   AGE
tcp-service-pool   false      False         1               24m
```

4. Crea un file denominato `cilium-bgp-advertisement-loadbalancer.yaml` con una risorsa `CiliumBGPAdvertisement` per pubblicizzare l'indirizzo IP del bilanciatore del carico per il servizio che creerai nel passaggio successivo. Se non utilizzi un BGP di Cilium, puoi ignorare questo passaggio. L'indirizzo IP del bilanciatore del carico utilizzato per il servizio deve essere instradabile sulla rete on-premises per consentire all'utente di interrogare il servizio nella Passaggio finale.
   - Il campo `advertisementType` è impostato su `Service` e `service.addresses` è impostato su `LoadBalancerIP` per pubblicizzare solo l'`LoadBalancerIP` per i servizi di tipo `LoadBalancer`.
   - Il campo `selector` è configurato in modo che corrisponda al nome del servizio che creerai in un passaggio successivo. Con questa configurazione, verrà pubblicizzato solo l'`LoadBalancerIP` per i servizi con lo stesso nome `tcp-sample-service`.

```
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. Applica la risorsa `CiliumBGPAdvertisement` al cluster. Se non utilizzi un BGP di Cilium, puoi ignorare questo passaggio.

```
kubectl apply -f cilium-bgp-advertisement-loadbalancer.yaml
```

6. Definisci un'applicazione di esempio in un file denominato `tcp-sample-app.yaml`. L'esempio seguente utilizza una semplice implementazione NGINX con una 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. Applica l'implementazione al cluster.

```
kubectl apply -f tcp-sample-app.yaml
```

8. Definisci un servizio di tipo LoadBalancer per l'implementazione in un file denominato `tcp-sample-service.yaml`.
   - Puoi richiedere un indirizzo IP specifico dal pool di IP del bilanciatore del carico con l'annotazione `lbipam.cilium.io/ips` sull'oggetto del servizio. Puoi rimuovere questa annotazione se non desideri richiedere un indirizzo IP specifico per il servizio.
   - Il campo della specifica `loadBalancerClass` è obbligatorio per impedire al provider cloud di AWS precedente di creare un Classic Load Balancer per il servizio. Nell'esempio seguente, questo è configurato su `io.cilium/bgp-control-plane` per utilizzare il piano di controllo BGP di Cilium come classe di bilanciatore del carico. In alternativa, questo campo può essere configurato su `io.cilium/l2-announcer` per utilizzare la Funzionalità L2 Announcements di Cilium (attualmente in versione beta e non supportata ufficialmente da 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. Applica il servizio al cluster. Il servizio verrà creato con un indirizzo IP esterno che puoi utilizzare per accedere all'applicazione.

```
kubectl apply -f tcp-sample-service.yaml
```

10. Verifica che il servizio sia stato creato correttamente e che gli sia stato assegnato un IP dal `CiliumLoadBalancerIPPool` creato nel passaggio precedente.

```
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 utilizzi Cilium in modalità kube-proxy sostitutivo, puoi confermare che Cilium sta gestendo il bilanciamento del carico per il servizio eseguendo il comando seguente. Nell'output seguente, gli indirizzi `10.86.2.x` sono gli indirizzi IP dei pod di backend per il servizio.

```
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. Conferma che Cilium stia pubblicizzando l'indirizzo IP sulla rete on-premises tramite BGP. Nell'esempio seguente ci sono cinque nodi ibridi, ciascuno dei quali pubblicizza `LB_IP_ADDRESS` per il servizio `tcp-sample-service` sulla rete 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. Accedi al servizio utilizzando l'indirizzo IP del bilanciatore del carico assegnato.

```
curl LB_IP_ADDRESS
```

Di seguito viene riportato un output di esempio.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

14 Eliminare tutte le risorse create.

```
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
```

Configurazione delle policy di rete Kubernetes per i nodi ibridi

AWS supporta le policy di rete Kubernetes (Livello 3/Livello 4) per il traffico in ingresso e in uscita dei pod quando si utilizza Cilium come CNI con EKS Hybrid Nodes. Se utilizzi cluster EKS con nodi nel cloud AWS, AWS supporta le CNI di Amazon VPC per le policy di rete Kubernetes.

Questo argomento descrive come configurare le policy di rete Cilium e Kubernetes con EKS Hybrid Nodes. Per informazioni dettagliate sulle policy di rete Kubernetes, consulta le Policy di rete Kubernetes nella documentazione di Kubernetes.

Configurazione delle policy di rete

Considerazioni

- AWS supporta le policy di rete Kubernetes upstream e le specifiche per l'ingresso e l'uscita dei pod. Attualmente AWS non supporta né `CiliumNetworkPolicy` né `CiliumClusterwideNetworkPolicy`.
- Il valore Helm `policyEnforcementMode` può essere utilizzato per controllare il comportamento predefinito di applicazione delle policy di Cilium. Il comportamento predefinito consente tutto il traffico in uscita e in ingresso. Quando un endpoint viene selezionato da una policy di rete, passa

allo stato predefinito di negazione, in cui è consentito solo il traffico esplicitamente consentito. Consulta la documentazione di Cilium per ulteriori informazioni su default policy mode e policy enforcement modes.

- Se stai passando da `policyEnforcementMode` a un'installazione Cilium esistente, devi riavviare l'agente Cilium DaemonSet per applicare la nuova modalità di applicazione delle policy.
- Utilizza `namespaceSelector` e `podSelector` per consentire o negare il traffico da/verso namespace e pod con etichette corrispondenti. `namespaceSelector` e `podSelector` possono essere usati con `matchLabels` o `matchExpressions` per selezionare namespace e pod in base alle relative etichette.
- Utilizza `ingress.ports` e `egress.ports` per consentire o negare il traffico da/verso porte e protocolli.
- Il campo `ipBlock` non può essere utilizzato per consentire o negare selettivamente il traffico da/verso gli indirizzi IP dei pod (#9209). L'uso dei selettori `ipBlock` per gli IP dei nodi è una funzionalità beta di Cilium e non è supportata da AWS.
- Consulta la Risorsa NetworkPolicy nella documentazione di Kubernetes per informazioni sui campi disponibili per le policy di rete Kubernetes.

Prerequisiti

- Cilium è installato seguendo le istruzioni contenute in the section called "Configurazione della CNI".
- Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.

Procedura

La procedura seguente imposta le policy di rete per un'applicazione di microservizi di esempio, in modo che i componenti possano comunicare solo con altri componenti necessari per il funzionamento dell'applicazione. La procedura utilizza l'applicazione di microservizi di esempio Istio Bookinfo.

L'applicazione Bookinfo è composta da quattro microservizi separati con le seguenti relazioni:

- productpage. Il microservizio productpage chiama i dettagli ed esamina i microservizi per popolare la pagina.
- details. Il microservizio details contiene informazioni sui libri.
- reviews. Il microservizio reviews contiene le recensioni dei libri. Chiama anche il microservizio di rating.

- ratings. Il microservizio ratings contiene le informazioni sulla classificazione dei libri che accompagnano la recensione del libro.

1. Crea un'applicazione di esempio.

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
```

2. Conferma che l'applicazione funzioni correttamente e annota l'indirizzo IP del pod per il microservizio productpage. Utilizzerai questo indirizzo IP del pod per interrogare ogni microservizio nei passaggi successivi.

```
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. Crea un pod che verrà utilizzato ovunque per testare le policy di rete. Ricorda che il pod viene creato nel namespace `default` con l'etichetta `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. Testa l'accesso al microservizio productpage. Nell'esempio seguente, utilizziamo l'indirizzo IP del pod productpage (`10.86.2.193`) per interrogare il microservizio. Sostituiscilo con l'indirizzo IP del pod productpage nel tuo ambiente.

```
curl -s http://10.86.2.193:9080/productpage | grep -o "<title>.*</title>"
```

```
<title>Simple Bookstore App</title>
```

5. Puoi uscire dal pod test curl digitando `exit` e puoi ricollegarti al pod eseguendo il seguente comando.

```
kubectl attach curl-pod -c curl-pod -i -t
```

6. Per dimostrare gli effetti delle policy di rete nei passaggi seguenti, creiamone innanzitutto una che neghi tutto il traffico per i microservizi BookInfo. Crea un file chiamato `network-policy-deny-bookinfo.yaml` che definisce la policy di negazione della rete.

```
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. Applica la policy di negazione della rete al cluster.

```
kubectl apply -f network-policy-default-deny-bookinfo.yaml
```

8. Testa l'accesso all'applicazione BookInfo. Nell'esempio seguente, utilizziamo l'indirizzo IP del pod productpage (`10.86.2.193`) per interrogare il microservizio. Sostituiscilo con l'indirizzo IP del pod productpage nel tuo ambiente.

```
curl http://10.86.2.193:9080/productpage --max-time 10
```

```
curl: (28) Connection timed out after 10001 milliseconds
```

9. Crea un file chiamato `network-policy-productpage.yaml` che definisce la policy di rete di productpage. La policy ha le seguenti regole:

- Consente il traffico in ingresso dai pod con l'etichetta `access: true` (il pod curl creato nel passaggio precedente)
- Consente il traffico TCP in uscita sulla porta `9080` per i microservizi details, reviews e ratings
- Consente il traffico TCP/UDP in uscita sulla porta `53` per CoreDNS che viene eseguita nel 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. Applica la policy di rete productpage al cluster.

```
kubectl apply -f network-policy-productpage.yaml
```

11. Connettiti al pod curl e testa l'accesso all'applicazione Bookinfo. Ora l'accesso al microservizio productpage è consentito, ma gli altri microservizi vengono comunque negati perché sono ancora soggetti alla policy di negazione della rete. Negli esempi seguenti, utilizziamo l'indirizzo IP del pod productpage (`10.86.2.193`) per interrogare il microservizio. Sostituiscilo con l'indirizzo IP del pod productpage nel tuo 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. Crea un file chiamato `network-policy-details.yaml` che definisce la policy di rete di details. La policy consente solo il traffico in ingresso dal microservizio 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. Crea un file chiamato `network-policy-reviews.yaml` che definisce la policy di rete di reviews. La policy consente solo il traffico in ingresso dal microservizio productpage e solo il traffico in uscita verso il microservizio ratings e 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. Crea un file chiamato `network-policy-ratings.yaml` che definisce la policy della rete di ratings. La policy consente solo il traffico in ingresso dai microservizi productpage e reviews.

```
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. Applica le policy di rete per details, reviews e ratings al cluster.

```
kubectl apply -f network-policy-details.yaml
kubectl apply -f network-policy-reviews.yaml
kubectl apply -f network-policy-ratings.yaml
```

16. Connettiti al pod curl e testa l'accesso all'applicazione Bookinfo. Negli esempi seguenti, utilizziamo l'indirizzo IP del pod productpage (`10.86.2.193`) per interrogare il microservizio. Sostituiscilo con l'indirizzo IP del pod productpage nel tuo ambiente.

```
kubectl attach curl-pod -c curl-pod -i -t
```

Testa il microservice details.

```
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"}
```

Testa il microservizio reviews.

```
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."}]}
```

Testa il microservizio ratings.

```
curl -s http://10.86.2.193:9080/api/v1/products/1/ratings
```

```
{"id": 1, "ratings": {"Reviewer1": 5, "Reviewer2": 4}}
```

17 Elimina tutte le risorse che hai creato con questa procedura.

```
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
```

Concetti per nodi ibridi

Con Amazon EKS Hybrid Nodes, unisci macchine fisiche o virtuali in esecuzione in ambienti on-premises o edge a cluster Amazon EKS in esecuzione nel cloud AWS. Questo approccio offre molti

vantaggi, ma introduce anche nuovi concetti e architetture di rete per chi ha dimestichezza con l'esecuzione di cluster Kubernetes in un unico ambiente di rete.

Le sezioni seguenti approfondiscono Kubernetes e i concetti di rete per i nodi ibridi EKS e descrivono in dettaglio come il traffico fluisce attraverso l'architettura ibrida. Queste sezioni richiedono che tu abbia una conoscenza di base delle reti Kubernetes, come i concetti di pod, nodi, servizi, piano di controllo Kubernetes, kubelet e kube-proxy.

Ti consigliamo di leggere queste pagine in ordine, iniziando con the section called "Concetti di rete", poi con the section called "Concetti di Kubernetes" e infine con the section called "Flussi di traffico".

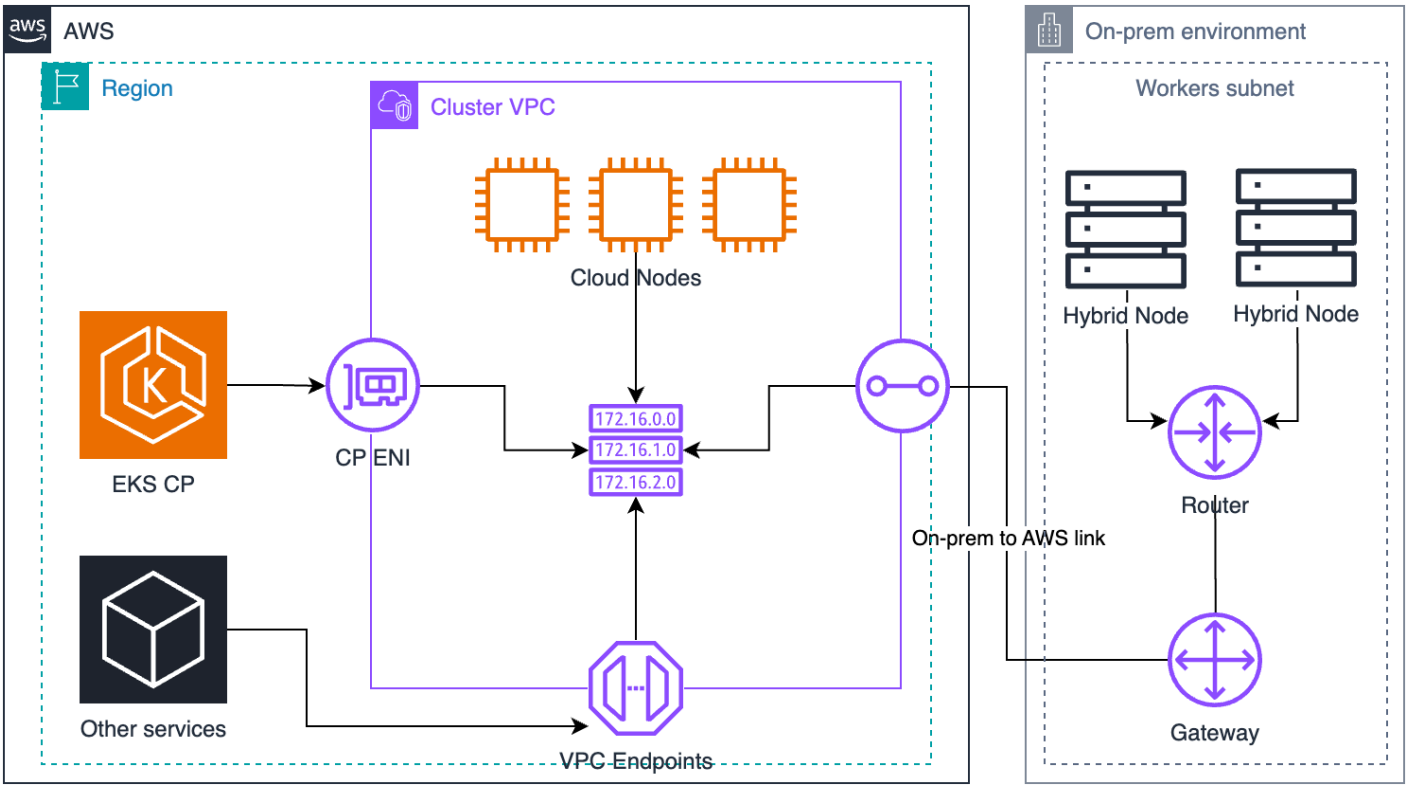
Argomenti

- Concetti di rete per nodi ibridi
- Concetti di Kubernetes per nodi ibridi
- Flussi di traffico di rete per nodi ibridi

Concetti di rete per nodi ibridi

Questa sezione descrive in dettaglio i concetti di rete principali e i vincoli da considerare durante la progettazione della topologia di rete per EKS Hybrid Nodes.

Concetti di rete per EKS Hybrid Nodes



VPC come hub di rete

Tutto il traffico che supera i limiti del cloud viene instradato attraverso il tuo VPC. Ciò include il traffico tra il piano di controllo EKS o i pod che arrivano AWS ai nodi ibridi o ai pod in esecuzione su di essi. Puoi considerare il VPC del cluster come all'hub di rete tra i nodi ibridi e il resto del cluster. Questa architettura offre pieno controllo del traffico e del relativo instradamento, ma ti impone anche la responsabilità di configurare correttamente instradamenti, gruppi di sicurezza e firewall per il VPC.

Dal piano di controllo (control-plane) EKS al VPC

Il piano di controllo EKS collega Elastic Network Interfaces (ENIs) al tuo VPC. Questi ENIs gestiscono il traffico da e verso il server API EKS. È possibile controllare il posizionamento del piano di controllo EKS ENIs durante la configurazione del cluster, poiché EKS si collega ENIs alle sottoreti passate durante la creazione del cluster.

EKS associa i gruppi di sicurezza a quelli ENIs che EKS collega alle sottoreti. Questi gruppi di sicurezza consentono il traffico da e verso il piano di controllo EKS attraverso il. ENIs Questo è importante per i nodi ibridi EKS perché è necessario consentire il traffico dai nodi ibridi e dai pod in esecuzione su di essi al piano ENIs di controllo EKS.

Reti di nodi remoti

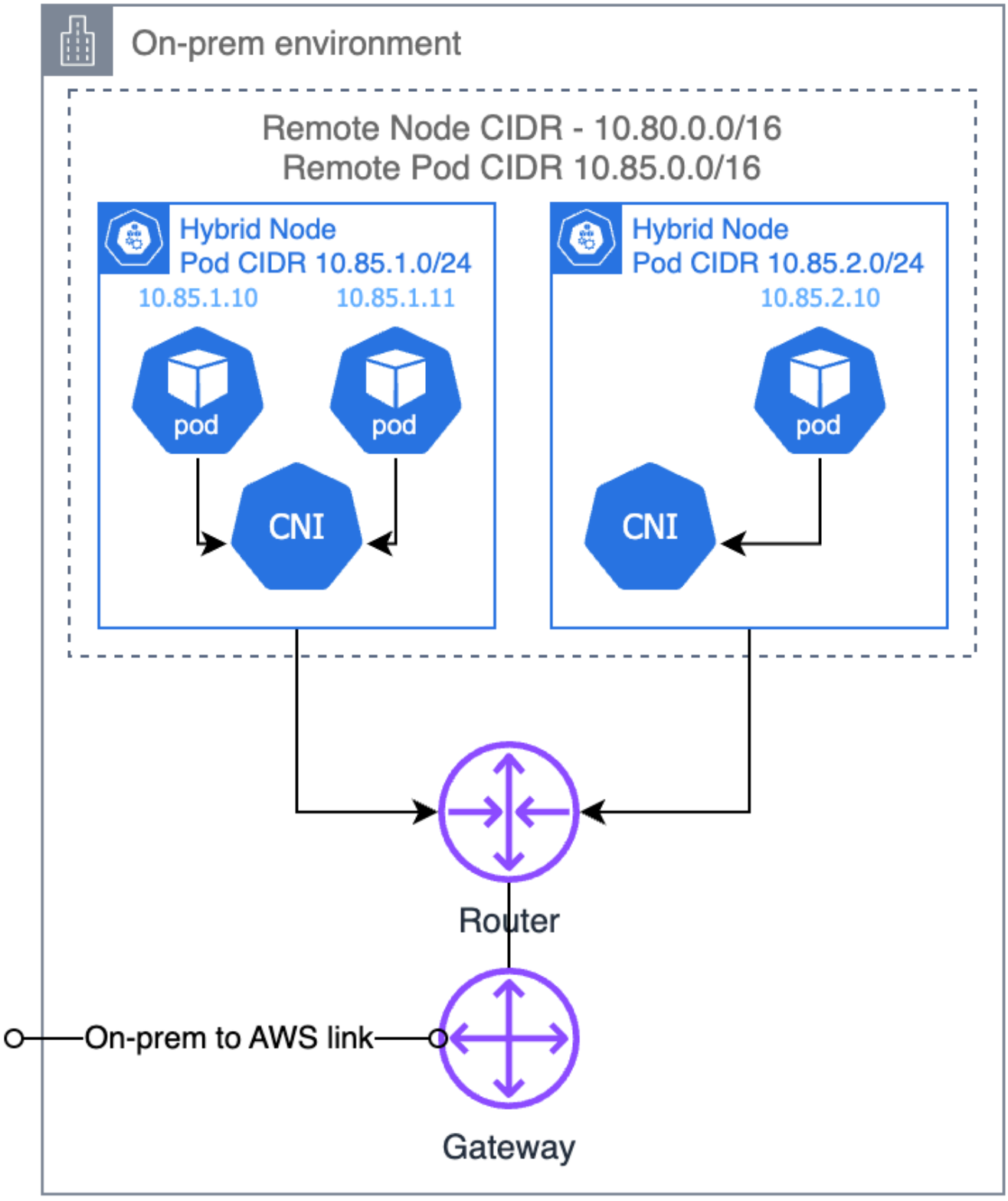
Le reti a nodi remoti, in particolare il nodo remoto CIDRs, sono gli intervalli IPs assegnati alle macchine utilizzate come nodi ibridi. Quando esegui il provisioning dei nodi ibridi, questi risiedono nel data center on-premises o nella posizione edge, che è un dominio di rete diverso dal piano di controllo (control-plane) EKS e dal VPC. Ogni nodo ibrido ha uno o più indirizzi IP da un CIDR del nodo remoto distinto dalle sottoreti del tuo VPC.

Il cluster EKS viene configurato con questi nodi remoti CIDRs in modo che EKS sappia come indirizzare tutto il traffico destinato ai nodi ibridi IPs attraverso il VPC del cluster, ad esempio le richieste all'API kubelet. Le connessioni all'`kubelet`API vengono utilizzate nei comandi`kubectl attach`,, `kubectl cp`, `kubectl exec`, `kubectl logs` e. `kubectl port-forward`

Reti di pod remoti

Le reti di pod remoti sono gli intervalli IPs assegnati ai pod in esecuzione sui nodi ibridi. In genere, bisogna configurare la Container Network Interface (CNI) con questi intervalli e la funzionalità di Gestione indirizzi IP (IPAM) della CNI si occupa di assegnare una parte di tali intervalli a ciascun nodo ibrido. Quando crei un pod, la CNI assegna un IP al pod dalla sezione allocata al nodo in cui il pod è stato pianificato.

Puoi configurare il cluster EKS con questi pod remoti CIDRs in modo che il piano di controllo EKS sappia instradare tutto il traffico destinato ai pod in esecuzione sui nodi ibridi attraverso il VPC del cluster, ad esempio la comunicazione con i webhook.



Dall'ambiente on-premises al VPC

La rete on-premises utilizzata per i nodi ibridi deve essere indirizzata al VPC usato per il cluster EKS. Sono disponibili diverse opzioni di connettività Network-to-Amazon VPC per connettere la rete locale a un VPC. Inoltre, puoi utilizzare la tua soluzione VPN.

È importante configurare correttamente il routing sul lato AWS cloud nel VPC e nella rete locale, in modo che entrambe le reti instradino il traffico giusto attraverso la connessione per le due reti.

Nel VPC, tutto il traffico diretto alle reti di nodi e pod remoti deve essere instradato attraverso la connessione alla rete on-premises (denominata "gateway"). Se alcune sottoreti hanno tabelle di routing diverse, devi configurare ogni tabella con gli instradamenti per i nodi ibridi e i pod in esecuzione su di essi. Questo vale per le sottoreti a cui è collegato il piano di controllo EKS e per ENIs le sottoreti che contengono EC2 nodi o pod che devono comunicare con i nodi ibridi.

Nella rete locale, è necessario configurare la rete per consentire il traffico da e verso il VPC del cluster EKS e AWS gli altri servizi richiesti per i nodi ibridi. Il traffico per il cluster EKS attraversa il gateway in entrambe le direzioni.

Vincoli di rete

Rete completamente instradata

Il vincolo principale è che il piano di controllo (control-plane) EKS e tutti i nodi, cloud o ibridi, devono formare una rete completamente instradata. Ciò significa che tutti i nodi devono essere in grado di raggiungersi a vicenda al terzo livello, tramite indirizzo IP.

Il piano di controllo (control-plane) EKS e i nodi cloud sono già raggiungibili a vicenda perché si trovano in una rete piatta (il VPC). I nodi ibridi, tuttavia, si trovano in un dominio di rete diverso. Ecco perché è necessario configurare un routing aggiuntivo nel VPC e nella rete on-premises per instradare il traffico tra i nodi ibridi e il resto del cluster. Se i nodi ibridi sono raggiungibili a vicenda e dal VPC, i nodi ibridi possono trovarsi in un'unica rete piatta o in più reti segmentate.

Pod remoto indirizzabile CIDRs

Affinché il piano di controllo (control-plane) EKS comunichi con i pod in esecuzione su nodi ibridi (ad esempio, webhook o Metrics Server) o affinché i pod in esecuzione su nodi cloud comunichino con quelli in esecuzione su nodi ibridi (comunicazione est-ovest del carico di lavoro), il CIDR del pod remoto deve essere instradabile dal VPC. Ciò significa che il VPC deve essere in grado di indirizzare il traffico verso il pod CIDRs attraverso il gateway verso la rete locale e che la rete locale deve essere in grado di indirizzare il traffico di un pod verso il nodo giusto.

È importante notare la distinzione tra i requisiti di routing dei pod nel VPC e on-premises. Il VPC deve solo sapere che tutto il traffico diretto a un pod remoto deve passare attraverso il gateway. Se disponi di un solo CIDR del pod remoto, è necessario un solo instradamento.

Questo requisito è valido per tutti gli hop della rete on-premises fino al router locale nella stessa sottorete dei nodi ibridi. Questo è l'unico router che deve conoscere la sezione CIDR del pod assegnata a ciascun nodo, assicurandosi che il traffico per un pod specifico venga recapitato al nodo in cui è stato pianificato.

Puoi scegliere di propagare queste route per il pod locale CIDRs dal router locale locale alle tabelle di routing VPC, ma non è necessario. Se il tuo pod locale CIDRs cambia frequentemente e le tabelle di routing VPC devono essere aggiornate per rispecchiare la modifica del CIDRs pod, ti consigliamo di propagare il pod locale CIDRs alle tabelle di routing VPC, ma questo non è comune.

Nota, il vincolo per rendere instradabile il pod locale è facoltativo. CIDRs Se non è necessario eseguire webhook sui nodi ibridi o disporre di pod sui nodi cloud, dialoga con i pod sui nodi ibridi, non è necessario configurare il routing per il pod sulla rete locale. CIDRs

Perché il pod locale CIDRs deve essere instradabile con nodi ibridi?

Quando si utilizza EKS con il VPC CNI per i nodi cloud, il VPC CNI esegue l'assegnazione direttamente IPs dal VPC ai pod. Ciò significa che non è necessario alcun routing speciale, poiché sia i cloud pod che il piano di controllo EKS possono raggiungere direttamente il Pod. IPs

Quando vengono eseguiti in locale (e insieme ad altri CNIs nel cloud), i pod vengono in genere eseguiti in una rete overlay isolata e il CNI si occupa della distribuzione del traffico tra i pod. Ciò avviene in genere tramite l'incapsulamento: il CNI converte il pod-to-pod traffico in node-to-node traffico, occupandosi dell'incapsulamento e del deincapsulamento su entrambe le estremità. In questo modo, non è necessaria alcuna configurazione aggiuntiva sui nodi e sui router.

La rete con nodi ibridi è unica perché presenta una combinazione di entrambe le topologie: il piano di controllo (control-plane) EKS e i nodi cloud (con la CNI del VPC) si aspettano una rete piatta che includa nodi e pod, mentre i pod in esecuzione su nodi ibridi si trovano in una rete sovrapposta utilizzando VXLAN per l'incapsulamento (impostazione predefinita in Cilium). I pod in esecuzione su nodi ibridi possono raggiungere il piano di controllo (control-plane) EKS e i pod in esecuzione su nodi cloud, presupponendo che la rete on-premises possa essere instradata verso il VPC. Tuttavia, senza il routing del pod CIDRs sulla rete locale, qualsiasi traffico che ritorna a un IP del pod locale alla fine verrà interrotto se la rete non sa come raggiungere la rete overlay e indirizzarlo ai nodi corretti.

Concetti di Kubernetes per nodi ibridi

Questa pagina descrive in dettaglio i concetti chiave di Kubernetes alla base dell'architettura di sistema dei nodi ibridi EKS.

Piano di controllo EKS nel VPC

Gli IP del piano di controllo EKS ENI sono memorizzati nell'oggetto `kubernetes` `Endpoints` nel namespace `default`. Quando EKS crea nuovi ENI o rimuove quelli più vecchi, EKS aggiorna questo oggetto in modo che l'elenco degli IP sia sempre aggiornato.

È possibile utilizzare questi endpoint tramite il Servizio `kubernetes`, anche nel namespace `default`. A questo servizio, di tipo `ClusterIP`, viene sempre assegnato il primo IP del servizio CIDR del cluster. Ad esempio, per il servizio CIDR `172.16.0.0/16`, l'IP del servizio sarà `172.16.0.1`.

In genere, questo è il modo in cui i pod (indipendentemente dal fatto che siano in esecuzione nel cloud o nei nodi ibridi) accedono al server API EKS Kubernetes. I pod utilizzano l'IP del servizio come IP di destinazione, che viene tradotto negli IP effettivi di uno degli ENI del piano di controllo EKS. L'eccezione principale è `kube-proxy`, perché imposta la traduzione.

Endpoint del server API EKS

L'IP del servizio `kubernetes` non è l'unico modo per accedere al server API EKS. EKS crea anche un nome DNS Route53 quando si crea il cluster. Questo è il campo `endpoint` del cluster EKS quando si richiama l'azione dell'API EKS `DescribeCluster`.

```
{
    "cluster": {
        "endpoint": "https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com",
        "name": "my-cluster",
        "status": "ACTIVE"
    }
}
```

In un cluster di accesso pubblico agli endpoint o in uno di accesso pubblico e privato agli endpoint, i nodi ibridi risolveranno questo nome DNS in un IP pubblico per impostazione predefinita, instradabile tramite Internet. In un cluster privato di accesso agli endpoint, il nome DNS si risolve negli IP privati del piano di controllo EKS ENI.

Ecco come `kubelet` e `kube-proxy` accedono al server dell'API Kubernetes. Se desideri che tutto il traffico del cluster Kubernetes fluisca attraverso il VPC, devi configurare il cluster in modalità di accesso privato o modificare il server DNS on-premises per risolvere l'endpoint del cluster EKS sugli IP privati del piano di controllo EKS ENI.

Endpoint **`kubelet`**

Il `kubelet` espone diversi endpoint REST, consentendo ad altre parti del sistema di interagire e raccogliere informazioni da ciascun nodo. Nella maggior parte dei cluster, la maggior parte del traffico diretto al server `kubelet` proviene dal piano di controllo, ma anche alcuni agenti di monitoraggio potrebbero interagire con esso.

Tramite questa interfaccia, `kubelet` gestisce varie richieste: recupero dei log (`kubectl logs`), esecuzione di comandi all'interno dei container (`kubectl exec`) e inoltro del traffico (`kubectl port-forward`). Ognuna di queste richieste interagisce con il runtime del container sottostante attraverso il `kubelet`, risultando perfetta per gli amministratori e gli sviluppatori del cluster.

Il consumatore più comune di questa API è il server API Kubernetes. Quando utilizzi uno dei comandi `kubectl` menzionati in precedenza, `kubectl` invia una richiesta API al server API, che quindi chiama l'API `kubelet` del nodo su cui è in esecuzione il pod. Questo è il motivo principale per cui l'IP del nodo deve essere raggiungibile dal piano di controllo EKS e perché, anche se i pod sono in funzione, non sarà possibile accedere ai loro log o `exec` se il percorso del nodo non è configurato correttamente.

IP del nodo

Quando il piano di controllo EKS comunica con un nodo, utilizza uno degli indirizzi riportati nello stato dell'oggetto `Node` (`status.addresses`).

Con i nodi cloud EKS, è normale che il kubelet riporti l'IP privato dell'istanza EC2 come un `InternalIP` durante la registrazione del nodo. Questo IP viene quindi convalidato dal Cloud Controller Manager (CCM) assicurandosi che appartenga all'istanza EC2. Inoltre, il CCM in genere aggiunge gli IP (come `ExternalIP`) pubblici e i nomi DNS (`InternalDNS`e `ExternalDNS`) dell'istanza allo stato del nodo.

Tuttavia, non esiste un CCM per i nodi ibridi. Quando si registra un nodo ibrido con EKS Hybrid Nodes CLI (`nodeadm`), configura il kubelet per riportare l'IP della macchina direttamente nello stato del nodo, senza 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
```

Se la macchina ha più IP, il kubelet ne selezionerà uno seguendo la propria logica. Puoi controllare l'IP selezionato con il flag `--node-ip`, che puoi passare in configurazione `nodeadm` in `spec.kubelet.flags`. Solo l'IP riportato nell'oggetto `Node` necessita di un percorso dal VPC. Le tue macchine possono avere altri IP che non sono raggiungibili dal cloud.

kube-proxy

`kube-proxy` è responsabile dell'implementazione dell'astrazione del servizio a livello di rete di ciascun nodo. Funziona come proxy di rete e bilanciatore del carico per il traffico destinato ai servizi Kubernetes. Monitorando continuamente il server API Kubernetes per rilevare eventuali modifiche relative ai servizi e agli endpoint, `kube-proxy` aggiorna dinamicamente le regole di rete dell'host sottostante per garantire che il traffico venga indirizzato correttamente.

In modalità `iptables`, `kube-proxy` programma diverse catene `netfilter` per gestire il traffico di servizio. Le regole formano la seguente gerarchia:

1. Catena KUBE-SERVICES: il punto di ingresso per tutto il traffico di servizio. Ha regole che corrispondono a ogni `ClusterIP` e porta del servizio.
2. Catene KUBE-SVC-XXX: le catene specifiche dei servizi hanno regole di bilanciamento del carico per ogni servizio.
3. Catene KUBE-SEP-XXX: le catene specifiche per endpoint hanno le regole effettive. `DNAT`

Esaminiamo cosa succede per un servizio `test-server` nel namespace `default`: * Service ClusterIP: `172.16.31.14` * Service port: `80` * Backing pods: `10.2.0.110`, `10.2.1.39`, e `10.2.2.254`

Quando controlliamo le regole `iptables` (usando `iptables-save 0 grep -A10 KUBE-SERVICES`):

1. Nella catena KUBE-SERVICES, troviamo una regola corrispondente al servizio:

```
-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
```

   - Questa regola corrisponde ai pacchetti destinati a 172.16.31.14:80
   - Il commento indica a cosa serve questa regola: `default/test-server cluster IP`
   - I pacchetti corrispondenti saltano alla catena `KUBE-SVC-XYZABC123456`
2. La catena KUBE-SVC-XYZABC123456 ha regole di bilanciamento del carico basate sulla probabilità:

```
-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
```

   - Prima regola: 33,3% di probabilità di saltare a `KUBE-SEP-POD1XYZABC`
   - Seconda regola: 50% di probabilità che il traffico rimanente (33,3% del totale) passi a `KUBE-SEP-POD2XYZABC`
   - Ultima regola: tutto il traffico rimanente (33,3% del totale) passa a `KUBE-SEP-POD3XYZABC`
3. Le singole catene KUBE-SEP-XXX eseguono il DNAT (Destination NAT):

```
-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
```

   - Queste regole DNAT riscrivono l'IP e la porta di destinazione per indirizzare il traffico verso pod specifici.
   - Ogni regola gestisce circa il 33,3% del traffico, garantendo un bilanciamento uniforme del carico tra `10.2.0.110`, `10.2.1.39` e `10.2.2.254`.

Questa struttura a catena a più livelli consente a `kube-proxy` di implementare in modo efficiente il bilanciamento e il reindirizzamento del carico di servizio mediante la manipolazione dei pacchetti a livello di kernel, senza richiedere un processo proxy nel percorso dei dati.

Impatto sulle operazioni di Kubernetes

Un'interruzione `kube-proxy` su un nodo impedisce a quel nodo di indirizzare correttamente il traffico del Servizio, causando timeout o connessioni non riuscite per i pod che si basano sui servizi del cluster. Ciò può essere particolarmente problematico quando un nodo viene registrato per la prima volta. Il CNI deve comunicare con il server dell'API Kubernetes per ottenere informazioni, come il pod CIDR del nodo, prima di poter configurare qualsiasi rete di pod. A tale scopo, utilizza l'IP del servizio `kubernetes`. Tuttavia, se `kube-proxy` non è riuscito ad avviarlo o a impostare le regole `iptables` corrette, le richieste inviate all'IP del servizio `kubernetes` non vengono tradotte negli IP effettivi degli ENI del piano di controllo EKS. Di conseguenza, il CNI entrerà in un ciclo di arresto e nessuno dei pod sarà in grado di funzionare correttamente.

Sappiamo che i pod utilizzano l'IP del servizio `kubernetes` per comunicare con il server dell'API Kubernetes, ma `kube-proxy` deve prima impostare le regole `iptables` per farlo funzionare.

Come `kube-proxy` comunica con il server API?

`kube-proxy` deve essere configurato per utilizzare gli IP/s effettivi del server dell'API Kubernetes o un nome DNS che li risolva. Nel caso di EKS, EKS configura il predefinito `kube-proxy` in modo che punti al nome DNS Route53 creato da EKS quando si crea il cluster. Puoi vedere questo valore nella ConfigMap `kube-proxy` nel namespace `kube-system`. Il contenuto di questa ConfigMap è un `kubeconfig` che viene iniettato nel pod `kube-proxy`, quindi cerca il campo `clusters0.cluster.server`. Questo valore corrisponderà al campo `endpoint` del cluster EKS (quando chiama l'API `DescribeCluster` di 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
```

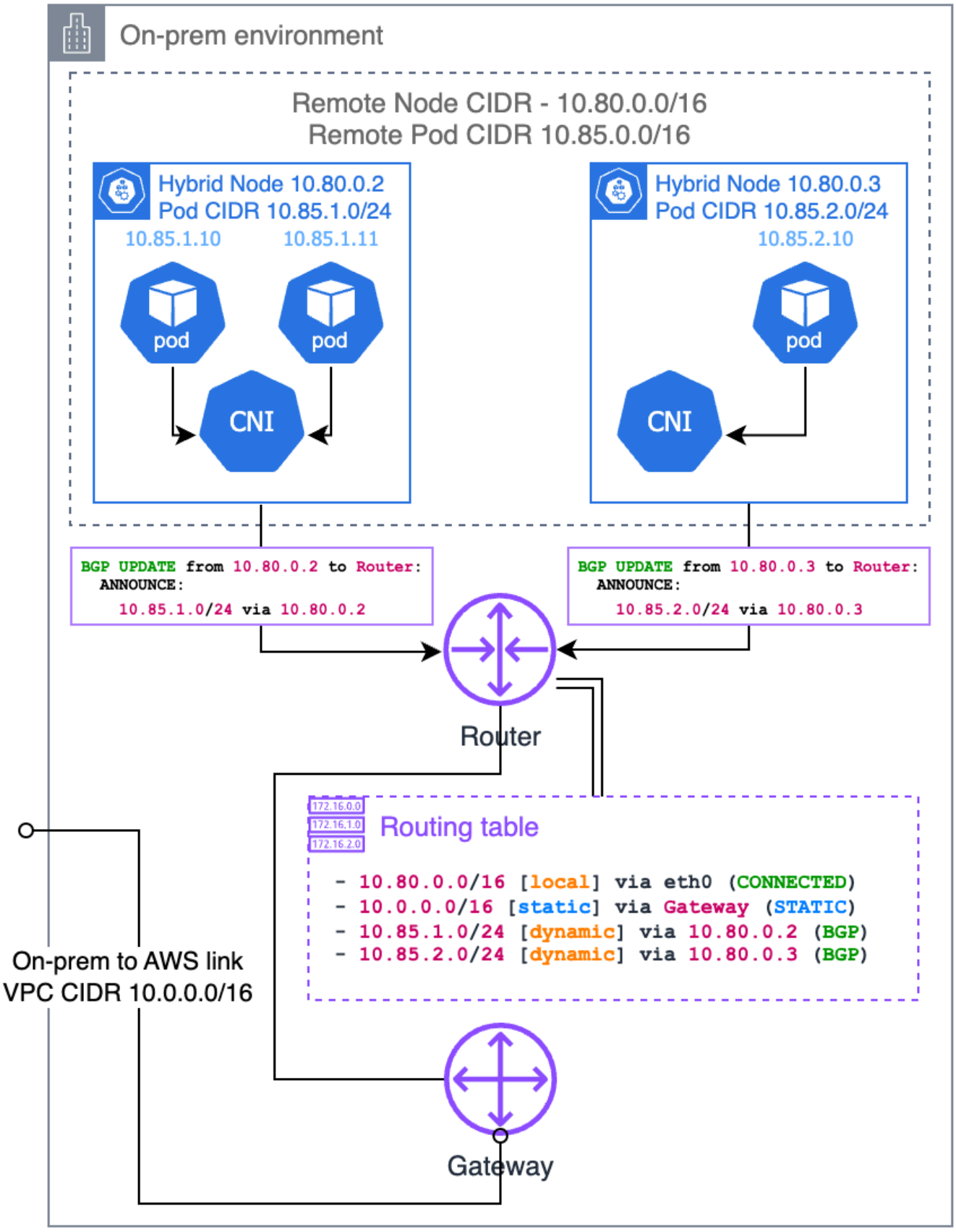
```
      name: default
    current-context: default
    users:
    - name: default
      user:
        tokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
kind: ConfigMap
metadata:
  name: kube-proxy
  namespace: kube-system
```

CIDR dei pod remoti instradabili

La pagina the section called "Concetti di rete" descrive in dettaglio i requisiti per eseguire webhook su nodi ibridi o per far sì che i pod in esecuzione su nodi cloud comunichino con quelli in esecuzione su nodi ibridi. Il requisito fondamentale è che il router on-premises debba sapere quale nodo è responsabile di un particolare pod IP. Esistono diversi modi per raggiungere questo obiettivo, tra cui Border Gateway Protocol (BGP), route statiche e proxy ARP (Address Resolution Protocol). Le diverse fasi vengono illustrate nelle sezioni seguenti.

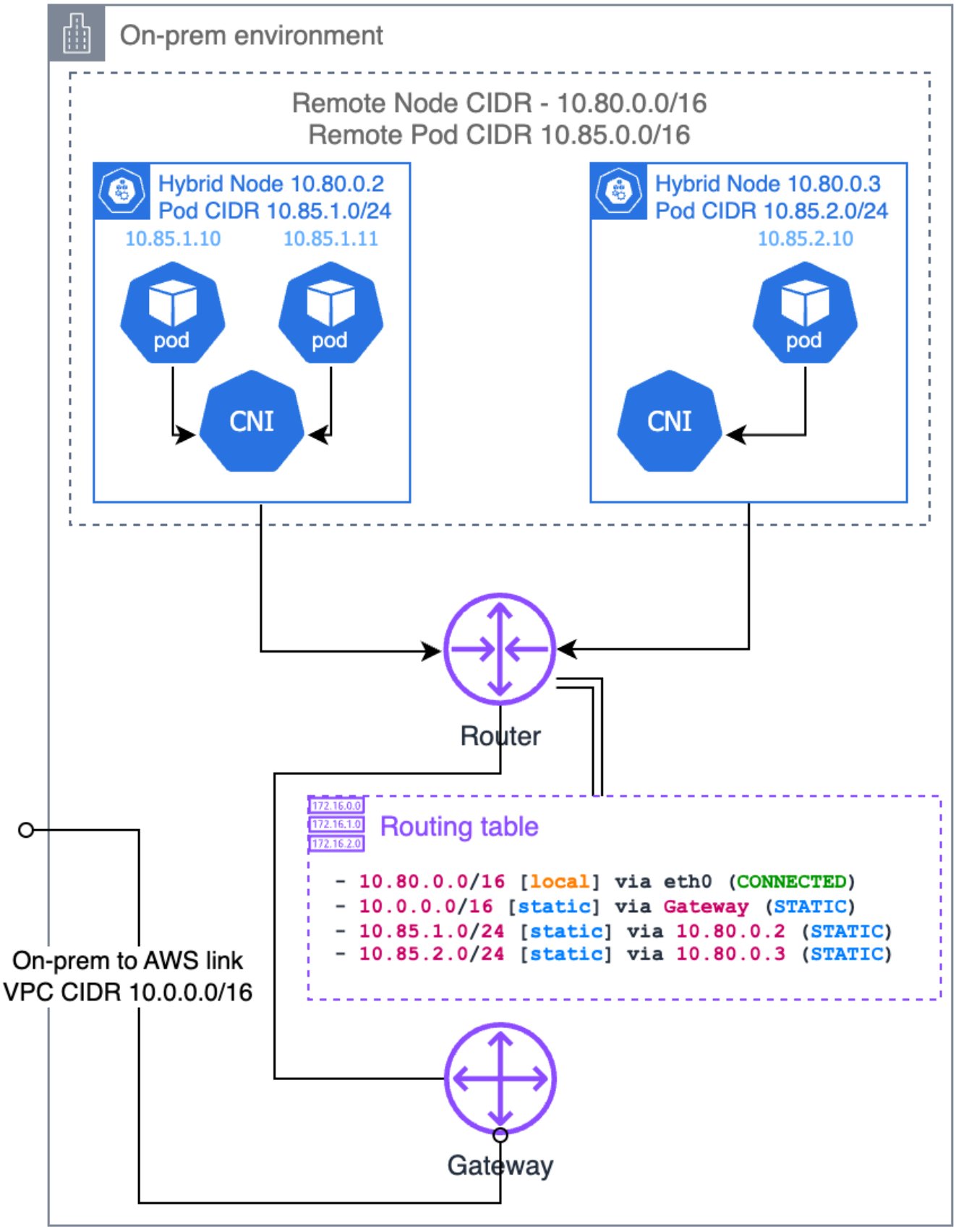
Border Gateway Protocol (BGP)

Se il tuo CNI lo supporta (come Cilium e Calico), puoi utilizzare la modalità BGP del tuo CNI per propagare le route ai tuoi pod CIDR per nodo dai nodi al router locale. Quando si utilizza la modalità BGP del CNI, quest'ultimo funge da router virtuale, quindi il router locale ritiene che il pod CIDR appartenga a una sottorete diversa e il nodo sia il gateway di accesso a quella sottorete.


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

Route statiche

In alternativa, puoi configurare percorsi statici nel router locale. Questo è il modo più semplice per indirizzare il pod CIDR on-premises al tuo VPC, ma è anche il più soggetto a errori e difficile da gestire. È necessario assicurarsi che i percorsi siano sempre aggiornati con i nodi esistenti e i pod CIDR a loro assegnati. Se il numero di nodi è ridotto e l'infrastruttura è statica, questa è un'opzione valida e elimina la necessità del supporto BGP nel router. Se scegli questa opzione, ti consigliamo di configurare il tuo CNI con la slice CIDR del pod che desideri assegnare a ciascun nodo invece di lasciare che sia l'IPAM a decidere.


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

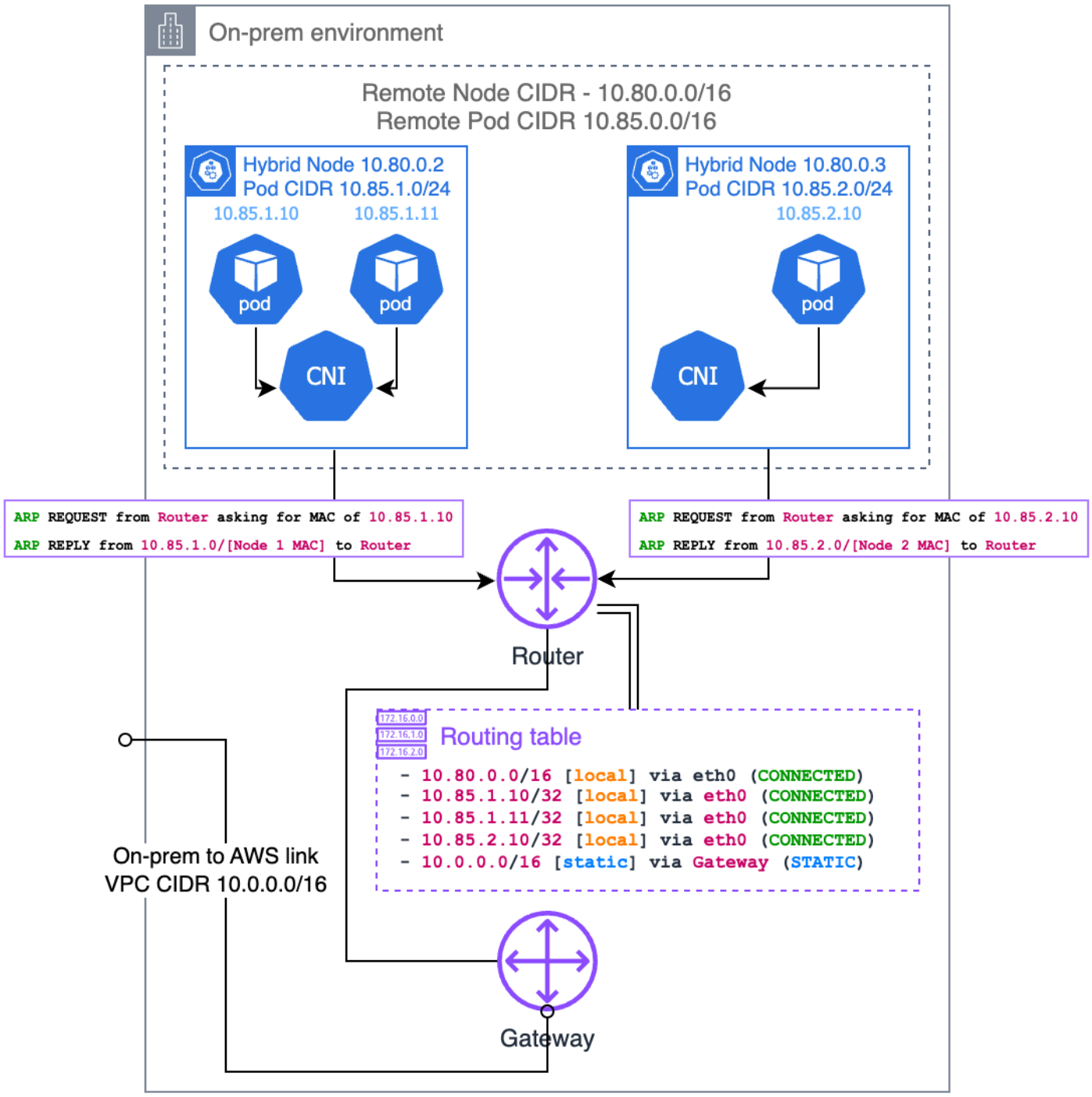
Proxy ARP (Address Resolution Protocol)

Il proxy ARP è un altro approccio per rendere instradabili gli IP dei on-premises, particolarmente utile quando i nodi ibridi si trovano sulla stessa rete di livello 2 del router locale. Con il proxy ARP abilitato, un nodo risponde alle richieste ARP per gli IP dei pod che ospita, anche se tali IP appartengono a una sottorete diversa.

Quando un dispositivo sulla rete locale tenta di raggiungere un pod IP, invia innanzitutto una richiesta ARP che chiede "Chi ha questo IP?". Il nodo ibrido che ospita il pod risponderà con il proprio indirizzo MAC, dicendo "Posso gestire il traffico per quell'IP". Questo crea un percorso diretto tra i dispositivi della rete locale e i pod senza richiedere la configurazione del router.

Perché ciò funzioni, il CNI deve supportare la funzionalità proxy ARP. Cilium ha un supporto integrato per il proxy ARP che puoi abilitare tramite la configurazione. La considerazione fondamentale è che il pod CIDR non deve sovrapporsi a nessun'altra rete dell'ambiente, poiché ciò potrebbe causare conflitti di routing.

Questo approccio presenta diversi vantaggi: * Non è necessario configurare il router con BGP o mantenere percorsi statici * Funziona bene in ambienti in cui non si ha il controllo sulla configurazione del router



Incapsulamento da pod a pod

Negli ambienti on-premises, i CNI utilizzano in genere protocolli di incapsulamento per creare reti overlay in grado di funzionare sulla rete fisica senza la necessità di riconfigurarla. Questa sezione spiega come funziona questo incapsulamento. Tieni presente che alcuni dettagli potrebbero variare a seconda del CNI che stai utilizzando.

L'incapsulamento avvolge i pacchetti della rete di pod originali all'interno di un altro pacchetto di rete che può essere instradato attraverso la rete fisica sottostante. Ciò consente ai pod di comunicare tra i nodi che eseguono lo stesso CNI senza che la rete fisica sappia come instradare quei pod CIDR.

Il protocollo di incapsulamento più comune utilizzato con Kubernetes è Virtual Extensible LAN (VXLAN), sebbene siano disponibili anche altri (come `Geneve`) a seconda del CNI utilizzato.

Incapsulamento VXLAN

VXLAN incapsula i frame Ethernet di livello 2 all'interno di pacchetti UDP. Quando un pod invia traffico a un altro pod su un nodo diverso, il CNI esegue le seguenti operazioni:

1. Il CNI intercetta i pacchetti dal Pod A.
2. Il CNI avvolge il pacchetto originale in un'intestazione VXLAN.
3. Questo pacchetto avvolto viene quindi inviato attraverso lo stack di rete normale del nodo al nodo di destinazione.
4. Il CNI sul nodo di destinazione scarta il pacchetto e lo consegna al Pod B.

Ecco cosa succede alla struttura dei pacchetti durante l'incapsulamento VXLAN:

Pacchetto originale da pod a 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    |                 |
+-----------------+---------------+-------------+-----------------+
```

Dopo l'incapsulamento 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)
 |
+-----------------+-------------+--------------+------------
+---------------------------+
```

Il VXLAN Network Identifier (VNI) distingue tra diverse reti di sovrapposizione.

Scenari di comunicazione dei pod

Pod sullo stesso nodo ibrido

Quando i pod sullo stesso nodo ibrido comunicano, in genere non è necessario alcun incapsulamento. Il CNI imposta percorsi locali che indirizzano il traffico tra i pod attraverso le interfacce virtuali interne del nodo:

```
Pod A -> veth0 -> node's bridge/routing table -> veth1 -> Pod B
```

Il pacchetto non lascia mai il nodo e non richiede l'incapsulamento.

Pod su diversi nodi ibridi

La comunicazione tra pod su diversi nodi ibridi richiede l'incapsulamento:

```
Pod A -> CNI -> [VXLAN encapsulation] -> Node A network -> router or gateway -> Node B
 network -> [VXLAN decapsulation] -> CNI -> Pod B
```

Ciò consente al traffico dei pod di attraversare l'infrastruttura di rete fisica senza che la rete fisica comprenda il routing IP dei pod.

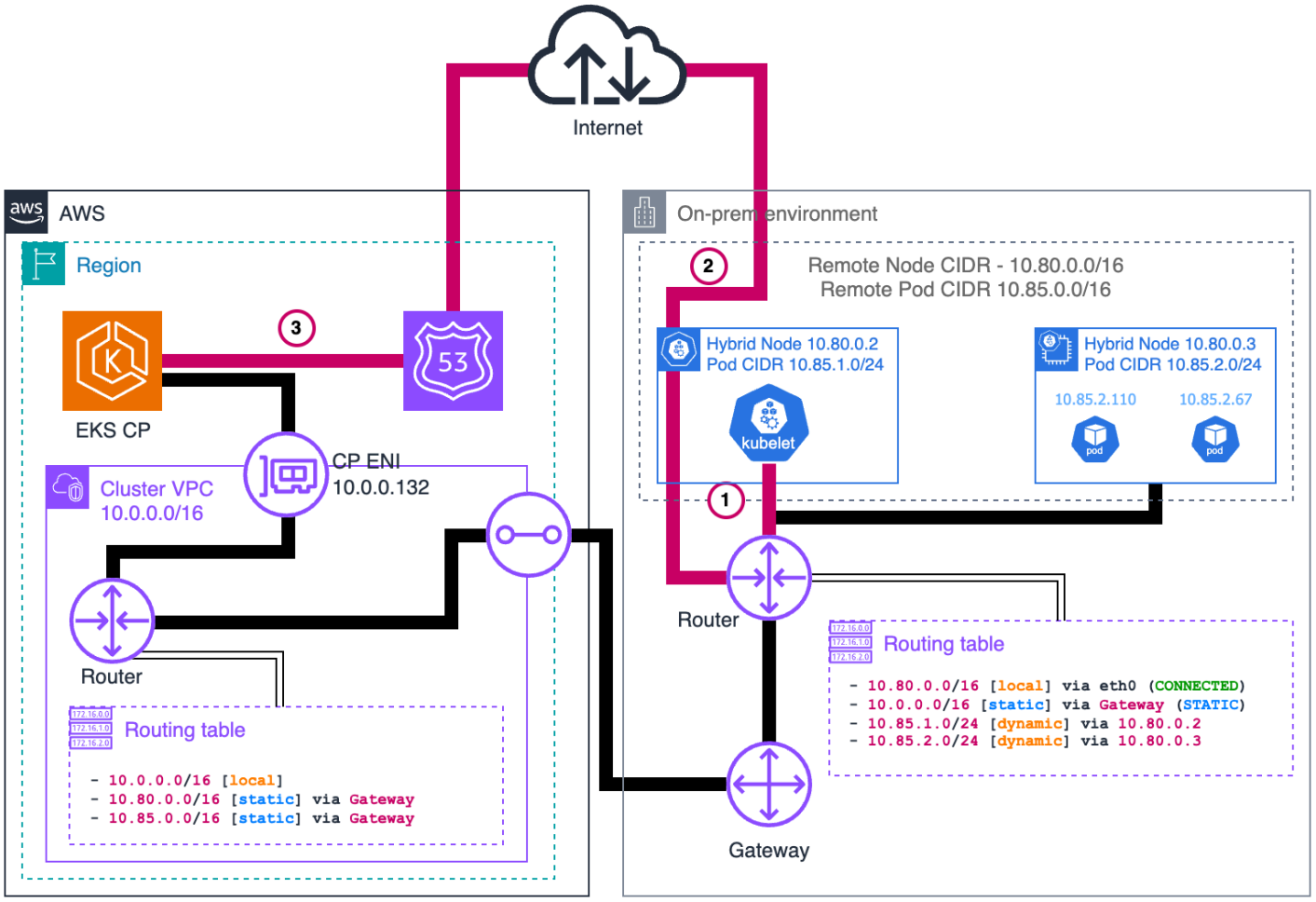
Flussi di traffico di rete per nodi ibridi

Questa pagina descrive in dettaglio i flussi di traffico di rete per i nodi ibridi EKS con diagrammi che mostrano i percorsi di end-to-end rete per i diversi tipi di traffico.

Sono coperti i seguenti flussi di traffico:

- the section called "Dal `kubelet` del nodo ibrido al piano di controllo (control-plane) EKS"
- the section called "Dal piano di controllo (control-plane) EKS al nodo ibrido (server `kubelet`)"
- the section called "I pod vengono eseguiti su nodi ibridi verso il piano di controllo (control-plane) EKS"
- the section called "Dal piano di controllo (control-plane) EKS ai pod in esecuzione su un nodo ibrido (webhook)"
- the section called "Pod-to-Pod in esecuzione su nodi ibridi"
- the section called "Da pod su nodi cloud a pod su nodi ibridi (traffico est-ovest)"

Dal **`kubelet`** del nodo ibrido al piano di controllo (control-plane) EKS



Richiesta

1**`kubelet`**. Avvia la richiesta

Quando il `kubelet` su un nodo ibrido deve comunicare con il piano di controllo (control-plane) EKS (ad esempio, per segnalare lo stato del nodo oppure ottenere le specifiche del pod), utilizza il file `kubeconfig` fornito durante la registrazione del nodo. Questo `kubeconfig` contiene l'URL dell'endpoint del server API (il nome DNS di Route53) anziché gli indirizzi IP diretti.

Il `kubelet` esegue una ricerca DNS per l'endpoint (ad esempio, `https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-west-2.eks.amazonaws.com`). In un cluster ad accesso pubblico, questo si risolve in un indirizzo IP pubblico (ad esempio, `54.239.118.52`) che appartiene al servizio EKS in esecuzione su AWS. Quindi, il `kubelet` crea una richiesta HTTPS sicura per questo endpoint. Il pacchetto iniziale appare come riportato di seguito:

```
+-------------------+--------------------+-----------------+
| IP Header         | TCP Header         | Payload         |
| Src: 10.80.0.2    | Src: 52390 (random) |                 |
| Dst: 54.239.118.52 | Dst: 443           |                 |
+-------------------+--------------------+-----------------+
```

2. Routing del router locale

Poiché l'IP di destinazione è un indirizzo IP pubblico e non fa parte della rete locale, il `kubelet` invia questo pacchetto al gateway predefinito (il router locale on-premises). Il router esamina l'IP di destinazione e determina che si tratta di un indirizzo IP pubblico.

Per il traffico pubblico, il router in genere inoltra il pacchetto a un gateway Internet o a un router di confine che gestisce il traffico in uscita verso internet. Questo valore è omesso nel diagramma e dipenderà da come è configurata la rete on-premises. Il pacchetto attraversa l'infrastruttura di rete on-premises e alla fine raggiunge la rete del provider di servizi internet.

3. Distribuzione al piano di controllo (control-plane) EKS

Il pacchetto viaggia attraverso la rete Internet pubblica e le reti di transito fino a raggiungere AWS la rete. AWS La rete indirizza il pacchetto all'endpoint del servizio EKS nella regione appropriata. Quando il pacchetto raggiunge il servizio EKS, viene inoltrato all'effettivo piano di controllo (control-plane) EKS del cluster.

Questo routing attraverso la rete internet pubblica è diverso dal percorso instradato dal VPC privato che vedremo in altri flussi di traffico. La differenza fondamentale è che quando si utilizza la modalità di accesso pubblico, il traffico dal `kubelet` on-premises (ma non dai pod) al piano di controllo (control-plane) EKS non passa attraverso il VPC, ma utilizza invece l'infrastruttura internet globale.

Risposta

Dopo che il piano di controllo (control-plane) EKS ha elaborato la richiesta del `kubelet`, invia una risposta:

3. Il piano di controllo (control-plane) EKS invia una risposta

Il piano di controllo (control-plane) EKS crea un pacchetto di risposta. Questo pacchetto ha l'IP pubblico come origine e l'IP del nodo ibrido come destinazione:

```
+-------------------+--------------------+-----------------+
| IP Header         | TCP Header         | Payload         |
```

```
| Src: 54.239.118.52 | Src: 443           |                 |
| Dst: 10.80.0.2     | Dst: 52390         |                 |
+--------------------+--------------------+-----------------+
```

2. Routing internet

Il pacchetto di risposta attraversa internet, seguendo il percorso di routing determinato dai provider di servizi internet, fino a raggiungere il router edge della rete on-premises.

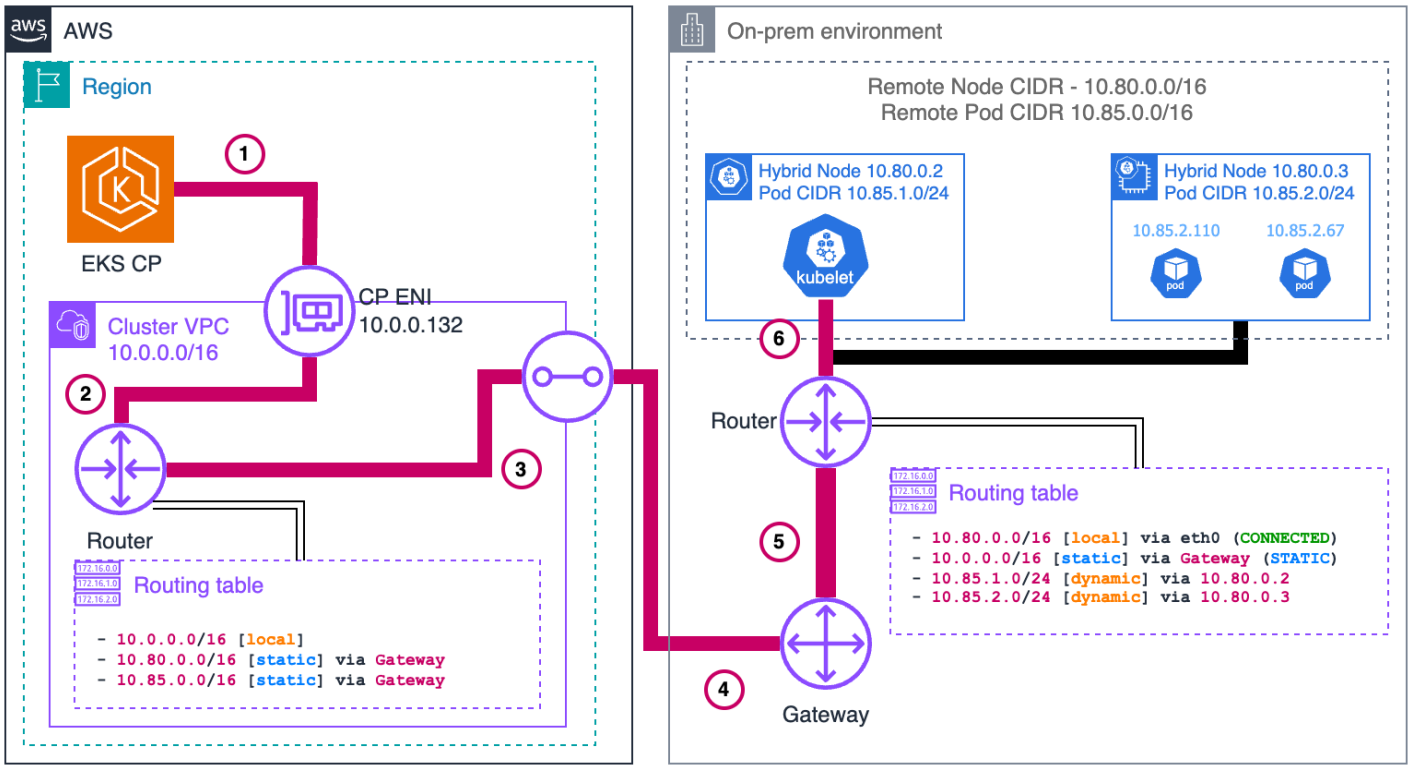
1. Distribuzione locale

Il router on-premises riceve il pacchetto e riconosce l'IP di destinazione (`10.80.0.2`) come appartenente alla rete locale. Inoltra il pacchetto attraverso l'infrastruttura di rete locale fino a raggiungere il nodo ibrido di destinazione, dove il `kubelet` riceve ed elabora la risposta.

Dal **`kube-proxy`** del nodo ibrido al piano di controllo (control-plane) EKS

Se abiliti l'accesso pubblico agli endpoint per il cluster, il traffico di ritorno utilizza la rete internet pubblica. Questo traffico proviene dal `kube-proxy` sul nodo ibrido verso il piano di controllo (control-plane) EKS e segue lo stesso percorso del traffico dal `kubelet` al piano di controllo (control-plane) EKS.

Dal piano di controllo (control-plane) EKS al nodo ibrido (server **`kubelet`**)



Richiesta

1. Il server API EKS Kubernetes avvia la richiesta

Il server API EKS Kubernetes recupera l'indirizzo IP del nodo (`10.80.0.2`) dallo stato dell'oggetto del nodo. Quindi indirizza questa richiesta tramite la sua interfaccia di rete elastica (ENI) nel VPC, poiché l'IP di destinazione appartiene al CIDR del nodo remoto configurato (`10.80.0.0/16`). Il pacchetto iniziale appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 67493 (random) |                 |
| Dst: 10.80.0.2  | Dst: 10250          |                 |
+-----------------+---------------------+-----------------+
```

2. Elaborazione di rete del VPC

Il pacchetto lascia l'ENI ed entra nel livello di rete del VPC, dove viene diretto al gateway della sottorete per un ulteriore instradamento.

3. Ricerca della tabella di routing del VPC

La tabella di routing del VPC per la sottorete contenente l'ENI del piano di controllo (control-plane) EKS ha un instradamento specifico (il secondo nel diagramma) per il CIDR del nodo remoto. In base a questa regola di routing, il pacchetto viene indirizzato al VPC-to-onprem gateway.

4. Transito oltre il limite

Il gateway trasferisce il pacchetto oltre i limiti del cloud tramite la connessione stabilita (come Direct Connect o VPN) alla rete on-premises.

5. Ricezione nella rete on-premises

Il pacchetto arriva al router on-premises locale che gestisce il traffico per la sottorete in cui si trovano i nodi ibridi.

6. Distribuzione finale

Il router locale identifica che l'indirizzo IP (`10.80.0.2`) di destinazione appartiene alla sua rete connessa direttamente e inoltra il pacchetto al nodo ibrido di destinazione, dove il `kubelet` riceve ed elabora la richiesta.

Risposta

Dopo che il `kubelet` del nodo ibrido ha elaborato la richiesta, restituisce una risposta seguendo lo stesso percorso in senso inverso:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 10250          |                 |
| Dst: 10.0.0.132 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

6. **`kubelet`** Invia la risposta

Il `kubelet` sul nodo ibrido (`10.80.0.2`) crea un pacchetto di risposta con l'IP di origine originale come destinazione. La destinazione non appartiene alla rete locale, quindi viene inviata al gateway predefinito dell'host, che è il router locale.

5. Routing del router locale

Il router determina l'appartenenza a `10.0.0.0/16` dell'IP di destinazione (`10.0.0.132`), che dispone di un instradamento che punta al gateway a cui si connette AWS.

4. Ritorno oltre il limite

Il pacchetto torna indietro attraverso la stessa connessione on-premises al VPC (come Direct Connect o VPN), superando il limite del cloud nella direzione opposta.

3. Routing del VPC

Quando il pacchetto arriva nel VPC, le tabelle di routing identificano che l'IP di destinazione appartiene a un CIDR del VPC. L'instradamento dei pacchetti all'interno del VPC.

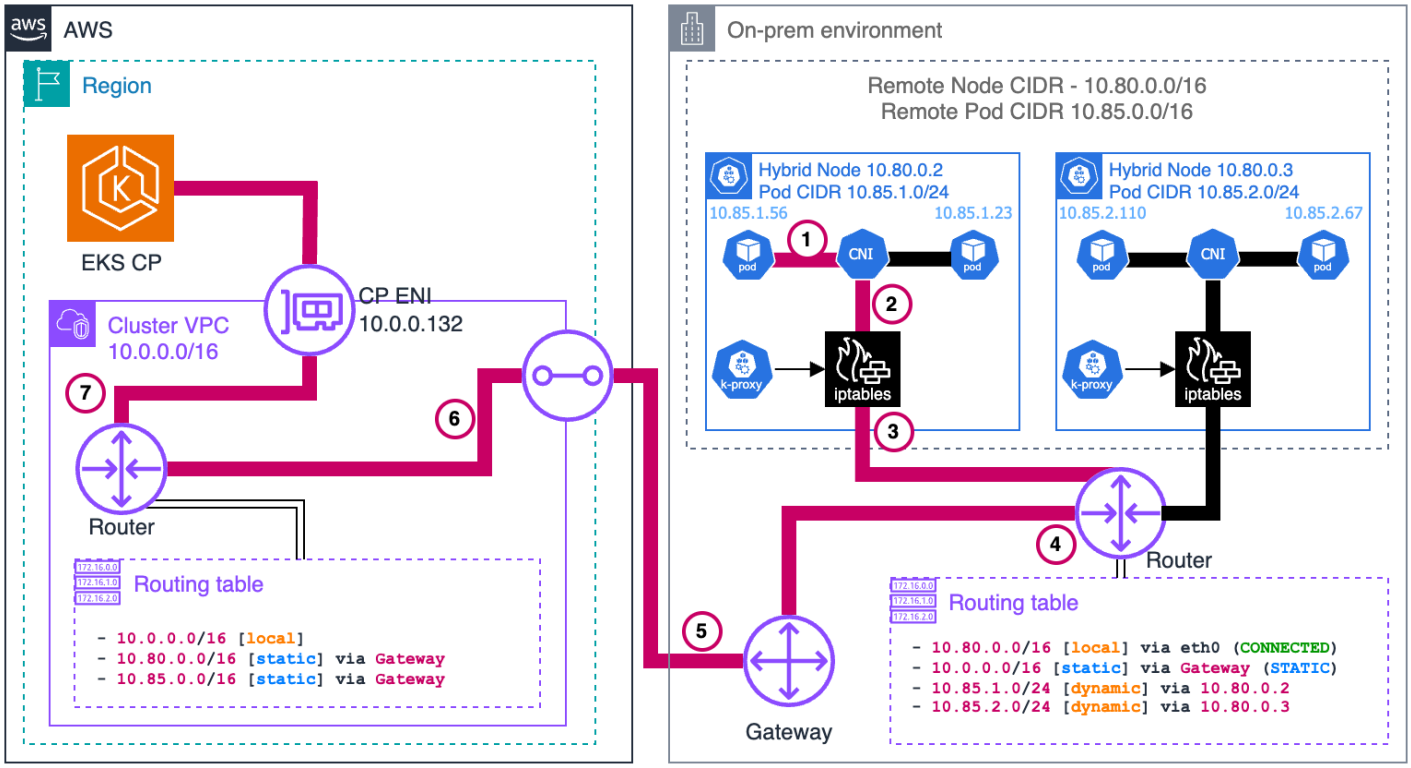
2. Distribuzione di rete del VPC

Il livello di rete del VPC inoltra il pacchetto alla sottorete con l'ENI del piano di controllo (control-plane) EKS (`10.0.0.132`).

1. Ricezione dell'ENI

Il pacchetto raggiunge l'ENI del piano di controllo (control-plane) EKS collegata al server API Kubernetes, completando il viaggio di andata e ritorno.

I pod vengono eseguiti su nodi ibridi verso il piano di controllo (control-plane) EKS



Senza NAT della CNI

Richiesta

In genere, i pod comunicano al server API Kubernetes tramite il servizio `kubernetes`. L'IP del servizio è il primo IP del servizio CIDR del cluster. Questa convenzione consente ai pod che devono essere eseguiti prima che CoreDNS sia disponibile per raggiungere il server API, ad esempio la Container Network Interface (CNI). Le richieste lasciano il pod con l'IP del servizio come destinazione. Ad esempio, se il servizio CIDR è `172.16.0.0/16`, l'IP del servizio è `172.16.0.1`.

1. Il pod avvia la richiesta

Il pod invia una richiesta all'IP del servizio `kubernetes` (`172.16.0.1`) sulla porta del server API (443) da una porta di origine casuale. Il pacchetto appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. Elaborazione della CNI

La CNI rileva che l'IP di destinazione non appartiene a nessun CIDR del pod che gestisce. Poiché il NAT in uscita è disabilitato, la CNI passa il pacchetto allo stack di rete host senza modificarlo.

3. Elaborazione della rete del nodo

Il pacchetto entra nello stack di rete del nodo dove gli hook `netfilter` attivano le regole `iptables` impostate da kube-proxy. Diverse regole si applicano nell'ordine seguente:

1. Il pacchetto colpisce per primo la catena `KUBE-SERVICES`, che contiene regole che corrispondono al ClusterIP e alla porta di ogni servizio.
2. La regola di corrispondenza passa alla catena `KUBE-SVC-XXX` del servizio `kubernetes` (pacchetti destinati a `172.16.0.1:443`), che contiene le regole di bilanciamento del carico.
3. La regola di bilanciamento del carico seleziona casualmente una delle `KUBE-SEP-XXX` catene per il piano di controllo ENI IPs (o). `10.0.0.132` `10.0.1.23`
4. La catena `KUBE-SEP-XXX` selezionata ha la regola effettiva che modifica l'IP di destinazione dall'IP del servizio a quello selezionato. Si chiama Destination Network Address Translation (DNAT).

Dopo aver applicato queste regole, supponendo che l'IP dell'ENI del piano di controllo (control-plane) EKS selezionato sia `10.0.0.132`, il pacchetto appare come indicato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Il nodo inoltra il pacchetto al gateway predefinito perché l'IP di destinazione non si trova nella rete locale.

4. Routing del router locale

Il router locale determina che l'IP di destinazione (`10.0.0.132`) appartiene al CIDR del VPC (`10.0.0.0/16`) e lo inoltra al gateway che si connette ad AWS.

5. Transito oltre il limite

Il pacchetto viaggia attraverso la connessione stabilita (come Direct Connect o VPN) oltre il limite del cloud fino al VPC.

6. Distribuzione di rete del VPC

Il livello di rete del VPC indirizza il pacchetto alla sottorete corretta in cui si trova l'ENI del piano di controllo (control-plane) EKS (`10.0.0.132`).

7. Ricezione dell'ENI

Il pacchetto raggiunge l'ENI del piano di controllo (control-plane) EKS collegato al server API Kubernetes.

Risposta

Dopo che il piano di controllo (control-plane) EKS ha elaborato la richiesta, invia una risposta al pod:

7. Il server API invia una risposta

Il server API EKS Kubernetes crea un pacchetto di risposta con l'IP di origine originale come destinazione. Il pacchetto appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Poiché l'IP di destinazione appartiene al CIDR del pod remoto configurato (`10.85.0.0/16`), lo invia tramite la sua ENI nel VPC con il router della sottorete come hop successivo.

6. Routing del VPC

La tabella di routing VPC contiene una voce per il pod remoto CIDR (`10.85.0.0/16`), che indirizza questo traffico verso il gateway. VPC-to-onprem

5. Transito oltre il limite

Il gateway trasferisce il pacchetto oltre i limiti del cloud tramite la connessione stabilita (come Direct Connect o VPN) alla rete on-premises.

4. Ricezione nella rete on-premises

Il pacchetto arriva al router on-premises locale.

3. Distribuzione al nodo

La tabella del router contiene una voce per `10.85.1.0/24` con `10.80.0.2` come hop successivo, che consegna il pacchetto al nostro nodo.

2. Elaborazione della rete del nodo

Poiché il pacchetto viene elaborato dallo stack di rete del nodo, `conntrack` (una parte di `netfilter`) abbina il pacchetto alla connessione inizialmente stabilita dal pod. Poiché il DNAT è stato originariamente applicato, `conntrack` inverte il DNAT riscrivendo l'IP di origine dall'IP dell'ENI del piano di controllo (control-plane) EKS all'IP del servizio `kubernetes`:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1. Elaborazione della CNI

La CNI identifica che l'IP di destinazione appartiene a un pod nella sua rete e consegna il pacchetto al namespace della rete del pod corretto.

Questo flusso mostra perché Remote Pod CIDRs deve essere correttamente instradabile dal VPC fino al nodo specifico che ospita ciascun pod: l'intero percorso di ritorno dipende dal corretto instradamento del pod su reti cloud e IPs locali.

Con NAT della CNI

Questo flusso è molto simile a quello senza NAT della CNI, ma con una differenza fondamentale: la CNI applica il NAT di origine (SNAT) al pacchetto prima di inviarlo allo stack di rete del nodo. Ciò modifica l'IP di origine del pacchetto nell'IP del nodo, permettendo al pacchetto di essere reindirizzato al nodo senza richiedere una configurazione di routing aggiuntiva.

Richiesta

1. Il pod avvia la richiesta

Il pod invia una richiesta all'IP del servizio `kubernetes` (`172.16.0.1`) sulla porta del server API EKS Kubernetes (443) da una porta di origine casuale. Il pacchetto appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. Elaborazione della CNI

La CNI rileva che l'IP di destinazione non appartiene a nessun CIDR del pod che gestisce. Poiché il NAT in uscita è abilitato, la CNI applica lo SNAT al pacchetto, cambiando l'IP di origine con l'IP del nodo prima di passarlo allo stack di rete del nodo:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Nota: CNI e `iptables` sono mostrati nell'esempio come blocchi separati per motivi di chiarezza, ma in pratica è possibile che alcuni CNIs utilizzino `iptables` il NAT.

3. Elaborazione della rete del nodo

Qui le `iptables` regole stabilite da si `kube-proxy` comportano come nell'esempio precedente, bilanciando il carico del pacchetto su uno dei piani di controllo EKS. ENIs Il pacchetto ora appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Il nodo inoltra il pacchetto al gateway predefinito perché l'IP di destinazione non si trova nella rete locale.

4. Routing del router locale

Il router locale determina che l'IP di destinazione (`10.0.0.132`) appartiene al CIDR del VPC (`10.0.0.0/16`) e lo inoltra al gateway che si connette ad AWS.

5. Transito oltre il limite

Il pacchetto viaggia attraverso la connessione stabilita (come Direct Connect o VPN) oltre il limite del cloud fino al VPC.

6. Distribuzione di rete del VPC

Il livello di rete del VPC indirizza il pacchetto alla sottorete corretta in cui si trova l'ENI del piano di controllo (control-plane) EKS (`10.0.0.132`).

7. Ricezione dell'ENI

Il pacchetto raggiunge l'ENI del piano di controllo (control-plane) EKS collegato al server API Kubernetes.

Risposta

Dopo che il piano di controllo (control-plane) EKS ha elaborato la richiesta, invia una risposta al pod:

7. Il server API invia una risposta

Il server API EKS Kubernetes crea un pacchetto di risposta con l'IP di origine originale come destinazione. Il pacchetto appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Poiché l'IP di destinazione appartiene al CIDR del nodo remoto configurato (`10.80.0.0/16`), lo invia tramite la sua ENI nel VPC con il router della sottorete come hop successivo.

6. Routing del VPC

La tabella di routing VPC contiene una voce per il nodo remoto CIDR (`10.80.0.0/16`), che indirizza questo traffico verso il gateway. VPC-to-onprem

5. Transito oltre il limite

Il gateway trasferisce il pacchetto oltre i limiti del cloud tramite la connessione stabilita (come Direct Connect o VPN) alla rete on-premises.

4. Ricezione nella rete on-premises

Il pacchetto arriva al router on-premises locale.

3. Distribuzione al nodo

Il router locale identifica che l'indirizzo IP (`10.80.0.2`) di destinazione appartiene alla sua rete connessa direttamente e inoltra il pacchetto al nodo ibrido di destinazione.

2. Elaborazione della rete del nodo

Poiché il pacchetto viene elaborato dallo stack di rete del nodo, `conntrack` (una parte di `netfilter`) abbina il pacchetto alla connessione stabilita inizialmente dal pod e, poiché il DNAT è stato originariamente applicato, inverte la situazione riscrivendo l'IP di origine dall'IP dell'ENI del piano di controllo (control-plane) EKS a quello del servizio `kubernetes`:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1. Elaborazione della CNI

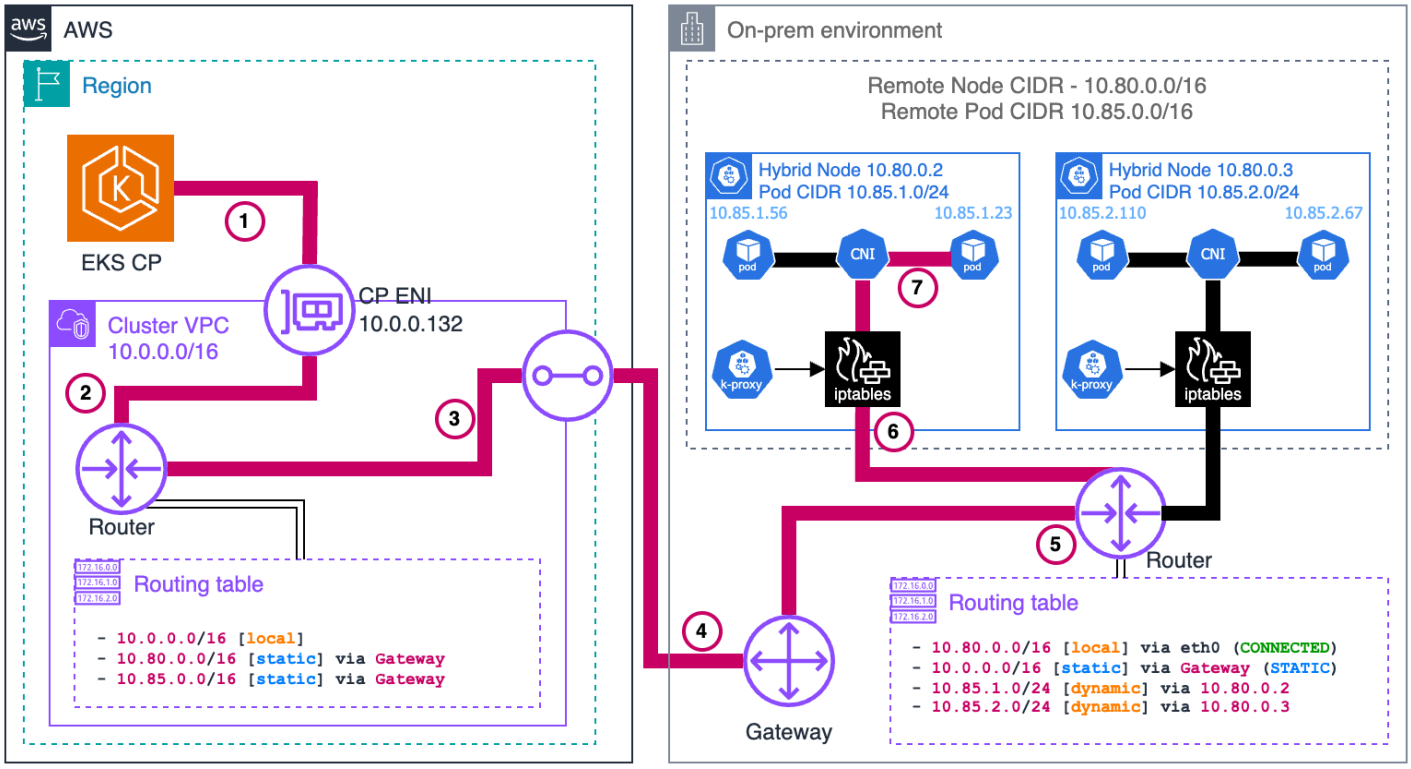
Il CNI identifica che questo pacchetto appartiene a una connessione a cui ha precedentemente applicato lo SNAT. Inverte lo SNAT, riportando l'IP di destinazione all'IP del pod:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

La CNI identifica che l'IP di destinazione appartiene a un pod nella sua rete e consegna il pacchetto al namespace della rete del pod corretto.

Questo flusso mostra come CNI Nat-ing può semplificare la configurazione consentendo il reindirizzamento dei pacchetti al nodo senza richiedere un routing aggiuntivo per il pod. CIDRs

Dal piano di controllo (control-plane) EKS ai pod in esecuzione su un nodo ibrido (webhook)



Questo schema di traffico si riscontra più comunemente con i webhook, in cui il piano di controllo (control-plane) EKS deve avviare direttamente le connessioni ai server webhook in esecuzione nei pod su nodi ibridi. Gli esempi includono la convalida e la modifica dei webhook di ammissione, che vengono chiamati dal server API durante i processi di convalida o mutazione delle risorse.

Richiesta

1. Il server API EKS Kubernetes avvia la richiesta

Quando un webhook è configurato nel cluster e un'operazione API pertinente lo attiva, il server API EKS Kubernetes deve stabilire una connessione diretta al pod del server webhook. Il server API cerca innanzitutto l'indirizzo IP del pod dalla risorsa Service o Endpoint associata al webhook.

Supponendo che il pod del webhook sia in esecuzione su un nodo ibrido con IP `10.85.1.23`, il server API EKS Kubernetes crea una richiesta HTTPS all'endpoint del webhook. Il pacchetto iniziale viene inviato tramite l'ENI del piano di controllo (control-plane) EKS nel VPC perché l'IP di destinazione `10.85.1.23` appartiene al CIDR del pod remoto configurato (`10.85.0.0/16`). Il pacchetto appare come riportato di seguito:

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

```
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 41892 (random) |                 |
| Dst: 10.85.1.23 | Dst: 8443           |                 |
+-----------------+---------------------+-----------------+
```

2. Elaborazione di rete del VPC

Il pacchetto lascia l'ENI del piano di controllo (control-plane) EKS ed entra nel livello di rete del VPC con il router della sottorete come hop successivo.

3. Ricerca della tabella di routing del VPC

La tabella di routing del VPC per la sottorete contenente l'ENI del piano di controllo (control-plane) EKS contiene un instradamento specifico per il CIDR del pod remoto (`10.85.0.0/16`). Questa regola di routing indirizza il pacchetto VPC-to-onprem verso il gateway (ad esempio, un Virtual Private Gateway per Direct Connect o connessioni VPN):

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

4. Transito oltre il limite

Il gateway trasferisce il pacchetto oltre i limiti del cloud tramite la connessione stabilita (come Direct Connect o VPN) alla rete on-premises. Il pacchetto mantiene gli indirizzi IP di origine e destinazione originali mentre attraversa questa connessione.

5. Ricezione nella rete on-premises

Il pacchetto arriva al router on-premises locale. Il router consulta la propria tabella di routing per determinare come raggiungere l'indirizzo 10.85.1.23. Perché ciò funzioni, la rete locale deve disporre di percorsi per il pod CIDRs che indirizzino il traffico verso il nodo ibrido appropriato.

In questo caso, la tabella di routing del router contiene una voce che indica che la sottorete `10.85.1.0/24` è raggiungibile tramite il nodo ibrido con IP `10.80.0.2`:

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

6. Distribuzione al nodo

In base alla voce della tabella di routing, il router inoltra il pacchetto al nodo ibrido (`10.80.0.2`). Quando il pacchetto arriva al nodo, ha lo stesso aspetto di quando il server API EKS Kubernetes lo ha inviato, con l'IP di destinazione che è ancora l'IP del pod.

7. Elaborazione della CNI

Lo stack di rete del nodo riceve il pacchetto e, dato che l'IP di destinazione non è quello del nodo, lo passa alla CNI per l'elaborazione. La CNI identifica che l'IP di destinazione appartiene a un pod in esecuzione localmente su questo nodo e inoltra il pacchetto al pod corretto tramite le interfacce virtuali appropriate:

```
Original packet -> node routing -> CNI -> Pod's network namespace
```

Il server webhook nel pod riceve la richiesta e la elabora.

Risposta

Dopo che il pod del webhook ha elaborato la richiesta, restituisce una risposta seguendo lo stesso percorso in senso inverso:

7. Il pod invia la risposta

Il pod del webhook crea un pacchetto di risposta con il proprio IP come origine e il richiedente originale (l'ENI del piano di controllo (control-plane) EKS) come destinazione:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.23 | Src: 8443           |                 |
| Dst: 10.0.0.132 | Dst: 41892          |                 |
+-----------------+---------------------+-----------------+
```

La CNI identifica che questo pacchetto va a una rete esterna (non a un pod locale) e lo passa allo stack di rete del nodo mantenendo l'IP di origine originale.

6. Elaborazione della rete del nodo

Il nodo determina che l'IP di destinazione (`10.0.0.132`) non si trova nella rete locale e inoltra il pacchetto al gateway predefinito (il router locale).

5. Routing del router locale

Il router locale consulta la sua tabella di routing e determina che l'IP di destinazione (`10.0.0.132`) appartiene al CIDR del VPC (`10.0.0.0/16`). Inoltra il pacchetto al gateway che si connette a AWS.

4. Transito oltre il limite

Il pacchetto torna indietro attraverso la stessa connessione on-premises al VPC, superando il limite del cloud nella direzione opposta.

3. Routing del VPC

Quando il pacchetto arriva nel VPC, le tabelle di routing identificano che l'IP di destinazione appartiene a una sottorete all'interno del VPC. Il pacchetto viene instradato di conseguenza all'interno del VPC.

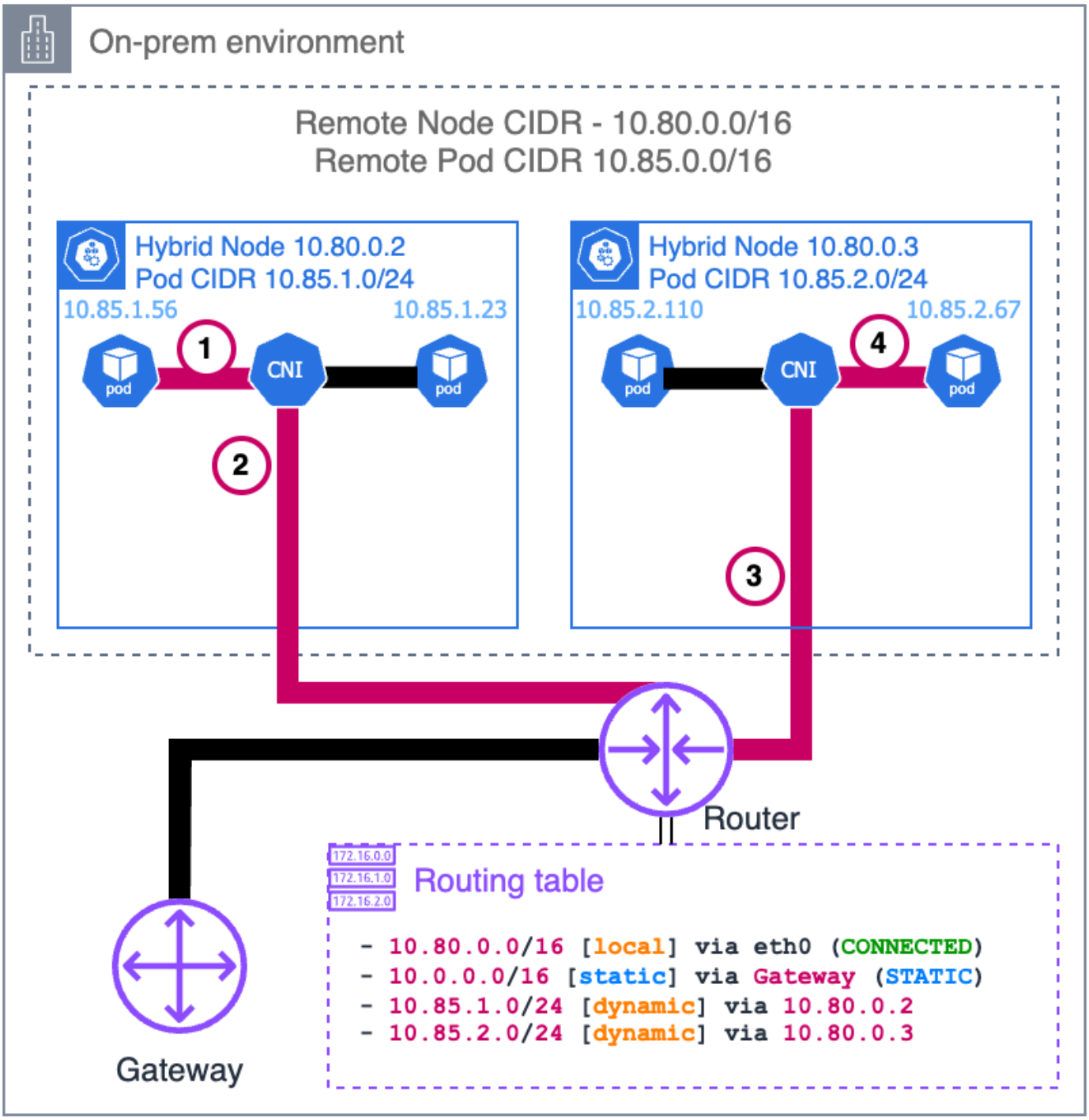
2. e 1. Ricezione dell'ENI del piano di controllo (control-plane) EKS

Il pacchetto raggiunge l'ENI collegata al server API Kubernetes, completando il viaggio di andata e ritorno. Il server API riceve la risposta del webhook e continua a elaborare la richiesta API originale in base a tale risposta.

Questo flusso di traffico dimostra perché il pod remoto CIDRs deve essere configurato e instradato correttamente:

- Il VPC deve disporre di percorsi per il pod remoto che CIDRs puntano al gateway locale
- La rete locale deve disporre di percorsi per i pod CIDRs che indirizzano il traffico verso i nodi specifici che ospitano tali pod
- Senza questa configurazione di routing, i webhook e altri servizi simili eseguiti in pod su nodi ibridi non sarebbero raggiungibili dal piano di controllo (control-plane) EKS.

Pod-to-Pod in esecuzione su nodi ibridi



Questa sezione spiega come i pod in esecuzione su diversi nodi ibridi comunicano tra loro. Questo esempio presuppone che il CNI utilizzi VXLAN per l'incapsulamento, operazione comune per CNIs

Cilium o Calico. Il processo complessivo è simile per altri protocolli di incapsulamento come Geneve o. IP-in-IP

Richiesta

1. Il pod A avvia la comunicazione

Il pod A (`10.85.1.56`) sul nodo 1 desidera inviare traffico al pod B (`10.85.2.67`) sul nodo 2. Il pacchetto iniziale appare come riportato di seguito:

```
+------------------+-----------------+-------------+-----------------+
| 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. La CNI intercetta ed elabora il pacchetto

Quando il pacchetto del pod A lascia il suo namespace di rete, la CNI lo intercetta. La CNI consulta la sua tabella di routing e determina i seguenti elementi: - L'IP di destinazione (`10.85.2.67`) appartiene al CIDR del pod - Questo IP non si trova sul nodo locale ma appartiene al nodo 2 (`10.80.0.3`) - Il pacchetto deve essere incapsulato con VXLAN.

La decisione di incapsulare è fondamentale perché la rete fisica sottostante non sa come indirizzare CIDRs direttamente i pod, ma solo come instradare il traffico tra i nodi. IPs

La CNI incapsula l'intero pacchetto originale all'interno di un frame VXLAN. Questo crea effettivamente un "pacchetto all'interno di un pacchetto" con nuovi header:

```
+-----------------+---------------+-------------+-----------
+--------------------------+
| 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)
    |
+-----------------+---------------+-------------+-----------
+--------------------------+
```

Punti chiave di questo incapsulamento: - Il pacchetto esterno viene indirizzato dal nodo 1 (`10.80.0.2`) al nodo 2 (`10.80.0.3`) - La porta UDP `8472` è la porta VXLAN utilizzata da Cilium

per impostazione predefinita - Il VXLAN Network Identifier (VNI) identifica a quale rete sovrapposta appartiene tale pacchetto - L'intero pacchetto originale (con l'IP del pod A come origine e l'IP del pod B come destinazione) è conservato intatto all'interno

Il pacchetto incapsulato entra ora nel normale stack di rete del nodo 1 e viene elaborato allo stesso modo di qualsiasi altro pacchetto:

1. Elaborazione della rete del nodo: lo stack di rete del nodo 1 indirizza il pacchetto in base alla sua destinazione (`10.80.0.3`)
2. Distribuzione della rete locale:
   - Se entrambi i nodi si trovano sulla stessa rete di livello 2, il pacchetto viene inviato direttamente al nodo 2
   - Se si trovano su sottoreti diverse, il pacchetto viene prima inoltrato al router locale
3. Gestione del router: il router inoltra il pacchetto in base alla sua tabella di routing, distribuendolo al nodo 2

3. Elaborazione dei nodi di ricezione

Quando il pacchetto incapsulato arriva al nodo 2 (`10.80.0.3`):

1. Lo stack di rete del nodo lo riceve e lo identifica come pacchetto VXLAN (porta UDP `4789`)
2. Il pacchetto viene passato all'interfaccia VXLAN della CNI per l'elaborazione

4. Decapsulazione VXLAN

La CNI sul nodo 2 elabora il pacchetto VXLAN:

1. Elimina gli header esterni (Ethernet, IP, UDP e VXLAN).
2. Estrae il pacchetto interno originale.
3. Il pacchetto è ora tornato alla sua forma originale:

```
+------------------+-----------------+-------------+-----------------+
| 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   |                 |
+------------------+-----------------+-------------+-----------------+
```

La CNI sul nodo 2 esamina l'IP di destinazione (`10.85.2.67`) e:

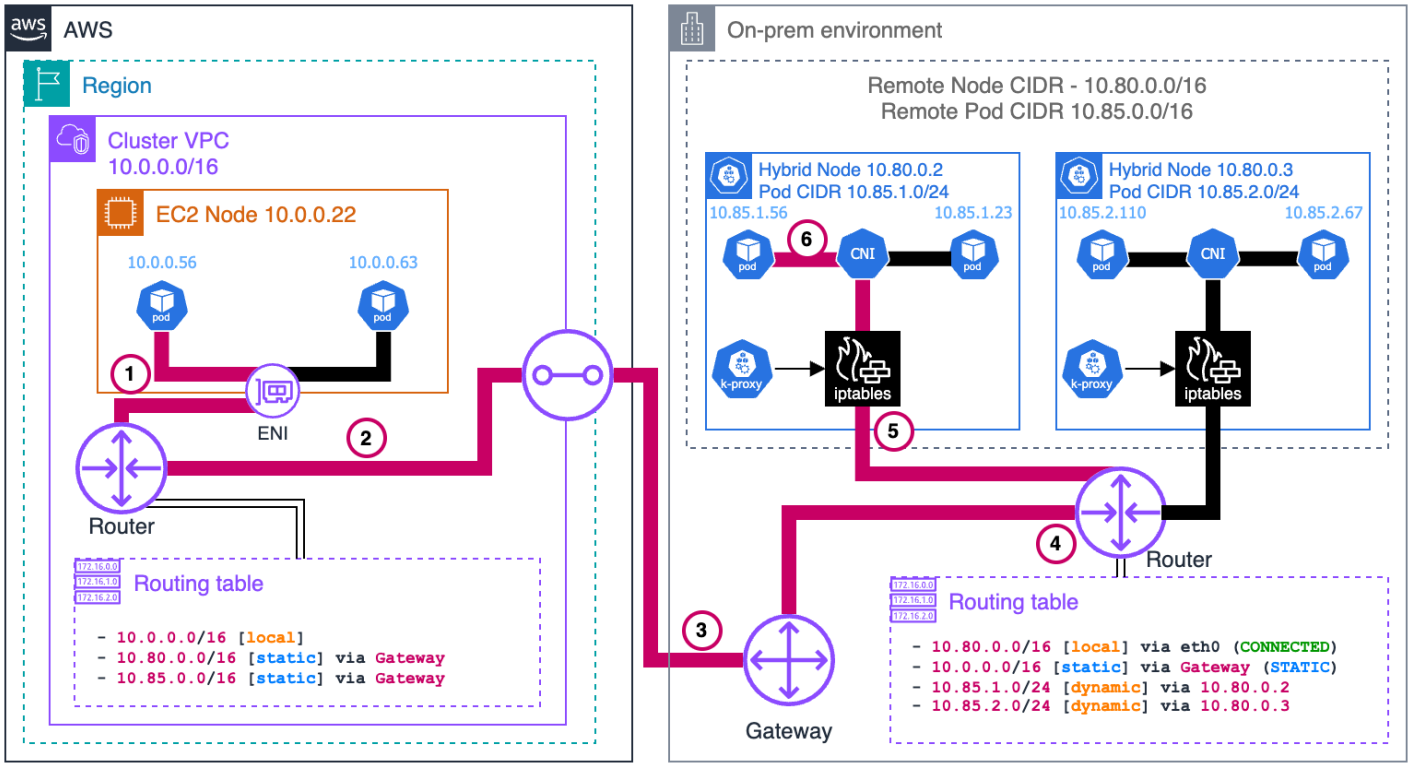
1. Identifica che questo IP appartiene a un pod locale.
2. Instrada il pacchetto attraverso le interfacce virtuali appropriate.
3. Fornisce il pacchetto al namespace di rete del pod B.

Risposta

Quando il pod B risponde al pod A, l'intero processo avviene al contrario:

4. Il pod B invia un pacchetto al pod A (`10.85.1.56`).
5. La CNI del nodo 2 lo incapsula con VXLAN, impostando la destinazione sul nodo 1 (`10.80.0.2`).
6. Il pacchetto incapsulato viene consegnato al nodo 1.
7. La CNI del nodo 1 lo decapsula e fornisce la risposta originale al pod A.

Da pod su nodi cloud a pod su nodi ibridi (traffico est-ovest)



Richiesta

1. Il pod A avvia la comunicazione

Il Pod A (`10.0.0.56`) sul EC2 nodo desidera inviare traffico al Pod B (`10.85.1.56`) sul nodo ibrido. Il pacchetto iniziale appare come riportato di seguito:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.56  | Src: 52390 (random) |                 |
| Dst: 10.85.1.56 | Dst: 8080           |                 |
+-----------------+---------------------+-----------------+
```

Con il VPC CNI, il Pod A ha un IP dal VPC CIDR ed è collegato direttamente a un ENI sull'istanza. EC2 Il namespace di rete del pod è connesso alla rete del VPC, quindi il pacchetto entra direttamente nell'infrastruttura di routing VPC.

2. Routing del VPC

La tabella di routing VPC contiene una route specifica per il Remote Pod CIDR (`10.85.0.0/16`), che indirizza questo traffico verso il gateway: VPC-to-onprem

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

In base a questa regola di routing, il pacchetto viene indirizzato verso il gateway che si connette alla rete on-premises.

3. Transito oltre il limite

Il gateway trasferisce il pacchetto oltre i limiti del cloud tramite la connessione stabilita (come Direct Connect o VPN) alla rete on-premises. Il pacchetto mantiene gli indirizzi IP di origine e di destinazione originali durante questo transito.

4. Ricezione nella rete on-premises

Il pacchetto arriva al router on-premises locale. Il router consulta la propria tabella di routing per determinare l'hop successivo per raggiungere l'indirizzo 10.85.1.56. Il router locale deve disporre di percorsi per il pod CIDRs che indirizzano il traffico verso il nodo ibrido appropriato.

La tabella del router contiene una voce che indica che la sottorete `10.85.1.0/24` è raggiungibile tramite il nodo ibrido con IP `10.80.0.2`:

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

5. Elaborazione della rete del nodo

Il router inoltra il pacchetto al nodo ibrido (`10.80.0.2`). Quando il pacchetto arriva al nodo, ha ancora l'IP del pod A come origine e l'IP del pod B come destinazione.

6. Elaborazione della CNI

Lo stack di rete del nodo riceve il pacchetto e, visto che l'IP di destinazione non è il proprio, lo passa alla CNI per l'elaborazione. La CNI identifica che l'IP di destinazione appartiene a un pod in esecuzione localmente su questo nodo e inoltra il pacchetto al pod corretto tramite le interfacce virtuali appropriate:

```
Original packet -> node routing -> CNI -> Pod B's network namespace
```

Il pod B riceve il pacchetto e lo elabora in base alle esigenze.

Risposta

6. Il pod B invia la risposta

Il pod B crea un pacchetto di risposta con il proprio IP come origine e l'IP del pod A come destinazione:

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 8080           |                 |
| Dst: 10.0.0.56  | Dst: 52390          |                 |
+-----------------+---------------------+-----------------+
```

La CNI identifica che questo pacchetto è destinato a una rete esterna e lo passa allo stack di rete del nodo.

5. Elaborazione della rete del nodo

Il nodo determina che l'IP di destinazione (`10.0.0.56`) non appartiene alla rete locale e inoltra il pacchetto al gateway predefinito (il router locale).

4. Routing del router locale

Il router locale consulta la sua tabella di routing e determina che l'IP di destinazione (`10.0.0.56`) appartiene al CIDR del VPC (`10.0.0.0/16`). Inoltra il pacchetto al gateway che si connette a AWS.

3. Transito oltre il limite

Il pacchetto torna indietro attraverso la stessa connessione on-premises al VPC, superando il limite del cloud nella direzione opposta.

2. Routing del VPC

Quando il pacchetto arriva nel VPC, il sistema di routing identifica che l'IP di destinazione appartiene a una sottorete all'interno del VPC. Il pacchetto viene instradato attraverso la rete VPC verso l'istanza che ospita il EC2 Pod A.

1. Il pod A riceve la risposta

Il pacchetto arriva all' EC2 istanza e viene consegnato direttamente al Pod A tramite l'ENI collegato. Poiché la CNI del VPC non utilizza reti sovrapposte per i pod nel VPC, non è necessario alcun decapsulazione aggiuntivo: il pacchetto arriva con le intestazioni originali intatte.

Questo flusso di traffico est-ovest dimostra perché il pod remoto CIDRs deve essere configurato correttamente e instradabile da entrambe le direzioni:

- Il VPC deve disporre di percorsi per il pod remoto che CIDRs puntano al gateway locale
- La rete locale deve disporre di percorsi per i pod CIDRs che indirizzano il traffico verso i nodi specifici che ospitano tali pod.

Riferimento **`nodeadm`** dei nodi ibridi

La CLI Amazon EKS Hybrid Nodes (`nodeadm`) semplifica l'installazione, la configurazione, la registrazione e la disinstallazione dei componenti dei nodi ibridi. Puoi includere il `nodeadm` nelle immagini del tuo sistema operativo per automatizzare il bootstrap dei nodi ibridi, consulta the section called "Preparazione del sistema operativo" per ulteriori informazioni.

La `nodeadm` versione per i nodi ibridi è diversa dalla `nodeadm` versione utilizzata per avviare le istanze EC2 Amazon come nodi nei cluster Amazon EKS. Consulta la documentazione e i riferimenti per la versione del `nodeadm` appropriata. Questa pagina di documentazione riguarda la versione del `nodeadm` per i nodi ibridi.

Il codice sorgente per i nodi ibridi `nodeadm` è pubblicato nel repository eks-hybrid. https://github.com/aws/ GitHub

> **Important**
>
> È necessario eseguire l'esecuzione `nodeadm` con un utente con privilegi. root/sudo

Scarica **`nodeadm`**

La versione con nodi ibridi di `nodeadm` è ospitata in Amazon S3 gestita da Amazon. CloudFront Per installare `nodeadm` su ciascun host on-premises puoi eseguire il seguente comando dai tuoi host on-premises.

Per host x86_64

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/nodeadm'
```

Per host ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/nodeadm'
```

Aggiungi l'autorizzazione per i file eseguibili al file binario scaricato su ciascun host.

```
chmod +x nodeadm
```

`nodeadm install`

Il comando `nodeadm install` viene utilizzato per installare gli artefatti e le dipendenze necessari per eseguire e unire nodi ibridi a un cluster Amazon EKS. Il comando `nodeadm install` può essere eseguito singolarmente su ciascun nodo ibrido o può essere eseguito durante le pipeline di creazione delle immagini per preinstallare le dipendenze dei nodi ibridi nelle immagini del sistema operativo.

Utilizzo

```
nodeadm install [KUBERNETES_VERSION] [flags]
```

Argomenti posizionali

(Obbligatorio) `KUBERNETES_VERSION` La versione major.minor di EKS Kubernetes da installare, ad esempio `1.32`

Flag

Name	Richiesto	Description
`-p`, `--credential-provider`	TRUE	Provider di credenziali da installare. I valori supportati sono `iam-ra` e `ssm`. Per ulteriori informazioni, consulta the section called "Preparazione delle credenziali".
`-s`, `--containerd-source`	FALSE	Origine per `containerd`. `nodeadm` supporta l'installazione di `containerd` dalla distribuzione del sistema operativo, i pacchetti Docker e la possibilità di saltare l'installazione di `containerd`. Valori `distro`- Questo è il valore predefinito. `nodeadm`installer à il `containerd` pacchetto più recente distribuito dal sistema operativo del nodo compatibile con la versione EKS Kubernetes. `distro`non è un valore supportato per i sistemi operativi Red Hat Enterprise Linux (RHEL). `docker`- `nodeadm` installer à l'ultimo `containerd`

Name	Richiesto	Description
		pacchetto creato e distribui to da Docker compatibile con la versione EKS Kubernete s. `docker`non è un valore supportato per Amazon Linux 2023. `none`: `nodeadm` non installer à il pacchetto `containerd` . Devi installare manualmen te `containerd` prima di eseguire `nodeadm init`.
`-r`, `--region`	FALSE	Speciifica la AWS regione per il download di artefatti come l'agente SSM. L'impostazione predefinita è `us-west-2` .
`-t`, `--timeout`	FALSE	Durata massima del comando di installazione. L'input segue il formato della durata. Ad esempio, `1h23m`. Il timeout per il download predefinito per il comando di installazione è impostato su 20 minuti.
`-h`, `--help`	FALSE	Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili.

Esempi

Installa la versione Kubernetes con AWS Systems `1.32` Manager (SSM) come provider di credenziali

```
nodeadm install 1.32 --credential-provider ssm
```

Installa la versione Kubernetes con AWS Systems `1.32` Manager (SSM) come provider di credenziali, Docker come origine containerd, con un timeout di download di 20 minuti.

```
nodeadm install 1.32 --credential-provider ssm --containerd-source docker --timeout 20m
```

Installa la versione Kubernetes con IAM Roles Anywhere come provider di credenziali `1.32` AWS

```
nodeadm install 1.32 --credential-provider iam-ra
```

`nodeadm config check`

Il comando `nodeadm config check` verifica la presenza di errori nella configurazione del nodo fornita. Questo comando può essere utilizzato per verificare e confermare la correttezza di un file di configurazione del nodo ibrido.

Utilizzo

```
nodeadm config check [flags]
```

Flag

Name	Richiesto	Description
`-c`, `--config-source`	TRUE	Origine della configurazione di nodeadm. Per i nodi ibridi, l'input deve seguire un URI con schema di file.
`-h`, `--help`	FALSE	Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili.

Esempi

```
nodeadm config check -c file://nodeConfig.yaml
```

`nodeadm init`

Il comando `nodeadm init` avvia e connette il nodo ibrido con il cluster Amazon EKS configurato. Consulta the section called "Configurazione del modo per attivazioni ibride SSM" o the section called "Configurazione del nodo per IAM Roles Anywhere" per i dettagli su come configurare il file `nodeConfig.yaml`.

Utilizzo

```
nodeadm init [flags]
```

Flag

Name	Richiesto	Description
`-c`, `--config-source`	TRUE	Origine della configurazione di `nodeadm`. Per i nodi ibridi, l'input deve seguire un URI con schema di file.
`-s`, `--skip`	FALSE	Fasi di `init` da saltare. Non è consigliabile saltare nessuna delle fasi a meno che non sia utile per risolvere un problema. Valori `install-validation` salta la verifica se il comando di installazione precedente è stato eseguito correttamente. `cni-validation` salta la verifica se le porte VXLAN della CNI di Cilium o di Calico sono aperte se il firewall è abilitato sul nodo

Name	Richiesto	Description
		`node-ip-validation` salta la verifica se l'IP del nodo rientra in un CIDR nelle reti dei nodi remoti
`-h`, `--help`	FALSE	Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili.

Esempi

```
nodeadm init -c file://nodeConfig.yaml
```

`nodeadm upgrade`

Il comando `nodeadm upgrade` aggiorna tutti gli artefatti installati alla versione più recente e avvia il nodo per configurare gli artefatti aggiornati e unirsi al cluster EKS su AWS. Upgrade è un comando che interrompe i carichi di lavoro in esecuzione sul nodo. Devi spostare i carichi di lavoro su un altro nodo prima di eseguire l'aggiornamento.

Utilizzo

```
nodeadm upgrade [KUBERNETES_VERSION] [flags]
```

Argomenti posizionali

(Obbligatorio) `KUBERNETES_VERSION` La versione major.minor di EKS Kubernetes da installare, ad esempio `1.32`

Flag

Name	Richiesto	Description
`-c`, `--config-source`	TRUE	Origine della configurazione di `nodeadm`. Per i nodi ibridi,

Name	Richiesto	Description
		l'input deve seguire un URI con schema di file.
`-t`, `--timeout`	FALSE	Timeout per il download degli artefatti. L'input segue il formato della durata. Ad esempio 1h23m. Il timeout per il download predefinito per il comando di aggiornamento è impostato su 10 minuti.

Name	Richiesto	Description
`-s`, `--skip`	FALSE	Fasi dell'aggiornamento da saltare. Non è consigliabile saltare nessuna Passaggio a meno che non sia utile per risolvere un problema. Valori `pod-validation` salta la verifica se tutti i pod sono in esecuzione sul nodo, tranne i set daemon e i pod statici. `node-validation` salta la verifica se il nodo è stato isolato. `init-validation` salta la verifica se il nodo è stato inizializzato correttamente prima di eseguire l'aggiorn amento. `containerd-major-version-upgrade` impedisce gli aggiornamenti delle versioni principali di containerd durante l'aggiornamento del nodo.
`-h`, `--help`	FALSE	Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili.

Esempi

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml
```

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml --timeout 20m
```

`nodeadm uninstall`

Il comando `nodeadm uninstall` arresta e rimuove gli artefatti che `nodeadm` installa durante `nodeadm install`, inclusi kubelet e containerd. Attenzione, il comando di disinstallazione non scarica né elimina i nodi ibridi dal cluster. Devi eseguire le operazioni di scaricamento ed eliminazione separatamente, consulta the section called "Elimina i nodi ibridi" per ulteriori informazioni. Per impostazione predefinita, `nodeadm uninstall` non proseguirà se sul nodo rimangono dei pod. Allo stesso modo, `nodeadm uninstall` non rimuove le dipendenze CNI o le dipendenze di altri componenti aggiuntivi di Kubernetes eseguiti sul cluster. Per rimuovere completamente l'installazione CNI dal tuo host, consulta le istruzioni contenute in the section called "Configurazione della CNI". Se utilizzi attivazioni ibride AWS SSM come provider di credenziali locale, il `nodeadm uninstall` comando annulla la registrazione degli host come istanze gestite tramite SSM. AWS

Utilizzo

```
nodeadm uninstall [flags]
```

Flag

Name	Richiesto	Description
`-s`, `--skip`	FALSE	Fasi di disinstallazione da saltare. Non è consigliabile saltare nessuna delle fasi a meno che non sia utile per risolvere un problema. Valori `pod-validation` salta la verifica se tutti i pod sono in esecuzione sul nodo, tranne i set daemon e i pod statici.

Name	Richiesto	Description
		`node-validation` salta la verifica se il nodo è stato isolato. `init-validation` salta la verifica se il nodo è stato inizializzato correttamente prima di eseguire la disinstallazione.
`-h`, `--help`	FALSE	Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili.

Name	Richiesto	Description
`-f`, `--force`	FALSE	Forza l'eliminazione delle directory aggiuntive che potrebbero contenere file rimanenti dai componenti Kubernetes e CNI. ATTENZIONE Ciò eliminerà tutti i contenuti nelle directory Kubernetes e CNI predefinite (`/var/lib/cni` , `/etc/cni/net.d` , ecc.). Non utilizzare questo flag se memorizzi i tuoi dati in queste posizioni. A partire dalla versione `v1.0.9` di nodeadm, il comando `./nodeadm uninstall --skip node-validation,pod-validation --force` non elimina più la directory `/var/lib/kubelet` . Questo perché può contenere volumi Pod e directory volume-subpath che a volte includono il filesystem dei nodi montato. Suggerimenti per una gestione sicura L'eliminazione dei percorsi montati può portare alla cancellazione accidentale dell'effettivo filesystem

Name	Richiesto	Description
		dei nodi montati. Prima di eliminare manualmente la directory `/var/lib/kubelet` , ispeziona attentamente tutti i montaggi attivi e smonta i volumi in modo sicuro per evitare la perdita di dati.

Esempi

```
nodeadm uninstall
```

```
nodeadm uninstall --skip node-validation,pod-validation
```

`nodeadm debug`

Il comando `nodeadm debug` può essere utilizzato per risolvere i problemi relativi ai nodi ibridi non integri o configurati in modo errato. Verifica che i seguenti requisiti siano soddisfatti.

- Il nodo ha accesso alla rete necessaria per ottenere le credenziali, AWS APIs
- Il nodo è in grado di ottenere AWS le credenziali per il ruolo IAM di Hybrid Nodes configurato,
- Il nodo ha accesso di rete all'endpoint dell'API EKS Kubernetes e ha la validità del certificato endpoint dell'API EKS Kubernetes.
- Il nodo è in grado di autenticarsi con il cluster EKS, la sua identità nel cluster è valida e ha accesso al cluster EKS tramite il VPC configurato per il cluster EKS.

Se vengono rilevati errori, l'output del comando suggerisce le procedure per la risoluzione dei problemi. Alcuni passaggi di conferma mostrano le procedure secondarie. Se queste non funzionano, l'output viene mostrato in una sezione stderr sotto l'errore di convalida.

Utilizzo

```
nodeadm debug [flags]
```

Flag

Name	Richiesto	Description
`-c`, `--config-source`	TRUE	Origine della configurazione di `nodeadm`. Per i nodi ibridi, l'input deve seguire un URI con schema di file.
`--no-color`	FALSE	Disattiva l'output a colori. Utile per l'automazione.
`-h`, `--help`	FALSE	Mostra un messaggio di aiuto con i parametri di flag, sottocomando e valore posizionale disponibili.

Esempi

```
nodeadm debug -c file://nodeConfig.yaml
```

Posizioni dei file nodeadm

installazione di nodeadm

Durante l'esecuzione di `nodeadm install`, vengono configurati i seguenti file e posizioni dei file.

Artifact	Path
CLI IAM Roles Anywhere	/usr/local/bin/aws_signing_helper
Binario Kubelet	/usr/bin/kubelet
Binario Kubectl	usr/local/bin/kubectl
Provider di credenziali ECR	/etc/eks/image-credential-provider/ecr-fornitore di credenziali
AWS Autenticatore IAM	/-iam-authenticator usr/local/bin/aws

Artifact	Path
CLI di configurazione SSM	/opt/ssm/ssm-setup-cli
Agente SSM	Su Ubuntu -/-ssm-agent snap/amazon-ssm-agent/current/amazon Su RHEL e 023 -/-ssm-agent AL2 usr/bin/amazon
Containerd	Su Ubuntu e 023 -/ AL2usr/bin/containerd Su RHEL - /bin/containerd
Iptables	Su Ubuntu e AL2 023 -/usr/sbin/iptables Su RHEL - /sbin/iptables
Plug-in CNI	/opt/cni/bin
tracciatore di artefatti installati	/opt/nodeadm/tracker

nodeadm init

Durante l'esecuzione di `nodeadm init`, vengono configurati i file e le posizioni seguenti dei file.

Name	Path
Kubelet kubeconfig	/var/lib/kubelet/kubeconfig
Configurazione di Kubelet	/.json etc/kubernetes/kubelet/config
Unità Kubelet systemd	/.servizio etc/systemd/system/kubelet
Configurazione del provider di credenziali di immagine	/.json etc/eks/image-credential-provider/config
File env Kubelet	/etc/eks/kubelet/environment
Certificati Kubelet	/.crt etc/kubernetes/pki/ca

Name	Path
Configurazione di Containerd	/.toml etc/containerd/config
Configurazione dei moduli del kernel Containerd	/.conf etc/modules-load.d/contianerd
AWS file di configurazione	/etc/aws/hybrid/config
AWS file di credenziali (se abilita il file delle credenziali)	/eks-hybrid/.aws/credentials
AWS unità di sistema di supporto alla firma	/etc/systemd/system/aws_signing_helper_update.service
File conf sysctl	/etc/sysctl.d/99-nodeadm.conf
Ca-certificates	/etc/ssl/certs/ca-certificati.crt
File chiave Gpg	/etc/apt/keyrings/docker.asc
File di origine del repository Docker	/.lista etc/apt/sources.list.d/docker

Configurazione del modo per attivazioni ibride SSM

Di seguito è riportato un esempio `nodeConfig.yaml` di utilizzo delle attivazioni ibride AWS SSM per le credenziali dei nodi ibridi.

```
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
```

Configurazione del nodo per IAM Roles Anywhere

Di seguito è riportato un esempio di `nodeConfig.yaml` credenziali AWS IAM Roles Anywhere per nodi ibridi.

Quando utilizzi AWS IAM Roles Anywhere come provider di credenziali locale, le `nodeName` credenziali utilizzate nella `nodeadm` configurazione devono essere in linea con le autorizzazioni previste per il ruolo IAM di Hybrid Nodes. Ad esempio, se le autorizzazioni per il ruolo IAM di Hybrid Nodes consentono a AWS IAM Roles Anywhere di assumere il ruolo solo quando il nome della sessione del ruolo è uguale al CN del certificato host, allora la `nodeName` `nodeadm` configurazione deve essere la stessa del CN dei tuoi certificati. Il `nodeName` che usi non può contenere più di 64 caratteri. Per ulteriori informazioni, consulta the section called "Preparazione delle credenziali".

```
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
```

Configurazione del nodo per personalizzare kubelet (Facoltativo)

Puoi passare la configurazione e i flag di kubelet nella tua configurazione `nodeadm`. Vedi l'esempio seguente su come aggiungere un'etichetta `abc.amazonaws.com/test-label` e una configurazione del nodo aggiuntive per l'impostazione di `shutdownGracePeriod` su 30 secondi.

```
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
```

Configurazione del nodo per personalizzare containerd (Facoltativo)

Puoi passare la configurazione containerd personalizzata nella tua configurazione `nodeadm`. La configurazione containerd per `nodeadm` accetta TOML in linea. Vedi l'esempio seguente su come configurare containerd per disabilitare l'eliminazione dei livelli di immagine decompressi nel content store 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**
>
> Le versioni 1.x e 2.x di Containerd utilizzano formati di configurazione diversi. Containerd 1.x utilizza la versione 2 di configurazione, mentre containerd 2.x utilizza la versione 3 di configurazione. Sebbene containerd 2.x rimanga retrocompatibile con la versione 2 di configurazione, la versione 3 è consigliata per prestazioni ottimali. Controlla la versione containerd in uso o `nodeadm` consulta i log di `containerd --version` installazione. Per

maggiori dettagli sul controllo delle versioni di configurazione, consulta https://containerd.io/releases/

Puoi anche usare la configurazione containerd per abilitare il supporto. SELinux Con SELinux enabled on containerd, assicurati che i pod programmati sul nodo abbiano il SecurityContext corretto e siano abilitati. seLinuxOptions Ulteriori informazioni sulla configurazione di un contesto di sicurezza sono disponibili in Kubernetes documentation.

> **Note**
>
> Red Hat Enterprise Linux (RHEL) 8 e RHEL 9 sono SELinux abilitati di default e impostati su strict sull'host. Amazon Linux 2023 è SELinux abilitato per impostazione predefinita e impostato in modalità permissiva. Quando SELinux è impostata la modalità permissiva sull'host, abilitarla su containerd non bloccherà le richieste ma le registrerà in base alla configurazione sull'host. SELinux

```
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
```

Risoluzione dei problemi relativi ai nodi ibridi

Questo argomento illustra alcuni errori comuni che si potrebbero verificare durante l'utilizzo di Amazon EKS Hybrid Nodes, e il modo in cui ripararli. Per altre informazioni sulla risoluzione dei problemi, consulta Risoluzione dei problemi il tag Knowledge Center per Amazon EKS su AWS re:POST. Se non riesci a risolvere il problema, contatta l' AWS assistenza.

Risoluzione dei problemi dei nodi con **`nodeadm debug`**. Puoi eseguire il comando `nodeadm debug` dai nodi ibridi per verificare che i requisiti di rete e le credenziali siano soddisfatti. Per ulteriori informazioni sul comando `nodeadm debug`, consulta the section called "nodeadm dei nodi ibridi".

Rileva i problemi con i tuoi nodi ibridi con le informazioni sul cluster. Le informazioni sul cluster Amazon EKS includono controlli approfonditi che rilevano problemi comuni con la configurazione dei nodi ibridi EKS nel cluster. È possibile visualizzare i risultati di tutti i controlli approfonditi da Console di gestione AWS, AWS CLI e. AWS SDKs Per ulteriori informazioni sulle informazioni sul cluster, consulta the section called "Approfondimenti sui cluster".

Risoluzione dei problemi di installazione dei nodi ibridi

I seguenti argomenti di risoluzione dei problemi riguardano l'installazione delle dipendenze dei nodi ibridi sugli host con il comando `nodeadm install`.

Comando **`nodeadm`** non riuscito per **`must run as root`**

Il comando `nodeadm install` deve essere eseguito con un utente che dispone di root o privilegi `sudo` sull'host. Se esegui `nodeadm install` con un utente che non dispone di root o privilegi `sudo`, nell'output `nodeadm` verrà visualizzato il seguente errore.

```
"msg":"Command failed","error":"must run as root"
```

Impossibile connettersi alle dipendenze

Il comando `nodeadm install` installa le dipendenze richieste per i nodi ibridi. Le dipendenze dei nodi ibridi includono`containerd`, `kubelet`, `kubectl`, e componenti AWS SSM o AWS IAM Roles Anywhere. Devi avere accesso da dove stai eseguendo `nodeadm install` per scaricare queste dipendenze. Per ulteriori informazioni sull'elenco delle posizioni a cui devi poter accedere, consulta the section called "Preparazione della rete". Se non disponi dell'accesso, nell'output `nodeadm install` verranno visualizzati errori simili ai seguenti.

```
"msg":"Command failed","error":"failed reading file from url: ...: max retries achieved
 for http request"
```

Impossibile aggiornare il gestore di pacchetti

Il comando `nodeadm install` esegue `apt update`, `yum update` o `dnf update` prima dell'installazione delle dipendenze dei nodi ibridi. Se questo passaggio non ha esito positivo, potresti

visualizzare errori simili ai seguenti. Per rimediare, puoi eseguire `apt update`, `yum update` o `dnf update` prima di eseguire `nodeadm install` oppure puoi provare a rieseguire `nodeadm install`.

```
failed to run update using package manager
```

Scadenza di un timeout o di un contesto superata

Durante l'esecuzione di `nodeadm install`, se riscontri problemi in varie fasi del processo di installazione con errori che indicano che è stata superata la scadenza di un timeout o di un contesto, è possibile che la connessione sia lenta e impedisca l'installazione delle dipendenze dei nodi ibridi prima del raggiungimento dei timeout. Per risolvere questi problemi, puoi provare a utilizzare il flag `--timeout` in `nodeadm` per estendere la durata dei timeout per il download delle dipendenze.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --timeout 20m0s
```

Risoluzione dei problemi di connessione dei nodi ibridi

Gli argomenti di risoluzione dei problemi in questa sezione riguardano il processo di connessione dei nodi ibridi ai cluster EKS con il comando `nodeadm init`.

Errori operativi o schema non supportato

Durante l'esecuzione di `nodeadm init`, se vedi errori relativi a `operation error` o`unsupported scheme`, controlla `nodeConfig.yaml` per assicurarti che sia formattato e passato correttamente a `nodeadm`. Per ulteriori informazioni sul formato e le opzioni per `nodeConfig.yaml`, consulta the section called "nodeadm dei nodi ibridi".

```
"msg":"Command failed","error":"operation error ec2imds: GetRegion, request canceled,
 context deadline exceeded"
```

Al ruolo IAM dei nodi ibridi mancano le autorizzazioni per l'azione **`eks:DescribeCluster`**

Durante l'esecuzione`nodeadm init`, `nodeadm` tenta di raccogliere informazioni sul cluster EKS richiamando l'azione EKS`DescribeCluster`. Se il ruolo IAM di Hybrid Nodes non dispone dell'autorizzazione per l'`eks:DescribeCluster`azione, devi passare l'endpoint dell'API Kubernetes, il bundle CA del cluster e il servizio IPv4 CIDR nella configurazione del nodo a cui passi durante

l'esecuzione. `nodeadm` `nodeadm init` Per ulteriori informazioni sulle autorizzazioni richieste per il ruolo IAM dei nodi ibridi, consulta the section called “Preparazione delle credenziali”.

```
"msg":"Command failed","error":"operation error EKS: DescribeCluster, https response
 error StatusCode: 403 ... AccessDeniedException"
```

Al ruolo IAM dei nodi ibridi mancano le autorizzazioni per l'azione **`eks:ListAccessEntries`**

Durante l'esecuzione`nodeadm init`, `nodeadm` tenta di verificare se il cluster EKS dispone di una voce di accesso di tipo `HYBRID_LINUX` associata al ruolo IAM di Hybrid Nodes richiamando l'azione EKS. `ListAccessEntries` Se il ruolo IAM di Hybrid Nodes non dispone dell'autorizzazione per l'`eks:ListAccessEntries`azione, è necessario passare il `--skip cluster-access-validation` flag quando si esegue il `nodeadm init` comando. Per ulteriori informazioni sulle autorizzazioni richieste per il ruolo IAM dei nodi ibridi, consulta the section called “Preparazione delle credenziali”.

```
"msg":"Command failed","error":"operation error EKS: ListAccessEntries, https response
 error StatusCode: 403 ... AccessDeniedException"
```

IP del nodo non presente nella rete di nodi remoti CIDR

Durante l'esecuzione`nodeadm init`, potresti riscontrare un errore se l'indirizzo IP del nodo non si trova all'interno della rete di nodi remoti specificata CIDRs. L'aspetto dell'errore sarà simile all'esempio seguente:

```
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]
```

Questo esempio mostra un nodo con IP 10.18.0.1 che tenta di unirsi a un cluster con la rete remota CIDRs 10.0.0.0/16 e 192.168.0.0/16. L'errore si verifica perché 10.18.0.1 non rientra in nessuno degli intervalli.

Conferma di aver configurato correttamente i tuoi indirizzi IP `RemoteNodeNetworks` per includere tutti gli indirizzi IP del nodo. Per ulteriori informazioni sulla configurazione delle reti, consulta the section called “Preparazione della rete”.

- Esegui il comando seguente nella regione in cui si trova il cluster per verificare le configurazioni `RemoteNodeNetwork`. Verifica che i blocchi CIDR elencati nell'output includano l'intervallo IP del nodo e corrispondano ai blocchi CIDR elencati nel messaggio di errore. Se non corrispondono,

conferma che il nome e la regione del cluster `nodeConfig.yaml` corrispondano al cluster desiderato.

```
aws eks describe-cluster --name CLUSTER_NAME --region REGION_NAME --query
 cluster.remoteNetworkConfig.remoteNodeNetworks
```

- Verifica di lavorare con il nodo desiderato:
  - Conferma di essere sul nodo corretto controllando che il nome host e l'indirizzo IP corrispondano a quelli che intendi registrare nel cluster.
  - Verifica che questo nodo si trovi nella rete on-premises corretta (quella il cui intervallo CIDR è stato registrato come `RemoteNodeNetwork` durante la configurazione del cluster).

Se l'IP del nodo non è ancora quello previsto, controlla quanto segue:

- Se utilizzi IAM Roles Anywhere, `kubelet` esegue una ricerca DNS su IAM Roles Anywhere `nodeName` e utilizza un IP associato al nome del nodo, se disponibile. Se conservi le voci DNS per i tuoi nodi, verifica che queste voci puntino all'interno della tua rete di nodi remoti. IPs CIDRs
- Se il tuo nodo ha più interfacce di rete, `kubelet` potresti selezionare un'interfaccia con un indirizzo IP esterno alla rete del nodo remoto CIDRs come impostazione predefinita. Per utilizzare un'interfaccia diversa, specifica il relativo indirizzo IP utilizzando il flag `--node-ip` `kubelet` nel tuo `nodeConfig.yaml`. Per ulteriori informazioni, consulta the section called "nodeadm dei nodi ibridi". Puoi visualizzare le interfacce di rete del tuo nodo e i relativi indirizzi IP eseguendo il seguente comando sul tuo nodo:

```
ip addr show
```

I nodi ibridi non vengono visualizzati nel cluster EKS

Se hai eseguito `nodeadm init` ed è stato completato ma i nodi ibridi non compaiono nel cluster, potrebbero esserci problemi con la connessione di rete tra i nodi ibridi e il piano di controllo EKS, potresti non avere configurato le autorizzazioni richieste per i gruppi di sicurezza o potresti non avere la mappatura richiesta del ruolo IAM di dei nodi ibridi sul controllo degli accessi basato su ruoli (RBAC) di Kubernetes. Puoi avviare il processo di debug controllando lo stato di `kubelet` e i log `kubelet` con i seguenti comandi. Esegui i seguenti comandi dai nodi ibridi che non sono riusciti a entrare a far parte del cluster.

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Impossibile comunicare con il cluster

Se il nodo ibrido non è in grado di comunicare con il piano di controllo del cluster, potresti visualizzare log simili ai seguenti.

```
"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 visualizzi questi messaggi, controlla quanto segue per assicurarti che soddisfi i requisiti dei nodi ibridi descritti in the section called "Preparazione della rete".

- Verifica che il VPC passato al cluster EKS abbia percorsi verso il tuo Transit Gateway (TGW) o Virtual Private Gateway (VGW) per il nodo locale e, facoltativamente, il pod. CIDRs
- Conferma di disporre di un gruppo di sicurezza aggiuntivo per il cluster EKS che disponga di regole in entrata per il nodo locale e, facoltativamente, per il pod. CIDRs CIDRs
- Verifica che il router on-premises sia configurato per consentire il traffico da e verso il piano di controllo EKS.

Non autorizzato

Se il nodo ibrido è in grado di comunicare con il piano di controllo EKS ma non è in grado di registrarsi, potresti visualizzare log simili ai seguenti. Nota che la differenza principale nei messaggi di log riportati di seguito è l'errore `Unauthorized`. Ciò indica che il nodo non è stato in grado di eseguire le proprie attività perché non dispone delle autorizzazioni richieste.

```
"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 visualizzi questi messaggi, controlla quanto segue per assicurarti che soddisfi i dettagli dei requisiti dei nodi in the section called "Preparazione delle credenziali" e the section called "Preparazione dell'accesso ai cluster".

- Verifica che l'identità dei nodi ibridi corrisponda al ruolo IAM dei nodi ibridi previsto. Questo può essere fatto eseguendo `sudo aws sts get-caller-identity` dai tuoi nodi ibridi.
- Verifica che il ruolo IAM dei nodi ibridi disponga delle autorizzazioni richieste.
- Verifica che nel tuo cluster sia presente una voce di accesso EKS per il tuo ruolo IAM Hybrid Nodes o conferma di avere una voce per il `aws-auth` ConfigMap tuo ruolo IAM Hybrid Nodes. Se utilizzi le voci di accesso EKS, conferma che la voce di accesso per il ruolo IAM dei nodi ibridi abbia il tipo di accesso `HYBRID_LINUX`. Se utilizzi il `aws-auth` ConfigMap, conferma che la voce immessa per il ruolo IAM Hybrid Nodes soddisfa i requisiti e la formattazione descritti inthe section called "Preparazione dell'accesso ai cluster".

I nodi ibridi sono registrati nel cluster EKS ma mostrano lo stato **`Not Ready`**

Se i nodi ibridi sono registrati correttamente nel cluster EKS, ma mostrano lo stato `Not Ready`, la prima cosa da verificare è lo stato dell'interfaccia CNI (Container Networking Interface). Se non hai installato una CNI, si prevede che i nodi ibridi abbiano lo status `Not Ready`. Quando una CNI è installata e funziona correttamente, i nodi vengono aggiornati allo stato `Ready`. Se hai tentato di installare una CNI ma non funziona correttamente, consulta the section called "Risoluzione dei problemi CNI dei nodi ibridi" su questa pagina.

Le richieste di firma dei certificati (CSRs) sono bloccate In sospeso

Dopo aver collegato i nodi ibridi al cluster EKS, se noti che ci sono nodi ibridi in sospeso CSRs , significa che i nodi ibridi non soddisfano i requisiti per l'approvazione automatica. CSRs per i nodi ibridi vengono approvati ed emessi automaticamente se i nodi CSRs per i nodi ibridi sono stati creati da un nodo con `eks.amazonaws.com/compute-type: hybrid` etichetta e il CSR ha i seguenti nomi alternativi del soggetto (SANs): almeno una SAN DNS uguale al nome del nodo e l'IP SANs appartengono alla rete del nodo remoto. CIDRs

Il profilo ibrido esiste già

Se hai modificato la configurazione `nodeadm` e tenti di registrare nuovamente il nodo con la nuova configurazione, potresti visualizzare un errore che indica che il profilo ibrido esiste già ma

il suo contenuto è cambiato. Invece di eseguire l'operazione `nodeadm init` tra le modifiche alla configurazione, esegui `nodeadm uninstall` seguito da `nodeadm install` e `nodeadm init`. Ciò garantisce una corretta pulizia con le modifiche alla configurazione.

```
"msg":"Command failed","error":"hybrid profile already exists at /etc/aws/hybrid/config
 but its contents do not align with the expected configuration"
```

Il nodo ibrido non è riuscito a risolvere l'API privata

Dopo l'esecuzione di `nodeadm init`, se visualizzi un errore nei log `kubelet` che indica che non è stato possibile contattare il server dell'API EKS Kubernetes perché si è verificato `no such host`, potresti dover modificare la voce DNS per l'endpoint dell'API EKS Kubernetes nella rete on-premises o a livello di host. Vedi Inoltro delle query DNS in entrata al tuo VPC nella documentazione di Route53. AWS

```
Failed to contact API server when waiting for CSINode publishing: Get ... no such host
```

Impossibile visualizzare i nodi ibridi nella console EKS

Se hai registrato i tuoi nodi ibridi ma non riesci a visualizzarli nella console EKS, controlla le autorizzazioni dell'IAM principale che stai utilizzando per visualizzare la console. L'IAM principale che stai utilizzando deve disporre di autorizzazioni IAM e Kubernetes minime specifiche per visualizzare le risorse nella console. Per ulteriori informazioni, consulta the section called "Accesso alle risorse di cluster".

Risoluzione dei problemi di esecuzione dei nodi ibridi

Se i nodi ibridi che sono stati registrati nel cluster EKS avevano lo stato `Ready` e poi sono passati allo stato `Not Ready`, esistono molti problemi che potrebbero aver contribuito a tale stato di non integrità, ad esempio la mancanza di risorse sufficienti per la CPU, la memoria o lo spazio su disco disponibile del nodo, oppure la disconnessione del nodo dal piano di controllo EKS. Puoi utilizzare i passaggi seguenti per risolvere i problemi dei nodi e, se non riesci a risolvere il problema, contatta il supporto AWS .

Esegui `nodeadm debug` dai nodi ibridi per verificare che i requisiti di rete e le credenziali siano soddisfatti. Per ulteriori informazioni sul comando `nodeadm debug`, consulta the section called "nodeadm dei nodi ibridi".

Ottieni lo stato del nodo

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

Controlla le condizioni e gli eventi del nodo

```
kubectl describe node NODE_NAME
```

Ottieni lo stato del pod

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

Controlla le condizioni e gli eventi del pod

```
kubectl describe pod POD_NAME
```

Controlla i log del pod

```
kubectl logs POD_NAME
```

Controlla i log di **`kubectl`**

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Probe liveness del pod non funziona o i webhook non funzionano

Se le applicazioni, i componenti aggiuntivi o i webhook in esecuzione sui nodi ibridi non si avviano correttamente, potresti riscontrare problemi di rete che bloccano la comunicazione con i pod. Affinché il piano di controllo EKS possa contattare i webhook in esecuzione su nodi ibridi, devi configurare il cluster EKS con una rete di pod remota e disporre di percorsi per il pod CIDR on-premises nella tabella di routing VPC con l'obiettivo come Transit Gateway (TGW), gateway privato virtuale (VPW) o altro gateway che si utilizza per connettere il VPC alla rete on-premises. Per ulteriori informazioni sui requisiti di rete per i nodi ibridi, consulta the section called "Preparazione della rete". Devi inoltre consentire a questo traffico di entrare nel firewall on-premises e assicurarti che il router sia in grado di

instradare correttamente i pod. Consulta the section called "Configurazione di webhook" per ulteriori informazioni sui requisiti per l'esecuzione di webhook su nodi ibridi.

Di seguito è riportato un messaggio di log del pod comune per questo scenario, dove l'indirizzo IP è l'IP del cluster per il servizio Kubernetes.

```
dial tcp <ip-address>:443: connect: no route to host
```

`kubectl logs`o **`kubectl exec`** comandi non funzionanti (comandi **`kubelet`** API)

Se`kubectl attach`,, `kubectl cp` `kubectl exec`, `kubectl logs`, e `kubectl port-forward` i comandi scadono mentre gli altri `kubectl` comandi hanno esito positivo, il problema è probabilmente correlato alla configurazione della rete remota. Questi comandi si connettono tramite il cluster all'endpoint `kubelet` sul nodo. Per ulteriori informazioni, consulta the section called "Endpoint `kubelet`".

Verifica che il nodo IPs e il pod IPs rientrino nella rete di nodi remoti e nella rete di pod remoti CIDRs configurati per il cluster. Usa i comandi seguenti per esaminare le assegnazioni IP.

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

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

Confrontali IPs con la rete remota configurata CIDRs per garantire un routing corretto. Per i requisiti relativi alla configurazione della rete, consulta the section called "Preparazione della rete".

Risoluzione dei problemi CNI dei nodi ibridi

Se riscontri problemi con l'avvio iniziale di Cilium o Calico con nodi ibridi, il più delle volte ciò è dovuto a problemi di rete tra i nodi ibridi o i pod CNI in esecuzione sui nodi ibridi e il piano di controllo EKS. Assicurati che il tuo ambiente soddisfi i requisiti di Preparazione della rete per i nodi ibridi. È utile suddividere il problema in parti.

Configurazione del cluster EKS

- Le configurazioni RemoteNodeNetwork e le RemotePodNetwork configurazioni sono corrette?

Configurazione VPC

- Esistono percorsi per RemoteNodeNetwork e RemotePodNetwork nella tabella di routing VPC con la destinazione del Transit Gateway o del Virtual Private Gateway?

Configurazione del gruppo di sicurezza

Esistono regole in entrata e in uscita per il e? RemoteNodeNetwork RemotePodNetwork

Rete locale

Esistono percorsi e accessi da e verso il piano di controllo EKS e da e verso i nodi ibridi e i pod in esecuzione sui nodi ibridi?

Configurazione CNI

Se si utilizza una rete overlay, la configurazione del pool IP per il CNI corrisponde a quella RemotePodNetwork configurata per il cluster EKS se si utilizzano i webhook?

Il nodo ibrido ha lo stato **`Ready`** senza una CNI installata

Se i nodi ibridi mostrano lo stato `Ready`, ma non è stata installata una CNI sul cluster, è possibile che sui nodi ibridi siano presenti vecchi artefatti CNI. Per impostazione predefinita, quando si disinstallano Cilium e Calico con strumenti come Helm, le risorse su disco non vengono rimosse dalle macchine fisiche o virtuali. Inoltre, le relative definizioni di risorse personalizzate (CRDs) CNIs potrebbero essere ancora presenti nel cluster a partire da una vecchia installazione. Per ulteriori informazioni, consulta le sezioni Eliminazione Cilium ed Eliminazione Calico di the section called "Configurazione della CNI".

Risoluzione dei problemi di Cilium

Se riscontri problemi nell'esecuzione di Cilium su nodi ibridi, consulta i passaggi per la risoluzione dei problemi nella documentazione di Cilium. Le sezioni seguenti trattano i problemi che potrebbero riguardare esclusivamente la distribuzione di Cilium sui nodi ibridi.

Cilium non si avvia

Se gli agenti Cilium eseguiti su ciascun nodo ibrido non si avviano, controlla se sono presenti errori nei log dei pod degli agenti Cilium. L'agente Cilium richiede la connettività all'endpoint dell'API EKS Kubernetes per iniziare. L'avvio dell'agente Cilium avrà esito negativo se questa connettività non è configurata correttamente. In questo caso, nei log del pod dell'agente Cilium verranno visualizzati messaggi di registro simili ai seguenti.

```
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"
```

L'agente Cilium viene eseguito sulla rete host. Il cluster EKS deve essere configurato con `RemoteNodeNetwork` per la connettività Cilium. Verifica di avere un gruppo di sicurezza aggiuntivo per il tuo cluster EKS con una regola in entrata per `RemoteNodeNetwork`, di disporre di percorsi nel VPC per `RemoteNodeNetwork` e che la tua rete on-premises sia configurata correttamente per consentire la connettività al piano di controllo EKS.

Se l'operatore Cilium è in esecuzione e alcuni dei vostri agenti Cilium sono in esecuzione ma non tutti, confermate di avere un pod disponibile IPs da allocare per tutti i nodi del cluster. Configurate la dimensione del vostro Pod allocabile CIDRs quando utilizzate il cluster pool IPAM nella configurazione di Cilium. `clusterPoolIPv4PodCIDRList` La dimensione CIDR per nodo è configurata con l'impostazione `clusterPoolIPv4MaskSize` nella configurazione di Cilium. Per ulteriori informazioni, consulta Expanding the cluster pool nella documentazione di Cilium.

Cilium BGP non funziona

Se utilizzi il piano di controllo di Cilium BGP per pubblicizzare i tuoi pod o gli indirizzi di servizio sulla tua rete on-premises, puoi utilizzare i seguenti comandi Cilium CLI per verificare se BGP sta pubblicizzando i percorsi verso le tue risorse. Per i passaggi per installare la CLI di Cilium, consulta Install the Cilium CLI nella documentazione di Cilium.

Se BGP funziona correttamente, dovresti avere dei nodi ibridi con `established` come stato della sessione nell'output. Potresti dover collaborare con il team di rete per identificare i valori corretti per Local AS, Peer AS e Peer Address del proprio ambiente.

```
cilium bgp peers
```

```
cilium bgp routes
```

Se utilizzate Cilium BGP per pubblicizzare i Servizi con tipo`LoadBalancer`, dovete avere la IPs stessa etichetta sia su voi che su Service, che deve essere utilizzata nel selettore del vostro`CiliumLoadBalancerIPPool`. `CiliumBGPAdvertisement` Di seguito è riportato un esempio. Nota, se utilizzi Cilium BGP per pubblicizzare i Servizi con tipo, i percorsi BGP potrebbero essere interrotti durante il riavvio IPs dell' LoadBalanceragente Cilium. Per ulteriori informazioni, consulta Failure Scenarios nella documentazione di Cilium.

Servizio

```
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 ] }
```

Risoluzione dei problemi di Calico

Se riscontri problemi nell'esecuzione di Calico su nodi ibridi, consulta troubleshooting steps nella documentazione di Calico. Le sezioni seguenti trattano i problemi che potrebbero riguardare esclusivamente la distribuzione di Calico sui nodi ibridi.

La tabella seguente riassume i componenti di Calico e se funzionano di default sul nodo o sulla rete di pod. Se hai configurato Calico per utilizzare NAT per il traffico dei pod in uscita, la rete on-premises deve essere configurata per indirizzare il traffico verso il nodo on-premises CIDR e le tabelle di routing VPC devono essere configurate con una route per il nodo on-premises CIDR con il gateway di transito (TGW) o il gateway privato virtuale (VGW) come destinazione. Se non hai configurato Calico per utilizzare il NAT per il traffico dei pod in uscita, la rete on-premises deve essere configurata per indirizzare il traffico verso il pod on-premises CIDR e le tabelle di routing VPC devono essere configurate con una route per il pod on-premises CIDR con il gateway di transito (TGW) o il gateway privato virtuale (VGW) come destinazione.

Componente	Rete
Server API Calico	Nodo
Controller per Kubernetes di Calico	Pod
Agente del nodo di Calico	Nodo
Calico `typha`	Nodo
Driver del nodo CSI di Calico	Pod
Operatore Calico	Nodo

Le risorse di Calico sono pianificate o in esecuzione su nodi isolati

Per impostazione predefinita, le risorse Calico che non funzionano come unità DaemonSet hanno tolleranze flessibili che ne consentono la pianificazione su nodi isolati che non sono pronti per la pianificazione o l'esecuzione dei pod. È possibile inasprire le tolleranze per le risorse diverse da DaemonSet Calico modificando l'installazione dell'operatore in modo da includere quanto segue.

```
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
```

Risoluzione dei problemi relativi alle credenziali

Sia per le attivazioni ibride AWS SSM che per AWS IAM Roles Anywhere, puoi verificare che le credenziali per il ruolo IAM di Hybrid Nodes siano configurate correttamente sui tuoi nodi ibridi

eseguendo il seguente comando dai tuoi nodi ibridi. Conferma che il nome del nodo e il nome del ruolo IAM dei nodi ibridi sono quelli che ti aspetti.

```
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>"
}
```

Risoluzione dei problemi di AWS Systems Manager (SSM)

Se utilizzi attivazioni ibride AWS SSM per le credenziali dei tuoi nodi ibridi, tieni presente le seguenti directory e artefatti SSM che vengono installati sui tuoi nodi ibridi da. `nodeadm` Per ulteriori informazioni sull'agente SSM, vedere Working with the SSM agent nella AWS Systems Manager User Guide.

Description	Location (Ubicazione)
Agente SSM	Ubuntu - `/snap/amazon-ssm-agent/current/amazon-ssm-agent` RHEL e 023 - AL2 `/usr/bin/amazon-ssm-agent`
Log di SSM Agent	`/var/log/amazon/ssm`
AWS credenziali	`/root/.aws/credentials`
CLI di configurazione SSM	`/opt/ssm/ssm-setup-cli`

Riavvio dell'agente SSM

Alcuni problemi possono essere risolti riavviando l'agente SSM. Puoi utilizzare i comandi seguenti per riavviarlo.

AL2023 e altri sistemi operativi

```
systemctl restart amazon-ssm-agent
```

Ubuntu

```
systemctl restart snap.amazon-ssm-agent.amazon-ssm-agent
```

Verifica la connettività agli endpoint SSM

Conferma di poter effettuare la connessione agli endpoint SSM dai nodi ibridi. Per un elenco degli endpoint SSM, consulta AWS Systems Manager endpoints and quotas. Sostituisci `us-west-2` nel comando seguente con la AWS Regione per l'attivazione ibrida AWS SSM.

```
ping ssm.us-west-2.amazonaws.com
```

Visualizza lo stato della connessione delle istanze SSM registrate

Puoi controllare lo stato della connessione delle istanze registrate con le attivazioni ibride SSM con il seguente comando AWS CLI. Sostituisci l'ID della macchina con quello della macchina dell'istanza.

```
aws ssm get-connection-status --target mi-012345678abcdefgh
```

Mancata corrispondenza del checksum CLI di installazione SSM

Durante l'esecuzione di `nodeadm install`, se riscontri un problema relativo alla mancata corrispondenza del checksum `ssm-setup-cli`, dovresti confermare che non ci siano installazioni SSM precedenti esistenti sul tuo host. Se sul tuo host sono presenti installazioni SSM precedenti, rimuovile ed esegui di nuovo `nodeadm install` per risolvere il problema.

```
Failed to perform agent-installation/on-prem registration: error while verifying
 installed ssm-setup-cli checksum: checksum mismatch with latest ssm-setup-cli.
```

`InvalidActivation` SSM

Se vedi un errore durante la registrazione dell'istanza con AWS SSM, conferma che, e i tuoi dati sono `region` correttiactivationCode. `activationId` `nodeConfig.yaml` La AWS regione del cluster EKS deve corrispondere alla regione dell'attivazione ibrida SSM. Se questi valori non sono configurati correttamente, potresti visualizzare un errore simile al seguente.

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 InvalidActivation
```

`ExpiredTokenException` SSM: Il token di sicurezza incluso nella richiesta è scaduto

Se l'agente SSM non è in grado di aggiornare le credenziali, potresti visualizzare un `ExpiredTokenException`. In questo scenario, se riesci a connetterti agli endpoint SSM dai tuoi nodi ibridi, potresti dover riavviare l'agente SSM per forzare l'aggiornamento delle credenziali.

```
"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"
```

Errore SSM nell'esecuzione del comando registra computer

Se vedi un errore durante la registrazione del computer con SSM, potresti dover eseguire di nuovo `nodeadm install` per assicurarti che tutte le dipendenze SSM siano installate correttamente.

```
"error":"running register machine command: , error: fork/exec /opt/aws/ssm-setup-cli:
 no such file or directory"
```

`ActivationExpired` SSM

Durante l'esecuzione di `nodeadm init`, se vedi un errore durante la registrazione dell'istanza con SSM a causa di un'attivazione scaduta, devi creare una nuova attivazione ibrida SSM, aggiornare `nodeConfig.yaml` con `activationCode` e `activationId` della nuova attivazione ibrida SSM ed eseguire di nuovo `nodeadm init`.

```
"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
```

SSM non è riuscito ad aggiornare le credenziali memorizzate nella cache

Se riscontri un errore nell'aggiornamento delle credenziali memorizzate nella cache, il file `/root/.aws/credentials` potrebbe essere stato eliminato sul tuo host. Controlla innanzitutto l'attivazione ibrida SSM e assicurati che sia attiva e che i nodi ibridi siano configurati correttamente per utilizzare l'attivazione. Controlla l'accesso dell'agente SSM a `/var/log/amazon/ssm` ed esegui di nuovo il comando `nodeadm init` una volta risolto il problema sul lato SSM.

```
"Command failed","error":"operation error SSM: DescribeInstanceInformation, get
 identity: get credentials: failed to refresh cached credentials"
```

Pulisci la SSM

Per rimuovere l'agente SSM dal tuo host, puoi eseguire i seguenti comandi.

```
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
```

Risoluzione problemi AWS IAM Roles Anywhere

Se utilizzi AWS IAM Roles Anywhere per le credenziali dei tuoi nodi ibridi, tieni presente le seguenti directory e artefatti che vengono installati sui tuoi nodi ibridi da. `nodeadm` Per ulteriori informazioni sulla risoluzione dei problemi di identità e accesso a IAM Roles Anywhere, consulta Risoluzione dei problemi di identità e accesso a AWS IAM Roles Anywhere nella Guida per l'utente di AWS IAM Roles Anywhere.

Description	Location (Ubicazione)
CLI IAM Roles Anywhere	`/usr/local/bin/aws_signing_helper`
Ubicazione e nome predefiniti del certificato	`/etc/iam/pki/server.pem`
Posizione e nome predefiniti della chiave	`/etc/iam/pki/server.key`

IAM Roles Anywhere non è riuscito ad aggiornare le credenziali memorizzate nella cache

Se riscontri un errore nell'aggiornamento delle credenziali memorizzate nella cache, esamina il contenuto di `/etc/aws/hybrid/config` e conferma che IAM Roles Anywhere sia stato configurato correttamente nella tua configurazione `nodeadm`. Confermare l'esistenza di `/etc/iam/pki`. Ogni nodo deve avere un certificato e una chiave univoci. Per impostazione predefinita, quando si utilizza IAM Roles Anywhere come fornitore di credenziali, `nodeadm` utilizza `/etc/iam/pki/server.pem` per la posizione e il nome del certificato e `/etc/iam/pki/server.key` per la chiave privata. Potresti dover creare le directory prima di inserire i certificati e le chiavi nelle directory con `sudo mkdir -p /etc/iam/pki`. Puoi verificare il contenuto del certificato con il comando seguente.

```
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 non è autorizzato a eseguire **`sts:AssumeRole`**

Nei log `kubelet`, se riscontri un problema di accesso negato all'operazione `sts:AssumeRole` quando utilizzi IAM Roles Anywhere, controlla la policy di fiducia del tuo ruolo IAM dei nodi ibridi per confermare che il principale del servizio IAM Roles Anywhere sia autorizzato ad assumere il ruolo IAM dei nodi ibridi. Verifica inoltre che l'ancora di fiducia ARN sia configurato correttamente nella policy di fiducia del ruolo IAM dei nodi ibridi e che il ruolo IAM dei nodi ibridi sia aggiunto al tuo profilo IAM Roles Anywhere.

```
could not get token: AccessDenied: User: ... is not authorized to perform:
 sts:AssumeRole on resource: ...
```

IAM Roles Anywhere non è autorizzato a impostare **`roleSessionName`**

Nei log `kubelet`, se riscontri un problema di accesso negato durante l'impostazione di `roleSessionName`, conferma che l'impostazione `acceptRoleSessionName` sia su true per il tuo profilo IAM Roles Anywhere.

```
AccessDeniedException: Not authorized to set roleSessionName
```

Risoluzione dei problemi del sistema operativo

RHEL

Errori di registrazione del gestore dei diritti o degli entitlement

Se `nodeadm install` è in esecuzione e non è possibile installare le dipendenze dei nodi ibridi a causa di problemi di registrazione degli entitlement, assicurati di aver impostato correttamente il nome utente e la password di Red Hat sull'host.

```
This system is not registered with an entitlement server
```

Ubuntu

GLIBC non trovato

Se utilizzi Ubuntu per il sistema operativo e IAM Roles Anywhere per il provider di credenziali con nodi ibridi e riscontri un problema relativo a GLIBC non trovato, puoi installare la dipendenza manualmente per risolverlo.

```
GLIBC_2.32 not found (required by /usr/local/bin/aws_signing_helper)
```

Esegui i comandi riportati qui di seguito per installare la dipendenza:

```
ldd --version
sudo apt update && apt install libc6
sudo apt install glibc-source
```

Bottlerocket

Se hai abilitato il container di amministrazione Bottlerocket, puoi accedervi con SSH per il debug avanzato e la risoluzione dei problemi con privilegi elevati. Le seguenti sezioni contengono comandi che devono essere eseguiti nel contesto dell'host Bottlerocket. Una volta che sei nel container di amministrazione, puoi eseguire `sheltie` per ottenere una shell root completa nell'host Bottlerocket.

```
sheltie
```

Puoi anche eseguire i comandi nelle seguenti sezioni dalla shell del container di amministrazione anteponendo a ogni comando il prefisso `sudo chroot /.bottlerocket/rootfs`.

```
sudo chroot /.bottlerocket/rootfs <command>
```

Utilizzo di logdog per la raccolta dei log

Bottlerocket fornisce l'utilità `logdog` per raccogliere in modo efficiente log e informazioni di sistema per la risoluzione dei problemi.

```
logdog
```

L'utilità `logdog` raccoglie i log da varie posizioni su un host Bottlerocket e li combina in un tarball. Per impostazione predefinita, il tarball viene creato in `/var/log/support/bottlerocket-logs.tar.gz` ed è accessibile dai contenitori host in `/.bottlerocket/support/bottlerocket-logs.tar.gz`.

Accesso ai log di sistema con journalctl

Puoi controllare lo stato dei vari servizi di sistema come `kubelet`, `containerd`, ecc. e visualizzarne i registri con i seguenti comandi. Il flag `-f` seguirà i log in tempo reale.

Per controllare lo stato del servizio `kubelet` e recuperare i log `kubelet`, puoi eseguire:

```
systemctl status kubelet
journalctl -u kubelet -f
```

Per controllare lo stato del servizio `containerd` e recuperare i log per l'istanza orchestrata `containerd`, puoi eseguire:

```
systemctl status containerd
journalctl -u containerd -f
```

Per controllare lo stato del servizio `host-containerd` e recuperare i log per l'istanza host `containerd`, puoi eseguire:

```
systemctl status host-containerd
journalctl -u host-containerd -f
```

Per recuperare i log per i container bootstrap e i container host, puoi eseguire:

```
journalctl _COMM=host-ctr -f
```

Utilizzo dell'archiviazione di dati delle applicazioni per il tuo cluster

Puoi utilizzare una gamma di servizi di AWS storage con Amazon EKS per le esigenze di storage delle tue applicazioni. Grazie a un' AWS ampia gamma di driver CSI (Container Storage Interface) supportati, puoi utilizzare facilmente Amazon EBS, Amazon S3, Amazon EFS, Amazon FSX e Amazon File Cache per le esigenze di storage delle tue applicazioni in esecuzione su Amazon EKS. Per gestire i backup del tuo cluster Amazon EKS, consulta Supporto di AWS backup per Amazon EKS.

Questo capitolo illustra le opzioni di archiviazione per i cluster Amazon EKS.

Argomenti

- Utilizzare il volume di archiviazione Kubernetes con Amazon EBS
- Utilizzo dell'archiviazione di file system elastici con Amazon EFS
- Usa l'archiviazione delle app ad alte prestazioni con Amazon FSx per Lustre
- Usa l'archiviazione delle app ad alte prestazioni con FSx per NetApp ONTAP
- Usa l'archiviazione di dati con Amazon FSx per OpenZFS
- Riduci al minimo la latenza con Amazon File Cache
- Accedere agli oggetti Amazon S3 con il driver CSI Mountpoint per Amazon S3
- Abilitare la funzionalità snapshot per i volumi CSI

Utilizzare il volume di archiviazione Kubernetes con Amazon EBS

> **Note**
>
> Novità: la modalità automatica di Amazon EKS automatizza le attività di routine per l'archiviazione a blocchi. Informazioni su come the section called "Implementazione di un carico di lavoro stateful".

Il driver Container Storage Interface (CSI) di Amazon Elastic Block Store (Amazon EBS) gestisce il ciclo di vita dei volumi Amazon EBS come archiviazione per i volumi Kubernetes creati. Il driver

CSI di Amazon EBS crea volumi Amazon EBS per questi tipi di volumi Kubernetes: volumi generici effimeri e volumi persistenti.

Considerazioni

- Non è necessario installare il controller CSI di Amazon EBS sui cluster della modalità automatica EKS.
- Non è possibile montare volumi Amazon EBS sui pod Fargate.
- Puoi eseguire il controller Amazon EBS CSI sui nodi Fargate, ma il nodo Amazon EBS CSI può essere eseguito `DaemonSet` solo su istanze Amazon. EC2
- I volumi Amazon EBS e il driver CSI di Amazon EBS non sono compatibili con Amazon EKS Hybrid Nodes.
- Verrà fornito supporto per la versione più recente del componente aggiuntivo e per una versione precedente. I bug o le vulnerabilità rilevati nella versione precedente verranno sottoposti a backport alla versione precedente in una nuova versione secondaria.
- La modalità automatica EKS richiede classi di archiviazione per utilizzare `ebs.csi.eks.amazonaws.com` come provisioner. Il driver CSI di Amazon EBS standard (`ebs.csi.aws.com`) gestisce i propri volumi separatamente. Per utilizzare i volumi esistenti con la modalità automatica EKS, eseguirne la migrazione tramite snapshot del volume verso una classe di archiviazione che utilizza il provisioner della modalità automatica.

> **Important**
>
> Per utilizzare la funzionalità snapshot del driver CSI di Amazon EBS, è prima necessario installare il controller di snapshot CSI. Per ulteriori informazioni, consulta the section called "Controller snapshot CSI".

Prerequisiti

- Un cluster esistente. Per visualizzare la versione della piattaforma richiesta, esegui il comando seguente.

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- Il driver CSI EBS richiede le autorizzazioni IAM. AWS

- AWS suggerisce di utilizzare EKS Pod Identities. Per ulteriori informazioni, consulta the section called "Panoramica sulla configurazione delle associazioni EKS Pod Identity".
- Per ulteriori informazioni sui ruoli IAM per gli account di servizio, consultare the section called "Gestore di identità OIDC IAM".

Passaggio 1: Creazione di un ruolo IAM

Il plug-in Amazon EBS CSI richiede le autorizzazioni IAM per effettuare chiamate per tuo AWS APIs conto. Se non viene soddisfatto questo prerequisito, il tentativo di installare il componente aggiuntivo e l'esecuzione di `kubectl describe pvc` riporteranno `failed to provision volume with StorageClass` insieme a un errore `could not create volume in EC2: UnauthorizedOperation`. Per ulteriori informazioni, consulta Configurazione dell'autorizzazione driver su GitHub.

> **Note**
>
> I pod avranno accesso alle autorizzazioni assegnate al ruolo IAM, a meno che non si blocchi l'accesso a IMDS. Per ulteriori informazioni, consulta the section called "Best practice".

La procedura seguente mostra come creare un ruolo IAM e allegare ad esso la policy AWS gestita. Per implementare questa procedura, è possibile utilizzare uno dei seguenti strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

> **Note**
>
> È possibile creare una policy autogestita con autorizzazioni limitate. Rivedere `AmazonEBSCSIDriverPolicy` e creare una policy IAM personalizzata con autorizzazioni ridotte.

> **Note**
>
> I passaggi specifici di questa procedura sono stati scritti per l'utilizzo del driver come componente aggiuntivo di Amazon EKS. Per utilizzare il driver come componente aggiuntivo autogestito occorre una procedura differente. Per ulteriori informazioni, consulta Configurare le autorizzazioni dei driver su GitHub.

`eksctl`

1. Crea un ruolo IAM e allega una policy. AWS mantiene una politica AWS gestita oppure puoi creare la tua politica personalizzata. È possibile creare un ruolo IAM e allegare la policy AWS gestita con il seguente comando. Sostituisci *`my-cluster`* con il nome del cluster. Il comando distribuisce uno AWS CloudFormation stack che crea un ruolo IAM e vi allega la policy 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
```

2. È possibile saltare questo passaggio se non si utilizza una chiave KMS personalizzata. Nel caso se ne utilizzi una per la crittografia dei volumi Amazon EBS, personalizzare il ruolo IAM secondo necessità. Ad esempio, completa le seguenti operazioni:

   a. Copia e incolla il codice seguente in un nuovo file `kms-key-for-encryption-on-ebs.json`. Sostituisci *`custom-key-arn`* con l'ARN della chiave KMS personalizzata.

```
{
      "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. Crea la policy. È possibile modificare *KMS_Key_For_Encryption_On_EBS_Policy* con un nome diverso. Se esegui questa operazione, tuttavia, assicurati di modificare il nome anche nelle fasi successive.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Collega la policy IAM al ruolo con il comando seguente. Sostituisci *111122223333* con l'ID del tuo account.

```
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 di gestione AWS

1. Apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Select trusted entity (Seleziona entità attendibile), esegui le operazioni seguenti:
   a. Nella sezione Tipo di identità attendibile, scegli Identità Web.
   b. Per Identity provider (Provider di identità), scegli URL del provider OpenID Connect per il cluster (come mostrato nella scheda Overview (Panoramica) in Amazon EKS).
   c. Per Pubblico, scegli `sts.amazonaws.com`.
   d. Scegli Next (Successivo).
5. Nella pagina Add permissions (Aggiungi autorizzazioni), esegui le operazioni seguenti:
   a. Nella casella Filtra policy, inserisci `AmazonEBSCSIDriverPolicy`.
   b. Seleziona la casella di controllo a sinistra della `AmazonEBSCSIDriverPolicy` restituita dalla ricerca.
   c. Scegli Next (Successivo).
6. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:
   a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio *`AmazonEKS_EBS_CSI_DriverRole`*.
   b. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.
   c. Scegli Crea ruolo.
7. Dopo aver creato il ruolo, sceglilo nella console in modo da aprirlo per la modifica.
8. Scegli la scheda Relazioni di attendibilità e quindi Modifica policy di attendibilità.
9. Trova la riga simile alla seguente:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

Inserisci una virgola alla fine della riga precedente, quindi aggiungine una seguente dopo quella precedente. Sostituisci *region-code* con la AWS regione in cui si trova il cluster. Sostituire *EXAMPLED539D4633E53DE1B71EXAMPLE* con l'ID del gestore OIDC del cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:ebs-csi-controller-sa"
```

10. Scegli Aggiorna policy per concludere.
11. Se utilizzi una chiave KMS personalizzata per la crittografia dei volumi Amazon EBS, personalizza il ruolo IAM secondo necessità. Ad esempio, completa le seguenti operazioni:
    a. Nel pannello di navigazione a sinistra, seleziona Policy.
    b. Nella pagina Policy, scegli Crea policy.
    c. Nella pagina Crea policy, scegli la scheda JSON.
    d. Copia e incolla il codice seguente nell'editor, sostituendo *custom-key-arn* con l'ARN della chiave KMS personalizzata.

```
{
      "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. Scegli Successivo: Tag.

f. Nella pagina Add tags (optional) (Aggiungi tag, facoltativo), seleziona Next: Review (Successivo: esamina).

g. Per Nome immettere un nome univoco per la policy (ad esempio *KMS_Key_For_Encryption_On_EBS_Policy*).

h. Scegli Crea policy.

i. Nel pannello di navigazione a sinistra, seleziona Ruoli.

j. Sceglila ***AmazonEKS_EBS_CSI_DriverRole***nella console per aprirla e modificarla.

k. Dall'elenco a discesa Aggiungi autorizzazioni, scegli Collega policy.

l. Nella casella Filtra policy, inserisci *KMS_Key_For_Encryption_On_EBS_Policy*.

m. Seleziona la casella di controllo a sinistra della *KMS_Key_For_Encryption_On_EBS_Policy* restituita dalla ricerca.

n. Scegli Collega policy.

AWS CLI

1. Visualizzare l'URL del gestore OIDC del cluster. Sostituire *my-cluster* con il nome del cluster. Se l'output dal comando è `None`, rivedi i Prerequisiti.

   ```
   aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
   output text
   ```

   Di seguito viene riportato un output di esempio:

   ```
   https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
   ```

2. Crea il ruolo IAM concedendo l'autorizzazione per l'operazione `AssumeRoleWithWebIdentity`.

a. Copiare i contenuti seguenti in un file denominato `aws-ebs-csi-driver-trust-policy.json`. Sostituisci *111122223333* con l'ID del tuo account. Sostituire *EXAMPLED539D4633E53DE1B71EXAMPLE* e *region-code* con i valori restituiti nella fase precedente.

```
{
      "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. Creare il ruolo. È possibile modificare *AmazonEKS_EBS_CSI_DriverRole* con un nome diverso. Se modifichi il valore, assicurati di modificarlo anche nelle fasi successive.

```
aws iam create-role \
      --role-name AmazonEKS_EBS_CSI_DriverRole \
      --assume-role-policy-document file://"aws-ebs-csi-driver-trust-policy.json"
```

3. Allega una politica. AWS mantiene una politica AWS gestita oppure è possibile creare una politica personalizzata. AWS Associa la politica gestita al ruolo con il seguente comando.

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

4. Se utilizzi una chiave KMS personalizzata per la crittografia dei volumi Amazon EBS, personalizza il ruolo IAM secondo necessità. Ad esempio, completa le seguenti operazioni:

   a. Copia e incolla il codice seguente in un nuovo file `kms-key-for-encryption-on-ebs.json`. Sostituisci *`custom-key-arn`* con l'ARN della chiave KMS personalizzata.

```
{
      "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. Crea la policy. È possibile modificare *`KMS_Key_For_Encryption_On_EBS_Policy`* con un nome diverso. Se esegui questa operazione, tuttavia, assicurati di modificare il nome anche nelle fasi successive.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Collega la policy IAM al ruolo con il comando seguente. Sostituisci *111122223333* con l'ID del tuo account.

```
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
```

Ora che è stato creato il ruolo IAM del driver CSI di Amazon EBS, è possibile passare alla sezione successiva. Quando si implementa il componente aggiuntivo con il ruolo IAM, questo crea e viene configurato per utilizzare un account di servizio denominato `ebs-csi-controller-sa`. L'account di servizio è associato a un `clusterrole` Kubernetes a cui sono assegnate le autorizzazioni Kubernetes richieste.

Fase 2: ottenere il driver CSI di Amazon EBS

Consigliamo di installare il driver CSI di Amazon EBS attraverso il componente aggiuntivo di Amazon EKS per migliorare la sicurezza e ridurre le attività. Per aggiungere un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS".

Important

Prima di aggiungere il driver Amazon EBS come componente aggiuntivo di Amazon EKS, verificare che nel cluster non sia installata una versione autogestita del driver. In tal caso, vediDisinstallazione di un driver Amazon EBS CSI autogestitosul GitHub.

Note

Per impostazione predefinita, il ruolo RBAC utilizzato dal CSI di EBS dispone delle autorizzazioni per modificare i nodi per supportare la funzionalità di rimozione

delle macchie. A causa delle limitazioni di Kubernetes RBAC, ciò consente anche di modificare qualsiasi altro nodo del cluster. Il grafico Helm ha un parametro (`node.serviceAccount.disableMutation`) che disabilita la modifica delle autorizzazioni RBAC del nodo per l'account del servizio. ebs-csi-node Se abilitate, le funzioni del driver come la rimozione delle macchie non funzioneranno.

In alternativa, se desideri un'installazione autogestita del driver CSI di Amazon EBS, consulta Installazione su. GitHub

Fase 3: implementazione di un'applicazione di esempio

Puoi implementare diverse app di esempio e modificarle in base alle tue esigenze. Per ulteriori informazioni, consulta Kubernetes Examples su. GitHub

Utilizzo dell'archiviazione di file system elastici con Amazon EFS

Amazon Elastic File System (Amazon EFS) fornisce un'archiviazione di file serverless e completamente elastica in modo da poter condividere i dati dei file senza dover fornire o gestire la capacità e le prestazioni di archiviazione. Il driver Amazon EFS Container Storage Interface (CSI) fornisce un'interfaccia CSI che consente ai cluster Kubernetes in esecuzione di AWS gestire il ciclo di vita dei file system Amazon EFS. Questo argomento illustra come implementare il driver CSI per Amazon EBS nel cluster Amazon EKS.

Considerazioni

- Il driver CSI per Amazon EFS non è compatibile con le immagini di container basate su Windows.
- Non è possibile utilizzare il provisioning dinamico dei volumi persistenti con nodi Fargate, ma è possibile utilizzare il provisioning statico.
- Il provisioning dinamico richiede un driver versione 1.2 o successiva. È possibile eseguire il provisioning statico di volumi persistenti utilizzando la versione `1.1` del driver su qualsiasi versione del cluster Amazon EKS supportata (consultare la pagina Amazon EKS supported versions).
- La versione 1.3.2 o successiva di questo driver supporta l'architettura Arm64, incluse le istanze basate su Amazon EC2 Graviton.
- La versione 1.4.2 o successiva di questo driver supporta l'uso di FIPS per il montaggio di file system.

- Osserva le quote delle risorse per Amazon EFS. Ad esempio, è possibile creare una quota di 1.000 punti di accesso per ogni file system Amazon EFS. Per ulteriori informazioni, consulta le Quote di risorse di Amazon EFS che non puoi modificare.
- A partire dalla versione 2.0.0, questo driver è passato dall'utilizzo di `stunnel` a `efs-proxy` per le connessioni TLS. Quando viene utilizzato `efs-proxy`, aprirà un numero di thread pari a uno più il numero di core del nodo su cui è in esecuzione.
- Il driver CSI per Amazon EFS non è compatibile con Amazon EKS Hybrid Nodes.

Prerequisiti

- Il driver CSI Amazon EFS necessita delle autorizzazioni AWS Identity and Access Management (IAM).
  - AWS suggerisce di utilizzare EKS Pod Identities. Per ulteriori informazioni, consulta the section called "Panoramica sulla configurazione delle associazioni EKS Pod Identity".
  - Per informazioni sui ruoli IAM per gli account di servizio e su come configurare un gestore OpenID Connect (OIDC) IAM per il cluster, consultare the section called "Gestore di identità OIDC IAM".
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":

> **Note**
>
> Un pod in esecuzione su Fargate monta automaticamente un file system Amazon EFS senza bisogno della procedura di installazione manuale del driver.

Passaggio 1: Creazione di un ruolo IAM

Il driver CSI per Amazon EFS richiede le autorizzazioni IAM per interagire con il file system. Crea un ruolo IAM e allega ad esso la policy AWS gestita richiesta. Per implementare questa procedura, è possibile utilizzare uno dei seguenti strumenti:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

> **Note**
>
> I passaggi specifici di questa procedura sono stati scritti per l'utilizzo del driver come componente aggiuntivo di Amazon EKS. Per informazioni dettagliate sulle installazioni autogestite, consulta Set up driver permission su GitHub.

`eksctl`

Se si utilizza Pod Identity

Eseguire i seguenti comandi per creare un ruolo IAM e un'associazione Pod Identity con `eksctl`. Sostituisci `my-cluster` con il nome del cluster. È anche possibile sostituire `AmazonEKS_EFS_CSI_DriverRole` con un nome diverso.

```
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 si utilizzano ruoli IAM per gli account di servizio

Eseguire i seguenti comandi per creare un ruolo IAM con `eksctl`. Sostituisci `my-cluster` con il nome del cluster. È anche possibile sostituire `AmazonEKS_EFS_CSI_DriverRole` con un nome diverso.

```
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 di gestione AWS

Esegui quanto segue per creare un ruolo IAM con Console di gestione AWS.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   a. Se si utilizza EKS Pod Identity:
      i. Nella sezione Tipo di entità affidabile, scegli AWS servizio.
      ii. Nel menu a discesa Servizio o caso d'uso, scegliere EKS.
      iii. Nella sezione Caso d'uso, scegliere EKS - Pod Identity.
      iv. Scegli Next (Successivo).

b. Se si utilizzano ruoli IAM per gli account di servizio:
   i. Nella sezione Tipo di identità attendibile, scegli Identità Web.
   ii. Per Identity provider (Provider di identità), scegli URL del provider OpenID Connect per il cluster (come mostrato nella scheda Overview (Panoramica) in Amazon EKS).
   iii. Per Pubblico, scegli `sts.amazonaws.com`.
   iv. Scegli Next (Successivo).

5. Nella pagina Add permissions (Aggiungi autorizzazioni), esegui le operazioni seguenti:
   a. Nella casella Filtra policy, inserisci `AmazonEFSCSIDriverPolicy`.
   b. Seleziona la casella di controllo a sinistra della `AmazonEFSCSIDriverPolicy` restituita dalla ricerca.
   c. Scegli Next (Successivo).
6. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:
   a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKS_EFS_CSI_DriverRole`.
   b. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.
   c. Scegli Crea ruolo.
7. Dopo aver creato il ruolo:
   a. Se si utilizza EKS Pod Identity:
      i. Aprire la Console Amazon EKS.
      ii. Nel riquadro di navigazione a sinistra, selezionare Cluster, quindi selezionare il nome del cluster per cui configurare l'associazione EKS Pod Identity.
      iii. Scegliere la scheda Accesso.
      iv. In Associazioni Pod Identity, scegliere Crea.
      v. Scegliere il menu a discesa Ruolo IAM e selezionare il ruolo appena creato.
      vi. Scegliere il campo Namespace Kubernetes e inserire `kube-system`.
      vii. Scegliere il campo Account del servizio Kubernetes e inserire `efs-csi-controller-sa`.
      viii. Scegli Create (Crea).
      ix. Per ulteriori informazioni sulla creazione di associazioni Pod Identity, consultare the section called "Crea un'associazione Pod Identity (AWS Console)".

b. Se si utilizzano ruoli IAM per gli account di servizio:

i. Scegliere il ruolo per aprirlo per la modifica.

ii. Scegli la scheda Relazioni di attendibilità e quindi Modifica policy di attendibilità.

iii. Trova la riga simile alla seguente:

```
"oidc.eks.region-code.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud":
 "sts.amazonaws.com"
```

Aggiungi la riga seguente sopra la riga precedente. Sostituiscilo `<region-code>` con la AWS regione in cui si trova il cluster. Sostituire `<EXAMPLED539D4633E53DE1B71EXAMPLE>` con l'ID del gestore OIDC del cluster.

```
"oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-*",
```

iv. Modifica l'operatore `Condition` da `"StringEquals"` a `"StringLike"`.

v. Scegli Aggiorna policy per concludere.

AWS CLI

Esegui i seguenti comandi per creare un ruolo IAM con AWS CLI.

Se si utilizza Pod Identity

1. Creare il ruolo IAM che concede le operazioni `AssumeRole` e `TagSession` al servizio `pods.eks.amazonaws.com`.

a. Copiare i seguenti contenuti in un file denominato `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. Crea il ruolo. Sostituisci `my-cluster` con il nome del cluster. È anche possibile sostituire `AmazonEKS_EFS_CSI_DriverRole` con un nome diverso.

```
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. Allega la policy AWS gestita richiesta al ruolo con il comando seguente.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

3. Eseguire il seguente comando per creare l'associazione Pod Identity. Sostituire `arn:aws:iam::<111122223333>:role/my-role` con il ruolo creato nei passaggi precedenti.

```
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. Per ulteriori informazioni sulla creazione di associazioni Pod Identity, consultare the section called "Crea un'associazione Pod Identity (AWS Console)".

Se si utilizzano ruoli IAM per gli account di servizio

1. Visualizzare l'URL del gestore OIDC del cluster. Sostituisci `my-cluster` con il nome del cluster. È anche possibile sostituire `AmazonEKS_EFS_CSI_DriverRole` con un nome diverso.

```
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
```

Di seguito viene riportato un output di esempio.

```
https://oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>
```

Se l'output dal comando è None, rivedi i Prerequisiti.

2. Crea il ruolo IAM che concede l'autorizzazione per l'operazione AssumeRoleWithWebIdentity.

   a. Copia i seguenti contenuti in un file denominato aws-efs-csi-driver-trust-policy.json. Sostituisci <111122223333> con l'ID del tuo account. Sostituire <EXAMPLED539D4633E53DE1B71EXAMPLE> e <region-code> con i valori restituiti nella fase precedente.

```
{
  "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. Creare il ruolo.

```
aws iam create-role \
```

```
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy.json"
```

3. Allega la politica AWS gestita richiesta al ruolo con il comando seguente.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

Fase 2: ottenere il driver CSI per Amazon EFS

Consigliamo di installare il driver Amazon EFS CSI attraverso il componente aggiuntivo di Amazon EKS. Per aggiungere un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS". Se non riesci a utilizzare il componente aggiuntivo Amazon EKS, ti consigliamo di segnalare un problema sul motivo per cui non puoi farlo all'archivio della roadmap GitHub di Containers.

> **Important**
>
> Prima di aggiungere il driver Amazon EFS come componente aggiuntivo di Amazon EKS, verificare che nel cluster non sia installata una versione autogestita del driver. In tal caso, consulta Disinstallazione del driver Amazon EFS CSI su. GitHub

In alternativa, se desideri un'installazione autogestita del driver Amazon EFS CSI, consulta Installazionesu GitHub.

Fase 3: Creazione di un file system Amazon EFS

Per creare un file system Amazon EFS, consulta Creazione di un sistema di file Amazon EFS per Amazon EKS su GitHub.

Fase 4: implementazione di un'applicazione di esempio

Puoi implementare diverse app di esempio e modificarle in base alle tue esigenze. Per ulteriori informazioni, consulta Esempi su. GitHub

Usa l'archiviazione delle app ad alte prestazioni con Amazon FSx per Lustre

Il driver Container Storage Interface (CSI) per Amazon FSx per Lustre mette a disposizione un'interfaccia CSI che permette ai cluster Amazon EKS di gestire il ciclo di vita dei file system Amazon FSx per Lustre. Per ulteriori informazioni, consulta Amazon FSx for Lustre User Guide.

Per informazioni su come implementare il driver CSI per Amazon FSx per Lustre nel cluster Amazon EKS e come verificarne il funzionamento, consulta the section called "Implementare il driver".

Implementa il driver FSx for Lustre

Questo argomento mostra come distribuire il driver CSI FSx for Lustre nel cluster Amazon EKS e verificarne il funzionamento. Ti consigliamo di utilizzare sempre la versione più recente del driver. Per le versioni disponibili, consulta la matrice di compatibilità delle specifiche CSI su. GitHub

> **Note**
>
> Il driver non è supportato su Fargate o Amazon EKS Hybrid Nodes.

Per descrizioni dettagliate dei parametri disponibili ed esempi completi che dimostrano le caratteristiche del driver, consultate il progetto driver FSx for Lustre Container Storage Interface (CSI) su. GitHub

Prerequisiti

- Un cluster esistente.
- Il componente aggiuntivo Amazon FSx CSI Driver EKS richiede l'agente EKS Pod Identity per l'autenticazione. Senza questo componente, il componente aggiuntivo fallirà con l'errore `Amazon EKS Pod Identity agent is not installed in the cluster`, impedendo le operazioni di volume. Installa l'agente Pod Identity prima o dopo la distribuzione del componente aggiuntivo FSx CSI Driver. Per ulteriori informazioni, consulta the section called "Configurazione dell'agente".
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione

rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.

- La versione `0.215.0` o quelle successive dello strumento a riga di comando `eksctl` deve essere installata sul dispositivo o nella AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":

Passaggio 1: Creazione di un ruolo IAM

Il plug-in Amazon FSx CSI richiede le autorizzazioni IAM per effettuare chiamate per tuo AWS APIs conto.

> **Note**
>
> I pod avranno accesso alle autorizzazioni assegnate al ruolo IAM, a meno che non si blocchi l'accesso a IMDS. Per ulteriori informazioni, consulta the section called "Best practice".

La procedura seguente mostra come creare un ruolo IAM e allegare ad esso la policy AWS gestita.

1. Crea un ruolo IAM e collega la policy AWS gestita con il seguente comando. Sostituisci `my-cluster` con il nome del cluster che desideri utilizzare. Il comando distribuisce uno AWS CloudFormation stack che crea un ruolo IAM e vi allega la policy IAM.

```
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
```

Visualizzerai diverse righe di output quando l'account del servizio viene creato. Le ultime righe dell'output sono simili a quelle riportate di seguito.

```
[#]  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"
```

Annota il nome dello AWS CloudFormation stack che è stato distribuito. Nell'output di esempio precedente, lo stack è denominato `eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa`.

Ora che hai creato il ruolo IAM del driver Amazon FSx CSI, puoi passare alla sezione successiva. Quando si implementa il componente aggiuntivo con il ruolo IAM, questo crea e viene configurato per utilizzare un account di servizio denominato `fsx-csi-controller-sa`. L'account di servizio è associato a un `clusterrole` Kubernetes a cui sono assegnate le autorizzazioni Kubernetes richieste.

Fase 2: Installare il driver Amazon FSx CSI

Ti consigliamo di installare il driver Amazon FSx CSI tramite il componente aggiuntivo Amazon EKS per migliorare la sicurezza e ridurre la quantità di lavoro. Per aggiungere un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS".

> **Important**
>
> Le installazioni preesistenti di driver Amazon FSx CSI nel cluster possono causare errori di installazione dei componenti aggiuntivi. Quando tenti di installare la versione del componente aggiuntivo Amazon EKS mentre esiste un driver FSx CSI non EKS, l'installazione avrà esito

negativo a causa di conflitti di risorse. Usa il flag `OVERWRITE` durante l'installazione per risolvere il problema.

```
aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
 --resolve-conflicts OVERWRITE
```

In alternativa, se desideri un'installazione autogestita del driver Amazon FSx CSI, consulta Installation on. GitHub

Passaggio 3: implementazione di una classe di archiviazione, una dichiarazione di volume persistente e un'app di esempio

Questa procedura utilizza il GitHub repository di driver FSx for Lustre Container Storage Interface (CSI) per utilizzare un volume Lustre con provisioning dinamico. FSx

1. Annota il gruppo di sicurezza per il cluster. Puoi vederlo nella Console di gestione AWS sezione Rete o utilizzando il seguente comando AWS CLI. Sostituisci `my-cluster` con il nome del cluster che desideri utilizzare.

   ```
   aws eks describe-cluster --name my-cluster --query
    cluster.resourcesVpcConfig.clusterSecurityGroupId
   ```

2. Crea un gruppo di sicurezza per il tuo FSx file system Amazon in base ai criteri indicati in Amazon VPC Security Groups nella Amazon FSx for Lustre User Guide. Per il VPC, seleziona il VPC del cluster come mostrato nella sezione Reti. Per i "gruppi di sicurezza associati ai client Lustre", usa il gruppo di sicurezza del cluster. Non specificare alcun valore nelle regole in uscita per consentire Tutto il traffico.
3. Scarica il manifesto della classe di storage con il comando seguente.

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
   examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
   ```

4. Modifica la sezione dei parametri del file `storageclass.yaml`. Sostituisci ogni valore esemplificativo con i tuoi valori.

   ```
   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`**— L'ID di sottorete in cui deve essere creato il file system Amazon FSx for Lustre. Amazon FSx for Lustre non è supportato in tutte le zone di disponibilità. Apri la console Amazon FSx for Lustre all'indirizzo per https://console.aws.amazon.com/fsx/ confermare che la sottorete che desideri utilizzare si trova in una zona di disponibilità supportata. La sottorete può includere i nodi oppure può essere una sottorete o un VPC differente:
  - Puoi verificare la presenza di sottoreti di nodi in Console di gestione AWS selezionando il gruppo di nodi nella sezione Elaborazione.
  - Se la sottorete specificata non è la stessa sottorete in cui sono presenti i nodi, è VPCs necessario essere connessi e assicurarsi di avere le porte necessarie aperte nei gruppi di sicurezza.
- **`securityGroupIds`** – l'ID del gruppo di sicurezza creato per il file system.
- **`deploymentType`** (facoltativo): il tipo di implementazione del file system. I valori validi sono `SCRATCH_1`, `SCRATCH_2`, `PERSISTENT_1` e `PERSISTENT_2`. Per ulteriori informazioni sui tipi di distribuzione, consulta Create your Amazon FSx for Lustre file system.
- altri parametri (facoltativo): per informazioni sugli altri parametri, consulta Modifica StorageClass su GitHub.

5. Crea il manifesto della classe di storage.

```
kubectl apply -f storageclass.yaml
```

Di seguito viene riportato un output di esempio:

```
storageclass.storage.k8s.io/fsx-sc created
```

6. Scaricare il manifesto della dichiarazione di volume persistente.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. (Facoltativo) Modificare il file `claim.yaml`. Cambia `1200Gi` in uno dei valori di incremento riportati di seguito in base ai tuoi requisiti di archiviazione e al `deploymentType` selezionato in un passaggio precedente.

```
storage: 1200Gi
```

   - `SCRATCH_2` e `PERSISTENT` – `1.2 TiB`, `2.4 TiB` o incrementi di 2,4 TiB su 2,4 TiB.
   - `SCRATCH_1` – `1.2 TiB`, `2.4 TiB`, `3.6 TiB` o incrementi di 3,6 TiB su 3,6 TiB.

8. Creare la dichiarazione di volume persistente.

```
kubectl apply -f claim.yaml
```

   Di seguito viene riportato un output di esempio:

```
persistentvolumeclaim/fsx-claim created
```

9. Verificare che il file system sia stato sottoposto a provisioning.

```
kubectl describe pvc
```

   Di seguito viene riportato un output di esempio:

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
[...]
```

   > **Note**
   >
   > `Status` può apparire `Pending` per 5-10 minuti, prima di passare a `Bound`. Non continuare con il passo successivo fino a quando `Status` non è `Bound`. Se nel campo `Status` viene visualizzato `Pending` per più di 10 minuti, utilizza i messaggi di avviso negli `Events` come riferimento per risolvere eventuali problemi.

10. Implementare un'applicazione di esempio.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-
driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11. Verificare che l'applicazione di esempio sia in esecuzione.

```
kubectl get pods
```

Di seguito viene riportato un output di esempio:

```
NAME      READY   STATUS              RESTARTS   AGE
fsx-app   1/1     Running             0          8s
```

12. Verifica che l'applicazione abbia montato correttamente il file system.

```
kubectl exec -ti fsx-app -- df -h
```

Di seguito viene riportato un output di esempio.

```
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. Verifica che i dati siano stati scritti nel file system FSx for Lustre dall'app di esempio.

```
kubectl exec -it fsx-app -- ls /data
```

Di seguito viene riportato un output di esempio.

```
out.txt
```

Questo output esemplificativo mostra che l'app di esempio ha eseguito correttamente la scrittura del file `out.txt` nel file system.

> **Note**
>
> Prima di eliminare il cluster, assicuratevi di eliminare il file system FSx for Lustre. Per ulteriori informazioni, consultate Clean up resources nella Guida FSx per l'utente di for Lustre.

Ottimizzazione delle prestazioni per for FSx Lustre

Quando si utilizza FSx per Lustre con Amazon EKS, è possibile ottimizzare le prestazioni applicando le ottimizzazioni Lustre durante l'inizializzazione del nodo. L'approccio consigliato consiste nell'utilizzare i dati utente del modello di avvio per garantire una configurazione coerente su tutti i nodi.

Queste ottimizzazioni includono:

- Ottimizzazioni di rete e RPC
- Gestione del modulo Lustre
- Ottimizzazioni LRU (Lock Resource Unit)
- Impostazioni di controllo della cache del client
- Controlli RPC per OST e MDC

Per istruzioni dettagliate sull'implementazione di queste ottimizzazioni delle prestazioni:

- Per ottimizzare le prestazioni per i nodi standard (non EFA), consulta the section called "Ottimizza (non EFA)" per uno script completo che può essere aggiunto ai dati utente del modello di avvio.
- Per ottimizzare le prestazioni per i nodi abilitati all'EFA, consulta the section called "Ottimizza (EFA)".

Ottimizza le prestazioni di Amazon FSx for Lustre sui nodi (EFA)

Questo argomento descrive come configurare l'ottimizzazione di Elastic Fabric Adapter (EFA) con Amazon EKS e Amazon for Lustre. FSx

> **Note**
>
> - Per informazioni sulla creazione e la distribuzione del driver CSI FSx for Lustre, consulta. the section called "Implementare il driver"
> - Per ottimizzare i nodi standard senza EFA, consultare the section called "Ottimizza (non EFA)".

Passaggio 1. Crea un cluster EKS

Crea un cluster utilizzando il file di configurazione fornito:

```
# Create cluster using efa-cluster.yaml
eksctl create cluster -f efa-cluster.yaml
```

Esempio: `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)"
```

Passaggio 2. Creazione di un gruppo di nodi

Creazione di un gruppo di nodi abilitato per EFA:

```
# Create node group using efa-ng.yaml
eksctl create nodegroup -f efa-ng.yaml
```

> **Important**
>
> === Modifica questi valori per il tuo ambiente nella sezione `# 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.
```

===

Esempio`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."
```

(Facoltativo) Passaggio 3. Verifica della configurazione EFA

SSH nel nodo:

```
# Get instance ID from EKS console or {aws} CLI
ssh -i /path/to/your-key.pem ec2-user@<node-internal-ip>
```

Verifica della configurazione EFA:

```
sudo lnetctl net show
```

Controlla i registri di configurazione:

```
sudo cat /var/log/cloud-init-output.log
```

Ecco un esempio di output previsto per `lnetctl net show`:

```
net:
    - net type: tcp
      ...
    - net type: efa
      local NI(s):
        - nid: xxx.xxx.xxx.xxx@efa
          status: up
```

Esempio di implementazioni

a. Creazione di claim.yaml

```
#claim.yaml

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fsx-claim-efa
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 4800Gi
  volumeName: fsx-pv
```

Applica la richiesta:

```
kubectl apply -f claim.yaml
```

b. Creazione di pv.yaml

Aggiorna`<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>
```

Applica il volume persistente:

```
kubectl apply -f pv.yaml
```

c. Creazione di 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
```

Applica il pod:

```
kubectl apply -f pod.yaml
```

Comandi di verifica aggiuntivi

Verifica i montaggi del pod e scrive sul filesystem:

```
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 sul nodo per verificare che il traffico stia passando attraverso EFA:

```
sudo lnetctl net show -v
```

L'output previsto mostrerà le interfacce EFA con le statistiche sul traffico.

Informazioni correlate

- the section called "Implementare il driver"
- the section called "Ottimizza (non EFA)"
- Prestazioni Amazon FSx for Lustre
- Elastic Fabric Adapter

Ottimizza le prestazioni di Amazon FSx for Lustre sui nodi (non EFA)

Puoi ottimizzare le prestazioni di Amazon FSx for Lustre applicando i parametri di ottimizzazione durante l'inizializzazione del nodo utilizzando i dati utente del modello di lancio.

> **Note**
>
> - Per informazioni sulla creazione e la distribuzione del driver CSI FSx for Lustre, consulta. the section called "Implementare il driver" Per ottimizzare le prestazioni con i nodi abilitati all'EFA, consultare the section called "Ottimizza (EFA)".

Perché utilizzare i dati utente del modello di avvio?

- Applica automaticamente le ottimizzazioni durante l'inizializzazione del nodo.
- Garantisce una configurazione coerente su tutti i nodi.
- Elimina la necessità di configurare manualmente il nodo.

Panoramica degli script di esempio

Lo script di esempio definito in questo argomento esegue le seguenti operazioni:

`# 1. Install Lustre client`

- Rileva automaticamente la versione del tuo sistema operativo Amazon Linux (AL).
- Installa il pacchetto client Lustre appropriato.

`# 2. Apply network and RPC tunings`

- Imposta `ptlrpcd_per_cpt_max=64` per l'elaborazione RPC parallela.
- Configura `ksocklnd credits=2560` per ottimizzare i buffer di rete.

`# 3. Load Lustre modules`

- Rimuove in modo sicuro i moduli Lustre esistenti, se presenti.
- Gestisce lo smontaggio dei filesystem esistenti.
- Carica nuovi moduli Lustre.

`# 4. Lustre Network Initialization`

- Inizializza la configurazione di rete Lustre.
- Imposta i parametri di rete richiesti.

`# 5. Mount FSx filesystem`

- È necessario modificare i valori per il tuo ambiente in questa sezione.

 # 6. Apply tunings

- Ottimizzazioni LRU (Lock Resource Unit):
  - `lru_max_age=600000`
  - `lru_size` calcolato in base al numero di CPU
- Controllo della cache del client: `max_cached_mb=64`
- Controlli RPC:
  - OST `max_rpcs_in_flight=32`
  - MDC `max_rpcs_in_flight=64`
  - MDC `max_mod_rpcs_in_flight=50`

 # 7. Verify tunings

- Verifica tutte le ottimizzazioni applicate.
- Segnala il successo o l'avviso per ogni parametro.

 # 8. Setup persistence

- È necessario modificare i valori per il tuo ambiente anche in questa sezione.
- Rileva automaticamente la versione del sistema operativo (AL2023) per determinare quale `Systemd` servizio applicare.
- Il sistema si avvia.
- `Systemd` avvia il servizio `lustre-tunings` (a causa di `WantedBy=multi-user.target`).
- Il servizio esegue `apply_lustre_tunings.sh` che:
  - Verifica se il filesystem è montato.
  - Monta il filesystem se non è montato.
  - Attende che il montaggio venga eseguito con successo (fino a cinque minuti).
  - Applica i parametri di ottimizzazione dopo il montaggio riuscito.
- Le impostazioni rimangono attive fino al riavvio.
- Il servizio si chiude dopo il completamento dello script.
  - Systemd contrassegna il servizio come "attivo (terminato)".
- Il processo si ripete al successivo riavvio.

Creazione di un modello di avvio

1. Apri la EC2 console Amazon all'indirizzo https://console.aws.amazon.com/ec2/.
2. Scegli Modelli di avvio.
3. Scegli Crea modello di avvio.
4. In Dettagli avanzati, individua la sezione Dati utente.
5. Incolla lo script qui di seguito, aggiornando tutto ciò che è necessario.

> **Important**
>
> Regola questi valori per il tuo ambiente sia nella sezione `# 5. Mount FSx filesystem` che nella funzione `setup_persistence()` di `apply_lustre_tunings.sh` nella sezione `# 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. Quando crei gruppi di nodi Amazon EKS, seleziona questo modello di avvio. Per ulteriori informazioni, consulta the section called "Crea".

Informazioni correlate

- the section called "Implementare il driver"
- the section called "Ottimizza (EFA)"
- Prestazioni Amazon FSx for Lustre

Usa l'archiviazione delle app ad alte prestazioni con FSx per NetApp ONTAP

NetApp Trident offre un'orchestrazione dinamica dello storage utilizzando un driver compatibile con CSI (Container Storage Interface). Ciò consente ai cluster Amazon EKS di gestire il ciclo di

vita dei volumi persistenti (PV) supportati dai file system Amazon FSx per NetApp ONTAP. Tieni presente che il driver CSI di Amazon FSx per NetApp ONTAP non è compatibile con Amazon EKS Hybrid Nodes. Per iniziare, consultare Use Trident with Amazon FSx for NetApp ONTAP nella documentazione di NetApp Trident.

Amazon FSx per NetApp ONTAP è un servizio di archiviazione che consente di avviare ed eseguire file system ONTAP completamente gestiti nel cloud. ONTAP è la tecnologia di file system di NetApp che fornisce un insieme ampiamente adottato di funzionalità di accesso ai dati e gestione dei dati. FSx per ONTAP offre le funzionalità, le prestazioni e le API dei file system NetApp on-premises con l'agilità, la scalabilità e l'agilità di un servizio AWS completamente gestito. Per ulteriori informazioni, consulta la Guida per l'utente di FSx per ONTAP.

Important

Se utilizzi Amazon FSx per NetApp ONTAP insieme al driver Amazon EBS CSI per il provisioning dei volumi EBS, devi specificare di non utilizzare dispositivi EBS nel file `multipath.conf`. Per i metodi supportati, consulta Configuration File Blacklist. Ecco un esempio.

```
defaults {
       user_friendly_names yes
       find_multipaths no
     }
     blacklist {
       device {
         vendor "NVME"
         product "Amazon Elastic Block Store"
       }
     }
```

Usa l'archiviazione di dati con Amazon FSx per OpenZFS

Amazon FSx per OpenZFS è un servizio di archiviazione di file interamente gestito che semplifica il trasferimento di dati su AWS da ZFS on-premise o altri file server basati su Linux. Puoi farlo senza modificare il codice dell'applicazione o la modalità di gestione dei dati. Offre un'archiviazione di file altamente affidabile, scalabile, efficiente e ricca di funzionalità basata sul file system OpenZFS open source. Combina queste funzionalità con l'agilità, la scalabilità e la semplicità di un servizio AWS

interamente gestito. Per ulteriori informazioni, consulta la Guida per l'utente di Amazon FSx per OpenZFS.

Il driver Container Storage Interface (CSI) di FSx per OpenZFS mette a disposizione un'interfaccia CSI che consente ai cluster EKS di gestire il ciclo di vita dei volumi Amazon FSx per OpenZFS. Tieni presente che il Driver CSI di Amazon FSx per OpenZFS non è compatibile con Amazon EKS Hybrid Nodes. Per implementare il driver CSI di FSx per OpenZFS nel tuo cluster Amazon EKS, consulta aws-fsx-openzfs-csi-driver su GitHub.

Riduci al minimo la latenza con Amazon File Cache

Amazon File Cache è una cache ad alta velocità completamente gestita su AWS che viene utilizzata per elaborare i dati dei file indipendentemente da dove sono archiviati. Amazon File Cache carica automaticamente i dati nella cache quando vi si accede per la prima volta e li rilascia quando non vengono utilizzati. Per ulteriori informazioni, consulta la Guida per l'utente di Amazon File Cache.

Il driver Container Storage Interface (CSI) di Amazon File Cache mette a disposizione un'interfaccia CSI che consente ai cluster Amazon EKS di gestire il ciclo di vita delle cache di file di Amazon. Tieni presente che il driver Amazon File Cache CSI non è compatibile con Amazon EKS Hybrid Nodes. Per implementare il driver CSI di Amazon File Cache nel tuo cluster Amazon EKS, consulta aws-file-cache-csi-driver su GitHub.

Accedere agli oggetti Amazon S3 con il driver CSI Mountpoint per Amazon S3

Con il driver dell'interfaccia di archiviazione del container (CSI) di Mountpoint per Amazon S3, le applicazioni Kubernetes possono accedere agli oggetti Amazon S3 tramite un'interfaccia di file system, raggiungendo una throughput aggregato elevato senza modificare alcun codice applicativo. Basato su Mountpoint per Amazon S3, il driver CSI presenta un bucket Amazon S3 come volume, a cui è possibile accedere tramite i container in Amazon EKS e i cluster Kubernetes autogestiti.

Considerazioni

- Il driver CSI di Mountpoint per Amazon S3 non è compatibile con le immagini di container basate su Windows.
- Il driver CSI di Mountpoint per Amazon S3 non è attualmente compatibile con Amazon EKS Hybrid Nodes.

- Il driver CSI di Mountpoint per Amazon S3 non supporta AWS Fargate. Tuttavia, sono supportati i container in esecuzione in Amazon EC2 (con Amazon EKS o un'installazione Kubernetes personalizzata).
- Il driver CSI di Mountpoint per Amazon S3 supporta solo il provisioning statico. Non è supportato il provisioning dinamico o la creazione di nuovi bucket.

> **Note**
>
> Il provisioning statico si riferisce all'utilizzo di un bucket Amazon S3 esistente specificato come `bucketName` in `volumeAttributes` nell'oggetto `PersistentVolume`. Per ulteriori informazioni, consulta Provisioning statico su GitHub.

- I volumi montati con il driver CSI di Mountpoint per Amazon S3 non supportano tutte le funzionalità del file system POSIX. Per i dettagli sul comportamento del file system, consulta Comportamento del file system di Mountpoint per Amazon S3 su GitHub.

Per i dettagli sull'implementazione del driver, consulta the section called "Implementazione del driver". Per informazioni dettagliate sulla rimozione del driver, consulta the section called "Rimozione del driver".

Implementazione del driver Mountpoint per Amazon S3

Con il driver dell'interfaccia di archiviazione del container (CSI) di Mountpoint per Amazon S3, le applicazioni Kubernetes possono accedere agli oggetti Amazon S3 tramite un'interfaccia di file system, raggiungendo una throughput aggregato elevato senza modificare alcun codice applicativo.

Questa procedura illustra come implementare il driver Amazon EKS CSI di Mountpoint per Amazon S3. Prima di procedere, consulta Considerazioni.

Prerequisiti

- Un provider di AWS Identity and Access Management (IAM) OpenID Connect (OIDC) esistente per il tuo cluster. Per determinare se si dispone già di un provider IAM o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
- Versione 2.12.3 o successiva della AWS CLI installata e configurata sul dispositivo o. AWS CloudShell
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di

Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":

Fase 1: Creazione di una policy IAM

Il driver CSI di Mountpoint per Amazon S3 richiede le autorizzazioni Amazon S3 per interagire con il file system. In questa sezione viene descritto come creare una policy IAM che conceda le autorizzazioni necessarie.

La seguente policy esemplificativa segue i suggerimenti delle autorizzazioni IAM per Mountpoint. In alternativa, puoi utilizzare la politica AWS gestita AmazonS3 FullAccess, ma questa politica gestita concede più autorizzazioni di quelle necessarie per Mountpoint.

Per ulteriori informazioni sulle autorizzazioni consigliate per Mountpoint, consulta Mountpoint IAM permissions on. GitHub

1. Apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel pannello di navigazione a sinistra, seleziona Policy.
3. Nella pagina Policy, scegli Crea policy.
4. Per Editor di policy, scegli JSON.
5. In Editor di policy, copia e incolla quanto segue:

> **Important**
>
> Sostituisci `amzn-s3-demo-bucket1` con il nome del bucket 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/*"
            ]
        }
    ]
}
```

I bucket di directory, introdotti con la classe di archiviazione Amazon S3 Express One Zone, utilizzano un meccanismo di autenticazione diverso dai bucket per uso generico. Invece di utilizzare le azioni `s3:*`, devi utilizzare l'azione `s3express:CreateSession`. Per ulteriori informazioni sui bucket, consulta Bucket di directory nella Guida per l'utente di Amazon S3.

Di seguito è riportato un esempio di policy dei privilegi minimi da utilizzare per un bucket di directory.

```
{
    "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. Scegli Next (Successivo).
7. Nella pagina Verifica e crea, assegna un nome alla policy. Questa procedura guidata di esempio utilizza il nome `AmazonS3CSIDriverPolicy`.

8. Scegli Crea policy.

Fase 2: creazione di un ruolo IAM

Il driver CSI di Mountpoint per Amazon S3 richiede le autorizzazioni Amazon S3 per interagire con il file system. In questa sezione viene descritto come creare un ruolo IAM per delegare queste autorizzazioni. Per creare questo ruolo, è possibile utilizzare uno di questi strumenti:

- the section called "eksctl"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

> **Note**
>
> La policy IAM `AmazonS3CSIDriverPolicy` è stata creata nella sezione precedente.

eksctl

creazione del ruolo IAM del driver CSI di Mountpoint per Amazon S3 con **`eksctl`**

Per creare il ruolo IAM e dell'account di servizio Kubernetes, esegui i comandi seguenti. Questi comandi allegano anche la policy IAM `AmazonS3CSIDriverPolicy` al ruolo, annotano l'account di servizio Kubernetes (`s3-csi-controller-sa`) con nome della risorsa Amazon (ARN) del ruolo IAM e aggiungono il nome dell'account di servizio Kubernetes alla policy di attendibilità per il ruolo 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 di gestione AWS

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Select trusted entity (Seleziona entità attendibile), esegui le operazioni seguenti:
   a. Nella sezione Tipo di identità attendibile, scegli Identità Web.
   b. Per Identity provider (Provider di identità), scegli URL del provider OpenID Connect per il cluster (come mostrato nella scheda Overview (Panoramica) in Amazon EKS).

      Se non URLs ne viene visualizzato nessuno, consulta i Prerequisiti.
   c. Per Pubblico, scegli `sts.amazonaws.com`.
   d. Scegli Next (Successivo).
5. Nella pagina Add permissions (Aggiungi autorizzazioni), esegui le operazioni seguenti:
   a. Nella casella Filtra politiche, inserisci la politica di CSIDriver AmazonS3.

      > **Note**
      >
      > Questa policy è stata creata nella sezione precedente.

   b. Seleziona la casella di controllo a sinistra del risultato `AmazonS3CSIDriverPolicy` restituito dalla ricerca.
   c. Scegli Next (Successivo).
6. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:
   a. Per il nome del ruolo, inserisci un nome univoco per il tuo ruolo, ad esempio AmazonEKS_S3_CSI_. DriverRole
   b. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.
   c. Scegli Crea ruolo.
7. Dopo aver creato il ruolo, sceglilo nella console in modo da aprirlo per la modifica.

8. Scegli la scheda Trust relationships (Relazioni di attendibilità), quindi scegli Edit trust policy (Modifica policy di attendibilità).
9. Cercare il risultato finale simile al seguente:

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

   Inserisci una virgola alla fine della riga precedente, quindi aggiungine una seguente dopo quella riga. Sostituiscilo con la regione in cui si trova il cluster. *region-code* AWS Sostituire *EXAMPLED539D4633E53DE1B71EXAMPLE* con l'ID del gestore OIDC del cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:s3-csi-driver-sa"
```

10. Assicurati che l'operatore `Condition` sia impostato su `"StringEquals"`.
11. Scegli Aggiorna policy per concludere.

AWS CLI

1. Visualizza l'URL del provider OIDC del cluster. Sostituisci *my-cluster* con il nome del cluster. Se l'output dal comando è `None`, rivedi i Prerequisiti.

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

   Di seguito viene riportato un output di esempio.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Creare il ruolo IAM, concedendo all'account di servizio Kubernetes l'operazione `AssumeRoleWithWebIdentity`.
   a. Copiare i contenuti seguenti in un file denominato `aws-s3-csi-driver-trust-policy.json`. Sostituisci *111122223333* con l'ID del tuo account. Sostituire *EXAMPLED539D4633E53DE1B71EXAMPLE* e *region-code* con i valori restituiti nella fase precedente.

```
{
  "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. Creare il ruolo. È possibile modificare *AmazonEKS_S3_CSI_DriverRole* con un nome diverso. In tal caso, assicurati di modificarlo anche nelle fasi successive.

```
aws iam create-role \
  --role-name AmazonEKS_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"
```

3. Collega la policy IAM creata in precedenza al ruolo con il seguente comando.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole
```

> **Note**
>
> La policy IAM `AmazonS3CSIDriverPolicy` è stata creata nella sezione precedente.

4. Salta questo passaggio se stai installando il driver come componente aggiuntivo di Amazon EKS. Per le installazioni autogestite del driver, crea account di servizio Kubernetes annotati con l'ARN del ruolo IAM che hai creato.

a. Salvare i seguenti contenuti in un file denominato `mountpoint-s3-service-account.yaml`. Sostituisci *111122223333* con l'ID del tuo account.

```
---
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. Creare l'account del servizio Kubernetes nel cluster. L'account del servizio Kubernetes (`mountpoint-s3-csi-controller-sa`) è annotato con il nome del ruolo IAM che hai creato. *AmazonEKS_S3_CSI_DriverRole*

```
kubectl apply -f mountpoint-s3-service-account.yaml
```

> **Note**
>
> Quando implementi il plugin tramite questa procedura, il plugin crea ed è configurato per l'utilizzo di un account di servizio denominato `s3-csi-driver-sa`.

Fase 3: installazione del driver CSI di Mountpoint per Amazon S3

È possibile installare il driver CSI di Mountpoint per Amazon S3 tramite il componente aggiuntivo di Amazon EKS. Per aggiungere il componente aggiuntivo al cluster, è possibile utilizzare i seguenti strumenti:

- the section called "eksctl"
- the section called "Console di gestione AWS"
- the section called "AWS CLI"

In alternativa, è possibile installare il driver CSI Mountpoint per Amazon S3 come installazione autogestita. Per istruzioni su come eseguire un'installazione autogestita, consulta Installazione su GitHub.

A partire da `v1.8.0`, è possibile configurare i taint da tollerare per i pod del driver CSI. Per fare ciò, specifica un set personalizzato di taint da tollerare con `node.tolerations` o tollera tutti i taint con `node.tolerateAllTaints`. Per ulteriori informazioni, consulta Taint e tolleranze nella documentazione Kubernetes.

eksctl

Aggiungere il componente aggiuntivo CSI per Amazon S3 utilizzando **`eksctl`**

Eseguire il seguente comando seguente. Sostituisci *`my-cluster`* con il nome del cluster, *`111122223333`* con il tuo ID account e *`AmazonEKS_S3_CSI_DriverRole`* con il nome del ruolo IAM creato in precedenza.

```
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 si rimuove l'opzione *`--force`* e tutte le impostazioni del componente aggiuntivo Amazon EKS sono in conflitto con le impostazioni esistenti, l'aggiornamento del componente aggiuntivo Amazon EKS non avrà successo. Verrà quindi visualizzato un messaggio di errore con la procedura utile a risolvere il conflitto. Prima di specificare questa opzione, assicurati che il componente aggiuntivo di Amazon EKS non gestisca le impostazioni che devi gestire tu, perché questa opzione le sovrascrive. Per ulteriori informazioni su altre opzioni per questa impostazione, consulta Componenti aggiuntivi nella documentazione di `eksctl`. Per ulteriori informazioni sulla gestione dei campi Kubernetes di Amazon EKS, consulta the section called "Campi che puoi personalizzare".

È possibile personalizzare `eksctl` tramite file di configurazione. Per ulteriori informazioni, consulta Funzionamento con i valori di configurazione nella documentazione `eksctl`. L'esempio seguente mostra come tollerare tutti i taint.

```
# config.yaml
...

addons:
- name: aws-mountpoint-s3-csi-driver
```

```
  serviceAccountRoleARN: arn:aws: iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
  configurationValues: |-
    node:
      tolerateAllTaints: true
```

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Scegli il nome del cluster per cui configurare il componente aggiuntivo CSI di Mountpoint per Amazon S3.
4. Selezionare la scheda Componenti aggiuntivi.
5. Scegli Ottieni altri componenti aggiuntivi.
6. Nella pagina Seleziona componenti aggiuntivi, procedi come segue:
   a. Nella sezione Componenti aggiuntivi di Amazon EKS, seleziona la casella di controllo Driver CSI di Mountpoint per Amazon S3.
   b. Scegli Next (Successivo).
7. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, procedi come segue:
   a. Seleziona la Versione che desideri utilizzare.
   b. In Seleziona ruolo IAM, seleziona il nome di un ruolo IAM a cui è stata allegata la policy IAM del driver CSI di Mountpoint per Amazon S3.
   c. (Facoltativo) Aggiorna il Metodo di risoluzione dei conflitti dopo aver espanso le Impostazioni di configurazione opzionali. Se selezioni Sostituisci, una o più impostazioni per il componente aggiuntivo esistente possono essere sovrascritte con le impostazioni del componente aggiuntivo di Amazon EKS. Se non si abilita questa opzione e c'è un conflitto con le impostazioni esistenti, l'operazione non va a buon fine e viene visualizzato un messaggio di errore per aiutarti a risolvere il conflitto. Prima di selezionare questa opzione, assicurarsi che il componente aggiuntivo Amazon EKS non gestisca le impostazioni che devono essere gestite dall'utente.
   d. (Facoltativo) Configura le tolleranze nel campo Valori di configurazione dopo aver espanso le Impostazioni di configurazione opzionali.
   e. Scegli Next (Successivo).
8. Nella pagina Rivedi e aggiungi, scegli Crea. Una volta completata l'installazione, viene visualizzato il componente aggiuntivo.

AWS CLI

Per aggiungere il componente aggiuntivo Mountpoint for Amazon S3 CSI utilizzando la CLI AWS

Eseguire il seguente comando seguente. Sostituire *my-cluster* con il nome del cluster, *111122223333* con il proprio ID account e *AmazonEKS_S3_CSI_DriverRole* con il nome del ruolo creato in precedenza.

```
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
```

È possibile personalizzare il comando con il contrassegno `--configuration-values`. L'esempio alternativo seguente mostra come tollerare tutti i taint.

```
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}}'
```

Fase 4: configurazione di Mountpoint per Amazon S3

Nella maggior parte dei casi, è possibile configurare Mountpoint per Amazon S3 solo con un nome di bucket. Per istruzioni sulla configurazione di Mountpoint per Amazon S3, consulta Configurazione di Mountpoint per Amazon S3 su. GitHub

Fase 5: implementazione di un'applicazione esemplificativa

Puoi implementare il provisioning statico al driver su un bucket Amazon S3 esistente. Per ulteriori informazioni, consulta Provisioning statico attivo. GitHub

Rimozione del componente aggiuntivo Amazon EKS di Mountpoint per Amazon S3

Sono disponibili due opzioni per rimuovere il driver CSI di Mountpoint per Amazon S3.

- Mantenere il software aggiuntivo sul cluster – Questa opzione rimuove la gestione di Amazon EKS su qualsiasi impostazione. Inoltre, rimuove la possibilità per Amazon EKS di notificare gli

aggiornamenti e aggiornare automaticamente il componente aggiuntivo Amazon EKS dopo l'avvio di un aggiornamento. Tuttavia, mantiene il software aggiuntivo sul cluster. Questa opzione rende il componente aggiuntivo come autogestito anziché come un componente aggiuntivo Amazon EKS. Con questa opzione, non ci sono tempi di inattività per il componente aggiuntivo. I comandi in questa procedura utilizzano questa opzione.

- Rimuovere completamente il software aggiuntivo dal cluster – Consigliamo di rimuovere il componente aggiuntivo Amazon EKS dal cluster solo se non ci sono risorse nel cluster che dipendono dallo stesso. Per eseguire questa operazione, elimina `--preserve` dal comando utilizzato in questa procedura.

Se al componente aggiuntivo è associato un account IAM, l'account IAM non è rimosso.

È possibile utilizzare i seguenti strumenti per rimuovere il componente aggiuntivo CSI per Amazon S3:

- the section called "eksctl"
- the section called "Console di gestione AWS"
- the section called "CLI AWS"

eksctl

Per rimuovere il componente aggiuntivo CSI per Amazon S3 utilizzando **`eksctl`**

Sostituisci *`my-cluster`* con il nome del cluster ed eseguire il comando seguente.

```
eksctl delete addon --cluster my-cluster --name aws-mountpoint-s3-csi-driver --preserve
```

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Scegli il nome del cluster per cui desideri rimuovere il componente aggiuntivo CSI per Amazon EBS.
4. Seleziona la scheda Componenti aggiuntivi.
5. Scegli Driver CSI di Mountpoint per Amazon S3

6. Scegli Rimuovi.
7. Nella finestra di dialogo di conferma Rimuovi: aws-mountpoint-s3-csi-driver, procedi come segue:
   a. Se desideri che Amazon EKS interrompa la gestione delle impostazioni per il componente aggiuntivo, seleziona Mantieni sul cluster. Effettuare questa operazione se si desidera conservare il software aggiuntivo sul cluster. In questo modo sarà possibile gestire autonomamente tutte le impostazioni del componente aggiuntivo.
   b. Specificare `aws-mountpoint-s3-csi-driver`.
   c. Selezionare Rimuovi.

CLI AWS

Rimuovere il componente aggiuntivo CSI di Amazon S3 utilizzando AWS CLI

Sostituisci *`my-cluster`* con il nome del cluster ed eseguire il comando seguente.

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver --preserve
```

Abilitare la funzionalità snapshot per i volumi CSI

La funzionalità snapshot consente di eseguire copie point-in-time dei dati. Affinché funzioni in Kubernetes, questa funzionalità richiede sia un driver CSI con supporto per snapshot (come il driver CSI di Amazon EBS), sia un controller di snapshot CSI. Il controller di snapshot è disponibile come componente aggiuntivo Amazon EKS o come installazione autogestita.

Di seguito sono elencati alcuni punti da considerare quando si utilizza il controller di snapshot CSI.

- Il controller di snapshot deve essere installato insieme a un driver CSI con funzionalità di snapshot. Per istruzioni sull'installazione del driver CSI di Amazon EBS, consultare la pagina the section called "Amazon EBS".
- Kubernetes non supporta snapshot di volumi serviti tramite migrazione CSI, come i volumi Amazon EBS che utilizzano una `StorageClass` con provisioner `kubernetes.io/aws-ebs`. I volumi devono essere creati con una `StorageClass` che fa riferimento al provisioner del driver CSI, `ebs.csi.aws.com`.
- La modalità automatica Amazon EKS non include il controller di snapshot. La funzionalità di archiviazione della modalità automatica EKS è compatibile con il controller di snapshot.

Consigliamo di installare il controller di snapshot CSI attraverso il componente aggiuntivo gestito di Amazon EKS. Questo componente aggiuntivo include le definizioni di risorse personalizzate (Custom Resource Definitions, CRD) necessarie per creare e gestire snapshot su Amazon EKS. Per aggiungere un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS".

In alternativa, se si desidera un'installazione autogestita del controller di snapshot CSI, consultare la pagina Usage nel `external-snapshotter` upstream Kubernetes su GitHub.

Configurazione della rete per cluster Amazon EKS

Il cluster Amazon EKS viene creato in un VPC. Le reti pod sono fornite dal plug-in della CNI (Container Network Interface) di Amazon VPC per i nodi eseguiti sull'infrastruttura AWS. Se gestisci nodi sulla tua infrastruttura, consulta the section called "Configurazione della CNI". Per ulteriori informazioni sulle reti disponibili per il cluster, puoi consultare gli argomenti seguenti.

Argomenti

- Aggiunta di una sottorete VPC esistente a un cluster Amazon EKS dalla console di gestione
- Visualizzazione di requisiti di rete di Amazon EKS per VPC e sottoreti
- Creazione di un Amazon VPC per il cluster Amazon EKS.
- Visualizzazione dei requisiti relativi al gruppo di sicurezza Amazon EKS per cluster
- Gestione di componenti aggiuntivi di rete per i cluster Amazon EKS

Aggiunta di una sottorete VPC esistente a un cluster Amazon EKS dalla console di gestione

1. Accedi al tuo cluster nella console di gestione.
2. Dalla scheda Rete, seleziona Gestisci risorse VPC.
3. Dal menu a discesa Sottoreti, seleziona sottoreti aggiuntive dal VPC del tuo cluster.

Per creare una nuova sottorete VPC:

- Consulta i requisiti per le sottoreti EKS.
- Consulta Create a Subnet nella Guida per l'utente di Amazon Virtual Private Cloud.

Visualizzazione di requisiti di rete di Amazon EKS per VPC e sottoreti

Durante la creazione di un cluster, in genere si specificano un VPC e due sottoreti che si trovano in zone di disponibilità diverse. Questo argomento offre una panoramica dei requisiti e delle considerazioni specifiche di Amazon EKS per il VPC e le sottoreti utilizzate con il cluster. Se non

si dispone di un VPC da utilizzare con Amazon EKS, si veda the section called "Crea un VPC". Se stai creando un cluster locale o esteso su AWS Outposts, consulta the section called "Creazione di un VPC e delle sottoreti" invece di questo argomento. Il contenuto di questo argomento si applica ai cluster Amazon EKS con nodi ibridi. Per ulteriori requisiti di rete per i nodi ibridi, consulta the section called "Preparazione della rete".

Considerazioni e requisiti relativi al VPC

Durante la creazione di un cluster, il VPC specificato deve soddisfare i requisiti e le considerazioni seguenti:

- Il VPC deve disporre di un numero sufficiente di indirizzi IP da mettere a disposizione per il cluster, i nodi e le altre risorse Kubernetes da creare. In caso contrario, è possibile provare ad aumentare il numero di indirizzi IP disponibili.

  È possibile farlo aggiornando la configurazione del cluster per modificare le sottoreti e i gruppi di sicurezza utilizzati dal cluster. È possibile eseguire l' Console di gestione AWS aggiornamento dalla versione più recente della AWS CLI e dalla `eksctl` versione `v0.164.0-rc.0` o successiva. AWS CloudFormation Potrebbe essere necessario eseguire questa operazione per fornire alle sottoreti più indirizzi IP disponibili per aggiornare in modo corretto una versione del cluster.

  > **Important**
  >
  > Tutte le sottoreti aggiunte devono appartenere allo stesso set fornito originariamente al momento della AZs creazione del cluster. Le nuove sottoreti devono soddisfare tutti gli altri requisiti, ad esempio devono disporre di sufficienti indirizzi IP.
  > Supponiamo ad esempio che hai creato un cluster e specificato quattro sottoreti.
  > Nell'ordine in cui le hai specificate, la prima sottorete si trova nella zona di disponibilità `us-west-2a`, la seconda e la terza si trovano nella zona di disponibilità `us-west-2b` e la quarta si trova nella zona di disponibilità `us-west-2c`. Se desideri modificare le sottoreti, devi fornire almeno una sottorete in ciascuna delle tre zone di disponibilità e le sottoreti devono trovarsi nello stesso VPC delle sottoreti originali.

  Se hai bisogno di più indirizzi IP rispetto ai blocchi CIDR nel VPC, puoi aggiungere altri blocchi CIDR associando ulteriori blocchi di routing interdominio senza classi (CIDR) al VPC. L'associazione di blocchi di CIDR privati (RFC 1918) e pubblici (non RFC 1918) può avvenire prima o dopo la creazione del cluster.

È possibile aggiungere nodi che utilizzano il nuovo intervallo CIDR subito dopo averlo aggiunto. Tuttavia, poiché il piano di controllo riconosce il nuovo intervallo CIDR solo dopo il completamento della riconciliazione, il riconoscimento dell'intervallo CIDR associato a un VPC da parte del cluster può richiedere fino a un'ora. È quindi possibile eseguire i comandi `kubectl attachkubectl cp`,`kubectl exec`,`kubectl logs`, e (questi `kubectl port-forward` comandi utilizzano il`kubelet API`) per nodi e pod nel nuovo blocco CIDR. Inoltre, se si dispone di pod che funzionano come backend webhook, devi attendere il completamento della riconciliazione del piano di controllo.

- Evita le sovrapposizioni degli intervalli di indirizzi IP quando connetti il tuo cluster EKS ad altri VPCs tramite Transit Gateway, peering VPC o altre configurazioni di rete. I conflitti di CIDR si verificano quando il servizio CIDR del cluster EKS si sovrappone al CIDR di un VPC connesso. In questi scenari, gli indirizzi ServiceIP hanno la priorità sulle risorse VPCs connesse allo stesso indirizzo IP, sebbene il routing del traffico possa diventare imprevedibile e le applicazioni potrebbero non riuscire a connettersi alle risorse previste.

  Per evitare conflitti CIDR, assicurati che il tuo servizio EKS CIDR non si sovrapponga a nessun VPC connesso CIDRs e mantieni un registro centralizzato di tutte le assegnazioni CIDR. Se si riscontrano sovrapposizioni di CIDR, è possibile utilizzare un gateway di transito con un VPC di servizi condivisi. Per ulteriori informazioni, consulta Isolato VPCs con servizi condivisi e modelli di conservazione degli indirizzi IP routabili VPC di Amazon EKS in una rete ibrida. Inoltre, consulta la VPCs sezione Comunicazione trasversale della pagina Considerazioni su VPC e sottorete nella Guida alle best practice di EKS.
- Se vuoi che Kubernetes assegni indirizzi `IPv6` a pod e servizi, associa un intervallo CIDR `IPv6` al tuo VPC. Per ulteriori informazioni, consulta Associare un blocco IPv6 CIDR al tuo VPC nella Amazon VPC User Guide. Non è possibile utilizzare indirizzi `IPv6` con pod e servizi in esecuzione su nodi ibridi e non è possibile utilizzare nodi ibridi con cluster configurati con la famiglia di indirizzi IP `IPv6`.
- Il VPC deve supportare il nome host `DNS` e la risoluzione `DNS`. In caso contrario, i nodi non possono registrarsi al cluster. Per ulteriori informazioni, consulta Attributi DNS nel VPC nella Guida per l'utente di Amazon VPC.
- Il VPC potrebbe richiedere l'utilizzo di endpoint VPC. AWS PrivateLink Per ulteriori informazioni, consulta the section called "Considerazioni e requisiti relativi alle sottoreti".

Se il cluster è stato creato con Kubernetes `1.14` o versioni precedenti, Amazon EKS aggiunge il tag seguente al VPC:

Chiave	Valore
kubernetes.io/cluster/ *my-cluster*	owned

Questo tag è stato utilizzato solo da Amazon EKS, per cui è possibile rimuoverlo senza influire sui servizi. Non è necessario con la versione 1.15 del cluster o con versioni successive.

Considerazioni e requisiti relativi alle sottoreti

Durante la creazione di un cluster, Amazon EKS crea da 2 a 4 interfacce di rete elastiche nelle sottoreti specificate. Queste interfacce di rete consentono la comunicazione tra il cluster e il VPC. Queste interfacce di rete abilitano anche le funzionalità di Kubernetes che utilizzano. kubelet API Le connessioni all'kubeletAPI vengono utilizzate nei comandikubectl attach,, kubectl cpkubectl exec, kubectl logs e. kubectl port-forward Ogni interfaccia di rete creata da Amazon EKS ha il testo Amazon EKS *cluster-name* nella sua descrizione.

Amazon EKS può creare le proprie interfacce di rete in qualsiasi sottorete specificata al momento della creazione di un cluster. Puoi modificare le sottoreti in cui Amazon EKS crea le interfacce di rete dopo la creazione del cluster. Quando aggiorni la versione Kubernetes di un cluster, Amazon EKS elimina le interfacce di rete originali e ne crea di nuove Queste interfacce di rete potrebbero essere create all'interno delle stesse sottoreti o in sottoreti diverse rispetto alle interfacce di rete originali. Per verificare in quali sottoreti vengono create le interfacce di rete, puoi limitare il numero di sottoreti specificate a due quando crei un cluster o aggiornare le sottoreti dopo aver creato il cluster.

Requisiti relativi alla sottorete per i cluster

Le sottoreti specificate al momento della creazione o dell'aggiornamento di un cluster devono soddisfare i seguenti requisiti:

- Ciascuna sottorete deve disporre di almeno sei indirizzi IP per l'uso da parte di Amazon EKS. Tuttavia, consigliamo di utilizzare almeno 16 indirizzi IP.
- Le due sottoreti devono trovarsi in almeno due zone di disponibilità diverse.
- Le sottoreti non possono risiedere in AWS Outposts o Wavelength. AWS Tuttavia, se sono presenti nel VPC, puoi implementare nodi autogestiti e risorse Kubernetes per questi tipi di sottoreti. Per ulteriori informazioni sulla gestione dei nodi, consulta the section called "Nodi autogestiti".
- Le sottoreti possono essere pubbliche o private. Tuttavia, si consiglia di specificare sottoreti private, se possibile. Una sottorete pubblica è una sottorete con una tabella di routing che

include un instradamento a un gateway Internet, mentre la sottorete privata non comprende tale instradamento.

- Le sottoreti non possono risiedere nelle seguenti zone di disponibilità:

AWS Regione	Nome della Regione	Zona di disponibilità non consentita IDs
`us-east-1`	Stati Uniti orientali (Virginia settentrionale)	`use1-az3`
`us-west-1`	Stati Uniti occidentali (Californ ia settentrionale)	`usw1-az2`
`ca-central-1`	Canada (Centrale)	`cac1-az3`

IP address family usage by component

La tabella seguente contiene la famiglia di indirizzi IP utilizzata da ciascun componente di Amazon EKS. È possibile utilizzare una network address translation (NAT) o un altro sistema di compatibilità per connettersi a questi componenti da indirizzi IP di origine in famiglie con il valore "No" per una voce di tabella.

La funzionalità può variare a seconda dell'impostazione della famiglia IP (`ipFamily`) del cluster. Questa impostazione modifica il tipo di indirizzi IP utilizzati per l'intervallo CIDR assegnato da Kubernetes ai servizi. Un cluster con il valore di impostazione di IPv4 viene definito IPv4 cluster, mentre un cluster con il valore di impostazione di IPv6 viene definito IPv6 cluster.

Componente	IPv4 indirizzi	IPv6 indirizzi	Indirizzi dual stack
Endpoint pubblico dell'API EKS	Sì[1,3]	Sì[1,3]	Sì[1,3]
Endpoint VPC di API EKS	Sì	No	No
Endpoint pubblico di API di autentica	Sì[1]	Sì[1]	Sì[1]

Componente	IPv4 indirizzi	IPv6 indirizzi	Indirizzi dual stack
zione EKS (EKS Pod Identity)			
Endpoint VPC di API di autenticazione EKS (EKS Pod Identity)	Sì[1]	Sì[1]	Sì[1]
`IPv4` Endpoint pubblico del cluster Kubernetes[2]	Sì	No	No
`IPv4` Endpoint privato del cluster Kubernetes[2]	Sì	No	No
`IPv6` Endpoint pubblico del cluster Kubernetes[2]	Sì[1,4]	Sì[1,4]	Sì[4]
`IPv6` Endpoint privato del cluster Kubernetes[2]	Sì[1,4]	Sì[1,4]	Sì[4]
Sottoreti del cluster Kubernetes	Sì[2]	No	Sì[2]
Indirizzi IP primari del nodo	Sì[2]	No	Sì[2]
Intervallo CIDR del cluster per gli indirizzi IP del servizio	Sì[2]	Sì[2]	No
Indirizzi IP del pod da CNI di VPC	Sì[2]	Sì[2]	No

Componente	IPv4 indirizzi	IPv6 indirizzi	Indirizzi dual stack
Emittente IRSA OIDC URLs	Sì[1,3]	Sì[1,3]	Sì[1,3]

Note

[1] L'endpoint è dual-stack con gli indirizzi sia `IPv4` che `IPv6`. Le applicazioni esterne AWS, i nodi del cluster e i pod all'interno del cluster possono raggiungere questo endpoint tramite l'uno o l'altro. `IPv4` `IPv6`

[2] Si sceglie tra un cluster `IPv4` e un cluster `IPv6` nell'impostazione della famiglia IP (`ipFamily`) del cluster quando se ne crea uno e questa impostazione non può essere modificata. È invece necessario scegliere un'impostazione diversa quando si crea un altro cluster e si migrano i carichi di lavoro.

[3] L'endpoint dual-stack è stato introdotto nell'agosto 2024. Per utilizzare gli endpoint dual-stack con la AWS CLI, consulta la configurazione degli endpoint Dual-stack e FIPS nella and Tools Reference Guide. AWS SDKs Di seguito sono elencati i nuovi endpoint:

endpoint pubblico dell'API EKS

```
eks.region.api.aws
```

Emittente IRSA OIDC URLs

```
oidc-eks.region.api.aws
```

[4] L'endpoint del cluster dual-stack è stato introdotto nell'ottobre 2024. EKS crea il seguente endpoint per i nuovi cluster creati dopo questa data e selezionati `IPv6` nell'impostazione della famiglia IP (IpFamily) del cluster:

Endpoint del cluster public/private EKS

```
eks-cluster.region.api.aws
```

Requisiti relativi alla sottorete per i nodi

È possibile implementare nodi e risorse Kubernetes nelle stesse sottoreti specificate al momento della creazione del cluster. Tuttavia, non è necessario. di tali risorse può avvenire anche in sottoreti non specificate precedentemente. L'implementazione di nodi in sottoreti diverse non comporta la creazione di interfacce di rete cluster in tali sottoreti da parte di Amazon EKS. Qualsiasi sottorete in cui si implementano nodi e risorse Kubernetes deve soddisfare i requisiti seguenti:

- Le sottoreti devono mettere a disposizione un numero sufficiente di indirizzi IP per implementare tutti i nodi e le risorse Kubernetes.
- Se si desidera che Kubernetes assegni indirizzi `IPv6` a pod e servizi, è necessario disporre di un intervallo CIDR `IPv6` e un intervallo CIDR `IPv4` associati alla sottorete. Per ulteriori informazioni, consulta Associare un blocco IPv6 CIDR alla sottorete nella Amazon VPC User Guide. Le tabelle di instradamento associate alle sottoreti devono includere instradamenti a indirizzi `IPv4` e `IPv6`. Per maggiori informazioni, consulta Routes (Instradamenti) nella Guida dell'utente di Amazon VPC. Ai pod viene assegnato solo un indirizzo `IPv6`, mentre alle interfacce di rete create da Amazon EKS per il cluster e ai nodi vengono assegnati un indirizzo `IPv4` e uno `IPv6`.
- Se è necessario garantire ai pod un accesso in entrata da Internet, assicurarsi di disporre di almeno una sottorete pubblica con un numero sufficiente di indirizzi IP disponibili su cui implementare i bilanciatori di carico e gli ingressi. È possibile implementare i load balancer nelle sottoreti pubbliche. I bilanciatori di carico possono bilanciare il carico sui pod di sottoreti private o pubbliche. Consigliamo di implementare i nodi su sottoreti private, se possibile.
- Se si prevede di implementare i nodi in una sottorete pubblica, quest'ultima deve essere configurata per assegnare automaticamente indirizzi `IPv4` pubblici o indirizzi `IPv6`. Se si implementano nodi in una sottorete privata a cui è associato un blocco CIDR `IPv6`, anche questa sottorete deve essere configurata per assegnare automaticamente indirizzi `IPv6`. Se hai utilizzato il AWS CloudFormation modello fornito da Amazon EKS per distribuire il tuo VPC dopo il 26 marzo 2020, questa impostazione è abilitata. Se hai utilizzato i modelli per implementare il VPC prima di tale data o utilizzi un VPC personale, devi abilitare questa impostazione manualmente. Per il modello, consulta the section called "Crea un VPC". Per ulteriori informazioni, consulta Modificare l'attributo di IPv4 indirizzamento pubblico per la sottorete e Modificare l'attributo di IPv6 indirizzamento per la sottorete nella Amazon VPC User Guide.
- Se la sottorete in cui distribuisci un nodo è una sottorete privata e la relativa tabella di routing non include un percorso verso un dispositivo NAT (Network Address Translation) () o un gateway di sola uscita (`IPv4`)`IPv6`, aggiungi endpoint VPC utilizzando al tuo VPC. AWS PrivateLink Gli endpoint VPC sono necessari per tutti i AWS servizi con cui i nodi e i pod devono comunicare.

Alcuni esempi includono Amazon ECR, Elastic Load Balancing, CloudWatch Amazon AWS , Security Token Service e Amazon Simple Storage Service (Amazon S3). L'endpoint deve includere la sottorete in cui si trovano i nodi. Non tutti i AWS servizi supportano gli endpoint VPC. Per ulteriori informazioni, consulta Cos'è? AWS PrivateLink e AWS servizi che si integrano con AWS PrivateLink. Per un elenco di altri requisiti Amazon EKS, consulta the section called "Cluster privati".

- Se si desidera implementare i load balancer in una sottorete, quest'ultima deve presentare il tag seguente:
  - Sottoreti private

Chiave	Valore
`kubernetes.io/role/internal-elb`	`1`

  - Sottoreti pubbliche

Chiave	Valore
`kubernetes.io/role/elb`	`1`

In passato, quando si creava un cluster Kubernetes che è la versione `1.18` e precedenti, Amazon EKS aggiungeva il seguente tag a tutte le sottoreti specificate.

Chiave	Valore
`kubernetes.io/cluster/` *`my-cluster`*	`shared`

Adesso, se crei un nuovo cluster Kubernetes, Amazon EKS non aggiunge alcun tag alle sottoreti. Se il tag era applicato a sottoreti utilizzate da un cluster che in precedenza era una versione precedente a `1.19`, il tag non veniva rimosso automaticamente dalle sottoreti quando il cluster veniva aggiornato a una versione più recente. La versione `2.1.1` o precedente del AWS Load Balancer Controller richiede questo tag. Se si utilizza una versione più recente di Load Balancer Controller, è possibile rimuovere il tag senza interrompere i servizi. Per ulteriori informazioni sul controller, consulta the section called " AWS Controller Load Balancer".

Se hai distribuito un VPC `eksctl` utilizzando o uno qualsiasi dei modelli AWS CloudFormation VPC di Amazon EKS, si applica quanto segue:

- In data 26 marzo 2020 o successiva. Gli indirizzi `IPv4` pubblici vengono assegnati automaticamente dalle sottoreti pubbliche ai nuovi nodi implementati nelle sottoreti pubbliche.
- Prima del 26 marzo 2020: gli indirizzi `IPv4` pubblici non vengono assegnati automaticamente dalle sottoreti pubbliche ai nuovi nodi implementati nelle sottoreti pubbliche.

Questa modifica influisce sui nuovi gruppi di nodi implementati nelle sottoreti pubbliche nei modi seguenti:

- Gruppi di nodi gestiti : se il gruppo di nodi viene implementato in una sottorete pubblica in data 22 aprile 2020 o successiva, la sottorete pubblica deve disporre dell'assegnazione automatica degli indirizzi IP pubblici abilitati. Per ulteriori informazioni, consulta Modifica dell'attributo di IPv4 indirizzamento pubblico per la sottorete.
- Gruppi di nodi autogestiti Linux, Windows o Arm: se il gruppo di nodi viene implementato in una sottorete pubblica in data 26 marzo 2020 o successiva, la sottorete pubblica deve disporre dell'assegnazione automatica degli indirizzi IP pubblici abilitati. Altrimenti, i nodi devono essere avviati con un indirizzo IP pubblico. Per ulteriori informazioni, vedere Modifica dell'attributo di IPv4 indirizzamento pubblico per la sottorete o Assegnazione di un indirizzo pubblico IPv4 durante l'avvio dell'istanza.

Considerazioni e requisiti relativi alle sottoreti condivisi

Puoi utilizzare la condivisione VPC per condividere sottoreti con altri AWS account all'interno delle stesse Organizzazioni. AWS È possibile creare cluster Amazon EKS in sottoreti condivise, tenendo a mente le seguenti considerazioni:

- Il proprietario della sottorete VPC deve condividere una sottorete con un account partecipante prima che tale account possa creare un cluster Amazon EKS al suo interno.
- Non è possibile avviare le risorse utilizzando il gruppo di sicurezza predefinito per il VPC, in quanto questo appartiene al proprietario. Inoltre, i partecipanti non possono avviare le risorse utilizzando i gruppi di sicurezza di proprietà di altri partecipanti o del proprietario.
- In una sottorete condivisa, il partecipante e il proprietario controllano separatamente i gruppi di sicurezza all'interno di ciascun account. Il proprietario della sottorete può visualizzare i gruppi di sicurezza che sono stati creati dai partecipanti, ma non può eseguire operazioni sugli stessi.

Se il proprietario della sottorete desidera rimuovere o modificare questi gruppi di sicurezza, è il partecipante che ha creato il gruppo di sicurezza che deve eseguire l'operazione.

- Se un cluster viene creato da un partecipante, valgono le seguenti considerazioni:
  - Il ruolo IAM del cluster e i ruoli IAM del nodo devono essere creati in quell'account. Per ulteriori informazioni, consultare the section called “Ruolo IAM del cluster” e the section called “Ruolo IAM del nodo”.
  - Tutti i nodi devono essere creati dallo stesso partecipante, inclusi i gruppi di nodi gestiti.
- Il proprietario del VPC condiviso non può visualizzare, aggiornare o eliminare un cluster creato da un partecipante nella sottorete condivisa. Ciò si aggiunge alle risorse VPC alle quali ogni account ha un accesso diverso. Per ulteriori informazioni, consulta la sezione Responsabilità e autorizzazioni per proprietari e partecipanti nella Guida per l'utente di Amazon VPC.
- Se si utilizza la funzionalità rete personalizzata del plugin CNI di Amazon VPC, è necessario utilizzare le mappature degli ID delle zone di disponibilità elencate nell'account del proprietario per creare ciascun `ENIConfig`. Per ulteriori informazioni, consulta the section called “Rete personalizzata”.

Per ulteriori informazioni sulla condivisione delle sottoreti VPC, consulta la pagina Condivisione del VPC con altri account nella Guida per l'utente di Amazon VPC.

Creazione di un Amazon VPC per il cluster Amazon EKS.

Puoi usare Amazon Virtual Private Cloud (Amazon VPC) per lanciare AWS risorse in una rete virtuale che hai definito. Questa rete virtuale è simile a una rete tradizionale da gestire all'interno del proprio data center, ma con i vantaggi dell'infrastruttura scalabile di Amazon Web Services. Prima di implementare cluster Amazon EKS di produzione, è preferibile approfondire le nozioni relative al servizio Amazon VPC. Per ulteriori informazioni, consulta la Guida utente Amazon VPC.

Un cluster Amazon EKS, i nodi e le risorse Kubernetes vengono implementati su un VPC. Se desideri utilizzare un VPC esistente con Amazon EKS, tale VPC dovrà soddisfare i requisiti descritti nella sezione the section called “Requisiti del VPC e delle sottoreti”. Questo argomento descrive come creare un VPC che soddisfi i requisiti di Amazon EKS utilizzando un modello fornito AWS CloudFormation da Amazon EKS. Dopo l'implementazione di un modello, puoi visualizzare le risorse create dal modello per sapere esattamente quali risorse ha creato e la configurazione di tali risorse. Se usi nodi ibridi, il tuo VPC deve avere dei percorsi nella tabella di routing per la tua rete on-premises. Per ulteriori informazioni sui requisiti di rete per i nodi ibridi, consulta the section called “Preparazione della rete”.

Prerequisiti

Per creare un VPC destinato ad Amazon EKS, devi disporre delle autorizzazioni IAM necessarie alla creazione delle risorse per Amazon VPC. Queste risorse sono sottoreti VPCs, gruppi di sicurezza, tabelle e percorsi di routing e gateway Internet e NAT. Per ulteriori informazioni, consulta la sezione Creare un VPC con una sottorete pubblica nella Guida per l'utente di Amazon VPC e l'elenco completo di Azioni disponibile nella Guida di riferimento per l'autorizzazione del servizio.

È possibile creare un VPC con sottoreti sia pubbliche che private, solo pubbliche o solo private.

Sottoreti pubbliche e private

Questo VPC dispone di due sottoreti pubbliche e due private. Alle sottoreti pubbliche è associata una tabella di routing con un routing a un gateway Internet. Tuttavia, la tabella di routing mento di una sottorete privata non ha una route a un gateway Internet. Sia la sottorete pubblica che la sottorete privata vengono distribuite nella stessa zona di disponibilità. Le altre sottoreti pubbliche e private vengono distribuite in una seconda zona di disponibilità nella stessa regione. AWS Si consiglia questa opzione per la maggior parte delle implementazioni,

dal momento che consente di distribuire i nodi in sottoreti private. Questa opzione consente a Kubernetes di implementare i bilanciatori del carico nelle sottoreti pubbliche, in grado di bilanciare il carico del traffico nei pod in esecuzione sui nodi nelle sottoreti private. Gli indirizzi `IPv4` pubblici vengono assegnati automaticamente ai nodi implementati nelle sottoreti pubbliche, tuttavia gli indirizzi `IPv4` pubblici non vengono assegnati ai nodi implementati nelle sottoreti private.

È anche possibile assegnare indirizzi `IPv6` anche ai nodi nelle sottoreti pubbliche e private. I nodi nelle sottoreti private possono comunicare con il cluster e altri servizi. AWS I pod possono comunicare con Internet tramite un gateway NAT utilizzando gli indirizzi `IPv4` o tramite un gateway Internet in uscita utilizzando gli indirizzi `IPv6` implementati in ciascuna zona di disponibilità. Viene implementato un gruppo di sicurezza che blocca il traffico in ingresso proveniente da origini diverse rispetto al cluster o ai nodi, consentendo al contempo il traffico in uscita da qualsiasi origine.
Le sottoreti vengono contrassegnate in modo che Kubernetes possa implementare i sistemi di bilanciamento del carico.

a. Apri la AWS CloudFormation console.

b. Dalla barra di navigazione, seleziona una AWS regione che supporta Amazon EKS.

c. Scegli Crea stack, Con nuove risorse (standard).

d. In Prerequisite - Prepare template (Prerequisito . Preparazione del modello), assicurarsi che Template is ready (Il modello è pronto) sia selezionato e quindi in Specify template (Specifica modello), selezionare Amazon S3 URL(URL Amazon S3).

e. Puoi creare un VPC che supporta solo `IPv4` o un VPC che supporta sia `IPv4` che `IPv6`. Incolla uno dei seguenti elementi URLs nell'area di testo sotto l'URL di Amazon S3 e scegli Avanti:

- `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. Nella pagina Specifica i dettagli dello stack immetti i relativi parametri, quindi scegli Next (Successivo).

- Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, puoi utilizzare il nome del modello utilizzato nella fase precedente. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.
- VpcBlock: scegli una gamma `IPv4` CIDR per il tuo VPC. A ogni nodo, pod e bilanciatore del carico che viene implementato viene assegnato un indirizzo `IPv4` da questo blocco. I valori `IPv4` di default forniscono indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificare questo comportamento. Per ulteriori informazioni, consultare VPC e dimensionamento della sottorete nella Guida per l'utente di Amazon VPC. È inoltre possibile aggiungere ulteriori blocchi CIDR al VPC una volta creato. Se si crea un VPC `IPv6`, gli intervalli CIDR `IPv6` vengono assegnati automaticamente dallo spazio Global Unicast Address di Amazon.
- PublicSubnet01Block: specifica un blocco `IPv4` CIDR per la sottorete pubblica 1. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo. Se si crea un VPC `IPv6`, questo blocco viene specificato automaticamente all'interno del modello.

- PublicSubnet02Block: specifica un blocco `IPv4` CIDR per la sottorete pubblica 2. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo. Se si crea un VPC `IPv6`, questo blocco viene specificato automaticamente all'interno del modello.
- PrivateSubnet01Block: specifica un blocco `IPv4` CIDR per la sottorete privata 1. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo. Se si crea un VPC `IPv6`, questo blocco viene specificato automaticamente all'interno del modello.
- PrivateSubnet02Block: specifica un blocco `IPv4` CIDR per la sottorete privata 2. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo. Se si crea un VPC `IPv6`, questo blocco viene specificato automaticamente all'interno del modello.

b. (Facoltativo) Nella pagina Configure stack options (Configura opzioni stack), aggiungere tag alle risorse dello stack, quindi scegliere Next (Successivo).

c. Nella pagina Revisione, scegliere Crea pila.

d. Quando viene creata la pila, selezionala nella console e scegli Outputs (Uscite).

e. Registra il VpcIdper il VPC che è stato creato. Questo valore sarà necessario al momento della creazione del cluster e dei nodi.

f. Registra il SubnetIdsper le sottoreti che sono state create e se le hai create come sottoreti pubbliche o private. Saranno necessari almeno due valori al momento della creazione del cluster e dei nodi.

g. Se è stato creato un VPC `IPv4`, ignorare questo passaggio. Se è stato creato un VPC `IPv6`, è necessario abilitare l'opzione di assegnazione automatica dell'indirizzo `IPv6` per le sottoreti pubbliche create dal modello. Tale impostazione è già abilitata per le sottoreti private. Per abilitare l'impostazione, completare la procedura seguente:

   i. Apri la console Amazon VPC all'indirizzo. https://console.aws.amazon.com/vpc/

   ii. Nel pannello di navigazione a sinistra, seleziona Subnets (Sottoreti).

   iii. Seleziona una delle sottoreti pubbliche (***`stack-name`***/SubnetPublic01 o***`stack-name`***/SubnetPublic02 contiene la parola public) e scegli Azioni, Modifica impostazioni di sottorete.

   iv. Seleziona la casella di controllo Abilita l'assegnazione automatica IPv6 dell'indirizzo, quindi scegli Salva.

   v. Esegui nuovamente i passaggi precedenti per l'altra sottorete pubblica.

Solo sottoreti pubbliche

Questo VPC dispone di tre sottoreti pubbliche che vengono distribuite in diverse zone di disponibilità in una regione. AWS Tutti i nodi vengono assegnati automaticamente a indirizzi `IPv4` pubblici e possono inviare e ricevere traffico Internet tramite un gateway Internet. Viene implementato un gruppo di sicurezza che nega tutto il traffico in ingresso e consente tutto il traffico in uscita. Le sottoreti vengono contrassegnate in modo che Kubernetes possa implementare i sistemi di bilanciamento del carico.

a. Apri la AWS CloudFormation console.
b. Dalla barra di navigazione, seleziona una AWS regione che supporta Amazon EKS.
c. Scegliere Create stack (Crea pila), With new resources (standard) (Con nuove risorse (standard)).
d. In Preparazione del modello, assicurarsi che Template is ready (Il modello è pronto) sia selezionato e quindi in Origine del modello, selezionare Amazon S3 URL.
e. Incollare il seguente URL nell'area di testo URL Amazon S3 e scegliere Successivo:

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

a. Nella pagina Specify Details (Specifica dettagli) immetti i relativi parametri, quindi scegli Next (Successivo).
   - Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, è possibile chiamarlo *`amazon-eks-vpc-sample`*. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.
   - VpcBlock: scegli un blocco CIDR per il tuo VPC. A ogni nodo, pod e bilanciatore del carico che viene implementato viene assegnato un indirizzo `IPv4` da questo blocco. I valori `IPv4` di default forniscono indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificare questo comportamento. Per ulteriori informazioni, consultare VPC e dimensionamento della sottorete nella Guida per l'utente di Amazon VPC. È inoltre possibile aggiungere ulteriori blocchi CIDR al VPC una volta creato.
   - Subnet01Block: specifica un blocco CIDR per la sottorete 1. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.

- Subnet02Block: specifica un blocco CIDR per sottorete 2. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.
- Subnet03Block: specifica un blocco CIDR per sottorete 3. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.

b. (Facoltativo) Nella pagina Options (Opzioni), contrassegna con dei tag le risorse della pila. Scegli Next (Successivo).

c. Nella pagina Review (Revisione) scegli Create (Crea).

d. Quando viene creata la pila, selezionala nella console e scegli Outputs (Uscite).

e. Registra il VpcIdper il VPC che è stato creato. Questo valore sarà necessario al momento della creazione del cluster e dei nodi.

f. Registra il SubnetIdsper le sottoreti che sono state create. Saranno necessari almeno due valori al momento della creazione del cluster e dei nodi.

g. (Facoltativo) Qualsiasi cluster implementato su questo VPC è in grado di assegnare indirizzi `IPv4` privati ai tuoi pod e servizi. Se si desidera che i cluster implementati in questo VPC assegnino indirizzi `IPv6` privati a pod e servizi, è necessario apportare aggiornamenti al VPC, alla sottorete, alle tabelle di routing e ai gruppi di sicurezza. Per ulteriori informazioni, consulta Migrate existing VPCs from IPv4 to IPv6 in the Amazon VPC User Guide. Amazon EKS richiede che le sottoreti abbiano l'opzione degli indirizzi `Auto-assign IPv6` abilitata. Per impostazione predefinita, tale opzione è disattivata.

Solo sottoreti private

Questo VPC dispone di tre sottoreti private che vengono distribuite in diverse zone di disponibilità della regione. AWS Le risorse implementate nelle sottoreti non possono accedere a Internet, né viceversa. Il modello crea endpoint VPC utilizzando diversi AWS servizi a cui AWS PrivateLink i nodi in genere devono accedere. Per consentire ai nodi di accedere a Internet in uscita, puoi aggiungere un gateway NAT pubblico nella zona di disponibilità di ogni sottorete dopo la creazione del VPC. Viene creato un gruppo di sicurezza che blocca tutto il traffico in entrata, ad eccezione delle risorse implementate nelle sottoreti. Un gruppo di sicurezza consente inoltre tutto il traffico in uscita. Le sottoreti vengono contrassegnate in modo che Kubernetes possa implementare i sistemi di bilanciamento del carico interno. Se stai creando un VPC con questa configurazione, consulta the section called "Cluster privati" per ulteriori requisiti e considerazioni.

a. Apri la AWS CloudFormation console.

b. Dalla barra di navigazione, seleziona una AWS regione che supporta Amazon EKS.

c. Scegliere Create stack (Crea pila), With new resources (standard) (Con nuove risorse (standard)).

d. In Preparazione del modello, assicurarsi che Template is ready (Il modello è pronto) sia selezionato e quindi in Origine del modello, selezionare Amazon S3 URL.

e. Incollare il seguente URL nell'area di testo URL Amazon S3 e scegliere Successivo:

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-fully-private-vpc.yaml
```

a. Nella pagina Specify Details (Specifica dettagli), immetti i relativi parametri, quindi scegli Next (Successivo).

- Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, è possibile chiamarlo *`amazon-eks-fully-private-vpc`*. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.
- VpcBlock: scegli un blocco CIDR per il tuo VPC. A ogni nodo, pod e bilanciatore del carico che viene implementato viene assegnato un indirizzo `IPv4` da questo blocco. I valori `IPv4` di default forniscono indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificare questo comportamento. Per ulteriori informazioni, consultare VPC e dimensionamento della sottorete nella Guida per l'utente di Amazon VPC. È inoltre possibile aggiungere ulteriori blocchi CIDR al VPC una volta creato.
- PrivateSubnet01Block: specifica un blocco CIDR per la sottorete 1. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.
- PrivateSubnet02Block: specifica un blocco CIDR per la sottorete 2. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.
- PrivateSubnet03Block: specifica un blocco CIDR per la sottorete 3. Il valore di default fornisce indirizzi IP sufficienti per la maggior parte delle implementazioni. In caso contrario, è comunque possibile modificarlo.

b. (Facoltativo) Nella pagina Options (Opzioni), contrassegna con dei tag le risorse della pila. Scegli Next (Successivo).

c. Nella pagina Review (Revisione) scegli Create (Crea).

d. Quando viene creata la pila, selezionala nella console e scegli Outputs (Uscite).

e. Registra il VpcIdper il VPC che è stato creato. Questo valore sarà necessario al momento della creazione del cluster e dei nodi.

f. Registra il SubnetIdsper le sottoreti che sono state create. Saranno necessari almeno due valori al momento della creazione del cluster e dei nodi.

g. (Facoltativo) Qualsiasi cluster implementato su questo VPC è in grado di assegnare indirizzi `IPv4` privati ai tuoi pod e servizi. Se si desidera che i cluster implementati in questo VPC assegnino indirizzi `IPv6` privati a pod e servizi, è necessario apportare aggiornamenti al VPC, alla sottorete, alle tabelle di routing e ai gruppi di sicurezza. Per ulteriori informazioni, consulta Migrate existing VPCs from IPv4 to IPv6 in the Amazon VPC User Guide. Amazon EKS richiede che le sottoreti abbiano l'opzione degli indirizzi `Auto-assign IPv6`abilitata (disattivata per impostazione predefinita).

Visualizzazione dei requisiti relativi al gruppo di sicurezza Amazon EKS per cluster

In questo argomento vengono descritti i requisiti relativi al gruppo di sicurezza per un cluster Amazon EKS.

Gruppo di sicurezza del cluster predefinito

Quando si crea un cluster, Amazon EKS crea un gruppo di sicurezza denominato `eks-cluster-sg-`*`my-cluster`*`-`*`uniqueID`* . Questo gruppo di sicurezza presenta le regole predefinite seguenti:

Tipo di regola	Protocollo	Porte	Origine	Destinazione
In entrata	Tutti	Tutti	Personale	
In uscita	Tutti	Tutti		0.0.0.0/0(`IPv4`) o ::/0 (`IPv6`)
In uscita	Tutti	Tutti		Self (per il traffico EFA)

Il gruppo di sicurezza predefinito include una regola in uscita che consente il traffico di Elastic Fabric Adapter (EFA) con la destinazione dello stesso gruppo di sicurezza. Ciò abilita il traffico EFA all'interno del cluster, il che è vantaggioso per i carichi di lavoro IA/ML e calcolo ad alte prestazioni (HPC). Per ulteriori informazioni, consulta Elastic Fabric Adapter per carichi di lavoro IA/ML e HPC su Amazon EC2 nella Guida per l'utente di Amazon Elastic Compute Cloud.

> **Important**
>
> Se il cluster non necessita della regola in uscita, è possibile rimuoverla. Se la rimuovi, dovrai comunque avere le regole minime elencate in Limitazione del traffico del cluster. Se rimuovi la regola in entrata, Amazon EKS la ricrea ogni volta che il cluster viene aggiornato.

Amazon EKS aggiunge i seguenti tag al gruppo di sicurezza. Se rimuovi i tag, Amazon EKS li aggiunge nuovamente al gruppo di sicurezza ogni volta che il cluster viene aggiornato.

Chiave	Valore
`kubernetes.io/cluster/` *`my-cluster`*	`owned`
`aws:eks:cluster-name`	*`my-cluster`*
`Name`	`eks-cluster-sg-` *`my-cluster`* *`-uniqueid`*

Amazon EKS crea le risorse riportate di seguito e le associa automaticamente a questo gruppo di sicurezza:

- 2-4 interfacce di rete elastiche (indicate nel resto del documento come interfacce di rete) create insieme al cluster.
- Interfacce di rete dei nodi in qualsiasi gruppo di nodi gestito creato.

Le regole predefinite consentono il flusso del traffico tra il cluster e i nodi e il traffico in uscita verso qualsiasi destinazione. Quando crei un cluster, puoi specificare i gruppi di sicurezza personali se lo desideri. In tal caso, Amazon EKS associa i gruppi di sicurezza specificati alle interfacce di rete create per il cluster, Tuttavia, non li associa ai gruppi di nodi che crei.

È possibile specificare l'ID del gruppo di sicurezza del cluster inConsole di gestione AWS all'interno della sezione Reti. In alternativa, è possibile eseguire il comando AWS CLI seguente.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Limitazione del traffico cluster

Se è necessario limitare le porte aperte tra il cluster e i nodi, puoi rimuovere la regola in uscita predefinita e aggiungere le regole minime necessarie per il cluster. Se rimuovi la regola in ingresso predefinita, Amazon EKS la ricrea ogni volta che il cluster viene aggiornato.

Tipo di regola	Protocollo	Porta	Destinazione
In uscita	TCP	443	Gruppo di sicurezza del cluster
In uscita	TCP	10250	Gruppo di sicurezza del cluster
In uscita (DNS)	TCP e UDP	53	Gruppo di sicurezza del cluster

È inoltre necessario aggiungere regole per il traffico seguente:

- Qualsiasi protocollo e porta che si prevede di utilizzare per la comunicazione tra i nodi.
- Accesso a Internet in uscita, in modo che i nodi possano accedere alle API di Amazon EKS per l'introspezione del cluster e la registrazione dei nodi al momento del lancio. Se i nodi non hanno accesso a Internet, consulta Implementazione di cluster privati con accesso Internet limitato per ulteriori considerazioni.
- Accesso ai nodi per estrarre le immagini del container da Amazon ECR o da altre API dei registri del container da cui è necessario estrarre le immagini, ad esempio DockerHub. Per ulteriori informazioni, consultare AWS Intervalli di indirizzi IP nella sezione AWS Riferimenti generali.
- Accesso dei nodi ad Amazon S3.
- Sono necessarie regole separate per gli indirizzi `IPv4` e `IPv6`.

- Se utilizzi nodi ibridi, devi aggiungere un gruppo di sicurezza aggiuntivo al cluster per consentire la comunicazione con i nodi on-premises e i pod. Per ulteriori informazioni, consulta the section called "Preparazione della rete".

Se stai pianificando una limitazione delle regole, ti consigliamo di testare accuratamente tutti i pod prima di applicare le regole modificate a un cluster di produzione.

Se precedentemente hai implementato un cluster con Kubernetes `1.14` e una versione della piattaforma `eks.3` o precedente, considera quanto segue:

- È probabile che siano presenti anche piani di controllo (control-plane) e gruppi di sicurezza dei nodi. Al momento della creazione, questi gruppi contenevano le regole con restrizioni elencate nella tabella precedente, che adesso possono essere rimosse in quanto non più necessarie. Tuttavia, è necessario assicurarsi che il gruppo di sicurezza del cluster contenga le regole incluse in tali gruppi.
- Se il cluster è stato implementato utilizzando direttamente l'API o è stato creato tramite uno strumento come AWS CLI o AWS CloudFormation senza specificare alcun gruppo di sicurezza alla creazione di cluster, quindi il gruppo di sicurezza predefinito per VPC è stato applicato a interfacce di rete del cluster create da Amazon EKS.

Gruppi di sicurezza condivisi

Amazon EKS supporta gruppi di sicurezza condivisi.

- Le associazioni VPC dei gruppi di sicurezza associano i gruppi di sicurezza a più VPC nello stesso account e nella stessa regione.
  - Scopri come associare gruppi di sicurezza a più VPC nella Guida per l'utente di Amazon VPC.
- I gruppi di sicurezza condivisi consentono di condividere gruppi di sicurezza con altri account AWS. Gli account devono appartenere alla stessa organizzazione AWS.
  - Scopri come condividere i gruppi di sicurezza con le organizzazioni nella Guida per l'utente di Amazon VPC.
- I gruppi di sicurezza sono sempre limitati a una singola regione AWS.

Considerazioni per Amazon EKS

- EKS ha gli stessi requisiti dei gruppi di sicurezza condivisi o multi-VPC dei gruppi di sicurezza standard.

Gestione di componenti aggiuntivi di rete per i cluster Amazon EKS

Per il cluster Amazon EKS sono disponibili diversi componenti aggiuntivi di rete.

Componenti aggiuntivi integrati

> **Note**
>
> Quando crei un cluster EKS:
>
> - Utilizzo della AWS console: i componenti aggiuntivi integrati (come CoredNS, kube-proxy, ecc.) vengono installati automaticamente come componenti aggiuntivi Amazon EKS. Questi possono essere facilmente configurati e aggiornati tramite la AWS console, la CLI o. SDKs
> - Utilizzo di altri metodi (CLI SDKs, ecc.): gli stessi componenti aggiuntivi integrati vengono installati come versioni autogestite che vengono eseguite come normali distribuzioni Kubernetes. Questi richiedono configurazioni e aggiornamenti manuali poiché non possono essere gestiti tramite strumenti. AWS
>
> Consigliamo di utilizzare i componenti aggiuntivi di Amazon EKS anziché le versioni autogestite per semplificare la gestione dei componenti aggiuntivi e consentire la configurazione e gli aggiornamenti centralizzati tramite i servizi. AWS

Plug-in della CNI di Amazon VPC per Kubernetes

Questo componente aggiuntivo CNI crea interfacce di rete elastiche e le collega ai tuoi nodi Amazon. EC2 Il componente aggiuntivo, inoltre, assegna un indirizzo `IPv4` o `IPv6` privato dal VPC a ogni pod e servizio. Per impostazione predefinita, nel cluster è stato installato questo componente aggiuntivo. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC". Se utilizzi nodi ibridi, la CNI del VPC è ancora installata per impostazione predefinita, ma non può essere eseguita sui nodi ibridi con una regola di anti-affinità. Per ulteriori informazioni sulle opzioni della CNI per i nodi ibridi, consulta the section called "Configurazione della CNI".

CoreDNS

CoreDNS è un server DNS flessibile ed estensibile che può fungere da cluster DNS Kubernetes. CoreDNS fornisce la risoluzione dei nomi per tutti i pod del cluster. Per impostazione predefinita, nel cluster è stato installato questo componente aggiuntivo. Per ulteriori informazioni, consulta the section called "CoreDNS".

`kube-proxy`

Questo componente aggiuntivo mantiene le regole di rete sui tuoi EC2 nodi Amazon e consente la comunicazione di rete con i tuoi Pod. Per impostazione predefinita, nel cluster è stato installato questo componente aggiuntivo. Per ulteriori informazioni, consulta the section called "kube-proxy".

Componenti aggiuntivi di rete opzionali AWS

AWS Load Balancer Controller

Quando si distribuiscono oggetti di servizio Kubernetes di tipo diverso`loadbalancer`, il controller crea Network Load Balancer. AWS Quando crei oggetti di ingresso Kubernetes, il controller crea Application Load Balancer. AWS Consigliamo di utilizzare questo controller per il provisioning di Network Load Balancer, anziché utilizzare il controller Cloud Provider legacy integrato in Kubernetes. Per ulteriori informazioni, consultare la documentazione sul Controller del load balancer AWS.

AWS Gateway API Controller

Questo controller consente di connettere i servizi su più cluster Kubernetes utilizzando l'API gateway Kubernetes. Il controller collega i servizi Kubernetes in esecuzione su EC2 istanze Amazon, contenitori e funzioni serverless utilizzando il servizio Amazon VPC Lattice. Per ulteriori informazioni, consulta documentazione Controller API Gateway AWS.

Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS".

Assegna IPs ai pod con Amazon VPC CNI

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

> **Tip**
>
> Con la modalità automatica di Amazon EKS, non devi installare o aggiornare componenti aggiuntivi di rete. Auto Mode include funzionalità di pod networking e bilanciamento del carico.
> Per ulteriori informazioni, consulta Modalità automatica EKS.

Il plug-in CNI di Amazon VPC per il componente aggiuntivo Kubernetes viene implementato su ogni nodo Amazon EC2 del cluster Amazon EKS. Il componente aggiuntivo crea interfacce di rete elastiche e le collega ai nodi Amazon EC2. Il componente aggiuntivo, inoltre, assegna un indirizzo `IPv4` o `IPv6` privato dal VPC a ogni pod.

Una versione del componente aggiuntivo viene implementata con ogni nodo Fargate nel cluster, ma non viene aggiornata sui nodi Fargate. Altri plug-in CNI compatibili sono disponibili per l'uso sui cluster Amazon EKS, ma questo è l'unico plug-in CNI supportato da Amazon EKS per i nodi eseguiti sull'infrastruttura. AWS Per ulteriori informazioni sugli altri plug-in CNI compatibili, consulta the section called "Plug-in CNI alternativi". Il CNI di VPC non è supportato per l'utilizzo con nodi ibridi. Per ulteriori informazioni sulle opzioni della CNI per i nodi ibridi, consulta the section called "Configurazione della CNI".

La tabella seguente elenca la versione più recente del componente aggiuntivo di Amazon EKS disponibile per ogni versione di Kubernetes.

Versioni del CNI di Amazon VPC

Versione di Kubernetes	Tipo di VPC di Amazon EKS versione CNI
1,35	v1.21.1-eksbuild.3
1,34	v1.21.1-eksbuild.3
1.33	v1.21.1-eksbuild.3
1,32	v1.21.1-eksbuild.3
1.31	v1.21.1-eksbuild.3
1.30	v1.21.1-eksbuild.3

Versione di Kubernetes	Tipo di VPC di Amazon EKS versione CNI
1,29	v1.21.1-eksbuild.3

Important

Se gestisci autonomamente questo componente aggiuntivo, le versioni nella tabella potrebbero non essere le stesse di quelle autogestite disponibili. Per ulteriori informazioni sull'aggiornamento del componente aggiuntivo del tipo autogestito, consulta the section called "Aggiornamento (autogestito)".

Important

Per eseguire l'aggiornamento al CNI VPC versione v1.12.0 o successiva, devi prima eseguire l'aggiornamento alla versione v1.7.0. È consigliabile aggiornare solo una versione secondaria alla volta.

Considerazioni

Di seguito sono riportate alcune considerazioni sull'utilizzo della funzionalità.

- Le versioni sono specificate come `major-version.minor-version.patch-version-eksbuild.build-number`.
- Verifica della compatibilità delle versioni per le varie funzionalità. Alcune funzionalità di ogni versione del plug-in CNI di Amazon VPC per Kubernetes richiedono versioni specifiche di Kubernetes. Quando si utilizzano diverse funzionalità di Amazon EKS ed è necessaria una versione specifica del componente aggiuntivo, questa viene annotata nella documentazione relativa alle funzionalità. A meno che tu non abbia un motivo specifico per eseguire una versione più vecchia, consigliamo di eseguire la versione più recente.

Creazione di CNI di Amazon VPC (componente aggiuntivo di Amazon EKS)

Utilizza i seguenti passaggi per creare il plugin CNI di Amazon VPC per il componente aggiuntivo Kubernetes di Amazon EKS.

Prima di iniziare, esamina le seguenti informazioni. Per ulteriori informazioni, consulta the section called "Considerazioni".

Prerequisiti

Di seguito sono riportati i seguenti prerequisiti per il plugin CNI di Amazon VPC per Kubernetes di Amazon EKS.

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.
- Un provider di AWS Identity and Access Management (IAM) OpenID Connect (OIDC) esistente per il tuo cluster. Per determinare se si dispone già di un provider IAM o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
- Un ruolo IAM con la policy IAM Amazoneks_CNI_Policy (se il cluster utilizza la famiglia) o una policy (se il cluster utilizza `IPv4` la famiglia) ad esso allegata. IPv6 `IPv6` Per ulteriori informazioni sul ruolo CNI di VPC, consulta the section called "Configurare per IRSA". Per informazioni sulla politica, consulta. IPv6 the section called "Creare una policy IAM per i cluster che utilizzano la famiglia `IPv6`"

> **Important**
>
> Plugin CNI di Amazon VPC per versioni Kubernetes `v1.16.0` per implementare `v1.16.1` versioni `v1.0.0` con specifiche di CNI. Per ulteriori informazioni sulle `v1.0.0` specifiche CNI, vedere la specifica CNI (Container Network Interface) su. GitHub

Procedura

Una volta completati i seguenti prerequisiti, segui i passaggi di seguito per creare il componente aggiuntivo.

1. Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

   Di seguito viene riportato un output di esempio:

```
v1.16.4-eksbuild.2
```

2. Scopri qual è il tipo di componente aggiuntivo attualmente installato sul cluster. A seconda dello strumento con cui hai creato il cluster, al momento potresti non avere il componente aggiuntivo del tipo Amazon EKS installato sul cluster. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

   Se è restituito il numero di versione, sul cluster è installato il componente aggiuntivo del tipo Amazon EKS e non è necessario completare i passaggi rimanenti di questa procedura. Se è restituito un errore, sul cluster non è installato il componente aggiuntivo del tipo Amazon EKS. Completa i passaggi rimanenti di questa procedura per installarlo.

3. Salva la configurazione del componente aggiuntivo attualmente installato.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. Crea il componente aggiuntivo utilizzando la AWS CLI. Se desideri utilizzare Console di gestione AWS o `eksctl` creare il componente aggiuntivo, consulta the section called "Creare un componente aggiuntivo" e specifica il nome del componente `vpc-cni` aggiuntivo. Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato.
   - Sostituisci *my-cluster* con il nome del cluster.
   - *v1.20.3-eksbuild.1*Sostituiscilo con la versione più recente elencata nella tabella delle versioni più recenti per la versione del cluster. Per la tabella con la versione più recente, consulta the section called "Versioni del CNI di Amazon VPC".
   - Sostituiscilo *111122223333* con l'ID del tuo account e *AmazonEKSVPCCNIRole* con il nome di un ruolo IAM esistente che hai creato. Per specificare un ruolo è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC 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 hai applicato impostazioni personalizzate al tuo attuale componente aggiuntivo che sono in conflitto con le impostazioni predefinite del componente aggiuntivo di Amazon EKS, la creazione potrebbe non riuscire. Se la creazione non riesce, riceverai un errore che può aiutare a risolvere il problema. In alternativa, puoi aggiungere `--resolve-conflicts OVERWRITE` al

comando precedente. Ciò consente al componente aggiuntivo di sovrascrivere le impostazioni personalizzate esistenti. Una volta creato il componente aggiuntivo, è possibile aggiornarlo con le tue impostazioni personalizzate.

5. Verifica che la versione più recente del componente aggiuntivo per la versione del cluster sia stata aggiunta al cluster. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

La creazione del componente aggiuntivo potrebbe richiedere alcuni secondi.

Di seguito viene riportato un output di esempio.

```
v1.20.3-eksbuild.1
```

6. Se hai creato impostazioni personalizzate per il componente aggiuntivo originale, prima di creare il componente aggiuntivo di Amazon EKS, utilizza la configurazione che hai salvato precedentemente per aggiornare il componente aggiuntivo di Amazon EKS con le tue impostazioni personalizzate. Segui la procedura riportata in the section called "Aggiornamento (componente aggiuntivo EKS)".
7. (Facoltativo) Installa `cni-metrics-helper` sul tuo cluster. Raccoglie le informazioni sull'interfaccia di rete elastica e sull'indirizzo IP, le aggrega a livello di cluster e pubblica le metriche su Amazon. CloudWatch Per ulteriori informazioni, consulta on. cni-metrics-helper GitHub

Aggiornamento di CNI di Amazon VPC (componente aggiuntivo Amazon EKS)

Aggiorna il tipo Amazon EKS del componente aggiuntivo del plugin CNI di Amazon VPC per Kubernetes del componente aggiuntivo Amazon EKS. Se non hai aggiunto il tipo di componente aggiuntivo Amazon EKS al cluster, è possibile installarlo seguendo the section called "Crea".
In alternativa, aggiorna l'altro tipo di installazione CNI di VPC seguendo the section called "Aggiornamento (autogestito)".

1. Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

Di seguito viene riportato un output di esempio.

```
v1.20.0-eksbuild.1
```

Confronta la versione con la tabella delle versioni più recenti all'indirizzo the section called "Versioni del CNI di Amazon VPC". Se la versione restituita è la stessa della versione Kubernetes del cluster presente nella tabella delle versioni più recenti, la versione più recente è già installata nel cluster e non è necessario completare il resto di questa procedura. Se nell'output ricevi un errore invece di un numero di versione, il componente aggiuntivo del tipo Amazon EKS non è installato sul cluster. È necessario creare il componente aggiuntivo prima di poterlo aggiornare con questa procedura. Per creare il tipo Amazon EKS del componente aggiuntivo CNI di VPC, segui questa pagina the section called "Crea".

2. Salva la configurazione del componente aggiuntivo attualmente installato.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

3. Aggiorna il tuo componente aggiuntivo utilizzando la AWS CLI. Se desideri utilizzare Console di gestione AWS o `eksctl` aggiornare il componente aggiuntivo, consulta. the section called "Aggiornamento di un componente aggiuntivo" Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato.
   - Sostituisci *`my-cluster`* con il nome del cluster.
   - Sostituiscilo *`v1.20.0-eksbuild.1`* con la versione più recente elencata nella tabella delle versioni più recenti della versione del cluster.
   - Sostituiscilo *`111122223333`* con l'ID del tuo account e *`AmazonEKSVPCCNIRole`* con il nome di un ruolo IAM esistente che hai creato. Per creare un ruolo IAM per CNI di VPC, consulta the section called "Passaggio 1: Creazione di un ruolo IAM per il plug-in CNI di Amazon VPC per Kubernetes". Per specificare un ruolo è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".
   - L'opzione `--resolve-conflicts PRESERVE` mantiene i valori di configurazione esistenti per il componente aggiuntivo. Se sono stati configurati valori personalizzati per le impostazioni del componente aggiuntivo e non si utilizza questa opzione, Amazon EKS sovrascrive i valori personalizzati con quelli predefiniti. Se utilizzi questa opzione, è preferibile testare eventuali modifiche ai campi e ai valori su un cluster non di produzione prima di aggiornare il componente aggiuntivo sul cluster di produzione. Se modifichi questo valore in `OVERWRITE`,

tutte le impostazioni vengono modificate nei valori predefiniti di Amazon EKS. Se hai impostato valori personalizzati per un'impostazione qualsiasi, è possibile che siano sovrascritti con i valori predefiniti di Amazon EKS. Se modifichi questo valore in none, Amazon EKS non modifica il valore di alcuna impostazione, ma l'aggiornamento potrebbe non riuscire. Se l'aggiornamento non riesce, riceverai un messaggio di errore che ti aiuterà a risolvere il conflitto.

- Se non viene aggiornata un'impostazione di configurazione, rimuovere `--configuration-values '{`*`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`*`}'` dal comando. Se stai aggiornando un'impostazione di configurazione, sostituiscila *`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`* con l'impostazione che desideri impostare. In questo esempio, la variabile di ambiente AWS_VPC_K8S_CNI_EXTERNALSNAT è impostata su `true`. Il valore specificato deve essere valido per lo schema di configurazione. Se non conosci lo schema di configurazione`aws eks describe-addon-configuration --addon-name vpc-cni --addon-version` *`v1.20.0-eksbuild.1`* , eseguilo sostituendolo *`v1.20.0-eksbuild.1`* con il numero di versione del componente aggiuntivo di cui vuoi vedere la configurazione. Lo schema viene restituito nell'output. Se disponi di una configurazione personalizzata esistente, desideri rimuoverla tutta e ripristinare i valori per tutte le impostazioni ai valori predefiniti di Amazon EKS, rimuovila *`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`* dal comando, in modo che sia vuota. `{}` Per una spiegazione di ciascuna impostazione, consulta Variabili di configurazione CNI su. 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"}}'
```

Il completamento dell'aggiornamento potrebbe richiedere alcuni secondi.

4. Conferma che la versione del componente aggiuntivo sia stata aggiornata. Sostituisci *`my-cluster`* con il nome del cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

Il completamento dell'aggiornamento potrebbe richiedere alcuni secondi.

Di seguito viene riportato un output di esempio.

```
{
    "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
\"}}"
    }
}
```

Risoluzione dei problemi

Quando si aggiorna VPC CNI da una versione precedente alla v1.13.2, è necessario sostituire tutti i nodi del cluster dopo l'aggiornamento. Le versioni precedenti alla v1.13.2 utilizzano il backend iptables-legacy per inserire le regole iptables necessarie per il corretto funzionamento, come il codice sorgente NAT (SNAT).

La versione v1.13.2 è stata una release importante che ha introdotto iptables-wrapper, che rileva automaticamente il backend iptables appropriato (iptables-legacy o iptables-nft) per l'inserimento di catene e regole. Questa modifica è in linea con la decisione iniziale di Kubernetes di abbandonare il backend legacy a causa delle limitazioni delle prestazioni.

La sostituzione dei nodi dopo un aggiornamento da una versione precedente alla v1.13.2 di VPC CNI è necessaria perché l'introduzione di regole nei backend iptables-legacy e iptables-nft può portare a comportamenti imprevisti per il traffico proveniente da fonti non primarie. ENIs

Aggiornamento di CNI di Amazon VPC (componente aggiuntivo autogestito)

> **Important**
>
> Consigliamo di aggiungere al cluster il componente aggiuntivo del tipo Amazon EKS anziché quello del tipo autogestito. Se non sono ben chiare le differenze tra i due tipi, consulta the section called "Componenti aggiuntivi Amazon EKS". Per ulteriori informazioni sull'aggiunta di un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Se non riesci a utilizzare il componente aggiuntivo Amazon EKS, ti consigliamo di segnalare un problema sul motivo per cui non puoi farlo all'archivio della roadmap GitHub di Containers.

1. Verifica che sul cluster non sia installato il tipo di Amazon EKS del componente aggiuntivo. Sostituisci *my-cluster* con il nome del cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    addon.addonVersion --output text
   ```

   Se è restituito un messaggio di errore, sul cluster non è installato il componente aggiuntivo del tipo Amazon EKS. Per gestire automaticamente il componente aggiuntivo, completa i passaggi rimanenti di questa procedura per aggiornare il componente aggiuntivo. Se viene restituito il numero di versione, sul cluster è installato il tipo Amazon EKS del componente aggiuntivo. Per aggiornarlo, utilizza la procedura descritta in the section called "Aggiornamento di un componente aggiuntivo" anziché questa. Se non conosci bene le differenze tra i due tipi di componente aggiuntivo, consulta the section called "Componenti aggiuntivi Amazon EKS".
2. Scopri qual è la versione dell'immagine di container attualmente installata sul cluster.

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
    cut -d : -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.20.0-eksbuild.1
   ```

   L'output potrebbe non includere il numero di build.

3. Effettua il backup delle impostazioni correnti in modo da poter riprodurre la medesima configurazione dopo avere aggiornato la versione.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

Per esaminare le versioni disponibili e acquisire familiarità con le modifiche nella versione a cui desideri eseguire l'aggiornamento, consulta le versioni su. GitHub Tieni presente che ti consigliamo di eseguire l'aggiornamento alla stessa. `major` `minor`. `patch`versione elencata nella tabella delle ultime versioni disponibili, anche se le versioni successive sono disponibili su GitHub. Per la tabella delle versioni più recenti disponibili, consulta the section called "Versioni del CNI di Amazon VPC". Le versioni di build elencate nella tabella non sono specificate nelle versioni autogestite elencate in GitHub. Aggiorna la versione completando le attività in una delle seguenti opzioni:

- Se non disponi di impostazioni personalizzate per il componente aggiuntivo, esegui il comando sotto l'`To apply this release:`intestazione GitHub relativa alla versione a cui stai eseguendo l'aggiornamento.
- Se disponi di impostazioni personalizzate, scarica il file manifesto con il comando seguente. Passa *`https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.0/config/master/aws-k8s-cni.yaml`* all'URL della versione a GitHub cui stai effettuando l'aggiornamento.

```
curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.3/config/
master/aws-k8s-cni.yaml
```

Se necessario, modifica il manifesto con le impostazioni personalizzate del backup che hai effettuato in una fase precedente, quindi applica il file modificato al tuo cluster. Se i tuoi nodi non hanno accesso ai repository privati Amazon ECR di Amazon EKS da cui sono estratte le immagini (consulta le righe che iniziano con `image:` nel manifesto), dovrai scaricare le immagini, copiarle nel tuo repository e modificare il manifesto per estrarre le immagini dal repository. Per ulteriori informazioni, consulta the section called "Copia di un'immagine in un repository".

```
kubectl apply -f aws-k8s-cni.yaml
```

4. Verifica che la nuova versione sia ora installata nel cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Di seguito viene riportato un output di esempio:

```
v1.20.3
```

5. (Facoltativo) Installa `cni-metrics-helper` sul tuo cluster. Raccoglie le informazioni sull'interfaccia di rete elastica e sull'indirizzo IP, le aggrega a livello di cluster e pubblica le metriche su Amazon. CloudWatch Per ulteriori informazioni, consulta on. cni-metrics-helper GitHub

Configurare il plug-in CNI di Amazon VPC per l'utilizzo di IRSA

Il plug-in CNI di Amazon VPC per Kubernetes è il plug-in per reti Pod nei cluster Amazon EKS. Il plug-in è responsabile dell'allocazione degli indirizzi IP VPC sui pod Kubernetes e della configurazione delle reti necessarie per i Pod in ogni nodo.

> **Note**
>
> Il plug-in CNI di Amazon VPC supporta anche Identità di Pod di Amazon EKS. Per ulteriori informazioni, consulta the section called "Assegnazione di un ruolo IAM".

Il plug-in:

- Richiede le autorizzazioni di AWS Identity and Access Management (IAM). Se il cluster utilizza la `IPv4` famiglia, le autorizzazioni sono specificate nella politica `AmazonEKS_CNI_Policy` AWS gestita. Se il cluster utilizza la famiglia `IPv6`, le autorizzazioni devono essere aggiunte a una policy IAM creata dall'utente; per le istruzioni, consulta the section called "Creare una policy IAM per i cluster che utilizzano la famiglia `IPv6`". È possibile collegare questa policy al ruolo del nodo IAM Amazon EKS o a un ruolo IAM separato. Per istruzioni su come allegare la policy al ruolo IAM del nodo Amazon EKS, consulta the section called "Ruolo IAM del nodo". Si consiglia di assegnarlo a un ruolo separato, come descritto in questo argomento.
- Crea ed è configurato per l'utilizzo di un account di servizio Kubernetes denominato `aws-node` quando viene implementato. L'account di servizio è associato a un `clusterrole` Kubernetes di nome `aws-node`, a cui vengono assegnate le autorizzazioni Kubernetes richieste.

> **Note**
>
> I Pod per il plug-in CNI di Amazon VPC per Kubernetes hanno accesso alle autorizzazioni assegnate al ruolo IAM del nodo Amazon EKS, a meno che non si blocchi l'accesso al servizio di metadati a IMDS. Per ulteriori informazioni, consulta Limitazione dell'accesso al profilo dell'istanza assegnato al nodo worker.

- Richiede un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.
- Richiede un provider AWS Identity and Access Management (IAM) OpenID Connect (OIDC) esistente per il cluster. Per determinare se si dispone già di un provider IAM o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

Passaggio 1: Creazione di un ruolo IAM per il plug-in CNI di Amazon VPC per Kubernetes

1. Determinare la famiglia IP del cluster.

   ```
   aws eks describe-cluster --name my-cluster | grep ipFamily
   ```

   Di seguito viene riportato un output di esempio.

   ```
   "ipFamily": "ipv4"
   ```

   L'output potrebbe invece restituire `ipv6`.

2. Creare il ruolo IAM. Puoi usare `eksctl` or `kubectl` e la AWS CLI per creare il tuo ruolo IAM.

   eksctl

   - Creare un ruolo IAM e collegare ad esso la policy IAM con il comando corrispondente alla famiglia IP del cluster. Il comando crea e distribuisce uno AWS CloudFormation stack che crea un ruolo IAM, allega la policy specificata e annota l'account del servizio `aws-node` Kubernetes esistente con l'ARN del ruolo IAM creato.
     - `IPv4`

       Sostituire *`my-cluster`* con il proprio valore.

       ```
       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`

  Sostituire *`my-cluster`* con il proprio valore. Sostituisci *`111122223333`* con il tuo ID account e *`AmazonEKS_CNI_IPv6_Policy`* con il nome della policy `IPv6`. Se non disponi di una policy `IPv6`, consulta the section called "Creare una policy IAM per i cluster che utilizzano la famiglia `IPv6`" per crearne una. Per utilizzare `IPv6` con il cluster, questo deve soddisfare diversi requisiti. Per ulteriori informazioni, consulta 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 la CLI AWS

i. Visualizzare l'URL del provider OIDC del cluster.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

Di seguito viene riportato un output di esempio.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

Se non viene restituito alcun output, devi creare un provider OIDC IAM per il cluster.

ii. Copia i seguenti contenuti in un file denominato *`vpc-cni-trust-policy.json`*. Sostituisci *`111122223333`* con il tuo ID account e *`EXAMPLED539D4633E53DE1B71EXAMPLE`* con l'output restituito nel passaggio precedente. Sostituisci *`region-code`* con la AWS regione in cui si trova il 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",
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
node"
                }
            }
        }
    ]
}
```

iii. Crea il ruolo. Puoi sostituire *AmazonEKSVPCCNIRole* con un nome a tua scelta.

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

iv. Allega la policy IAM richiesta al ruolo. Esegui il comando corrispondente alla famiglia IP del cluster.

- IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSVPCCNIRole
```

- IPv6

  Sostituisci *111122223333* con il tuo ID account e *AmazonEKS_CNI_IPv6_Policy* con il nome della policy IPv6. Se non disponi di una policy IPv6, consulta the section

called "Creare una policy IAM per i cluster che utilizzano la famiglia IPv6" per crearne una. Per utilizzare IPv6 con il cluster, questo deve soddisfare diversi requisiti. Per ulteriori informazioni, consulta the section called "IPv6".

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSVPCCNIRole
```

v. Esegui il comando seguente per annotare l'account di servizio aws-node con l'ARN del ruolo IAM creato in precedenza. Sostituire i valori di esempio con i propri valori.

```
kubectl annotate serviceaccount \
    -n kube-system aws-node \
    eks.amazonaws.com/role-arn=arn:aws: iam::111122223333:role/
AmazonEKSVPCCNIRole
```

3. (Facoltativo) Configura il tipo di endpoint del servizio AWS Security Token utilizzato dal tuo account del servizio Kubernetes. Per ulteriori informazioni, consulta the section called "Endpoint STS".

Passaggio 2: Nuova implementazione dei Pod del plug-in CNI di Amazon VPC per Kubernetes

1. Eliminare e ricreare i Pod esistenti associati all'account di servizio per applicare le variabili di ambiente delle credenziali. L'annotazione non viene applicata ai Pod attualmente in esecuzione senza l'annotazione. Il comando seguente elimina i aws-node DaemonSet Pod esistenti e li distribuisce con l'annotazione dell'account di servizio.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

2. Confermare che tutti i Pod sono stati riavviati.

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. Descrivere uno dei Pod e verificare che le variabili di ambiente AWS_WEB_IDENTITY_TOKEN_FILE e AWS_ROLE_ARN esistano. Sostituiscilo *cpjw7* con il nome di uno dei tuoi Pod restituito nell'output del passaggio precedente.

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\|
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

Di seguito viene riportato un output di esempio.

```
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
```

Vengono restituiti due set di risultati duplicati perché il Pod contiene due container. I due container hanno gli stessi valori.

Se il Pod utilizza l'endpoint AWS regionale, nell'output precedente viene restituita anche la riga seguente.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Passaggio 3: rimozione della policy CNI dal ruolo IAM del nodo

Se al tuo ruolo IAM del nodo Amazon EKS è attualmente associata la IPv6 policy `AmazonEKS_CNI_Policy` IAM (`IPv4`) o una policy e hai creato un ruolo IAM separato, invece hai collegato la policy ad esso e l'hai assegnata all'account di servizio `aws-node` Kubernetes, ti consigliamo di rimuovere la policy dal tuo ruolo di nodo con il comando AWS CLI che corrisponde alla famiglia IP del tuo cluster. Sostituisci *`AmazonEKSNodeRole`* con il nome del ruolo del nodo.

- `IPv4`

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:
 iam::aws:policy/AmazonEKS_CNI_Policy
```

- `IPv6`

  Sostituisci *`111122223333`* con il tuo ID account e *`AmazonEKS_CNI_IPv6_Policy`* con il nome della policy `IPv6`.

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:
 iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

Creare una policy IAM per i cluster che utilizzano la famiglia **`IPv6`**

Se hai creato un cluster che utilizza la famiglia `IPv6` e sul cluster è configurata la versione `1.10.1` o successiva del componente aggiuntivo del plug-in CNI di Amazon VPC per Kubernetes, devi creare una policy IAM da assegnare a un ruolo IAM. Se disponi di un cluster che non è stato configurato con la famiglia `IPv6` al momento della creazione, prima di utilizzare `IPv6`, devi creare un nuovo cluster. Per ulteriori informazioni sull'uso di `IPv6` con il cluster, consulta the section called "IPv6".

1. Copia il testo seguente e salvalo in un file denominato `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. Creare la policy IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

Ulteriori informazioni sulle modalità e sulla configurazione di CNI di VPC

Il plugin CNI di Amazon VPC fornisce una rete per pod. Utilizza la tabella seguente per ulteriori informazioni sulle funzionalità di rete disponibili.

Caratteristica di rete	Ulteriori informazioni
Configura il cluster per assegnare indirizzi IPv6 a cluster, pod e servizi	the section called "IPv6"
Utilizza la traduzione degli indirizzi di rete di origine IPv4 per i pod	the section called "Traffico in uscita"
Limita il traffico di rete verso e dai pods	the section called "Limitare il traffico"
Personalizza l'interfaccia di rete secondaria nei nodi	the section called "Rete personalizzata"
Aumenta gli indirizzi IP per il nodo	the section called "Aumentare gli indirizzi IP"
Utilizza gruppi di sicurezza per il traffico di rete dei pod	the section called "Gruppi di sicurezza per pod"
Utilizza interfacce di rete multiple per pod	the section called "Interfacce multiple"

Scopri IPv6 gli indirizzi di cluster, pod e servizi

Si applica a: Pods con EC2 istanze Amazon e Fargate Pod

Per impostazione predefinita, Kubernetes assegna indirizzi `IPv4` a pod e servizi. Invece di assegnare indirizzi `IPv4` a pod e servizi, è possibile configurare il cluster in modo da assegnare loro indirizzi `IPv6`. Amazon EKS non supporta pod o servizi con doppio stack, anche se Kubernetes lo fa. Di conseguenza, non è possibile assegnare sia indirizzi `IPv4` che `IPv6` a pod e servizi.

È possibile selezionare la famiglia IP che si desidera utilizzare per il cluster al momento della creazione. Non sarà possibile modificare la famiglia dopo aver creato il cluster.

Per un tutorial sull'implementazione di un cluster `IPv6` di Amazon EKS, consultare the section called "Implementazione".

Di seguito sono riportate alcune considerazioni sull'utilizzo della funzionalità:

`IPv6` Supporto funzionalità

- Nessun supporto per Windows: i Pod e i servizi Windows non sono supportati.
- EC2 Nodi basati su Nitro: puoi utilizzarli solo `IPv6` con nodi AWS Amazon o EC2 Fargate basati su Nitro.
- EC2 e nodi Fargate supportati: puoi utilizzarlo `IPv6` con EC2 nodi the section called "Gruppi di sicurezza per pod" Amazon e nodi Fargate.
- Outpost non supportati: non è possibile usare `IPv6` con Amazon EKS su AWS Outposts.
- FSx for Lustre non è supportato: non the section called "Amazon FSx per Lustre" è supportato.
- Rete personalizzata non supportata: per ridurre l'esaurimento dell'indirizzo IP, è possibile utilizzare the section called "Rete personalizzata" anche se in precedenza è stato usato `IPv6`. Non è possibile utilizzare una rete personalizzata con `IPv6`. Se si utilizza una rete personalizzata per l'isolamento della rete, potrebbe essere necessario continuare a utilizzare la rete personalizzata e la famiglia `IPv4` per i cluster.

Assegnazioni dell'indirizzo IP

- Servizi Kubernetes: ai servizi Kubernetes viene assegnato solo un indirizzo `IPv6`. A loro non vengono assegnati IPv4 indirizzi.
- Pod: ai pod vengono assegnati un IPv6 indirizzo e un indirizzo locale dell'host IPv4 . L' IPv4 indirizzo locale dell'host viene assegnato utilizzando un plug-in CNI locale dell'host concatenato con VPC CNI e l'indirizzo non viene segnalato al piano di controllo di Kubernetes. Viene utilizzato solo quando un pod deve comunicare con IPv4 risorse esterne in un altro Amazon VPC o su Internet. L' IPv4 indirizzo locale dell'host arriva SNATed (tramite VPC CNI) all' IPv4 indirizzo principale dell'ENI primario del nodo di lavoro.
- Pod e servizi: i pod e i servizi ricevono solo indirizzi `IPv6`, non quelli `IPv4`. Quando i Pod devono comunicare con endpoint `IPv4` esterni, utilizzano il NAT sul nodo stesso. Questa funzionalità NAT integrata elimina la necessità di e. DNS64 NAT64 Per il traffico che richiede un accesso pubblico a Internet, il traffico del pod è l'indirizzo di rete di origine tradotto in un indirizzo IP pubblico.
- Indirizzi di instradamento: quando un Pod comunica all'esterno del VPC, il suo indirizzo `IPv6` originale viene preservato (non tradotto nell'indirizzo `IPv6` del nodo). Questo traffico viene instradato direttamente attraverso un gateway Internet o gateway Internet egress-only.
- Nodi: a tutti i nodi viene assegnato un indirizzo `IPv4` e `IPv6`.

- Pod Fargate: ogni Pod Fargate riceve un indirizzo `IPv6` dal CIDR specificato per la sottorete in cui è stato implementato. L'unità hardware sottostante che esegue i Fargate Pods riceve un `IPv6` indirizzo univoco `IPv4` e assegnato alla sottorete in cui è installata l'unità hardware. CIDRs

Come utilizzare **`IPv6`** con EKS

- Creare un nuovo cluster: è necessario creare un nuovo cluster e specificare che si vuole utilizzare la famiglia `IPv6` per quel cluster. Non è possibile abilitare la famiglia `IPv6` per un cluster aggiornato da una versione precedente. Per istruzioni su come creare un nuovo cluster, consultare Considerations.
- Usare un CNI di VPC recente: implementare la versione CNI di Amazon VPC o successiva `1.10.1`. Per impostazione predefinita, viene distribuita questa versione o una versione successiva. Dopo aver implementato il componente aggiuntivo, non sarà possibile eseguire il downgrade del componente aggiuntivo CNI di Amazon VPC a una versione inferiore a `1.10.1` senza rimuovere tutti i nodi in tutti i gruppi di nodi nel cluster.
- Configura VPC CNI per **`IPv6`**: Se utilizzi EC2 nodi Amazon, devi configurare il componente aggiuntivo Amazon VPC CNI con delega del prefisso IP e. `IPv6` Se durante la creazione del cluster si sceglie la famiglia `IPv6`, la versione `1.10.1` del componente aggiuntivo utilizza per impostazione predefinita questa configurazione. Questo è il caso di un componente aggiuntivo autogestito o Amazon EKS. Per ulteriori informazioni sulla delega dei prefissi IP, consultare the section called "Aumentare gli indirizzi IP".
- Configurare indirizzi **`IPv4`** e **`IPv6`**: quando si crea un cluster, il VPC e le sottoreti specificate devono avere un intervallo CIDR `IPv6` assegnato al VPC e alle sottoreti specificate. Devono inoltre disporre di un blocco CIDR `IPv4` assegnato. Questo perché, anche se si desidera utilizzare solo `IPv6`, il funzionamento di un VPC richiede comunque un blocco CIDR `IPv4`. Per ulteriori informazioni, consulta Associare un blocco IPv6 CIDR al tuo VPC nella Amazon VPC User Guide.
- Assegna automaticamente IPv6 gli indirizzi ai nodi: quando crei i nodi, devi specificare le sottoreti configurate per l'assegnazione automatica degli indirizzi. `IPv6` In caso contrario, non sarà possibile implementare i nodi. Per impostazione predefinita, questa configurazione è disabilitata. Per ulteriori informazioni, consulta Modificare l'attributo di IPv6 indirizzamento per la sottorete nella Amazon VPC User Guide.
- Impostare le tabelle di routing per usare **`IPv6`**: le tabelle di routing assegnate alle sottoreti devono avere percorsi per gli indirizzi `IPv6`. Per ulteriori informazioni, consulta Migrare a IPv6 nella Guida per l'utente di Amazon VPC.

- Impostare gruppi di sicurezza per **IPv6**: i gruppi di sicurezza devono consentire gli indirizzi IPv6. Per ulteriori informazioni, consulta Migrare a IPv6 nella Guida per l'utente di Amazon VPC.
- Configura il load balancer: utilizza la versione 2.3.1 o successiva del Load Balancer Controller per bilanciare il AWS carico delle applicazioni HTTP utilizzando il traffico di rete o utilizzando the section called “Bilanciamento del carico di applicazione” i IPv6 to Pods con entrambi i sistemi di bilanciamento the section called “Bilanciamento del carico di rete” del carico in modalità IP, ma non in modalità istanza. Per ulteriori informazioni, consulta the section called “ AWS Controller Load Balancer”.
- Aggiungere una policy IAM **IPv6**: è necessario allegare una policy IAM IPv6 al ruolo IAM del nodo o del CNI. Tra i due, consigliamo di allegarlo a un ruolo IAM CNI. Per ulteriori informazioni, consultare the section called “Creare una policy IAM per i cluster che utilizzano la famiglia IPv6” e the section called “Passaggio 1: Creazione di un ruolo IAM per il plug-in CNI di Amazon VPC per Kubernetes”.
- Valuta tutti i componenti: esegui una valutazione approfondita delle tue applicazioni, dei componenti aggiuntivi di Amazon EKS e AWS dei servizi con cui ti integri prima di distribuire IPv6 i cluster. Questo per garantire che tutto funzioni come previsto con IPv6.

Implementare un cluster **IPv6** Amazon EKS e nodi Amazon Linux gestiti

In questo tutorial, distribuisci un IPv6 Amazon VPC, un cluster Amazon EKS con IPv6 la famiglia e un gruppo di nodi gestiti con nodi Amazon EC2 Amazon Linux. Non puoi distribuire nodi Amazon EC2 Windows in un IPv6 cluster. È inoltre possibile implementare nodi Fargate nel cluster, sebbene tali istruzioni non siano fornite in questo argomento per semplicità.

Prerequisiti

Prima di iniziare questo tutorial, completare le seguenti operazioni:

Installare e configurare gli strumenti e le risorse elencati di seguito, che sono necessari per creare e gestire un cluster Amazon EKS.

- # consigliabile acquisire familiarità con tutte le impostazioni e implementare un cluster con le impostazioni che soddisfano i requisiti specifici. Per ulteriori informazioni, consultare the section called “Creazione di un cluster ”, the section called “Gruppi di nodi gestiti” e le considerazioni per questo argomento. Alcune impostazioni possono essere abilitate solo quando viene creato il cluster.

- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Il responsabile della sicurezza IAM che stai utilizzando deve disporre delle autorizzazioni per lavorare con i ruoli IAM di Amazon EKS, i ruoli collegati ai servizi AWS CloudFormation, un VPC e le risorse correlate. Per ulteriori informazioni, consultare le pagine Actions e Using service-linked roles nella Guida per l'utente di IAM.
- Se si utilizza eksctl, installare la versione `0.215.0` o successiva sul computer. Per installarlo o aggiornarlo, consulta la sezione Installation nella documentazione di `eksctl`.
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente. Se si utilizza la AWS CloudShell, potrebbe essere necessario installare la versione 2.12.3 o successiva o 1.27.160 o successiva della CLI AWS , poiché la versione CLI predefinita installata in potrebbe essere una versione precedente. AWS AWS CloudShell

È possibile utilizzare eksctl o la CLI per implementare un cluster `IPv6`.

Distribuisci un cluster con eksctl IPv6

a. Crea il file `ipv6-cluster.yaml`. Copia il comando seguente sul tuo dispositivo. Apportare le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:
   - Sostituisci *`my-cluster`* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.

- Sostituisci *region-code* con qualsiasi AWS regione supportata da Amazon EKS. Per un elenco di AWS regioni, consulta gli endpoint e le quote di Amazon EKS nella guida di riferimento AWS generale.
- Il valore per `version` con la versione del tuo cluster. Per ulteriori informazioni, consultare la pagina Amazon EKS supported versions.
- Sostituisci *my-nodegroup* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura.
- Sostituisci *t3.medium* con qualsiasi tipo di istanza di 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. Creare il cluster.

```
eksctl create cluster -f ipv6-cluster.yaml
```

La creazione di cluster richiede diversi minuti. Non procedere fino a quando non viene visualizzata l'ultima riga di output, simile alla seguente.

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

c. Confermare che ai pod predefiniti siano assegnati indirizzi IPv6.

```
kubectl get pods -n kube-system -o wide
```

Di seguito viene riportato un output di esempio.

```
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. Confermare che ai servizi di default siano assegnati indirizzi IPv6.

```
kubectl get services -n kube-system -o wide
```

Di seguito viene riportato un output di esempio.

```
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. (Facoltativo) Implementare un'applicazione di esempio o implementare AWS Load Balancer Controller e un'applicazione di esempio per eseguire il bilanciamento del carico delle applicazioni HTTP con the section called "Bilanciamento del carico di applicazione" o del traffico di rete con the section called "Bilanciamento del carico di rete" verso i pod IPv6.

f. Una volta terminato con il cluster e i nodi creati per questo tutorial, è necessario eliminare le risorse create con il seguente comando.

```
eksctl delete cluster my-cluster
```

Implementa un IPv6 cluster con CLI AWS

> **Important**
>
> - È necessario che tutti i passaggi di questa guida siano completati dallo stesso utente. Esegui il comando seguente per controllare l'utente corrente:
>
>   ```
>   aws sts get-caller-identity
>   ```
>
> - È necessario completare tutti i passaggi di questa procedura nella stessa shell. Diversi passaggi utilizzano le variabili impostate nelle fasi precedenti. Le fasi che utilizzano le variabili non funzioneranno correttamente se i relativi valori sono impostati in una shell diversa. Se utilizzate il AWS CloudShellper completare la procedura seguente, ricordate che se non interagite con esso utilizzando la tastiera o il puntatore per circa 20-30 minuti, la sessione di shell termina. I processi in esecuzione non sono considerati interazioni.
> - Le istruzioni sono scritte per la shell Bash e per altre shell potrebbero essere necessarie ulteriori regolazioni.

Sostituire tutti i valori di esempio nelle fasi di questa procedura con i propri valori.

a. Emettere i seguenti comandi per impostare alcune variabili utilizzate nelle fasi successive. Sostituiscilo *region-code* con la AWS regione in cui desideri distribuire le risorse. Il valore può essere qualsiasi AWS regione supportata da Amazon EKS. Per un elenco di AWS regioni, consulta gli endpoint e le quote di Amazon EKS nella guida di riferimento AWS generale. Sostituisci *my-cluster* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster. Sostituisci *my-nodegroup* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura. Sostituisci *111122223333* con l'ID del tuo account.

```
export region_code=region-code
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
```

b. Creare un Amazon VPC con sottoreti pubbliche e private che soddisfino i requisiti Amazon EKS e IPv6.

   i. Esegui il comando seguente per impostare una variabile per il nome AWS CloudFormation dello stack. Puoi sostituire *my-eks-ipv6-vpc* con un nome a tua scelta.

   ```
   export vpc_stack_name=my-eks-ipv6-vpc
   ```

   ii. Crea un IPv6 VPC utilizzando un AWS CloudFormation modello.

   ```
   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
   ```

   La creazione dello stack richiede alcuni minuti. Eseguire il seguente comando seguente. Non passare alla fase successiva finché l'output del comando non è CREATE_COMPLETE.

   ```
   aws cloudformation describe-stacks --region $region_code --stack-name
    $vpc_stack_name --query Stacks[].StackStatus --output text
   ```

   iii. Recupera le IDs sottoreti pubbliche che sono state create.

```
aws cloudformation describe-stacks --region $region_code --stack-name
 $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
 text
```

Di seguito viene riportato un output di esempio.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE
```

iv. Abilitare l'opzione di assegnazione automatica dell'indirizzo IPv6 per le sottoreti pubbliche create.

```
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. Recupera i nomi delle sottoreti e dei gruppi di sicurezza creati dal modello dallo AWS CloudFormation stack distribuito e memorizzali in variabili da utilizzare in un passaggio successivo.

```
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. Crea un ruolo IAM del cluster e collegalo alla policy gestita IAM richiesta dall'AMI per Amazon EKS. I cluster Kubernetes gestiti da Amazon EKS effettuano chiamate ad altri AWS servizi per tuo conto per gestire le risorse che utilizzi con il servizio.

i. Per creare il file `eks-cluster-role-trust-policy.json`, emetti il seguente comando:

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

ii. Eseguire questo comando per impostare una variabile per il nome del ruolo. Puoi sostituire *myAmazonEKSClusterRole* con un nome a tua scelta.

```
export cluster_role_name=myAmazonEKSClusterRole
```

iii. Crea il ruolo.

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

iv. Recuperare l'ARN del ruolo IAM e memorizzarlo in una variabile per una fase successiva.

```
CLUSTER_IAM_ROLE=$(aws iam get-role --role-name $cluster_role_name --
query="Role.Arn" --output text)
```

v. Allega la policy IAM gestita da Amazon EKS richiesta al ruolo.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

d. Creare il 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. NOTA: è possibile ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

   La creazione del cluster richiede diversi minuti. Eseguire il seguente comando seguente. Non passare alla fase successiva fino a quando l'output del comando non è `ACTIVE`.

```
aws eks describe-cluster --region $region_code --name $cluster_name --query
 cluster.status
```

e. Creare o aggiornare un file `kubeconfig` per il cluster in modo che sia possibile comunicare con il cluster.

```
aws eks update-kubeconfig --region $region_code --name $cluster_name
```

   Per impostazione predefinita, il file `config` viene creato in `~/.kube` o la configurazione del nuovo cluster viene aggiunta a un file `config` esistente in `~/.kube`.

f. Creare un ruolo IAM del nodo.

   i. Per creare il file `vpc-cni-ipv6-policy.json`, emetti il seguente comando:

```
{
    "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. Creare la policy IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

iii. Per creare il file `node-role-trust-relationship.json`, emetti il seguente comando:

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

iv. Eseguire questo comando per impostare una variabile per il nome del ruolo. Puoi sostituire *`AmazonEKSNodeRole`* con un nome a tua scelta.

```
export node_role_name=AmazonEKSNodeRole
```

v. Creare il ruolo IAM.

```
aws iam create-role --role-name $node_role_name --assume-role-policy-document
 file://"node-role-trust-relationship.json"
```

vi. Allegare la policy IAM al ruolo IAM.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy \
    --role-name $node_role_name
```

> **Important**
>
> Per semplicità, in questo tutorial la policy è allegata a questo ruolo IAM. Tuttavia, in un cluster di produzione, si consiglia di allegare la policy a un ruolo IAM separato. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".

vii. Allegare al ruolo IAM le due policy gestite IAM richieste.

```
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
```

viii Recuperare l'ARN del ruolo IAM e memorizzarlo in una variabile per una fase successiva.

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --query="Role.Arn" --
output text)
```

g. Creare un gruppo di nodi gestito.

i. Visualizza le IDs sottoreti che hai creato in un passaggio precedente.

```
echo $subnets
```

Di seguito viene riportato un output di esempio.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE,subnet-0377963d69EXAMPLE,subnet-0c05f819
```

ii. Creare il gruppo di nodi. Sostituisci *0a1a56c486EXAMPLE*, *099e6ca77aEXAMPLE*, *0377963d69EXAMPLE*, and *0c05f819d5EXAMPLE* con i valori restituiti nell'output della fase precedente. Assicurati di rimuovere le virgole tra le sottoreti IDs dall'output precedente nel comando seguente. Puoi sostituire *t3.medium* con qualsiasi tipo di istanza di 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
```

La creazione del gruppo di nodi richiede alcuni minuti. Eseguire il seguente comando seguente. Non passare alla fase successiva finché l'output restituito non è `ACTIVE`.

```
aws eks describe-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --query nodegroup.status --output text
```

h. Confermare che ai pod predefiniti siano assegnati indirizzi `IPv6` nella colonna `IP`.

```
kubectl get pods -n kube-system -o wide
```

Di seguito viene riportato un output di esempio.

```
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. Confermare che ai servizi di default siano assegnati gli indirizzi `IPv6` nella colonna `IP`.

```
kubectl get services -n kube-system -o wide
```

Di seguito viene riportato un output di esempio.

```
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. (Facoltativo) Implementare un'applicazione di esempio o implementare AWS Load Balancer Controller e un'applicazione di esempio per eseguire il bilanciamento del carico delle applicazioni HTTP con the section called "Bilanciamento del carico di applicazione" o del traffico di rete con the section called "Bilanciamento del carico di rete" verso i pod `IPv6`.

k. Una volta terminato con il cluster e i nodi creati per questo tutorial, è necessario eliminare le risorse create con i seguenti comandi. Assicurarsi di non utilizzare nessuna delle risorse al di fuori di questo tutorial prima di eliminarle.

i. Se si sta completando questo passaggio in una shell diversa da quella in cui sono state completate le fasi precedenti, impostare i valori di tutte le variabili utilizzate nelle fasi precedenti sostituendo i valori di esempio con quelli specificate durante le fasi precedenti. Se si sta completando questo passaggio nella stessa shell in cui sono state completate le fasi precedenti, andare alla fase successiva.

```
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. Eliminare il gruppo di nodi.

```
aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name
```

L'eliminazione richiede pochi minuti. Eseguire il seguente comando seguente. Non passare alla fase successiva se viene restituito un output.

```
aws eks list-nodegroups --region $region_code --cluster-name $cluster_name --query
 nodegroups --output text
```

iii. Elimina il cluster.

```
aws eks delete-cluster --region $region_code --name $cluster_name
```

L'eliminazione del cluster richiede alcuni minuti. Prima di continuare, accertarsi che il cluster sia eliminato con il comando seguente:

```
aws eks describe-cluster --region $region_code --name $cluster_name
```

Non procedere alla fase successiva finché l'output non è simile al seguente:

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-cluster.
```

iv. Eliminare le risorse IAM create. Sostituisci *AmazonEKS_CNI_IPv6_Policy* con il nome scelto, se è stato scelto un nome diverso da quello usato nei passaggi precedenti.

```
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. Elimina lo AWS CloudFormation stack che ha creato il VPC.

```
aws cloudformation delete-stack --region $region_code --stack-name $vpc_stack_name
```

Abilitare l'accesso a Internet in uscita per i pod

Si applica a: nodi Fargate Linux `IPv4`, nodi Linux con istanze Amazon EC2

Le informazioni contenute in questa sezione non sono applicabili al cluster se l'implementazione è avvenuta tramite la famiglia `IPv6`, in quanto gli indirizzi `IPv6` non sono convertiti in rete. Per ulteriori informazioni sull'uso di `IPv6` con il cluster, consulta the section called "IPv6".

Per impostazione predefinita, a ciascun pod nel cluster viene assegnato un indirizzo `IPv4` privato da un routing interdominio senza classi (CIDR) associato al VPC in cui è implementato il pod. I pod nello stesso VPC comunicano tra loro utilizzando questi indirizzi IP privati come endpoint. Quando un pod comunica a un qualsiasi indirizzo `IPv4` che non è incluso in un intervallo CIDR associato al VPC, il plugin CNI di Amazon VPC (per Linux o Windows) converte l'indirizzo `IPv4` del pod nell'indirizzo `IPv4` privato principale dell'interfaccia di rete elastica principale del nodo su cui è in esecuzione il pod, ovvero * per impostazione predefinita.

> **Note**
>
> Per i nodi Windows, ci sono ulteriori dettagli da considerare. Per impostazione predefinita, il plugin CNI VPC per Windows è definito con una configurazione di rete in cui il traffico verso una destinazione all'interno dello stesso VPC è escluso per SNAT. Ciò significa che la comunicazione VPC interna ha SNAT disabilitata e l'indirizzo IP assegnato al pod è indirizzabile all'interno del VPC. Tuttavia, il traffico verso una destinazione esterna al VPC ha l'IP di origine del pod tradotto tramite SNAT nell'indirizzo IP primario dell'interfaccia di rete elastica (ENI) dell'istanza. Questa configurazione predefinita per Windows garantisce che il pod possa accedere alle reti esterne al VPC allo stesso modo dell'istanza host.

A causa di questo comportamento:

- I pod possono comunicare con le risorse Internet solo se il nodo su cui sono in esecuzione dispone di un indirizzo IP pubblico o elastico a esso assegnato e si trova in una sottorete pubblica. La tabella di routing associata a una sottorete pubblica dispone di un instradamento verso un gateway

Internet. Per questo motivo, ti consigliamo di implementare i nodi nelle sottoreti private, quando possibile.

- Per le versioni del plugin precedenti alla `1.8.0`, le risorse che si trovano nelle reti o nei VPC collegati al VPC del cluster tramite peering VPC, VPC di transito o AWS Direct Connect non possono avviare una comunicazione con i pod oltre le interfacce di rete elastiche secondarie. I pod possono tuttavia avviare comunicazioni con tali risorse e ricevere risposte.

Se una delle seguenti affermazioni è vera nel tuo ambiente, modifica la configurazione predefinita con il comando che segue.

- Si dispone di risorse nelle reti o nei VPC collegati al VPC del cluster tramite peering VPC, VPC di transito o AWS Direct Connect che devono avviare la comunicazione con i pod tramite un indirizzo `IPv4`, con la versione del plugin precedente alla `1.8.0`.
- I pod si trovano in una sottorete privata e devono comunicare in uscita a Internet. La sottorete dispone di una route a un gateway NAT.

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

> **Note**
>
> Le variabili di configurazione `AWS_VPC_K8S_CNI_EXTERNALSNAT` e `AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS` di CNI non sono applicabili ai nodi Windows. La disabilitazione di SNAT non è supportata per Windows. Per quanto riguarda l'esclusione di un elenco di CIDR `IPv4` da SNAT, è possibile definirla specificando il parametro `ExcludedSnatCIDRs` nello script di bootstrap di Windows. Per ulteriori informazioni su questo parametro, consulta la sezione the section called "Parametri di configurazione dello script di bootstrap".

Rete host

*Se la specifica di un pod contiene `hostNetwork=true` (il valore predefinito è `false`), il suo indirizzo IP non viene convertito in un indirizzo diverso. Per impostazione predefinita, questo comportamento è presente per i pod di `kube-proxy` e del plugin CNI di Amazon VPC per Kubernetes in esecuzione sul cluster. Per questi pod, l'indirizzo IP corrisponde a quello principale del nodo, poiché l'indirizzo IP del pod non viene convertito. Per ulteriori informazioni sull'impostazione

`hostNetwork` dei pod, consultare PodSpec v1 core nella documentazione di riferimento dell'API Kubernetes.

Limita il traffico di Pod con le policy di rete di Kubernetes

Panoramica di

Per impostazione predefinita, in Kubernetes non ci sono restrizioni per indirizzi IP, porte o connessioni tra qualsiasi Pod nel cluster o tra i pod e le risorse in qualsiasi altra rete. È possibile utilizzare la policy di rete di Kubernetes per limitare il traffico di rete da e verso i pod. Per ulteriori informazioni, consulta Network Policies nella documentazione Kubernetes.

Politica di rete standard

È possibile utilizzare lo standard `NetworkPolicy` per segmentare il pod-to-pod traffico nel cluster. Queste policy di rete operano ai livelli 3 e 4 del modello di rete OSI, consentendoti di controllare il flusso di traffico a livello di indirizzo IP o di porta all'interno del tuo cluster Amazon EKS. Le policy di rete standard sono limitate al livello di namespace.

Casi d'uso

- Segmenta il traffico di rete tra i carichi di lavoro per garantire che solo le applicazioni correlate possano comunicare tra loro.
- Isola i tenant a livello di namespace utilizzando policy per imporre la separazione della rete.

Esempio

Nella policy riportata di seguito, il traffico in uscita dai pod delle webapp nello spazio dei nomi sun è limitato.

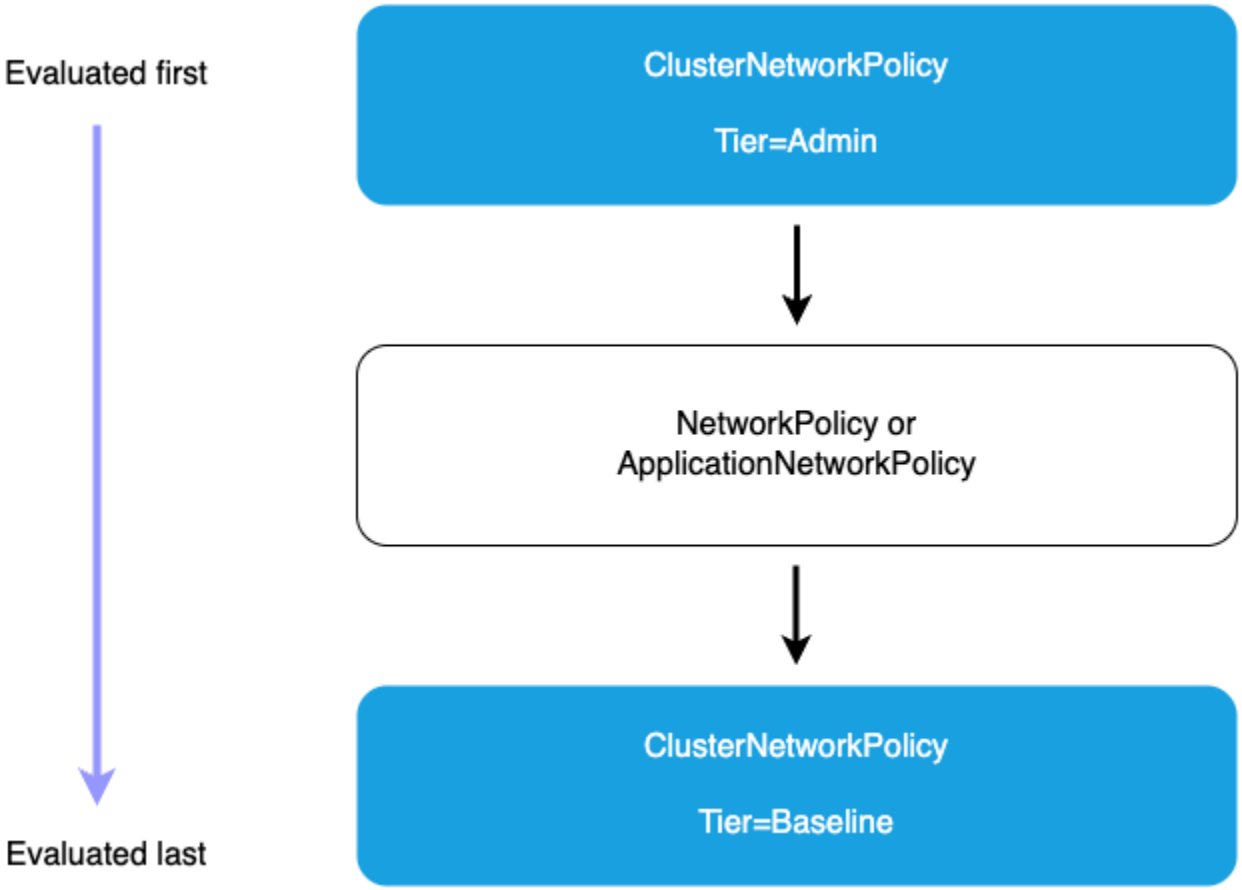
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: webapp-egress-policy
  namespace: sun
spec:
  podSelector:
    matchLabels:
      role: webapp
  policyTypes:
```

```
  - 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
```

La politica si applica ai pod con l'etichetta nello spazio dei nomi. `role: webapp` `sun`

- Traffico consentito: pod con l'etichetta `role: frontend` nello spazio dei nomi sulla porta TCP `moon` `8080`
- Traffico consentito: pod con il ruolo di etichetta: frontend nello spazio dei nomi sulla porta TCP `stars` `8080`
- Traffico bloccato: tutto il resto del traffico in uscita dai pod viene negato implicitamente `webapp`

Politica di rete dell'amministratore (o del cluster)



È possibile utilizzare il `ClusterNetworkPolicy` per applicare uno standard di sicurezza di rete che si applica all'intero cluster. Invece di definire e mantenere ripetutamente una policy distinta per ogni namespace, è possibile utilizzare un'unica policy per gestire centralmente i controlli di accesso alla rete per i diversi carichi di lavoro del cluster, indipendentemente dal relativo namespace.

Casi d'uso

- Gestisci centralmente i controlli di accesso alla rete per tutti (o un sottoinsieme di) carichi di lavoro nel tuo cluster EKS.
- Definisci una posizione di sicurezza di rete predefinita in tutto il cluster.
- Estendi gli standard di sicurezza organizzativi all'ambito del cluster in un modo più efficiente dal punto di vista operativo.

Esempio

Nella politica riportata di seguito, puoi bloccare in modo esplicito il traffico del cluster proveniente da altri namespace per impedire l'accesso di rete a uno spazio dei nomi di carico di lavoro riservato.

```
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
```

Note importanti

Le politiche di rete nel plug-in Amazon VPC CNI per Kubernetes sono supportate nelle configurazioni elencate di seguito.

- Versione 1.21.0 (o successiva) del plug-in Amazon VPC CNI per policy di rete standard e amministrative.
- Cluster configurato per indirizzi `IPv4` o `IPv6`.
- È possibile utilizzare le policy di rete con i gruppi di sicurezza per pod. Con le policy di rete, è possibile controllare tutte le comunicazioni all'interno del cluster. Con i gruppi di sicurezza per Pods, puoi controllare l'accesso ai AWS servizi dalle applicazioni all'interno di un Pod.
- È possibile utilizzare le policy di rete con le reti personalizzate e la delega del prefisso.

Considerazioni

Architettura

- Quando applichi il plug-in Amazon VPC CNI per le policy di rete Kubernetes al tuo cluster con il plug-in Amazon VPC CNI per Kubernetes, puoi applicare le policy solo ai nodi Amazon Linux. EC2 Non è possibile applicare le policy ai nodi Fargate o Windows.
- Le policy di rete si applicano solo a indirizzi `IPv4` o `IPv6`, ma non a entrambi. In un cluster `IPv4`, il CNI di VPC assegna l'indirizzo `IPv4` ai pod e applica le policy `IPv4`. In un cluster `IPv6`, il CNI di VPC assegna l'indirizzo `IPv6` ai pod e applica le policy `IPv6`. Tutte le regole di policy di rete `IPv4` applicate a un cluster `IPv6` vengono ignorate. Tutte le regole di policy di rete `IPv6` applicate a un cluster `IPv4` vengono ignorate.

Policy di rete

- Le policy di rete vengono applicate solo ai pod che fanno parte di un'implementazione. Ai pod autonomi che non dispongono di un set `metadata.ownerReferences` non possono essere applicate policy di rete.
- È possibile applicare più policy di rete allo stesso pod. Quando sono configurate due o più policy che selezionano lo stesso pod, al pod vengono applicate tutte le policy.
- Il numero massimo di combinazioni di porte e protocolli per un singolo intervallo di indirizzi IP (CIDR) è 24 per tutte le policy di rete. `namespaceSelector` CIDRsSelettori come resolve to one o più. Se più selettori si risolvono in un unico CIDR o si specifica lo stesso CIDR diretto più volte nella stessa policy di rete o in diverse policy di rete, tutti questi fattori vengono conteggiati ai fini di questo limite.
- Per ognuno dei servizi di Kubernetes, la porta del servizio deve essere uguale a quella del container. Se si utilizzano porte con un nome assegnato, è necessario utilizzare lo stesso nome anche nelle specifiche del servizio.

Politiche di rete di amministrazione

1. Politiche del livello di amministrazione (valutate per prime): tutte le politiche di livello amministratore ClusterNetworkPolicies vengono valutate prima di qualsiasi altra politica. All'interno del livello di amministrazione, le politiche vengono elaborate in ordine di priorità (prima il numero di priorità più basso). Il tipo di azione determina cosa succede dopo.
   - Nega azione (priorità massima): quando una politica di amministrazione con un'azione Rifiuta corrisponde al traffico, tale traffico viene immediatamente bloccato indipendentemente da qualsiasi altra politica. Non vengono elaborate ClusterNetworkPolicy ulteriori NetworkPolicy

regole. Ciò garantisce che i controlli di sicurezza a livello di organizzazione non possano essere sostituiti da politiche a livello di namespace.

- Consenti azioni: dopo la valutazione delle regole Deny, le politiche di amministrazione con le azioni Consenti vengono elaborate in ordine di priorità (inizia con il numero di priorità più basso). Quando un'azione Consenti corrisponde, il traffico viene accettato e non viene effettuata alcuna ulteriore valutazione delle politiche. Queste politiche possono concedere l'accesso su più namespace in base a selettori di etichette, fornendo un controllo centralizzato su quali carichi di lavoro possono accedere a risorse specifiche.
- Pass action: le azioni Pass Action nelle policy del livello di amministrazione delegano il processo decisionale ai livelli inferiori. Quando il traffico corrisponde a una regola Pass, la valutazione salta tutte le regole rimanenti del livello di amministratore per quel traffico e passa direttamente al livello. NetworkPolicy Ciò consente agli amministratori di delegare in modo esplicito il controllo di determinati modelli di traffico ai team applicativi. Ad esempio, è possibile utilizzare le regole Pass per delegare la gestione del traffico all'interno del namespace agli amministratori del namespace, mantenendo al contempo controlli rigorosi sull'accesso esterno.

2. Livello di policy di rete: se nessuna politica del livello di amministratore corrisponde a Deny o Allow, o se è stata soddisfatta un'azione Pass, vengono successivamente valutate le risorse con ambito namespace. NetworkPolicy Queste politiche forniscono un controllo granulare all'interno dei singoli namespace e sono gestite dai team delle applicazioni. Le politiche con ambito namespace possono essere solo più restrittive delle politiche di amministrazione. Non possono ignorare la decisione di rifiuto di una politica di amministrazione, ma possono limitare ulteriormente il traffico consentito o approvato dalle politiche di amministrazione.
3. Politiche di amministrazione di livello base: se nessuna politica di amministrazione o con ambito del namespace corrisponde al traffico, viene valutato il livello di base. ClusterNetworkPolicies Queste forniscono posizioni di sicurezza predefinite che possono essere sostituite da policy con ambito namespace, consentendo agli amministratori di impostare impostazioni predefinite a livello di organizzazione e offrendo al contempo ai team la flessibilità necessaria per personalizzarle in base alle esigenze. Le politiche di base vengono valutate in ordine di priorità (per primo il numero di priorità più basso).
4. Negazione predefinita (se nessuna policy corrisponde): questo deny-by-default comportamento garantisce che siano consentite solo le connessioni esplicitamente consentite, mantenendo un solido livello di sicurezza.

Migrazione

- Se il cluster utilizza attualmente una soluzione di terze parti per la gestione delle policy di rete di Kubernetes, è possibile utilizzare le stesse policy con il plug-in CNI di Amazon VPC per Kubernetes. Tuttavia, è necessario rimuovere la soluzione esistente in modo che non gestisca le stesse policy.

> **Warning**
>
> Dopo aver rimosso una soluzione di policy di rete, si consiglia di sostituire tutti i nodi a cui era stata applicata la soluzione di policy di rete. Questo perché le regole del traffico potrebbero essere ignorate da un pod della soluzione se questa si interrompe all'improvviso.

Installazione

- La funzionalità di policy di rete crea e richiede una `PolicyEndpoint` Custom Resource Definition (CRD) denominata `policyendpoints.networking.k8s.aws`. Gli oggetti `PolicyEndpoint` della Custom Resource sono gestiti da Amazon EKS. È sconsigliabile modificare o eliminare queste risorse.
- Se esegui pod che utilizzano le credenziali IAM del ruolo di istanza o ti connetti all' EC2 IMDS, fai attenzione a verificare le politiche di rete che bloccherebbero l'accesso all'IMDS. EC2 Potrebbe essere necessario aggiungere una policy di rete per consentire l'accesso all'IMDS. EC2 Per ulteriori informazioni, consulta Metadati dell'istanza e dati utente nella Amazon EC2 User Guide.

  I pod che utilizzano ruoli IAM per gli account di servizio o EKS Pod Identity non accedono EC2 a IMDS.
- Il plug-in CNI di Amazon VPC per Kubernetes non applica le policy di rete alle interfacce di rete aggiuntive per ogni pod, ma solo all'interfaccia principale per ogni pod (`eth0`). Ciò influisce sulle seguenti architetture:
  - Pod `IPv6` con la variabile `ENABLE_V4_EGRESS` impostata su `true`. Questa variabile abilita la `IPv4` funzione di uscita per connettere i IPv6 pod a `IPv4` endpoint come quelli esterni al cluster. La funzione `IPv4` di uscita funziona creando un'interfaccia di rete aggiuntiva con un indirizzo di loopback locale. IPv4
  - In caso di utilizzo di plugin di rete concatenati come Multus. Poiché questi plug-in aggiungono interfacce di rete a ciascun pod, le policy di rete non vengono applicate ai plug-in di rete concatenati.

Limitare il traffico di rete dei Pod con le policy di rete Kubernetes

È possibile utilizzare le policy di rete di Kubernetes per limitare il traffico di rete da e verso i pod. Per ulteriori informazioni, consulta Network Policies nella documentazione Kubernetes.

È necessario configurare quanto segue per utilizzare questa funzionalità:

1. Impostare l'applicazione delle policy all'avvio del Pod. Si effettua nel container `aws-node` del CNI `DaemonSet` di VPC.
2. Abilitare il parametro della policy di rete per il componente aggiuntivo.
3. Configurare il cluster per l'utilizzo della policy di rete di Kubernetes

Prima di iniziare, esaminare le seguenti informazioni. Per ulteriori informazioni, consulta the section called "Considerazioni".

Prerequisiti

Di seguito sono indicati i prerequisiti per la funzionalità:

Versione minima del cluster

Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base. Il cluster deve eseguire una delle versioni Kubernetes e della piattaforma elencate nella tabella seguente. Sono supportate anche tutte le versioni di Kubernetes e della piattaforma successive a quelle elencate. Puoi controllare la tua versione attuale di Kubernetes *`my-cluster`* sostituendo il seguente comando con il nome del cluster e quindi eseguendo il comando modificato:

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Versione di Kubernetes	Versione della piattaforma
`1.27.4`	`eks.5`
`1.26.7`	`eks.6`

Versione minima VPC CNI

Per creare sia policy di rete Kubernetes standard che policy di rete di amministrazione, devi eseguire `1.21` la versione del plugin VPC CNI. È possibile verificare la versione attuale con il seguente comando.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Se la versione è precedente alla `1.21`, consulta la pagina the section called "Aggiornamento (componente aggiuntivo EKS)" per eseguire l'aggiornamento alla versione `1.21` o successiva.

Versione minima del kernel Linux

I nodi devono avere una versione del kernel Linux `5.10` o successiva. È possibile verificare la versione del kernel con `uname -r`. Se utilizzi le versioni più recenti di Amazon Linux ottimizzato per Amazon EKS, Amazon Linux accelerato ottimizzato per Amazon EKS e Bottlerocket AMIs, dispongono già della versione AMIs del kernel richiesta.

La versione `v20231116` o successive delle AMI Amazon Linux accelerate ottimizzate per Amazon EKS dispone della versione del kernel `5.10`.

Passaggio 1: Impostare l'applicazione delle policy all'avvio del Pod

Il plug-in CNI di Amazon VPC per Kubernetes configura le policy di rete per i pod parallelamente al provisioning dei pod. Fino a quando tutte le policy non saranno configurate per il nuovo pod, i container nel nuovo si avvieranno con una policy di autorizzazione predefinita. Questa è chiamata modalità standard. Una policy di autorizzazione predefinita prevede che tutto il traffico in entrata e in uscita sia consentito da e verso i nuovi pod. Ad esempio, ai pod non verrà applicata alcuna regola del firewall (tutto il traffico è consentito) finché il nuovo pod non verrà aggiornato con le policy attive.

Con la variabile `NETWORK_POLICY_ENFORCING_MODE` impostata su `strict`, i pod che utilizzano il CNI di VPC iniziano con una policy di negazione predefinita, poi vengono configurate le policy. Questa è detta modalità rigorosa. In modalità rigorosa, è necessario disporre di una policy di rete per ogni endpoint a cui i pod devono accedere nel cluster. Questo requisito si applica ai pod CoreDNS. La policy di negazione predefinita non è configurata per i pod con rete host.

È possibile modificare la policy di rete predefinita impostando la variabile di ambiente `NETWORK_POLICY_ENFORCING_MODE` su `strict` nel container `aws-node` del CNI di VPC `DaemonSet`.

```
env:
  - name: NETWORK_POLICY_ENFORCING_MODE
    value: "strict"
```

Passaggio 2: Abilitare il parametro della policy di rete per il componente aggiuntivo

Per impostazione predefinita, per i parametri la funzionalità di policy di rete utilizza la porta `8162` sul nodo. Inoltre, la funzionalità utilizza la porta `8163` per le sonde sanitarie. Se esegui un'altra applicazione sui nodi o all'interno dei pod che deve utilizzare queste porte, l'app non viene eseguita. Dalla versione VPC CNI `v1.14.1` o successiva, puoi modificare queste porte.

Per abilitare il parametro della policy di rete per il componente aggiuntivo, procedere come indicato di seguito.

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui desideri configurare il componente aggiuntivo CNI di Amazon VPC.
3. Selezionare la scheda Componenti aggiuntivi.
4. Selezionare la casella nella parte superiore destra della casella del componente aggiuntivo e scegli Modifica.
5. Nella pagina Configura **`Amazon VPC CNI`**:
   a. Selezionare una `v1.14.0-eksbuild.3` o versione successiva nell'elenco Versione.
   b. Scegliere Impostazioni di configurazione facoltative.
   c. Inserisci la chiave JSON `"enableNetworkPolicy":` e il valore `"true"` nei Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. Se questa chiave e questo valore sono gli unici dati nella casella di testo, racchiudi la chiave e il valore tra parentesi graffe `{ }`.

      L'esempio seguente ha la funzionalità delle policy di rete abilitata e i parametri e le sonde dell'integrità sono impostati sui numeri di porta predefiniti:

      ```
      {
          "enableNetworkPolicy": "true",
          "nodeAgent": {
              "healthProbeBindAddr": "8163",
      ```

```
        "metricsBindAddr": "8162"
    }
}
```

Helm

Se il plug-in CNI di Amazon VPC per Kubernetes è stato installato attraverso `helm`, è possibile aggiornare la configurazione per scrivere le porte.

1. Esegui il seguente comando per modificare le porte. Imposta il numero di porta nel valore, rispettivamente, della chiave `nodeAgent.metricsBindAddr` o `nodeAgent.healthProbeBindAddr`.

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
 nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/aws-vpc-
cni
```

kubectl

1. Apri la `DaemonSet` del `aws-node` nell'editor.

```
kubectl edit daemonset -n kube-system aws-node
```

2. Sostituire i numeri di porta negli argomenti del comando seguente nel `args:` del container `aws-network-policy-agent` nel manifesto del daemonset `aws-node` della CNI del VPC.

```
    - args:
            - --metrics-bind-addr=:8162
            - --health-probe-bind-addr=:8163
```

Passaggio 3: Configurare il cluster per l'utilizzo delle policy di rete di Kubernetes

È possibile impostarlo per un componente aggiuntivo di Amazon EKS o come un componente aggiuntivo autogestito.

Componenti aggiuntivi di Amazon EKS

Utilizzando la AWS CLI, puoi configurare il cluster per utilizzare le politiche di rete Kubernetes eseguendo il comando seguente. Sostituire `my-cluster` con il nome del cluster e l'ARN del ruolo IAM con il ruolo che si sta utilizzando.

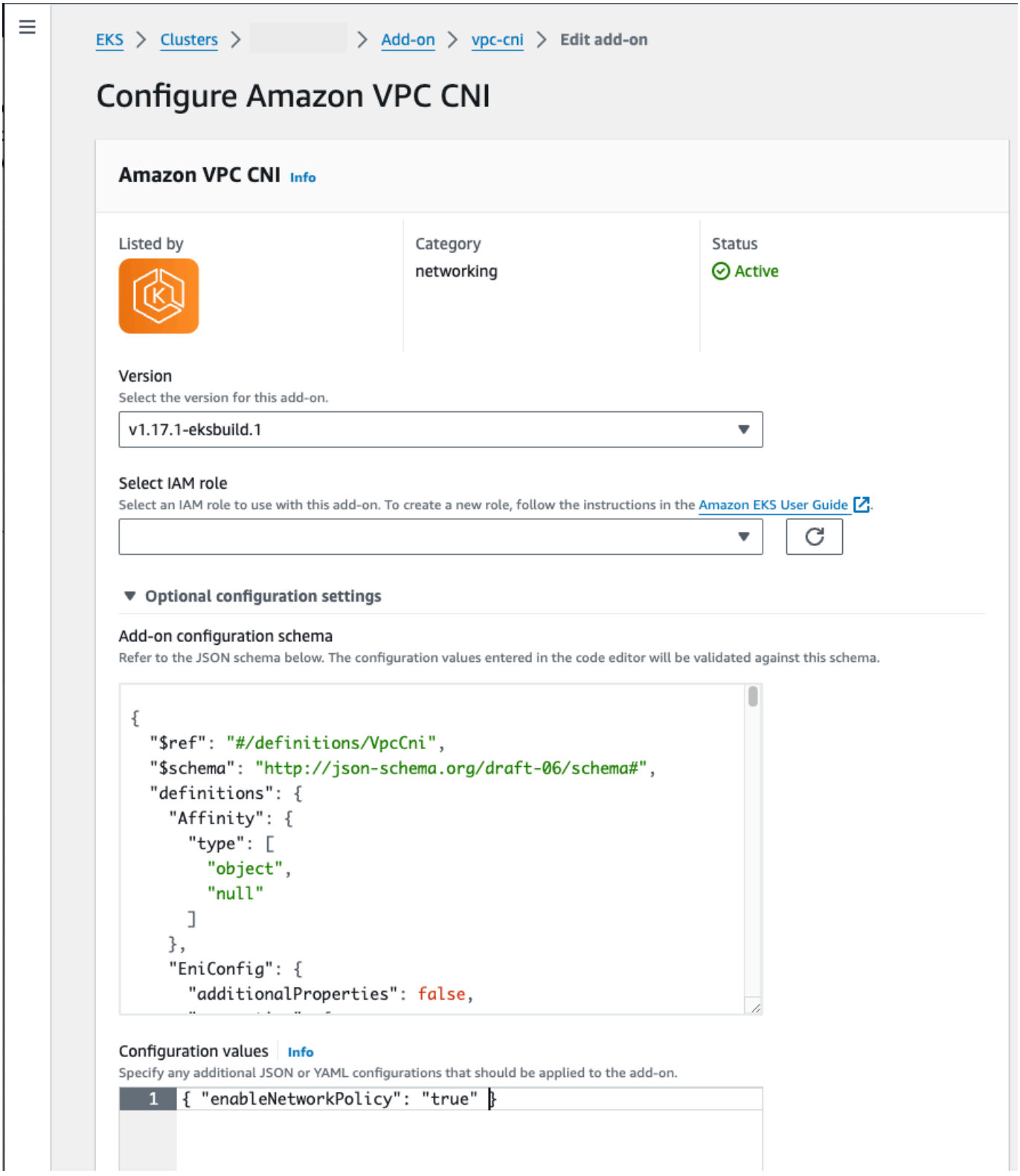
```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"enableNetworkPolicy":
 "true"}'
```

Per configurarlo utilizzando la console di AWS gestione, segui i passaggi seguenti:

1. Aprire la Console Amazon EKS.
2. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui desideri configurare il componente aggiuntivo CNI di Amazon VPC.
3. Selezionare la scheda Componenti aggiuntivi.
4. Selezionare la casella nella parte superiore destra della casella del componente aggiuntivo e scegli Modifica.
5. Nella pagina Configura **`Amazon VPC CNI`**:
   a. Selezionare una `v1.14.0-eksbuild.3` o versione successiva nell'elenco Versione.
   b. Scegliere Impostazioni di configurazione facoltative.
   c. Inserisci la chiave JSON `"enableNetworkPolicy":` e il valore `"true"` nei Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. Se questa chiave e questo valore sono gli unici dati nella casella di testo, racchiudi la chiave e il valore tra parentesi graffe `{ }`. Di seguito è riportato l'esempio dell'abilitazione della policy di rete:

      ```
      { "enableNetworkPolicy": "true" }
      ```

      Lo screenshot seguente mostra un esempio di tale scenario.


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" }

Componenti aggiuntivi autogestiti

Helm

Se il plug-in CNI di Amazon VPC per Kubernetes è stato installato attraverso `helm`, è possibile aggiornare la configurazione per abilitare la policy di rete.

1. Esegui il comando seguente per abilitare la policy di rete.

```
helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. Apri la `ConfigMap` del `amazon-vpc-cni` nell'editor.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

2. Aggiungi la seguente riga ai `data` nella `ConfigMap`.

```
enable-network-policy-controller: "true"
```

   Dopo aver aggiunto la riga, la `ConfigMap` dovrebbe essere simile al seguente esempio.

```
apiVersion: v1
 kind: ConfigMap
 metadata:
  name: amazon-vpc-cni
  namespace: kube-system
 data:
  enable-network-policy-controller: "true"
```

3. Apri la `DaemonSet` del `aws-node` nell'editor.

```
kubectl edit daemonset -n kube-system aws-node
```

   a. Sostituire il valore `false` con `true` nell'argomento del comando `--enable-network-policy=false` in `args:` nel container `aws-network-policy-agent` nel manifesto del daemonset `aws-node` della CNI di VPC.

```
    - args:
       - --enable-network-policy=true
```

Passaggio 4. Fasi successive

Dopo aver completato la configurazione, confermare che i pod `aws-node` siano in esecuzione sul cluster.

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

Di seguito viene riportato un output di esempio.

```
aws-node-gmqp7                                          2/2     Running   1 (24h ago)
 24h
aws-node-prnsh                                          2/2     Running   1 (24h ago)
 24h
```

Nei pod `aws-node` sono presenti 2 container nelle versioni `1.14` e successive. Nelle versioni precedenti e se le policy di rete sono disabilitate, nei pod `aws-node` è presente un solo container.

Ora è possibile implementare le policy di rete di Kubernetes per il cluster.

Per implementare le policy di rete Kubernetes, puoi creare Kubernetes `NetworkPolicy` o `ClusterNetworkPolicy` oggetti e distribuirli nel tuo cluster. `NetworkPolicy`gli oggetti sono limitati a un namespace, mentre gli `ClusterNetworkPolicy` oggetti possono essere estesi all'intero cluster o a più namespace. Le policy vengono implementate per consentire o negare il traffico tra Pod in base a selettori di etichette, namespace e intervalli di indirizzi IP. Per ulteriori informazioni sulla creazione di oggetti `NetworkPolicy`, consultare Network Policies nella documentazione di Kubernetes.

L'applicazione di oggetti `NetworkPolicy` di Kubernetes viene implementata con Extended Berkeley Packet Filter (eBPF). Relativo alle implementazioni basate su `iptables`, offre caratteristiche di latenza e prestazioni inferiori, tra cui un utilizzo ridotto della CPU e l'evitamento delle ricerche sequenziali. Inoltre, le sonde eBPF forniscono l'accesso a dati contestuali che aiutano a eseguire il debug di problemi complessi a livello di kernel e a migliorare l'osservabilità. Amazon EKS supporta un esportatore basato su eBPF che sfrutta le sonde per registrare i risultati delle policy su ogni nodo ed esportare i dati in raccoglitori di log esterni per facilitare il debug. Per ulteriori informazioni, consulta eBPF documentation.

Disattiva le politiche di rete Kubernetes per il traffico di rete Amazon EKS Pod

Disattiva le policy di rete Kubernetes per interrompere la limitazione del traffico di rete Amazon EKS Pod

1. Elenca tutte le politiche di rete Kubernetes.

```
kubectl get netpol -A
```

2. Elimina ogni policy di rete Kubernetes. È necessario eliminare tutte le politiche di rete prima di disabilitarle.

```
kubectl delete netpol <policy-name>
```

3. Apri aws-node DaemOnset nel tuo editor.

```
kubectl edit daemonset -n kube-system aws-node
```

4. Sostituisci il valore `true` con `false` nell'argomento del comando `--enable-network-policy=true` in `args:` nel container `aws-network-policy-agent` nel manifesto del daemonset `aws-node` della CNI di VPC.

```
    - args:
       - --enable-network-policy=true
```

Risoluzione dei problemi delle politiche di rete Kubernetes per Amazon EKS

Questa è la guida alla risoluzione dei problemi per la funzionalità di policy di rete di CNI di Amazon VPC.

Questa guida illustra:

- informazioni su installazione, autorizzazioni di CRD e RBAC the section called "Nuovo CRD `policyendpoints` e autorizzazioni"
- log da esaminare durante la diagnosi dei problemi relativi alle politiche di rete the section called "Log delle policy di rete"
- esecuzione della raccolta di strumenti SDK eBPF per la risoluzione dei problemi
- Problemi noti e risoluzioni the section called "Problemi noti e risoluzioni"

> **Note**
>
> Tieni presente che le politiche di rete vengono applicate solo ai pod creati da Implementazioni di Kubernetes. Per ulteriori limitazioni delle politiche di rete nel CNI di VPC, consultare the section called "Considerazioni".

È possibile risolvere i problemi e analizzare le connessioni di rete che utilizzano le policy di rete leggendo i log della policy di rete ed eseguendo gli strumenti da eBPF SDK.

Nuovo CRD **`policyendpoints`** e autorizzazioni

- CRD: `policyendpoints.networking.k8s.aws`
- API di Kubernetes: `apiservice` ha chiamato `v1.networking.k8s.io`
- Risorsa di Kubernetes: `Kind: NetworkPolicy`
- RBAC: `ClusterRole` ha chiamato `aws-node` (CNI di VPC), `ClusterRole` ha chiamato `eks:network-policy-controller` (controller delle politiche di rete nel piano di controllo del cluster EKS)

Per la politica di rete, CNI di VPC crea un nuovo `CustomResourceDefinition` (CRD) chiamato `policyendpoints.networking.k8s.aws`. Il CNI VPC deve disporre delle autorizzazioni per creare il CRD e creare CustomResources (CR) di questo e dell'altro CRD installato dal VPC CNI (). `eniconfigs.crd.k8s.amazonaws.com` Entrambi sono disponibili nel file su. CRDs `crds.yaml` GitHub In particolare, CNI di VPC deve disporre delle autorizzazioni verbali "ottieni", "elenco" e "guarda" per `policyendpoints`.

La politica di rete di Kubernetes fa parte dei `apiservice` chiamato `v1.networking.k8s.io` e ciò è `apiversion: networking.k8s.io/v1` nei file YAML della politica. `DaemonSet` di CNI di VPC deve disporre delle autorizzazioni per utilizzare questa parte dell'API Kubernetes.

Le autorizzazioni di CNI di VPC sono disponibili in un `ClusterRole` chiamato `aws-node`. Si noti che gli oggetti `ClusterRole` non sono raggruppati in namespace. Di seguito, è illustrato il `aws-node` di un 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
```

Inoltre, un nuovo controller è avviato nel piano di controllo di ciascun cluster EKS. Il controller utilizza le autorizzazioni del `ClusterRole` chiamato `eks:network-policy-controller`. Di seguito, è illustrato il `eks:network-policy-controller` di un 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
```

Log delle policy di rete

Ciascuna decisione da parte di CNI di VPC relativamente al fatto che le connessioni sono consentite o negate da una policy di rete è registrata da log di flusso. I log delle policy di rete su ogni nodo includono i log di flusso per ogni pod che dispone di una policy di rete. I log delle policy di rete sono archiviati in `/var/log/aws-routed-eni/network-policy-agent.log`. Di seguito è riportato un esempio del file `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"}
```

I log della policy di rete sono disabilitati per impostazione predefinita. Per abilitare i log della policy di rete, segui i passaggi riportati di seguito:

> **Note**
>
> I log delle policy di rete richiedono 1 vCPU aggiuntiva per il container `aws-network-policy-agent` nel manifesto `aws-node DaemonSet` di CNI di VPC.

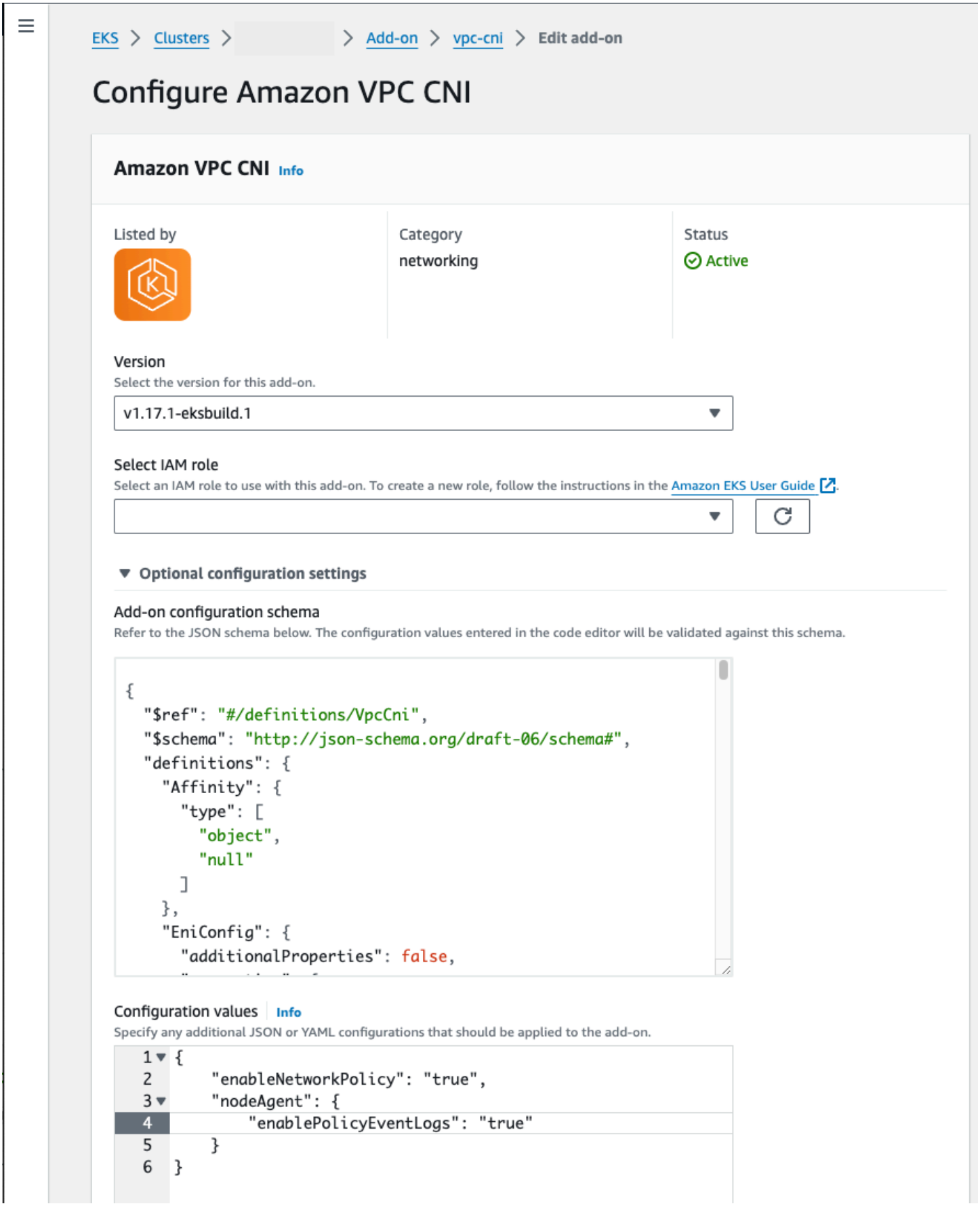
Componenti aggiuntivi di Amazon EKS

Console di gestione AWS

a. Aprire la Console Amazon EKS.
b. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui desideri configurare il componente aggiuntivo CNI di Amazon VPC.
c. Selezionare la scheda Componenti aggiuntivi.
d. Selezionare la casella nella parte superiore destra della casella del componente aggiuntivo e scegli Modifica.
e. Nella pagina Configura ***Amazon VPC CNI***:
   i. Seleziona una `v1.14.0-eksbuild.3` o versione successiva nell'elenco a discesa Versione.
   ii. Scegli Impostazioni di configurazione facoltative.
   iii. Inserisci la chiave JSON di primo livello `"nodeAgent":` e il valore, che è un oggetto con una chiave`"enablePolicyEventLogs":` e un valore di `"true"` nei Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. L'esempio seguente mostra i criteri di rete e i registri dei criteri di rete sono abilitati e i registri dei criteri di rete vengono inviati a CloudWatch Logs:

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true"
    }
}
```

Lo screenshot seguente mostra un esempio di tale scenario.


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. Esegui il seguente comando AWS CLI. Sostituisci `my-cluster` con il nome del cluster e l'ARN del ruolo IAM con il ruolo che stai utilizzando.

```
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"}}'
```

Componenti aggiuntivi autogestiti

Helm

Se il plugin CNI di Amazon VPC per Kubernetes è stato installato attraverso`helm`, è possibile aggiornare la configurazione per scrivere i log della policy di rete.

a. Esegui il comando seguente per abilitare la policy di rete.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

Se il plugin CNI di Amazon VPC per Kubernetes è stato installato attraverso`kubectl`, è possibile aggiornare la configurazione per scrivere i log della policy di rete.

a. Apri la `DaemonSet` del `aws-node` nell'editor.

```
kubectl edit daemonset -n kube-system aws-node
```

b. Sostituisci il valore `false` con `true` nell'argomento del comando `--enable-policy-event-logs=false` in `args:` nel container `aws-network-policy-agent` nel manifesto `aws-node` `DaemonSet` di CNI di VPC.

```
    - args:
       - --enable-policy-event-logs=true
```

Invia i log delle policy di rete ad Amazon CloudWatch Logs

Puoi monitorare i log delle politiche di rete utilizzando servizi come Amazon CloudWatch Logs. Puoi utilizzare i seguenti metodi per inviare i log delle politiche di rete a Logs. CloudWatch

Per i cluster EKS, i log delle policy si troveranno in `/aws/eks/`*`cluster-name`*`/cluster/`, mentre per i cluster K8S autogestiti i log verranno collocati in `/aws/k8s-cluster/cluster/`.

Invio dei log della policy di rete con il plugin CNI di Amazon VPC per Kubernetes

Se si abilitano le policy di rete, ai pod viene aggiunto un secondo container `aws-node` per un agente del nodo. Questo agente del nodo può inviare i registri delle politiche di rete a Logs. CloudWatch

> **Note**
>
> Solo i log delle policy di rete vengono inviati dall'agente del nodo. Gli altri log creati dalla CNI di VPC non sono inclusi.

Prerequisiti

- Aggiungi le seguenti autorizzazioni come strofa o policy separata al ruolo IAM che stai utilizzando per la CNI di VPC.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

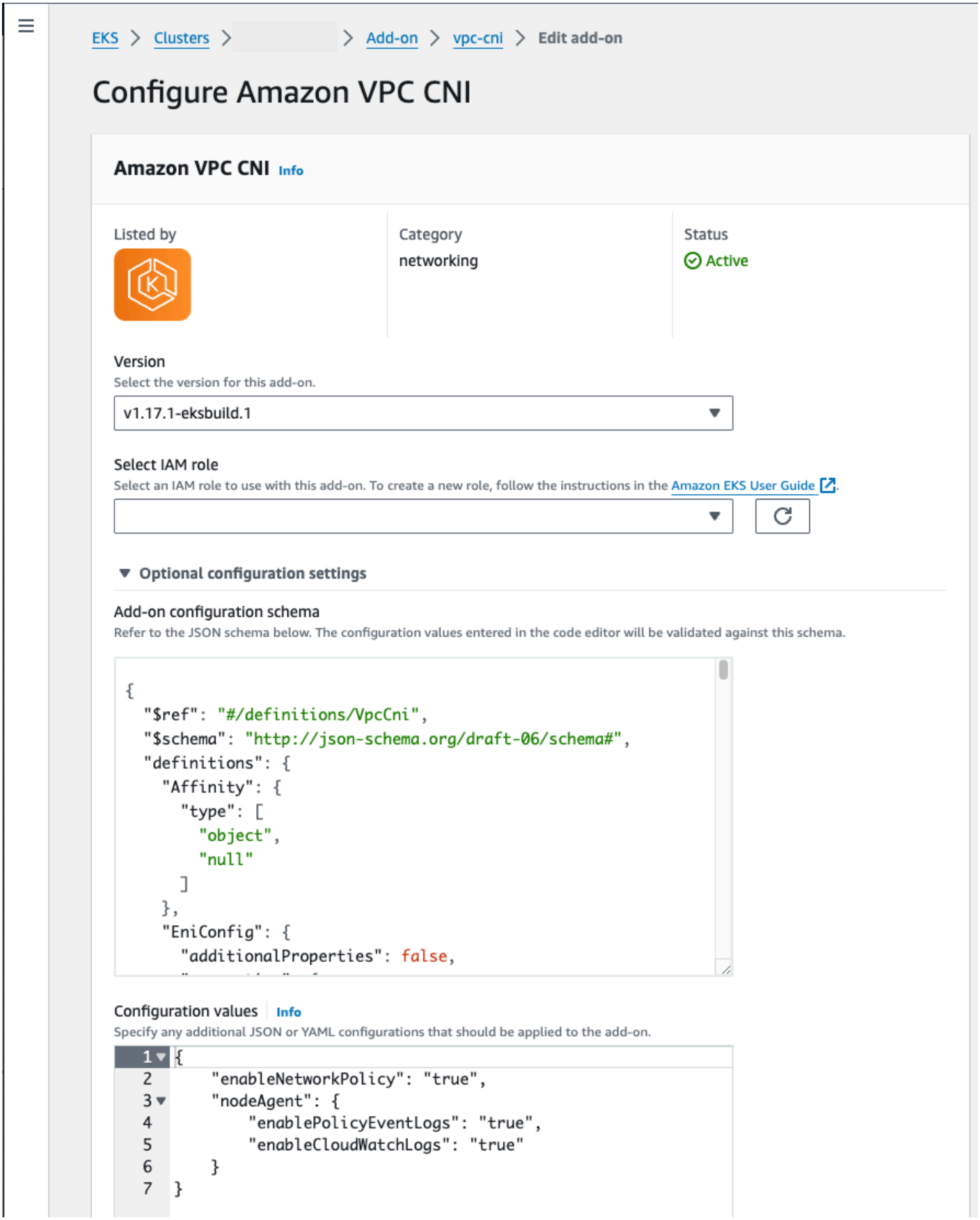
Componenti aggiuntivi di Amazon EKS

Console di gestione AWS

a. Aprire la Console Amazon EKS.
b. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui desideri configurare il componente aggiuntivo CNI di Amazon VPC.
c. Selezionare la scheda Componenti aggiuntivi.
d. Selezionare la casella nella parte superiore destra della casella del componente aggiuntivo e scegli Modifica.
e. Nella pagina Configura ***Amazon VPC CNI***:
   i. Seleziona una `v1.14.0-eksbuild.3` o versione successiva nell'elenco a discesa Versione.
   ii. Scegli Impostazioni di configurazione facoltative.
   iii. Inserisci la chiave JSON di primo livello `"nodeAgent":` e il valore, che è un oggetto con una chiave`"enableCloudWatchLogs":` e un valore di `"true"` nei Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. L'esempio seguente mostra i criteri di rete e i registri dei criteri di rete sono abilitati e i log vengono inviati a Logs: CloudWatch

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
    }
}
```

Lo screenshot seguente mostra un esempio di tale scenario.


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. Esegui il seguente comando AWS CLI. Sostituisci `my-cluster` con il nome del cluster e l'ARN del ruolo IAM con il ruolo che stai utilizzando.

```
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"}}'
```

Componenti aggiuntivi autogestiti

Timone

Se hai installato il plug-in Amazon VPC CNI per Kubernetes tramite`helm`, puoi aggiornare la configurazione per inviare i log delle politiche di rete a Logs. CloudWatch

a. Esegui il comando seguente per abilitare i log delle politiche di rete e inviarli a Logs. CloudWatch

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set
 nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/aws-
vpc-cni
```

kubectl

a. Apri la `DaemonSet` del `aws-node` nell'editor.

```
kubectl edit daemonset -n kube-system aws-node
```

b. Sostituisci `false` con `true` in due argomenti del comando `--enable-policy-event-logs=false` e `--enable-cloudwatch-logs=false` nel `args:` nel container `aws-network-policy-agent` nel manifesto `aws-node` `DaemonSet` di CNI di VPC.

```
    - args:
       - --enable-policy-event-logs=true
       - --enable-cloudwatch-logs=true
```

Invio dei log delle policy di rete con Fluent Bit **`DaemonSet`**

Se stai utilizzando Fluent Bit in `DaemonSet` per inviare i log dai nodi, è possibile aggiungere una configurazione che includa i log delle policy di rete provenienti dalle policy di rete. È possibile utilizzare la configurazione di esempio seguente:

```
    [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 ebPF incluso

Il plugin CNI di Amazon VPC per Kubernetes installa la raccolta di strumenti dell'SDK eBPF sui nodi. È possibile utilizzare gli strumenti dell'SDK eBPF per identificare i problemi relativi alle policy di rete. Ad esempio, il comando seguente elenca i programmi in esecuzione sul nodo.

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

Per eseguire questo comando, è possibile utilizzare qualsiasi metodo di connessione al nodo.

Problemi noti e risoluzioni

Le seguenti sezioni descrivono i problemi noti con la funzionalità di policy di rete CNI di Amazon VPC e le relative soluzioni.

Registri delle politiche di rete generati nonostante l'impostazione sia impostata su false enable-policy-event-logs

Problema: CNI di VPC EKS genera log delle politiche di rete anche quando l'impostazione `enable-policy-event-logs` è impostata su `false`.

Soluzione: l'impostazione `enable-policy-event-logs` disabilita solo i log delle "decisioni" delle politiche, ma non disabilita tutta la registrazione degli agenti della policy di rete. Questo comportamento è documentato nel aws-network-policy-agent README on. GitHub Per disabilitare completamente la registrazione, potrebbe essere necessario modificare altre configurazioni di registrazione.

Problemi di pulizia della mappa della policy di rete

Problema: la rete `policyendpoint` è ancora esistente e non è ripulita dopo l'eliminazione dei pod.

Soluzione: questo problema è stato causato da un problema con il componente aggiuntivo di CNI di VPC versione 1.19.3-eksbuild.1. Per risolvere questo problema, esegui l'aggiornamento a una versione più recente del componente aggiuntivo CNI di VPC.

Le politiche di rete non sono applicate

Problema: la funzionalità della policy di rete è abilitata nel plugin CNI di Amazon VPC, ma le policy di rete non sono applicate correttamente.

Se crei una policy di rete `kind: NetworkPolicy` e questa non influisce sul pod, verifica che l'oggetto policyendpoint sia stato creato nello stesso namespace del pod. Se non ci sono oggetti `policyendpoint` nei namespace, il controller della policy di rete (parte del cluster EKS) non è stato in grado di creare regole di policy di rete per l'agente delle policy di rete (parte del CNI di VPC) da applicare.

Soluzione: la soluzione consiste nel correggere le autorizzazioni del CNI di VPC (`ClusterRole` : `aws-node`) e del controller delle policy di rete (`ClusterRole` : `eks:network-policy-controller`) e consentire queste azioni in qualsiasi strumento di applicazione delle policy come Kyverno. Assicurati che le policy di Kyverno non blocchino la creazione di oggetti `policyendpoint`. Vedi la sezione precedente per le autorizzazioni in cui sono necessarie le autorizzazioni in the section called "Nuovo CRD `policyendpoints` e autorizzazioni".

I pod non tornano allo stato di negazione predefinito dopo l'eliminazione della policy in modalità rigida

Problema: quando i criteri di rete sono abilitati in modalità rigida, i pod iniziano con una politica di negazione predefinita. Dopo l'applicazione delle politiche, il traffico è consentito verso gli endpoint specificati. Tuttavia, quando le policy sono eliminate, il pod non torna allo stato di negazione predefinito, invece passa allo stato di autorizzazione predefinito.

Soluzione: questo problema è stato risolto in CNI di VPC versione 1.19.3, che includeva l'agente di policy di rete versione 1.2.0. Dopo la correzione, con la modalità rigida abilitata, una volta rimosse le policy, il pod tornerà allo stato di negazione predefinito come previsto.

Latenza di avvio dei gruppi di sicurezza for pod

Problema: quando si utilizza la funzionalità dei gruppi di sicurezza per pod in EKS, la latenza di avvio dei pod aumenta.

Soluzione: la latenza è dovuta alla limitazione della velocità nel controller di risorse dovuta alla limitazione dell'API sull'`CreateNetworkInterface`API, che il controller di risorse VPC utilizza per creare un ramo per i pod. ENIs Verifica i limiti delle API del tuo account per questa operazione e, se necessario, valuta la possibilità di richiedere un aumento del limite.

FailedScheduling a causa dell'insufficienza di vpc.amazonaws.com/pod-eni

Problema: i pod non riescono a pianificarsi con l'errore: `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.`

Soluzione: come nel problema precedente, l'assegnazione dei gruppi di sicurezza ai pod aumenta la latenza di pianificazione dei pod e può superare la soglia CNI necessaria per aggiungere ogni ENI, causando il mancato avvio dei pod. Questo è il comportamento previsto quando si utilizzano i gruppi di sicurezza per pod. Considera le implicazioni di pianificazione durante la progettazione dell'architettura del carico di lavoro.

Problemi di connettività IPAM ed errori di segmentazione

Problema: si verificano molteplici errori, tra cui problemi di connettività IPAM, richieste di limitazione ed errori di segmentazione:

- `Checking for IPAM connectivity …`
- `Throttling request took 1.047064274s`
- `Retrying waiting for IPAM-D`
- `panic: runtime error: invalid memory address or nil pointer dereference`

Soluzione: questo problema si verifica se si installa `systemd-udev` su AL2 023, poiché il file viene riscritto con una politica non conforme. Ciò può accadere quando si esegue l'aggiornamento a un `releasever` diverso che ha un pacchetto aggiornato o si aggiorna manualmente il pacchetto stesso. Evita l'installazione o l'aggiornamento `systemd-udev` sui nodi AL2 023.

Errore di ricerca del dispositivo in base al nome non riuscito

Problema: messaggio di errore: `{"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})"}`

Soluzione: questo problema è stato identificato e risolto nelle versioni più recenti dell'agente di policy di rete CNI di Amazon VPC (v1.2.0). Per risolvere questo problema, effettua l'aggiornamento alla versione più recente di CNI di VPC.

Vulnerabilità di CVE nell'immagine Multus CNI

Problema: Enhanced EKS ImageScan CVE Report identifica le vulnerabilità nella versione dell'immagine Multus CNI v4.1.4-eksbuild.2_thick.

Soluzione: esegui l'aggiornamento alla nuova versione dell'immagine Multus CNI e alla nuova immagine del controller della policy di rete, che non presentano vulnerabilità. Lo scanner può essere aggiornato per risolvere le vulnerabilità rilevate nella versione precedente.

Le informazioni sul flusso NEGANO i verdetti nei log

Problema: i log delle politiche di rete mostrano i verdetti di NEGAZIONE: `{"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"}`

Soluzione: questo problema è stato risolto nella nuova versione di controller della policy di rete. Effettua l'aggiornamento alla versione più recente della piattaforma EKS per risolvere i problemi di registrazione.

Pod-to-pod problemi di comunicazione dopo la migrazione da Calico

Problema: dopo l'aggiornamento di un cluster EKS alla versione 1.30 e il passaggio da Calico ad Amazon VPC CNI per le policy di rete pod-to-pod, la comunicazione fallisce quando vengono applicate le policy di rete. La comunicazione è ripristinata quando le policy di rete sono eliminate.

Soluzione: l'agente di policy di rete nel CNI di VPC non può avere tante porte specificate quante ne ha Calico. Invece, utilizza gli intervalli di porte nelle policy di rete. Il numero massimo di combinazioni univoche di porte per ogni protocollo `ingress:` o `egress:` selettore in una politica di rete è 24. Utilizza gli intervalli di porte per ridurre il numero di porte uniche ed evitare questa limitazione.

L'agente della policy di rete non supporta i pod autonomi

Problema: le policy di rete applicate ai pod autonomi possono avere un comportamento incoerente.

Soluzione: attualmente l'agente della policy di rete supporta solo i pod distribuiti come parte di un set di implementazione/replica. Se le policy di rete sono applicate ai pod autonomi, potrebbero esserci

delle incongruenze nel comportamento. Questo è documentato nella parte superiore di questa paginathe section called "Considerazioni", nel e nel aws-network-policy-agent GitHub numero #327 successivo. GitHub Implementa i pod come parte di un'implementazione o di un set di replica per un comportamento coerente delle policy di rete.

Demo delle policy di rete con Stars per Amazon EKS

La demo consente di creare un servizio di front-end, di back-end e client sul cluster Amazon EKS. Inoltre, la demo consente di creare un'interfaccia utente grafica di gestione che mostra i percorsi di ingresso e uscita disponibili tra ogni servizio. Ti consigliamo di completare la demo su un cluster su cui non esegui carichi di lavoro di produzione.

Prima di creare eventuali policy di rete, tutti i servizi sono in grado di comunicare in modo bidirezionale. Dopo l'applicazione delle policy di rete, puoi vedere che il client può comunicare solo con il servizio di front-end, mentre il back-end accetta traffico solo dal front-end.

1. Applica i servizi di front-end, back-end, client e dell'interfaccia utente di gestione:

```
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. Visualizza tutti i pod nel cluster.

```
kubectl get pods -A
```

   Di seguito viene riportato un output di esempio:

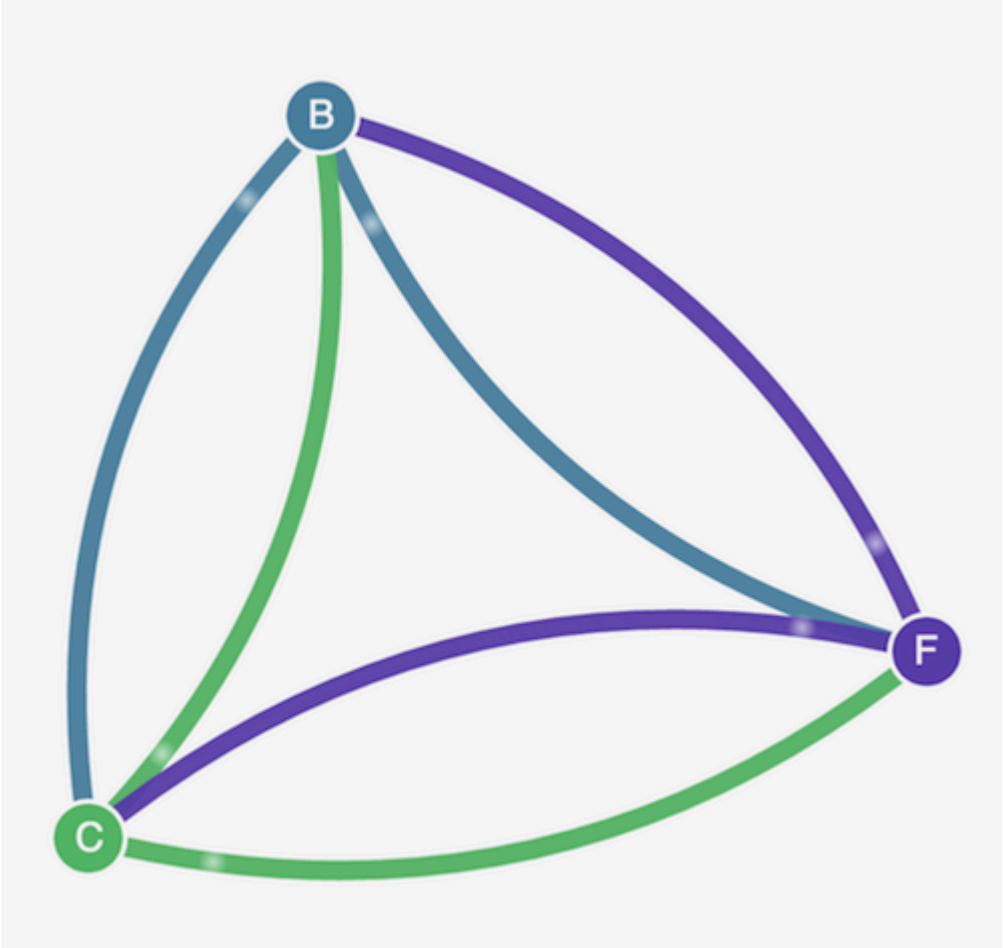
Nell'output, dovresti vedere i pod negli spazi dei nomi mostrati nel seguente output. I *NOMI* dei pod e il numero di pod nella colonna READY sono diversi da quelli del seguente output. Non continuare finché non consulta pod con nomi simili e tutti hanno il valore Running nella colonna STATUS.

```
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. Per connetterti all'interfaccia utente di gestione, connettiti all'EXTERNAL-IP del servizio in esecuzione sul cluster:

```
kubectl get service/management-ui -n management-ui
```

4. Apri un browser nella posizione del passaggio precedente. Dovresti visualizzare l'interfaccia utente di gestione seguente. Il nodo C è il servizio client, il nodo F è il servizio front-end e il nodo B è il servizio di back-end. Ogni nodo dispone di accesso di comunicazione completo a tutti gli altri nodi, come indicato dalle righe colorate in grassetto.



5. Applica la seguente policy di rete nei due spazi dei nomi `stars` e `client` per isolare tra loro i servizi:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

È possibile utilizzare i seguenti comandi per applicare la policy a entrambi gli spazi dei nomi:

```
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. Aggiorna il tuo browser. Poiché l'interfaccia utente di gestione non può più raggiungere i nodi, essi non vengono visualizzati nell'interfaccia utente.
7. Applica le seguenti policy di rete per consentire all'interfaccia utente di gestione di accedere ai servizi. Applica questa policy per consentire all'interfaccia utente di:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

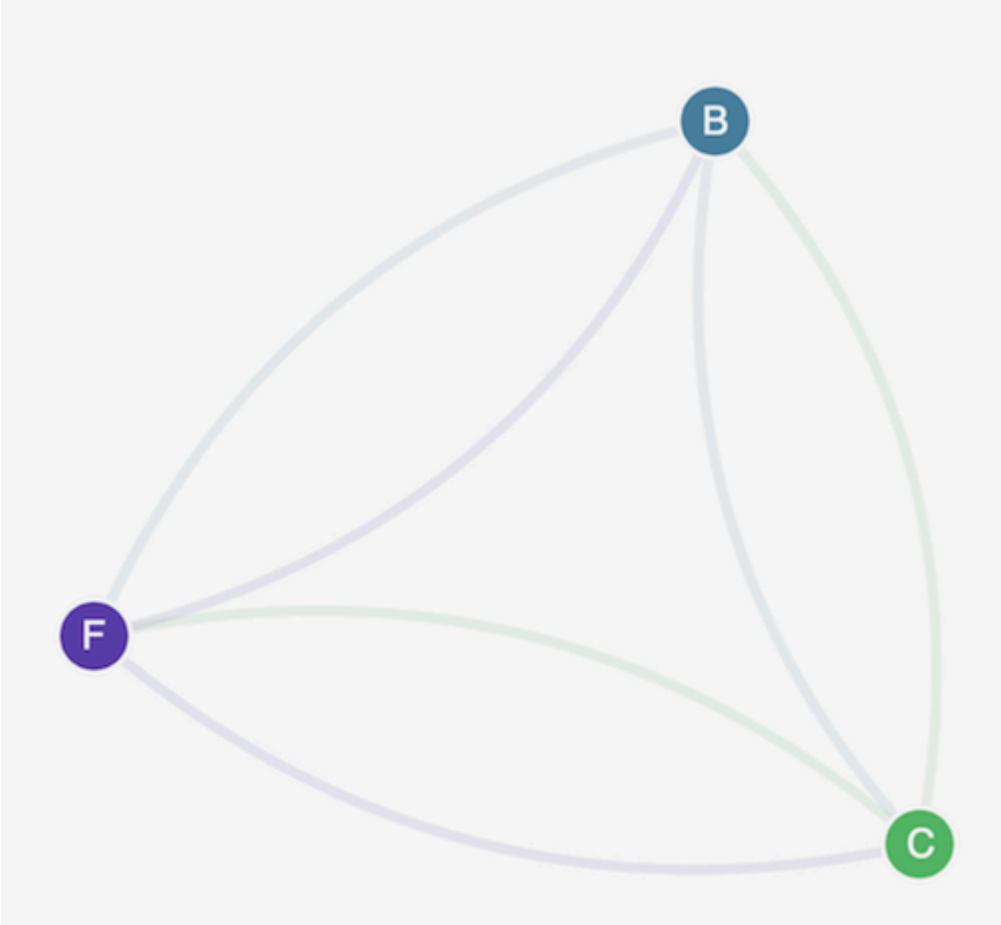
   Applica questa policy per consentire al client di:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

   È possibile utilizzare i seguenti comandi per applicare la policy a entrambe le policy:

```
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. Aggiorna il tuo browser. L'interfaccia utente di gestione può raggiungere nuovamente i nodi, ma i nodi non sono in grado di comunicare tra loro.

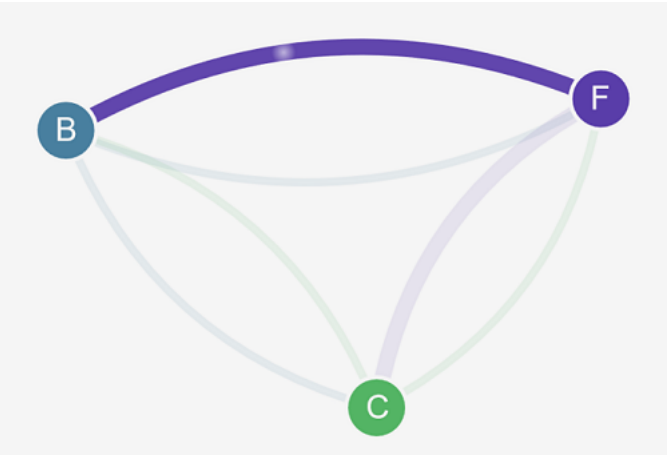


9. Applicare le seguenti policy di rete per consentire il traffico dal servizio front-end al servizio back-end:

```
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 Aggiorna il tuo browser. Puoi vedere che il front-end può comunicare con il back-end.

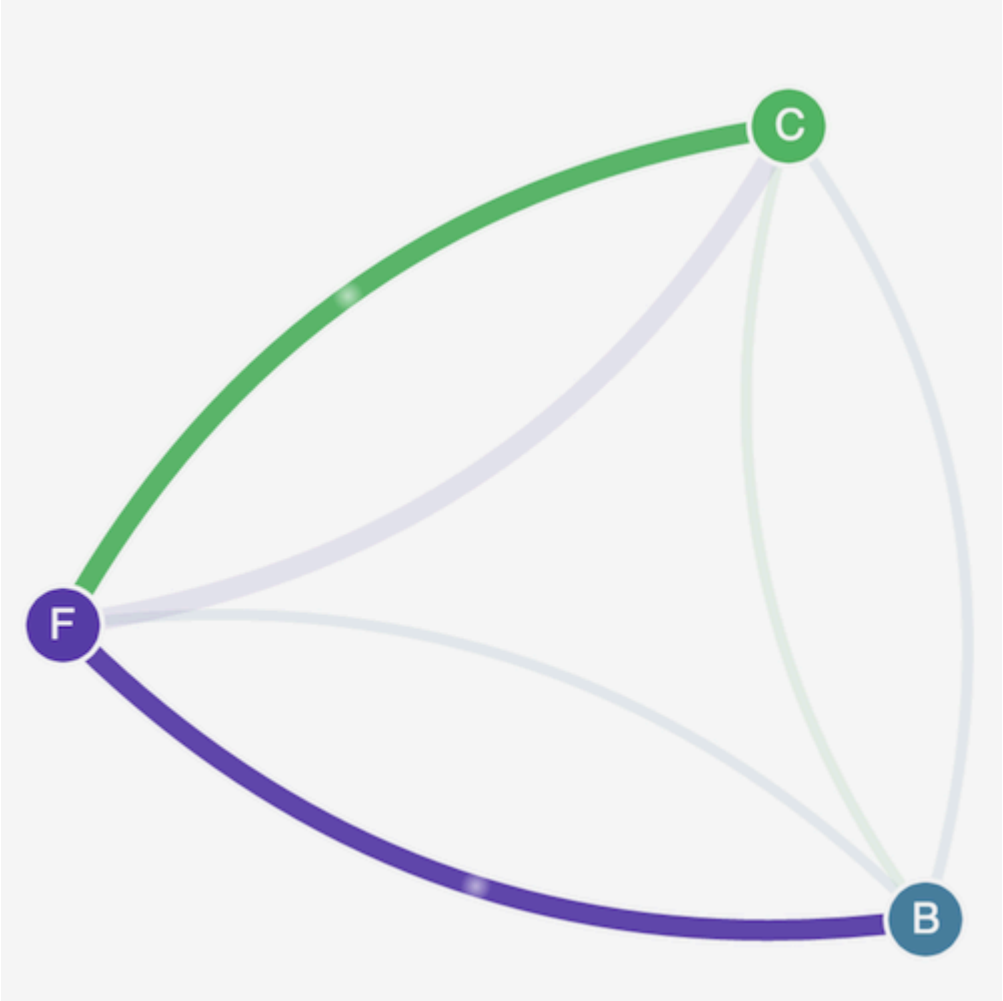


11 Applica la seguente policy di rete per consentire il traffico dal client al servizio di front-end:

```
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. Aggiorna il tuo browser. Puoi vedere che il client può comunicare con il servizio di front-end. Il servizio di front-end può ancora comunicare con il servizio di back-end.



13. (Facoltativo) Al termine della demo, puoi eliminare le relative risorse.

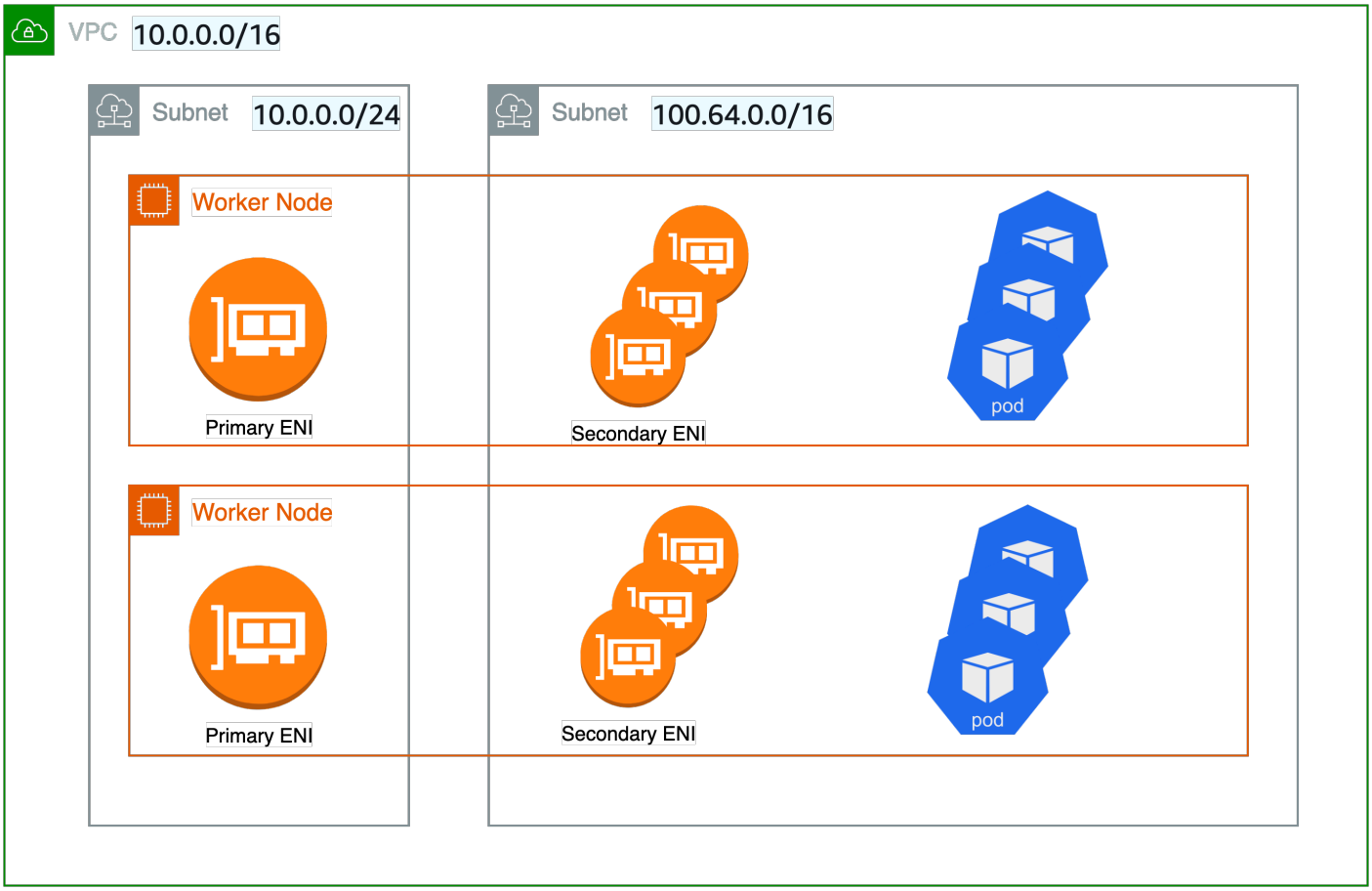
```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
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
```

Anche dopo l'eliminazione delle risorse, sui nodi possono essere presenti endpoint di policy di rete che potrebbero interferire in modo imprevisto con la rete nel cluster. L'unico modo sicuro per rimuovere queste regole è riavviare i nodi o terminare tutti i nodi e riciclarli. Per terminare tutti i nodi, impostare il conteggio desiderato del Gruppo Auto Scaling su 0, quindi eseguire il backup al numero desiderato oppure terminare semplicemente i nodi.

Implementazione dei pod in sottoreti alternative con reti personalizzate

Si applica a: nodi Fargate Linux `IPv4`, nodi Linux con istanze Amazon EC2



Per impostazione predefinita, il plug-in CNI di Amazon VPC per Kubernetes crea interfacce di rete elastiche secondarie (interfacce di rete) per il nodo Amazon EC2 nella stessa sottorete dell'interfaccia di rete primaria del nodo. Inoltre, associa all'interfaccia di rete secondaria gli stessi gruppi di sicurezza associati a quella primaria. È possibile far sì che il plugin crei interfacce di rete secondarie in una sottorete diversa o associ gruppi di sicurezza diversi alle interfacce di rete secondarie, o entrambe le cose, in base a uno dei motivi riportati di seguito:

- È disponibile un numero limitato di indirizzi `IPv4` nella sottorete in cui si trova l'interfaccia di rete primaria. Ciò potrebbe limitare il numero di Pod che è possibile creare nella sottorete. Utilizzando una sottorete diversa per le interfacce di rete secondarie, puoi aumentare il numero di indirizzi `IPv4` disponibili per i Pod.
- Per motivi di sicurezza, i tuoi Pod dovrebbero usare sottoreti o gruppi di sicurezza diversi rispetto all'interfaccia di rete primaria del nodo.
- I nodi sono configurati in sottoreti pubbliche e puoi collocare i Pod in sottoreti private. La tabella di instradamento associata a una sottorete pubblica include un percorso a un gateway Internet. La tabella di routing associata a una sottorete privata non include una route a un gateway Internet.

> **Tip**
>
> Puoi anche aggiungere una sottorete nuova o esistente direttamente al cluster Amazon EKS, senza utilizzare reti personalizzate. Per ulteriori informazioni, consulta the section called "Aggiunta di una sottorete VPC esistente a un cluster Amazon EKS dalla console di gestione".

Considerazioni

Di seguito sono riportate alcune considerazioni sull'utilizzo della funzionalità.

- L'abilitazione di una rete personalizzata non consente di assegnare ai Pod gli indirizzi IP associati all'interfaccia di rete primaria. Solo gli indirizzi IP delle interfacce di rete secondarie vengono assegnati ai Pod.
- Se il cluster utilizza la famiglia `IPv6`, non puoi usare la rete personalizzata.
- Se si prevede di utilizzare una rete personalizzata esclusivamente per ridurre l'esaurimento dell'indirizzo `IPv4`, è invece possibile creare un cluster tramite la famiglia `IPv6`. Per ulteriori informazioni, consulta the section called "IPv6".

- I Pod implementati nelle sottoreti specificate per le interfacce di rete secondarie possono utilizzare sottoreti e gruppi di sicurezza diversi rispetto all'interfaccia di rete primaria del nodo, a patto che si trovino nello stesso VPC del nodo.
- Per Fargate, le sottoreti sono controllate tramite il profilo Fargate. Per ulteriori informazioni, consulta the section called "Definisci i profili".

Personalizzare l'interfaccia di rete secondaria nei nodi di Amazon EKS

Prima di iniziare questo tutorial, completa le seguenti operazioni:

- Esaminare le considerazioni
- Familiarità con il modo in cui il plug-in CNI di Amazon VPC per Kubernetes crea interfacce di rete secondarie e assegna indirizzi IP ai Pod. Per ulteriori informazioni, consulta Allocazione di ENI su GitHub.
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Ti consigliamo di completare la procedura descritta in questo argomento in una shell Bash. In alternativa, puoi apportare alcune modifiche alla tua shell per alcuni comandi di script, come i caratteri di continuazione della riga, e per il modo in cui le variabili vengono impostate e utilizzate. Inoltre, le regole di escape e di utilizzo delle virgolette per la shell (interprete di comandi) potrebbero essere diverse. Per ulteriori informazioni, consulta Uso delle virgolette con le stringhe nella AWS CLI nella Guida per l'utente dell'interfaccia a riga di AWS comando.

Per questo tutorial, ti consigliamo di utilizzare i valori di esempio tranne quando è indicato di sostituirli. Puoi sostituire qualsiasi valore di esempio al completamento delle fasi dedicate alla creazione del cluster. Consigliamo di completare tutte le fasi nello stesso terminale, dato che le variabili impostate e utilizzate durante la procedura non sono presenti in terminali diversi.

I comandi in questo argomento sono formattati utilizzando le convenzioni elencate in Utilizzo degli esempi CLI AWS. Se esegui comandi dalla riga di comando su risorse che si trovano in una AWS regione diversa da quella predefinita definita nel profilo AWS CLI che stai utilizzando, devi aggiungerli `--region us-west-2` ai comandi, sostituendoli `us-west-2` con la tua AWS AWS regione.

Per implementare le reti personalizzate nel cluster di produzione, passa alla the section called "Passaggio 2: configurazione del VPC".

Passaggio 1: creazione di un VPC e di un cluster di test

Le procedure seguenti illustrano la creazione di un VPC e un cluster di test, oltre alla configurazione di una rete personalizzata per tale cluster. Consigliamo di non utilizzare il cluster di test per i carichi di lavoro di produzione poiché alcune funzionalità da utilizzare con il cluster di produzione non sono trattate in questo argomento. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ".

1. Eseguire il seguente comando per definire la variabile `account_id`.

```
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. Crea un VPC.
   a. Se esegui la distribuzione su un sistema di test, crea un VPC utilizzando un modello Amazon AWS CloudFormation EKS.

```
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. La AWS CloudFormation creazione dello stack richiede alcuni minuti. Utilizzare il comando seguente per verificare lo stato di implementazione dello stack.

```
aws cloudformation describe-stacks --stack-name my-eks-custom-networking-vpc --
query Stacks\[\].StackStatus  --output text
```

   Non passare alla Passaggio successiva finché l'output del comando non è `CREATE_COMPLETE`.

c. Definisci le variabili con i valori della sottorete privata IDs creata dal modello.

```
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. Definire le variabili con le zone di disponibilità delle sottoreti recuperate nel passaggio precedente.

```
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. Crea un ruolo IAM del cluster.

a. Per creare un file JSON della policy di attendibilità IAM, esegui il comando seguente.

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

b. Crea il ruolo IAM del cluster Amazon EKS. Se necessario, inserisci il prefisso `eks-cluster-role-trust-policy.json` nel percorso del computer in cui hai eseguito la scrittura del file nella Passaggio precedente. Il comando associa il criterio di attendibilità creato nella Passaggio precedente al ruolo. Per creare un ruolo IAM, l'azione `iam:CreateRole` (autorizzazione) deve essere assegnata al principale IAM che sta creando il ruolo.

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

c. Allega la policy gestita di Amazon EKS denominata Amazon EKSCluster Policy al ruolo. Per allegare una policy IAM a un principale IAM, è necessario assegnare al principale che sta allegando la policy una delle azioni IAM (autorizzazioni) seguenti: `iam:AttachUserPolicy` o `iam:AttachRolePolicy`.

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

4. Crea un cluster Amazon EKS e configura il dispositivo per comunicare con esso.

a. Crea un 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**
>
> Potresti ricevere un messaggio di errore indicante che una delle zone di disponibilità nella richiesta non dispone di capacità sufficiente per creare un cluster Amazon EKS. In questo caso, l'output di errore contiene le zone di disponibilità in grado di supportare un nuovo cluster. Riprova a creare il cluster con almeno due sottoreti che si trovano nelle zone di disponibilità supportate per il tuo account. Per ulteriori informazioni, consulta the section called "Capacità insufficiente".

b. La creazione del cluster richiede diversi minuti. Utilizzare il comando seguente per verificare lo stato di implementazione del cluster.

```
aws eks describe-cluster --name my-custom-networking-cluster --query
 cluster.status
```

Non passare alla Passaggio successiva finché l'output del comando non è `"ACTIVE"`.

c. Configura `kubectl` per comunicare con il cluster.

```
aws eks update-kubeconfig --name my-custom-networking-cluster
```

Passaggio 2: configurazione del VPC

Questo tutorial richiede il VPC creato nella the section called "Passaggio 1: creazione di un VPC e di un cluster di test". Per un cluster di produzione, modificare i passaggi in base al VPC, sostituendo tutti i valori di esempio con i propri.

1. Accertati che la versione del plug-in CNI di Amazon VPC per Kubernetes installata sia l'ultima versione. Per determinare la versione più recente per il componente aggiuntivo del tipo Amazon EKS e aggiornarla, consulta la pagina the section called "Aggiornamento di un componente aggiuntivo". Per determinare la versione più recente per il componente aggiuntivo del tipo autogestito e aggiornarla, consulta la pagina the section called "CNI di Amazon VPC".
2. Recupera l'ID del VPC in cui si trova il cluster e memorizzarlo in una variabile per l'utilizzo nei passaggi successivi.

   ```
   vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query
    "cluster.resourcesVpcConfig.vpcId" --output text)
   ```
3. Associare un intervallo di routing interdominio senza classi (CIDR) aggiuntivo al VPC del cluster. L'intervallo CIDR non può sovrapporsi agli intervalli CIDR esistenti già associati.

   a. Visualizza i blocchi CIDR attualmente associati al tuo VPC.

   ```
   aws ec2 describe-vpcs --vpc-ids $vpc_id \
       --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
    CidrBlockState.State}' --out table
   ```

   Di seguito viene riportato un output di esempio:

   ```
   ---------------------------------
   |          DescribeVpcs         |
   +-----------------+-------------+
   |    CIDRBlock    |    State    |
   +-----------------+-------------+
   |  192.168.0.0/24 |  associated |
   +-----------------+-------------+
   ```

b. Associa un blocco CIDR aggiuntivo al VPC. Sostituire il valore dell'intervallo CIDR nel comando seguente. Per ulteriori informazioni, consulta Associare blocchi IPv4 CIDR aggiuntivi al tuo VPC nella Amazon VPC User Guide.

```
aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 192.168.1.0/24
```

c. Verifica che il nuovo blocco sia associato.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --query
 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

Di seguito viene riportato un output di esempio.

```
----------------------------------
|          DescribeVpcs          |
+-----------------+--------------+
|    CIDRBlock    |    State     |
+-----------------+--------------+
|  192.168.0.0/24 |  associated  |
|  192.168.1.0/24 |  associated  |
+-----------------+--------------+
```

Non andare al passaggio successivo finché il nuovo `State` dell'intervallo CIDR non è `associated`.

4. Crea tutte le sottoreti che vuoi utilizzare in ogni zona di disponibilità in cui si trovano le sottoreti esistenti. Specificare un intervallo CIDR all'interno dell'intervallo CIDR associato al VPC in una Passaggio precedente.

a. Crea nuove sottoreti. Sostituire i valori dell'intervallo CIDR nel seguente comando. È necessario creare le sottoreti in un blocco CIDR VPC diverso rispetto alle sottoreti esistenti, ma all'interno delle stesse zone di disponibilità. In questo esempio si crea una sottorete nel nuovo blocco CIDR in ogni zona di disponibilità in cui esistono le sottoreti private attuali. Le IDs sottoreti create vengono memorizzate in variabili da utilizzare nelle fasi successive. I valori `Name` corrispondono a quelli assegnati alle sottoreti create utilizzando il modello Amazon EKS VPC in un passaggio precedente. I nomi non sono obbligatori. È possibile utilizzare nomi diversi.

```
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**
>
> Per impostazione predefinita, le nuove sottoreti sono implicitamente associate alla tabella di routing principale del VPC. Questa tabella di instradamento consente la comunicazione tra tutte le risorse implementate nel VPC, ad eccezione delle risorse con indirizzi IP esterni ai blocchi CIDR associate al VPC. Per modificare questo comportamento, associa la tabella di instradamento alle sottoreti. Per ulteriori informazioni, consulta Tabelle di instradamento per sottoreti nella Guida per l'utente di Amazon VPC.

b. Per visualizzare le sottoreti attuali nel tuo VPC.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" \
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

Di seguito viene riportato un output di esempio:

```
-------------------------------------------------------------------
|                          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       |
+------------------+--------------------+---------------------------+
```

Come puoi vedere, le sottoreti create nel blocco CIDR `192.168.1.0` si trovano nelle stesse zone di disponibilità di quelle create nel blocco CIDR `192.168.0.0`.

Passaggio 3: configurazione delle risorse Kubernetes

1. Imposta la variabile di `AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG` ambiente su `true` in. `aws-node` DaemonSet

```
kubectl set env daemonset aws-node -n kube-system
 AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

2. Recupera l'ID del gruppo di sicurezza del cluster e memorizzalo in una variabile da utilizzare nel passaggio successivo. Amazon EKS crea automaticamente questo gruppo di sicurezza durante la creazione del cluster.

```
cluster_security_group_id=$(aws eks describe-cluster --name my-custom-networking-
cluster --query cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
```

3. Creare una risorsa personalizzata `ENIConfig` per ogni sottorete in cui vuoi implementare i Pod.

   a. Crea un file univoco per ogni configurazione di interfaccia di rete.

   I comandi seguenti creano file `ENIConfig` separati per le due sottoreti create in una Passaggio precedente. Il valore di `name` deve essere univoco. Il nome corrisponde alla zona di disponibilità in cui si trova la sottorete. Il gruppo di sicurezza del cluster viene assegnato 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
```

Per un cluster di produzione, puoi apportare le seguenti modifiche ai comandi precedenti:

- Sostituire $cluster_security_group_id con l'ID di un gruppo di sicurezza esistente da utilizzare per ciascun `ENIConfig`.
- Quando possibile, consigliamo di assegnare a `ENIConfigs` lo stesso nome della zona di disponibilità che verrà utilizzato per `ENIConfig`. Per una serie di motivi, tuttavia, potrebbe essere necessario utilizzare nomi diversi per `ENIConfigs` e le zone di disponibilità. Ad esempio, se nella stessa zona di disponibilità sono presenti più di due sottoreti e si desidera utilizzarle entrambe con reti personalizzate, saranno necessari più `ENIConfigs` per la stessa zona di disponibilità. Dal momento che ogni `ENIConfig` richiede un nome univoco, non puoi assegnare lo stesso nome della zona di disponibilità a più `ENIConfigs`.

  Se i tuoi `ENIConfig` nomi non sono tutti uguali ai nomi delle zone di disponibilità, sostituisci $az_1 e $az_2 con i tuoi nomi nei comandi precedenti e annota i tuoi nodi con quelli successivi di questo tutorial. ENIConfig

  > **Note**
  >
  > Nel caso in cui non specifichi un gruppo di sicurezza valido da utilizzare con un cluster di produzione:

- se utilizzi la versione `1.8.0` o successiva del plug-in CNI di Amazon VPC per Kubernetes, vengono impiegati i gruppi di sicurezza associati all'interfaccia di rete elastica primaria del nodo.
- se utilizzi una versione del plug-in CNI di Amazon VPC per Kubernetes precedente alla `1.8.0`, il gruppo di sicurezza di default per il VPC viene assegnato alle interfacce di rete secondarie.

> **Important**
>
> - `AWS_VPC_K8S_CNI_EXTERNALSNAT=false` è un'impostazione predefinita per la configurazione del plugin CNI di Amazon VPC per Kubernetes. Per impostazione predefinita, il traffico destinato agli indirizzi IP che non si trovano all'interno di uno dei blocchi CIDR associati al VPC utilizza i gruppi di sicurezza e le sottoreti dell'interfaccia di rete primaria del nodo. Le sottoreti e i gruppi di sicurezza definiti in `ENIConfigs` al fine di creare interfacce di rete secondarie non vengono utilizzati per questo traffico. Per ulteriori informazioni su questa impostazione, consulta the section called "Traffico in uscita".
> - Se utilizzi gruppi di sicurezza anche per i Pod, il gruppo di sicurezza specificato in una `SecurityGroupPolicy` viene utilizzato al posto di quello specificato in `ENIConfigs`. Per ulteriori informazioni, consulta the section called "Gruppi di sicurezza per pod".

b. Applica ogni file delle risorse personalizzato creato in precedenza al cluster con i comandi seguenti.

```
kubectl apply -f $az_1.yaml
kubectl apply -f $az_2.yaml
```

4. Verifica che `ENIConfigs` sia stato creato.

```
kubectl get ENIConfigs
```

Di seguito viene riportato un output di esempio.

```
NAME         AGE
us-west-2a   117s
us-west-2d   105s
```

5. Se stai abilitando reti personalizzate su un cluster di produzione e hai un nome `ENIConfigs` diverso dalla Zona di disponibilità per cui le stai utilizzando, passa al passaggio successivo per distribuire i nodi Amazon EC2 .

   Consenti a Kubernetes di applicare automaticamente la zona `ENIConfig` di disponibilità a tutti i nuovi EC2 nodi Amazon creati nel tuo cluster.

a. Per il cluster di test in questo tutorial, passa alla Passaggio successiva.

   Per un cluster di produzione, controlla se esiste un'annotazione con la chiave `k8s.amazonaws.com/eniConfig` per la variabile di `ENI_CONFIG_ANNOTATION_DEF` ambiente nelle specifiche del contenitore per. `aws-node` DaemonSet

   ```
   kubectl describe daemonset aws-node -n kube-system | grep
    ENI_CONFIG_ANNOTATION_DEF
   ```

   Se viene restituito un output, l'annotazione è presente. Se non viene restituito alcun output, la variabile non viene impostata. Per un cluster di produzione, puoi scegliere di utilizzare la presente impostazione o quella riportata nel passaggio successivo. L'utilizzo di questa impostazione sovrascrive quella del passaggio successivo. In questo tutorial viene utilizzata l'impostazione riportata nel passaggio successivo.

b. Aggiorna il tuo `aws-node` DaemonSet `ENIConfig` per applicare automaticamente la zona di disponibilità a tutti i nuovi EC2 nodi Amazon creati nel tuo cluster.

   ```
   kubectl set env daemonset aws-node -n kube-system
    ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
   ```

Fase 4: Implementazione dei nodi Amazon EC2

1. Creare un ruolo IAM del nodo.

   a. Per creare un file JSON della policy di attendibilità IAM, esegui il comando seguente.

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

b. Creare un ruolo IAM e memorizzare il nome della risorsa Amazon (ARN) restituito in una variabile da utilizzare in una Passaggio successiva.

```
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. Allega al ruolo IAM le tre policy gestite IAM richieste.

```
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**
>
> Per semplicità, in questo tutorial la policy AmazonEKS_CNI_Policy è allegata al ruolo IAM del nodo. Tuttavia, in un cluster di produzione, consigliamo di allegare la policy a un ruolo IAM separato da utilizzare solo con il plug-in CNI di Amazon VPC per Kubernetes. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".

2. Creare uno dei seguenti tipi di gruppi di nodi. Per determinare il tipo di istanza da implementare, consulta the section called "Tipi di istanza Amazon EC2". Per questo tutorial, utilizza l'opzione Gestito, senza un modello di avvio o con un modello di avvio senza un ID AMI specificato. Se prevedi di utilizzare il gruppo di nodi per i carichi di lavoro di produzione, ti consigliamo di familiarizzare con tutte le opzioni del gruppo di nodi gestito e del gruppo di nodi autogestito prima di implementare il gruppo di nodi.
   - Gestito: implementare il gruppo di nodi utilizzando una delle opzioni seguenti:
     - Senza un modello di avvio o con un modello di avvio senza un ID AMI specificato: esegui il comando seguente. Per questo tutorial, utilizza i valori di esempio. Per un gruppo di nodi di produzione, sostituire tutti i valori di esempio con i propri. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura.

```
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
```

- Con un modello di avvio con un ID AMI specificato

  A. Determinare il numero massimo di Pod consigliato da Amazon EKS per i nodi. Segui le istruzioni riportate in Amazon EKS, il numero massimo di Pods consigliato per ogni tipo di EC2 istanza Amazon, `--cni-custom-networking-enabled` aggiungendole al passaggio 3 di quell'argomento. Annota l'output restituito per l'utilizzo in un passaggio successivo.

  B. Nel modello di avvio, specifica un ID AMI ottimizzato per Amazon EKS o un'AMI personalizzata sviluppata con l'AMI ottimizzato per Amazon EKS, quindi implementa il gruppo di nodi utilizzando un modello di avvio e fornisci i seguenti dati utente nel modello di avvio. Questi dati utente passano gli argomenti alle `NodeConfig` specifiche. Per ulteriori informazioni in merito NodeConfig, consulta il riferimento all'NodeConfig API. Puoi sostituire `20` con il valore del passaggio precedente (consigliato) o con un valore desiderato.

```
---
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 hai creato un'AMI personalizzata che non è stata sviluppata con quella ottimizzata per Amazon EKS, devi creare autonomamente la configurazione.

- Autogestito

i. Determinare il numero massimo di Pod consigliato da Amazon EKS per i nodi. Seguire le istruzioni riportate in the section called "Per ogni tipo di istanza Amazon EC2, Amazon EKS consiglia un numero massimo di pod", aggiungendo `--cni-custom-networking-enabled` alla Passaggio 3 di questo argomento. Annota l'output restituito per l'utilizzo in un passaggio successivo.

ii. Implementa il gruppo di nodi utilizzando le istruzioni in the section called "Amazon Linux".

> **Note**
>
> Se vuoi che i nodi presenti in un cluster di produzione supportino un numero significativamente più elevato di Pod, esegui nuovamente lo script in the section called "Per ogni tipo di istanza Amazon EC2, Amazon EKS consiglia un numero massimo di pod". Aggiungi inoltre l'opzione `--cni-prefix-delegation-enabled` al comando. Ad esempio, per un tipo di istanza `m5.large` viene restituito `110`. Per istruzioni su come abilitare questa funzionalità, consulta the section called "Aumentare gli indirizzi IP". Puoi utilizzare questa funzionalità con una rete personalizzata.

3. La creazione di gruppi di nodi richiede diversi minuti. Puoi verificare lo stato della creazione di un gruppo di nodi gestiti con il comando seguente.

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

Non andare al passaggio successivo finché l'output restituito non è `ACTIVE`.

4. Ai fini di questo tutorial, puoi ignorare questo passaggio.

Per un cluster di produzione, se è stato assegnato a `ENIConfigs` un nome diverso da quello della zona di disponibilità in cui lo stai utilizzando, devi annotare i nodi con il nome `ENIConfig` da utilizzare con il nodo. Questo passaggio non è necessario se disponi di una sola sottorete in ogni zona di disponibilità e hai assegnato a `ENIConfigs` gli stessi nomi delle zone di disponibilità. Questo perché il plug-in CNI di Amazon VPC per Kubernetes associa automaticamente il `ENIConfig` corretto al nodo se questa opzione è stata abilitata in una Passaggio precedente.

a. Ottieni l'elenco dei nodi nel cluster.

```
kubectl get nodes
```

Di seguito viene riportato un output di esempio:

```
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. Determina in quale zona di disponibilità si trova ogni nodo. Eseguire il comando seguente per ogni nodo restituito nella Passaggio precedente, sostituendo gli indirizzi IP in base all'output precedente.

```
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}'
```

Di seguito viene riportato un output di esempio.

```
[
    {
        "AvailabilityZone": "us-west-2d",
        "SubnetId": "subnet-Example5"
    }
]
```

c. Annota ogni nodo con la risorsa `ENIConfig` creata per l'ID della sottorete e la zona di disponibilità. Puoi annotare solo un nodo con un `ENIConfig`, ma più nodi con lo stesso `ENIConfig`. Sostituire i valori di esempio con i propri valori.

```
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 in un cluster di produzione erano presenti nodi con Pod in esecuzione prima di passare all'utilizzo della funzionalità di rete personalizzata, completare le attività seguenti:

a. Assicurati di avere dei nodi disponibili abilitati per la funzionalità di rete personalizzata.

b. Cordonare ed espellere i nodi per interrompere normalmente i Pod. Per ulteriori informazioni, consulta Safely Drain a Node (Espellere un nodo in modo sicuro) nella documentazione Kubernetes.

c. Termina i nodi. Se i nodi si trovano in un gruppo di nodi gestito esistente, puoi eliminare il gruppo di nodi. Eseguire il seguente comando seguente.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup
```

Solo i nuovi nodi registrati con l'etichetta `k8s.amazonaws.com/eniConfig` useranno la nuova funzionalità di rete personalizzata.

6. Verificare che ai Pod venga assegnato un indirizzo IP da un intervallo CIDR associato a una delle sottoreti create in una Passaggio precedente.

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

Di seguito viene riportato un output di esempio.

```
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>
```

Puoi vedere che ai coredns vengono assegnati indirizzi IP dall'intervallo CIDR `192.168.1.0` aggiunto al VPC. Senza una rete personalizzata, ai pod sarebbero stati assegnati indirizzi

provenienti dall'unico blocco CIDR originariamente associato al VPC, ossia il blocco CIDR `192.168.0.0`.

Se una `spec` del Pod contiene il valore `hostNetwork=true`, viene assegnato l'indirizzo IP principale del nodo, non un indirizzo proveniente dalle sottoreti aggiunte. Per impostazione predefinita, questo valore è impostato su `false`. Questo valore è impostato su `true` per il `kube-proxy` e i pod (`aws-node`) del plug-in CNI di Amazon VPC per Kubernetes in esecuzione sul cluster. Per questo motivo al `kube-proxy` e ai pod `aws-node` del plug-in non vengono assegnati indirizzi 192.168.1.x nell'output precedente. Per ulteriori informazioni sull'`hostNetwork`impostazione di un Pod, consulta PodSpec v1 core nel riferimento all'API Kubernetes.

Passaggio 5: eliminazione delle risorse del tutorial

Ti consigliamo di eliminare le risorse create dopo aver completato il tutorial. In seguito, potrai modificare i passaggi per abilitare la rete personalizzata di un cluster di produzione.

1. Se il gruppo di nodi creato era solo a scopo di test, eliminalo.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name
 my-nodegroup
```

2. Anche dopo che l'output della AWS CLI indica che il cluster è stato eliminato, il processo di eliminazione potrebbe non essere effettivamente completo. Questo processo di eliminazione richiede alcuni minuti. Verificare che il processo sia completo con il comando seguente.

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

   Non continuare finché l'output restituito non è simile al seguente.

```
An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup
 operation: No node group found for name: my-nodegroup.
```

3. Se il gruppo di nodi creato era solo a scopo di test, elimina il ruolo IAM del nodo.

   a. Scollega le policy dal ruolo.

```
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. Elimina il ruolo.

```
aws iam delete-role --role-name myCustomNetworkingNodeRole
```

4. Elimina il cluster.

```
aws eks delete-cluster --name my-custom-networking-cluster
```

Verifica l'eliminazione del cluster con il comando seguente.

```
aws eks describe-cluster --name my-custom-networking-cluster --query cluster.status
 --output text
```

Quando viene restituito un output simile al seguente, il cluster è stato eliminato correttamente.

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-custom-networking-cluster.
```

5. Elimina il ruolo IAM del cluster.

a. Scollega le policy dal ruolo.

```
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole --
policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

b. Elimina il ruolo.

```
aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
```

6. Elimina le sottoreti create in una Passaggio precedente.

```
aws ec2 delete-subnet --subnet-id $new_subnet_id_1
aws ec2 delete-subnet --subnet-id $new_subnet_id_2
```

7. Elimina il VPC creato.

```
aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
```

Assegnazione di più indirizzi IP ai nodi Amazon EKS con prefissi

Si applica a: nodi Linux e Windows con istanze Amazon EC2

Si applica a: sottoreti pubbliche e private

Ogni istanza Amazon EC2 supporta un numero massimo di interfacce di rete elastiche e un numero massimo di indirizzi IP che possono essere assegnati a ogni interfaccia di rete. Ogni nodo richiede un indirizzo IP per ogni interfaccia di rete. Tutti gli altri indirizzi IP disponibili possono essere assegnati a `Pods`. Ogni `Pod` necessita di un proprio indirizzo IP. Di conseguenza, si potrebbero avere nodi con risorse di calcolo e memoria disponibili, ma che non possono ospitare ulteriori `Pods` perché il nodo ha esaurito gli indirizzi IP a cui assegnare `Pods`.

È possibile aumentare sensibilmente il numero di indirizzi IP che i nodi possono assegnare ai `Pods` assegnando prefissi IP ai nodi, anziché assegnare singoli indirizzi IP secondari. Ogni prefisso include diversi indirizzi IP. Se non configuri il cluster per l'assegnazione del prefisso IP, il cluster deve effettuare più chiamate di interfaccia di programmazione dell'applicazione (API) di Amazon EC2 per la configurazione delle interfacce di rete e degli indirizzi IP necessari alla connettività del Pod. La frequenza di queste chiamate API, man mano che i cluster acquisiscono dimensioni maggiori, può portare a tempi di avvio di Pod e delle istanze più lunghi. Ciò si traduce in ritardi di dimensionamento per soddisfare la domanda di carichi di lavoro grandi e spigolosi e aggiunge costi generali di gestione in quanto è necessario eseguire il provisioning di cluster e VPC aggiuntivi per soddisfare i requisiti di scalabilità. Consultare Kubernetes Scalability thresholds su GitHub per ulteriori informazioni.

Compatibilità con le funzionalità del plug-in CNI di Amazon VPC per Kubernetes

È possibile utilizzare prefissi IP con le seguenti caratteristiche:

- Traslazione degli indirizzi di rete di origine IPv4: Per ulteriori informazioni, consulta the section called "Traffico in uscita".
- Indirizzi IPv6 verso cluster, pod e servizi: Per ulteriori informazioni, consulta the section called "IPv6".
- Limitazione del traffico tramite le policy di rete di Kubernetes: Per ulteriori informazioni, consulta the section called "Policy di Kubernetes".

L'elenco seguente fornisce informazioni sulle impostazioni del plug-in CNI di Amazon VPC applicabili. Per maggiori informazioni su ciascuna impostazione, consultare amazon-vpc-cni-k8s su GitHub.

- `WARM_IP_TARGET`
- `MINIMUM_IP_TARGET`
- `WARM_PREFIX_TARGET`

Considerazioni

Quando si utilizza questa funzionalità, considerare i seguenti aspetti:

- Ogni tipo di istanza Amazon EC2 supporta un numero massimo di Pod. Se il gruppo di nodi gestito è costituito da più tipi di istanza, a tutti i nodi del cluster viene applicato il numero minimo di Pod massimo per un'istanza del cluster.
- Per impostazione predefinita, il numero massimo di `Pods` che puoi eseguire su un nodo è pari a 110, ma puoi modificare questo numero. Se cambi il numero e disponi di un gruppo di nodi gestiti, il prossimo aggiornamento dell'AMI o del modello di avvio del gruppo di nodi comporta la creazione di nuovi nodi con il valore modificato.
- Quando passi dall'assegnazione di indirizzi IP all'assegnazione di prefissi IP, si consiglia di creare nuovi gruppi di nodi per aumentare il numero di indirizzi IP disponibili, anziché effettuare una sostituzione progressiva dei nodi esistenti. L'esecuzione di Pod su un nodo a cui sono stati assegnati sia indirizzi IP che prefissi può portare a un'incoerenza nella capacità degli indirizzi IP pubblicizzati, con un impatto sui carichi di lavoro futuri sul nodo. Per il metodo consigliato di eseguire la transizione, consulta la modalità Prefix Delegation per Linux in Amazon EKS Best Practices Guide.
- L'ambito del gruppo di sicurezza è a livello di nodo: Per ulteriori informazioni, consulta Gruppo di sicurezza.
- I prefissi IP assegnati a un'interfaccia di rete supportano un'elevata densità di Pod per nodo e offrono il miglior tempo di avvio.
- I prefissi IP e gli indirizzi IP sono associati alle interfacce di rete elastiche Amazon EC2 standard. Ai pod che richiedono gruppi di sicurezza specifici viene assegnato l'indirizzo IP primario di un'interfaccia di rete di filiali. È possibile combinare i Pod ottenendo gli indirizzi IP o gli indirizzi IP dai prefissi IP con i Pod che ottengono interfacce di rete di filiali sullo stesso nodo.
- Solo per i cluster con nodi Linux.

- Dopo aver configurato il componente aggiuntivo per assegnare prefissi alle interfacce di rete, non è possibile eseguire il downgrade del componente aggiuntivo del plug-in CNI di Amazon VPC per Kubernetes a una versione inferiore a `1.9.0` (o `1.10.1`) senza rimuovere tutti i nodi in tutti i gruppi di nodi nel cluster.
- Se si utilizzano anche gruppi di sicurezza per i Pod, con `POD_SECURITY_GROUP_ENFORCING_MODE=standard` e `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`, quando i Pod comunicano con endpoint esterni al VPC, vengono utilizzati i gruppi di sicurezza del nodo, anziché i gruppi di sicurezza assegnati ai Pod.

  Se si utilizzano anche gruppi di sicurezza per Pod, con `POD_SECURITY_GROUP_ENFORCING_MODE=strict`, quando i `Pods` comunicano con endpoint esterni al VPC, vengono utilizzati i gruppi di sicurezza di `Pod's`.

Aumentare gli indirizzi IP disponibili per il nodo Amazon EKS

Puoi aumentare sensibilmente il numero di indirizzi IP che i nodi possono assegnare ai Pod assegnando prefissi IP ai nodi, anziché assegnare singoli indirizzi IP secondari.

Prerequisiti

- Necessiti di un cluster esistente. Per implementarne uno, consulta the section called "Creazione di un cluster ".
- Le sottoreti in cui si trovano i tuoi nodi Amazon EKS devono avere un numero sufficiente di blocchi di routing interdominio senza classi (CIDR) `/28` (per i cluster `IPv4`) o `/80` (per i cluster `IPv6`). Puoi disporre solo di nodi Linux in un cluster `IPv6`. L'uso dei prefissi IP può non riuscire se gli indirizzi IP sono sparsi nella sottorete CIDR. Consigliamo quanto segue:
  - L'utilizzo di una prenotazione di sottorete CIDR in modo che, anche se alcuni indirizzi IP all'interno dell'intervallo riservato sono ancora in uso, al momento del rilascio, gli indirizzi IP non vengano riassegnati. Ciò assicura che i prefissi siano disponibili per l'assegnazione senza segmentazione.
  - Usa nuove sottoreti utilizzate specificamente per l'esecuzione di carichi di lavoro a cui sono assegnati i prefissi IP. I carichi di lavoro sia di Windows sia di Linux possono essere eseguiti nella stessa sottorete quando si assegnano prefissi IP.

- Per assegnare prefissi IP ai nodi, i nodi devono essere basati su Nitro. AWS Le istanze che non sono basate su Nitro continuano ad allocare singoli indirizzi IP secondari, ma hanno un numero significativamente inferiore di indirizzi IP da assegnare ai Pod rispetto alle istanze basate su Nitro.
- Solo per cluster con nodi Linux`IPv4`: se il cluster è configurato per la famiglia, devi avere installata la versione `1.9.0` o una versione successiva del componente aggiuntivo del plug-in CNI di Amazon VPC per Kubernetes. Puoi controllare la versione attuale con il seguente comando.

  ```
  kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/"
   -f 2
  ```

  Se il cluster è configurato per la famiglia `IPv6`, è necessario che sia installata la versione `1.10.1` del componente aggiuntivo. Se la versione del plugin è precedente a quelle richieste, è necessario aggiornarla. Per ulteriori informazioni, consulta le sezioni di aggiornamento di Assign IPs to Pods con Amazon VPC CNI.
- Solo per i cluster con nodi Windows
  - Per il cluster, devi abilitare il supporto Windows. Per ulteriori informazioni, consulta the section called "Abilitazione del supporto Windows".

Assegnare i prefissi degli indirizzi IP ai nodi

Configurare il cluster per assegnare i prefissi degli indirizzi IP ai nodi. Completare la procedura corrispondente al sistema operativo del nodo.

Linux

1. Abilita il parametro per assegnare prefissi alle interfacce di rete per Amazon VPC CNI. DaemonSet Quando implementi un cluster, la versione `1.10.1` o successiva del componente aggiuntivo del plug-in CNI di Amazon VPC per Kubernetes viene implementata con esso. Se crei il cluster con la famiglia `IPv6`, questa impostazione era su `true` di default. Se crei il cluster con la famiglia `IPv4`, questa impostazione era su `false` di default.

   ```
   kubectl set env daemonset aws-node -n kube-system ENABLE_PREFIX_DELEGATION=true
   ```

> **Important**
>
> Anche se la sottorete dispone di indirizzi IP disponibili, se non dispone di blocchi `/28` contigui disponibili, vedrai il seguente errore nei log del plug-in di Amazon VPC CNI per Kubernetes.
>
> ```
> InsufficientCidrBlocks: The specified subnet does not have enough free cidr
>  blocks to satisfy the request
> ```
>
> Ciò può verificarsi a causa della frammentazione degli indirizzi IP secondari esistenti distribuiti in una sottorete. Per risolvere questo errore, crea una nuova sottorete e avvia i Pod lì oppure utilizza una prenotazione CIDR di EC2 subnet Amazon per riservare spazio all'interno di una sottorete da utilizzare con l'assegnazione del prefisso. Per ulteriori informazioni, consulta la sezione relativa a Prenotazioni della CIDR per la sottorete nella Guida per l'utente di Amazon VPC.

2. Se intendi implementare un gruppo di nodi gestito senza un modello di avvio o con uno in cui non è specificato un ID AMI e se utilizzi una versione del componente aggiuntivo del plug-in CNI di Amazon VPC per Kubernetes corrispondente o successiva alle versioni riportate nei prerequisiti, vai al passaggio successivo. I gruppi di nodi gestiti calcolano automaticamente il numero massimo di Pod.

   Se stai implementando un gruppo di nodi autogestito o un gruppo di nodi gestito con un modello di avvio in cui è specificato un ID AMI, devi determinare il numero massimo di Pod consigliati da Amazon EKS per i nodi. Segui le istruzioni riportate in Amazon EKS, il numero massimo di Pods consigliato per ogni tipo di EC2 istanza Amazon, `--cni-prefix-delegation-enabled` da aggiungere al passaggio 3. Annotare l'output restituito per l'utilizzo in un passaggio successivo.

   > **Important**
   >
   > I gruppi di nodi gestiti applicano un numero massimo sul valore di `maxPods`. Per le istanze con meno di 30 v CPUs il numero massimo è 110 e per tutte le altre istanze il numero massimo è 250. Questo numero massimo viene applicato indipendentemente dal fatto che la delega del prefisso sia abilitata o meno.

3. Se usi un cluster configurato per `IPv6`, vai alla Passaggio successiva.

È possibile specificare i parametri una delle seguenti opzioni. Per determinare quale opzione è adatta a te e quale valore fornirti, consulta WARM_PREFIX_TARGET, WARM_IP_TARGET e MINIMUM_IP_TARGET su. GitHub

Puoi sostituire i valori di esempio con un valore maggiore di zero.

- `WARM_PREFIX_TARGET`

```
kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
```

- `WARM_IP_TARGET` o `MINIMUM_IP_TARGET` – Se uno dei due valori è impostato, sovrascrive qualsiasi valore impostato per `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. Crea uno dei seguenti tipi di gruppi di nodi con almeno un tipo di istanza Amazon EC2 Nitro Amazon Linux 2023. Per un elenco dei tipi di istanze Nitro, consulta Istanze costruite sul sistema Nitro nella Amazon EC2 User Guide. Questa funzionalità non è supportata da Windows. Per le opzioni che includono *110*, sostituirlo con il valore del passaggio 3 (consigliato) o con il proprio valore.

   - Gestione automatica: implementare il gruppo di nodi utilizzando le istruzioni in Create self-managed Amazon Linux nodes. Prima di creare lo CloudFormation stack, apri il file modello e modificalo `UserData` in modo che sia `NodeLaunchTemplate` come segue

```
...
            apiVersion: node.eks.aws/v1alpha1
            kind: NodeConfig
            spec:
              cluster:
                name: ${ClusterName}
                apiServerEndpoint: ${ApiServerEndpoint}
                certificateAuthority: ${CertificateAuthorityData}
                cidr: ${ServiceCidr}
              kubelet:
                config:
                  maxPods: 110
...
```

Se stai utilizzando `eksctl` per creare il gruppo di nodi, puoi usare il seguente comando.

```
eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-node
 110
```

- Gestito: implementare il gruppo di nodi utilizzando una delle opzioni seguenti:
  - Senza un modello di avvio o con un modello di avvio senza un ID AMI specificato: completare la procedura in Create a managed node group for your cluster. I gruppi di nodi gestiti calcolano automaticamente il valore massimo di `max-pods` consigliato da Amazon EKS.
  - Con un modello di avvio con un ID AMI specificato – nel modello di lancio, specificare un ID AMI ottimizzato per Amazon EKS o un'AMI personalizzata sviluppata con l'AMI ottimizzato Amazon EKS, quindi implementare il gruppo di nodi utilizzando un modello di avvio e fornire i seguenti dati utente nel modello di avvio. Questi dati utente passano un `NodeConfig` oggetto che deve essere letto dallo `nodeadm` strumento sul nodo. Per ulteriori informazioni in meritonodeadm, consulta la documentazione di 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 stai utilizzando `eksctl` per creare il gruppo di nodi, puoi usare il seguente comando.

    ```
    eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
    ```

Se hai creato un'AMI personalizzata che non è stata sviluppata con quella ottimizzata per Amazon EKS, devi creare autonomamente la configurazione.

> **Note**
>
> Se desideri assegnare anche indirizzi IP ai Pod di una sottorete diversa da quella dell'istanza, devi abilitare la funzionalità in questa Passaggio. Per ulteriori informazioni, consulta the section called "Rete personalizzata".

Windows

1. Abilita l'assegnazione di prefissi IP.
   a. Apri `amazon-vpc-cni ConfigMap` per la modifica.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. Aggiungi la seguente riga al file `data`:

   ```
     enable-windows-prefix-delegation: "true"
   ```

   c. Salva il file e chiudi l'editor.
   d. Accertati che la riga sia stata aggiunta a `ConfigMap`.

   ```
   kubectl get configmap -n kube-system amazon-vpc-cni -o "jsonpath={.data.enable-
   windows-prefix-delegation}"
   ```

   Se l'output restituito non è `true`, potrebbe essersi verificato un errore. Prova a completare di nuovo il passaggio.

   > **Important**
   >
   > Anche se la sottorete dispone di indirizzi IP disponibili, se non dispone di blocchi `/28` contigui disponibili, vedrai il seguente errore nei log del plug-in di Amazon VPC CNI per Kubernetes.

```
InsufficientCidrBlocks: The specified subnet does not have enough free cidr
 blocks to satisfy the request
```

Ciò può verificarsi a causa della frammentazione degli indirizzi IP secondari esistenti distribuiti in una sottorete. Per risolvere questo errore, crea una nuova sottorete e avvia i Pod lì oppure utilizza una prenotazione CIDR di EC2 subnet Amazon per riservare spazio all'interno di una sottorete da utilizzare con l'assegnazione del prefisso. Per ulteriori informazioni, consulta la sezione relativa a Prenotazioni della CIDR per la sottorete nella Guida per l'utente di Amazon VPC.

2. (Facoltativo) Specifica una configurazione aggiuntiva per controllare il comportamento di prescalabilità e scalabilità dinamica del cluster. Per ulteriori informazioni, consulta Opzioni di configurazione con la modalità di delega dei prefissi su Windows on. GitHub

   a. Apri `amazon-vpc-cni ConfigMap` per la modifica.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. Sostituite i valori di esempio con un valore maggiore di zero e aggiungete le voci necessarie alla `data` sezione di. `ConfigMap` Se imposti un valore per `warm-ip-target` o `minimum-ip-target`, il valore sostituisce qualsiasi valore impostato per `warm-prefix-target`.

   ```
   warm-prefix-target: "1"
   warm-ip-target: "5"
   minimum-ip-target: "2"
   ```

   c. Salva il file e chiudi l'editor.

3. Crea gruppi di nodi Windows con almeno un tipo di istanza Amazon EC2 Nitro. Per un elenco dei tipi di istanze Nitro, consulta Istanze costruite sul sistema Nitro nella Amazon EC2 User Guide. Per impostazione predefinita, il numero massimo di Pod che è possibile distribuire su un nodo è pari a 110. Se vuoi aumentare o diminuire questo numero, specifica quanto segue nei dati utente per la configurazione di bootstrap. Sostituisci *`max-pods-quantity`* con il tuo valore di pod massimo.

   ```
   -KubeletExtraArgs '--max-pods=max-pods-quantity'
   ```

   Se si sta implementando gruppi di nodi gestiti, questa configurazione deve essere aggiunta nel modello di avvio. Per ulteriori informazioni, consulta the section called "Modelli di avvio". Per

ulteriori informazioni sui parametri di configurazione dello script bootstrap di Windows, consultare the section called "Parametri di configurazione dello script di bootstrap".

Determinare il numero massimo di pod e gli indirizzi IP disponibili

1. Dopo aver implementato i nodi, visualizzare i nodi del cluster.

```
kubectl get nodes
```

Di seguito viene riportato un output di esempio:

```
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. Descrivi uno dei nodi per determinare il valore di `max-pods` per il nodo e il numero di indirizzi IP disponibili. Sostituisci *`192.168.30.193`* con l'indirizzo `IPv4` nel nome di uno dei tuoi nodi restituiti nell'output del passaggio precedente.

```
kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|
PrivateIPv4Address'
```

Di seguito viene riportato un output di esempio.

```
pods:                                110
vpc.amazonaws.com/PrivateIPv4Address:  144
```

Nell'output precedente, `110` è il numero massimo di Pod che Kubernetes distribuirà sul nodo, anche se gli indirizzi IP sono disponibili. *`144`*

Assegnazione dei gruppi di sicurezza ai singoli pod

Si applica a: nodi Linux con EC2 istanze Amazon

Si applica a: sottoreti private

I gruppi di sicurezza per Pods integrano i gruppi EC2 di sicurezza Amazon con Kubernetes Pods. Puoi utilizzare i gruppi EC2 di sicurezza Amazon per definire regole che consentano il traffico di rete in entrata e in uscita da e verso i Pods da distribuire ai nodi in esecuzione su molti EC2 tipi di istanze Amazon e Fargate. Per una descrizione dettagliata di questa funzionalità, consulta il post di blog Introduzione ai gruppi di sicurezza per i pod.

Compatibilità con il plugin CNI di Amazon VPC per le funzionalità di Kubernetes

È possibile utilizzare gruppi di sicurezza per pod con le seguenti funzionalità:

- IPv4 Source Network Address Translation - Per ulteriori informazioni, consulta. the section called "Traffico in uscita"
- IPv6 indirizzi a cluster, pod e servizi - Per ulteriori informazioni, consulta. the section called "IPv6"
- Limitazione del traffico tramite le policy di rete di Kubernetes: per ulteriori informazioni, consulta the section called "Policy di Kubernetes".

Considerazioni

Prima di implementare i gruppi di sicurezza per i pod, considera i limiti e le condizioni seguenti:

- I gruppi di sicurezza per Pods non possono essere utilizzati con nodi Windows o EKS Auto Mode.
- I gruppi di sicurezza per Pods possono essere utilizzati con cluster configurati per la `IPv6` famiglia che contiene EC2 nodi Amazon utilizzando la versione 1.16.0 o successiva del plug-in Amazon VPC CNI. È possibile utilizzare i gruppi di sicurezza per i pod con cluster configurati per la famiglia `IPv6` che contengono solo nodi Fargate, utilizzando la versione 1.7.7 o successive del plugin CNI di Amazon VPC. Per ulteriori informazioni, consulta the section called "IPv6"
- I gruppi di sicurezza per Pods sono supportati dalla maggior parte delle famiglie di EC2 istanze Amazon basate su Nitro, ma non da tutte le generazioni di una famiglia. Ad esempio, sono supportate la famiglia e le generazioni dell'istanza `m5`, `c5`, `r5`, `m6g`, `c6g` e `r6g`. Non è supportato alcun tipo di istanza nella famiglia `t`. Per un elenco completo dei tipi di istanze supportati, consulta il file limits.go su. GitHub I nodi devono essere uno dei tipi di istanza elencati che contengono `IsTrunkingCompatible: true` nel file.
- Se si utilizzano allo stesso tempo una rete personalizzata e gruppi di sicurezza per i pod, il gruppo di sicurezza specificato dai gruppi di sicurezza per pod è utilizzato in alternativa al gruppo di sicurezza specificato in `ENIConfig`.

- Se si utilizza la versione `1.10.2` o precedente del plugin CNI di Amazon VPC e si include l'impostazione `terminationGracePeriodSeconds` nelle specifiche del pod, il valore per l'impostazione non può essere zero.
- Se si utilizza la versione `1.10` o precedente del plugin CNI di Amazon VPC o la versione `1.11` con l'impostazione predefinita `POD_SECURITY_GROUP_ENFORCING_MODE=strict`, i servizi Kubernetes di tipo `NodePort` e `LoadBalancer` che utilizzano le destinazioni di istanza con `externalTrafficPolicy` impostato su `Local` non sono supportati con i pod a cui si assegnano i gruppi di sicurezza. Per ulteriori informazioni sull'utilizzo di un load balancer con target di istanza, consultare the section called "Bilanciamento del carico di rete".
- La fonte NAT è disabilitata per il traffico in uscita dai pod con gruppi di sicurezza assegnati, in modo che siano applicate le regole dei gruppi di sicurezza in uscita, se si utilizza la versione `1.10` o precedente del plugin CNI di Amazon VPC o la versione `1.11` con l'impostazione predefinita `POD_SECURITY_GROUP_ENFORCING_MODE=strict`. Per accedere a Internet, è necessario avviare pod con gruppi di sicurezza assegnati su nodi distribuiti in una sottorete privata configurata con un gateway NAT o un'istanza. I pod con gruppi di sicurezza assegnati distribuiti alle sottoreti pubbliche non sono in grado di accedere a Internet.

  Se si utilizza la versione `1.11` o successiva del plugin con `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, il traffico del pod destinato all'esterno del VPC è convertito nell'indirizzo IP dell'interfaccia di rete primaria dell'istanza. Per questo traffico sono utilizzate le regole nei gruppi di sicurezza per l'interfaccia di rete primaria, anziché le regole nei gruppi di sicurezza dei pod.
- Per utilizzare le policy di rete Calico con pod associati ai gruppi di sicurezza, è necessario utilizzare la versione `1.11.0` o successiva del plugin CNI di Amazon VPC e impostare `POD_SECURITY_GROUP_ENFORCING_MODE=standard`. In caso contrario, il flusso di traffico da e verso i Pod con gruppi di sicurezza associati non è soggetto all'applicazione delle policy di rete di Calico e si limita esclusivamente all'applicazione dei gruppi EC2 di sicurezza di Amazon. Per aggiornare la versione di CNI di Amazon VPC, consulta la sezione the section called "CNI di Amazon VPC"
- I pod in esecuzione su EC2 nodi Amazon che utilizzano gruppi di sicurezza in cluster che utilizzano NodeLocal DNSCachesono supportati solo con la versione `1.11.0` o successiva del plug-in Amazon VPC CNI e con =. `POD_SECURITY_GROUP_ENFORCING_MODE` `standard` Per aggiornare la versione del plug-in CNI di Amazon VPC, consulta la sezione the section called "CNI di Amazon VPC"

- I gruppi di sicurezza per pod potrebbero portare a un aumento della latenza di avvio dei pod per quelli con un alto tasso di abbandono. Ciò è dovuto alla limitazione della velocità nel controller delle risorse.
- L'ambito del gruppo EC2 di sicurezza è a livello di POD. Per ulteriori informazioni, consulta Security group.

  Se hai impostato `POD_SECURITY_GROUP_ENFORCING_MODE=standard` e `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`, il traffico destinato agli endpoint esterni al VPC utilizza i gruppi di sicurezza del nodo, non i gruppi di sicurezza del pod.

Configurazione del plugin CNI di Amazon VPC per Kubernetes per gruppi di sicurezza per i pod Amazon EKS

Se si utilizzano pod con istanze Amazon EC2, devi configurare il plugin CNI di Amazon VPC per Kubernetes per gruppi di sicurezza

Se si utilizzano solo i pod di Fargate e non si dispone di nodi Amazon EC2 nel cluster, consulta the section called "SecurityGroupPolicy".

1. Controlla la versione corrente del plugin CNI di Amazon VPC per Kubernetes con il comando seguente:

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
    cut -d : -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.7.6
   ```

   Se la versione del plugin CNI di Amazon VPC per Kubernetes è precedente a `1.7.7`, aggiorna il plugin alla versione `1.7.7` o successiva. Per ulteriori informazioni, consulta the section called "CNI di Amazon VPC"
2. Aggiungi la policy IAM gestita da AmazonEKSVPCResourceControlleral ruolo del cluster associato al cluster Amazon EKS. La policy consente al ruolo di gestire le interfacce di rete, i relativi indirizzi IP privati e i relativi allegati e distacchi da e verso le istanze di rete.

   a. Recupera il nome del ruolo IAM del cluster e archivialo in una variabile. Sostituisci *my-cluster* con il nome del tuo cluster.

```
cluster_role=$(aws eks describe-cluster --name my-cluster --query cluster.roleArn
 --output text | cut -d / -f 2)
```

b. Collegare la policy al ruolo.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSVPCResourceController --role-name $cluster_role
```

3. Abilita il componente aggiuntivo CNI di Amazon VPC per gestire le interfacce di rete per i pod, impostando la variabileENABLE_POD_ENI su `true` in DaemonSet `aws-node`. Una volta impostato questo parametro su `true`, per ciascun nodo nel cluster il componente aggiuntivo crea una risorsa `cninode` personalizzata. Il controller di risorse VPC crea e allega un'interfaccia di rete speciale chiamata interfaccia di rete trunk con la descrizione `aws-k8s-trunk-eni`.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
```

> **Note**
>
> L'interfaccia di rete trunk è inclusa nel numero massimo di interfacce di rete supportate dal tipo di istanza. Per un elenco del numero massimo di interfacce di rete supportate da ciascun tipo di istanza, consulta Indirizzi IP per interfaccia di rete per tipo di istanza nella Guida per l'utente di Amazon EC2. Se il nodo ha già allegato il numero massimo di interfacce di rete standard, il controller di risorse VPC riserverà uno spazio. Sarà necessario ridurre verticalmente i pod in esecuzione abbastanza da consentire al controller di scollegare ed eliminare un'interfaccia di rete standard, creare l'interfaccia di rete trunk e allegarla all'istanza.

4. Con il seguente comando è possibile vedere quale nodo ha una risorsa `CNINode` personalizzata. Se viene restituito l'output `No resources found`, attendere alcuni secondi e riprovare. La fase precedente richiede il riavvio del plugin CNI di Amazon VPC per pod Kubernetes, che richiede alcuni secondi.

```
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 si utilizzano versioni VPC CNI precedenti alla `1.15`, sono state utilizzate le etichette dei nodi anziché la risorsa `CNINode` personalizzata. Con il seguente comando, è possibile vedere quale nodo ha l'etichetta del nodo `aws-k8s-trunk-eni` impostata su `true`. Se viene restituito l'output `No resources found`, attendere alcuni secondi e riprovare. La fase precedente richiede il riavvio del plugin CNI di Amazon VPC per pod Kubernetes, che richiede alcuni secondi.

```
kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
```

Una volta creata l'interfaccia di rete trunk, ai pod possono essere assegnati indirizzi IP secondari dalla rete trunk o dalle interfacce di rete standard. L'interfaccia trunk viene eliminata automaticamente se il nodo viene eliminato.

Quando si distribuisce un gruppo di sicurezza per un pod in un passaggio successivo, il controller di risorse VPC crea un'interfaccia di rete speciale denominata interfaccia di rete di filiali con una descrizione di `aws-k8s-branch-eni` e vi associa i gruppi di sicurezza. Le interfacce di rete di filiali vengono create in aggiunta alle interfacce di rete standard e trunk allegate al nodo.

Se si utilizzano sonde liveness o readiness, è inoltre necessario disabilitare il demultiplexing precoce TCP, in modo che `kubelet` possa connettersi a pod su interfacce di rete di filiali tramite TCP. Per disabilitare demultiplexing precoce TCP, esegui il comando seguente:

```
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 non utilizzi il plugin CNI di Amazon VPC `1.11.0` o successivo per il componente aggiuntivo di Kubernetes e impostarlo `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, come descritto nel passaggio successivo, non devi eseguire il comando precedente.

5. Se il cluster utilizza `NodeLocal DNSCache` o desideri utilizzare la policy di rete Colico con i pod che dispongono dei propri gruppi di sicurezza, o se disponi di servizi Kubernetes di tipo `NodePort` e `LoadBalancer` che utilizzano le destinazioni di istanza con `externalTrafficPolicy` impostato su `Local` per pod a cui desideri assegnare i gruppi di sicurezza, dovrai utilizzare la

versione `1.11.0` o successiva del plugin CNI di Amazon VPC per il componente aggiuntivo di Kubernetes e dovrai abilitare la seguente impostazione:

```
kubectl set env daemonset aws-node -n kube-system
 POD_SECURITY_GROUP_ENFORCING_MODE=standard
```

IMPORTANTE: le regole del gruppo di sicurezza dei pod non si applicano al traffico tra pod o tra pod e servizi, esempio **`kubelet`** o **`nodeLocalDNS`**, che si trovano sullo stesso nodo. I pod che utilizzano gruppi di sicurezza differenti sullo stesso nodo non possono comunicare perché sono configurati in sottoreti diverse e l'instradamento è disabilitato tra tali sottoreti. Traffico in uscita dai pod verso gli indirizzi esterni al VPC è l'indirizzo di rete convertito nell'indirizzo IP dell'interfaccia di rete primaria dell'istanza (a meno che non sia stato impostato `AWS_VPC_K8S_CNI_EXTERNALSNAT=true`). Per questo traffico sono utilizzate le regole nei gruppi di sicurezza per l'interfaccia di rete primaria, anziché le regole nei gruppi di sicurezza dei pod. ** Affinché questa impostazione si applichi ai pod esistenti, è necessario riavviare i pod o i nodi su cui i pod sono in esecuzione.

6. Per scoprire come utilizzare una politica di gruppo di sicurezza per il pod, consulta the section called "SecurityGroupPolicy".

Utilizzo di una policy di gruppo di sicurezza per un pod Amazon EKS

Per utilizzare i gruppi di sicurezza per i pod, è necessario disporre di un gruppo di sicurezza esistente. I passaggi seguenti mostrano l'utilizzo delle policy di gruppo di sicurezza per un pod. Se non diversamente indicato, completa tutti i passaggi nello stesso terminale, poiché nei passaggi successivi sono utilizzate variabili che non persistono tra i terminali.

Se disponi di un Pod con EC2 istanze Amazon, devi configurare il plug-in prima di utilizzare questa procedura. Per ulteriori informazioni, consulta the section called "Configura".

1. Crea un namespace Kubernetes in cui distribuire le risorse . Puoi sostituirlo *`my-namespace`* con il nome di uno spazio dei nomi che desideri utilizzare.

```
kubectl create namespace my-namespace
```

2. Implementa una `SecurityGroupPolicy` Amazon EKS nel cluster.
   a. Copia i seguenti contenuti sul dispositivo. Puoi sostituirlo *`podSelector`* con `serviceAccountSelector` se preferisci selezionare i Pod in base alle etichette degli account di servizio. É necessario specificare uno selettore. Un `podSelector` vuoto (ad

esempio: `podSelector: {}`) seleziona tutti i pod nel namespace. Puoi *`my-role`* cambiare il nome del tuo ruolo. Un `serviceAccountSelector` vuoto seleziona tutti gli account di servizio nel namespace. Puoi sostituirlo *`my-security-group-policy`* con un nome per te `SecurityGroupPolicy` e *`my-namespace`* con lo spazio dei nomi in cui desideri crearlo. `SecurityGroupPolicy`

È necessario sostituirlo *`my_pod_security_group_id`* con l'ID di un gruppo di sicurezza esistente. Se non disponi ancora di un gruppo di sicurezza dovrai crearne uno. Per ulteriori informazioni, consulta i gruppi EC2 di sicurezza Amazon per le istanze Linux nella Amazon EC2 User Guide. Puoi specificare da 1 a 5 gruppi di sicurezza. IDs Se si specificano più ID, la combinazione di tutte le regole in tutti i gruppi di sicurezza sarà valida per i pod selezionati.

```
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

Il gruppo di sicurezza o i gruppi specificati per i pod devono soddisfare i criteri seguenti:

- Devono essere esistenti. Se non lo sono, quando si distribuisce un pod che corrisponde al selettore, il pod rimane bloccato nel processo di creazione. Se si descrive il pod, si otterrà un messaggio di errore simile al seguente: `An error occurred (InvalidSecurityGroupID.NotFound) when calling the CreateNetworkInterface operation: The securityGroup ID 'sg-05b1d815d1EXAMPLE' does not exist.`
- Devono consentire la comunicazione in ingresso dal gruppo di sicurezza applicato ai nodi (per `kubelet`) su tutte le porte per cui sono state configurate le sonde.

- Devono consentire la comunicazione in uscita tramite le porte 53 TCP e UDP a un gruppo di sicurezza assegnato ai pod (o ai nodi su cui sono eseguiti i pod) che eseguono CoreDNS. Il gruppo di sicurezza per i pod CoreDNS deve consentire il traffico in ingresso tramite le porte 53 TCP e UDP dal gruppo di sicurezza specificato.
- Devono disporre delle regole in entrata e in uscita necessarie per comunicare con altri pod.
- Devono disporre di regole che consentano ai pod di comunicare con il piano di controllo Kubernetes se si utilizza il gruppo di sicurezza con Fargate. Il modo più semplice per eseguire questa operazione consiste nello specificare il gruppo di sicurezza del cluster come uno dei gruppi di sicurezza.

Le policy dei gruppi di sicurezza si applicano solo ai nuovi pod pianificati. Non incidono sui pod in esecuzione.

b. Implementare la policy.

```
kubectl apply -f my-security-group-policy.yaml
```

3. Implementa un'applicazione di esempio con un'etichetta che corrisponda al valore *my-role* per *podSelector* specificato nel passaggio precedente.

a. Copia i seguenti contenuti sul dispositivo. Sostituisci i valori di esempio con i tuoi, quindi esegui il comando modificato. Se lo sostituisci*my-role*, assicurati che sia uguale al valore specificato per il selettore in un passaggio precedente.

```
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. Implementare l'applicazione con il seguente comando. Quando si implementa l'applicazione, il plugin CNI di Amazon VPC per Kubernetes corrisponde all'etichetta `role` e i gruppi di sicurezza specificati nel passaggio precedente sono applicati al pod.

```
kubectl apply -f sample-application.yaml
```

4. Visualizza i pod implementati con l'applicazione esemplificativa. Nella parte restante di questo argomento, questo terminale è indicato come `TerminalA`.

```
kubectl get pods -n my-namespace -o wide
```

Di seguito viene riportato un output di esempio.

```
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

Prova questi suggerimenti se qualche pod è bloccato.

- Se qualche pod è bloccato nello stato `Waiting`, esegui `kubectl describe pod` *`my-deployment-xxxxxxxxxx-xxxxx`* `-n` *`my-namespace`* . Se visualizzi `Insufficient permissions: Unable to create Elastic Network Interface.`, verifica di avere aggiunto la policy IAM al ruolo cluster IAM in un passaggio precedente.
- Se qualche pod è bloccato nello stato `Pending`, verifica che il tipo di istanza del nodo sia elencato in limits.go e che non sia già stato raggiunto il prodotto del numero massimo di interfacce di rete di filiali supportate dal tipo di istanza moltiplicato per il numero di nodi nel gruppo di nodi. Ad esempio, una istanza `m5.large` supporta nove interfacce di rete di filiali. Se il gruppo di nodi dispone di cinque nodi, è possibile creare un massimo di 45 interfacce di rete di filiali per il gruppo di nodi. Il 46° pod che si tenta di implementare si troverà in uno stato di `Pending` finché non viene eliminato un altro pod a cui sono associati gruppi di sicurezza.

Se si esegue `kubectl describe pod` *`my-deployment-xxxxxxxxxx-xxxxx`* `-n` *`my-namespace`* e si visualizza un messaggio simile al seguente, può essere ignorato in modo sicuro. Questo messaggio potrebbe essere visualizzato quando il plugin CNI di Amazon VPC per Kubernetes tenta di impostare la rete host e fallisce durante la creazione dell'interfaccia di rete. Il plug-in registra questo evento fino a quando non viene creata l'interfaccia di rete.

```
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
```

Non è possibile superare il numero massimo di pod eseguibili sul tipo di istanza. Per un elenco del numero massimo di Pod che puoi eseguire su ogni tipo di istanza, consulta eni-max-pods.txt on. GitHub Quando si elimina un pod a cui sono associati gruppi di sicurezza o si elimina il nodo su cui è in esecuzione il pod, il controller di risorse VPC elimina l'interfaccia di rete di filiali. Se elimini un cluster con pod che utilizzano pod per gruppi di sicurezza, il controller non elimina le interfacce di rete delle diramazioni, per cui dovrai eliminarle personalmente. Per informazioni su come eliminare le interfacce di rete, consulta Delete a network interface nella Amazon EC2 User Guide.

5. In un terminale separato, accedi alla shell di uno dei pod. Nella parte restante di questo argomento, questo terminale è indicato come `TerminalB`. Sostituiscilo *`5df6f7687b-4fbjm`* con l'ID di uno dei Pod restituiti nell'output del passaggio precedente.

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

6. Confermare il funzionamento dell'applicazione di esempio dalla shell in `TerminalB`.

```
curl my-app
```

Di seguito viene riportato un output di esempio.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

L'output è stato ricevuto in quanto tutti i pod che eseguono l'applicazione sono associati al gruppo di sicurezza creato. Tale gruppo contiene una regola che abilita il traffico tra tutti i pod a cui è associato il gruppo di sicurezza. Il traffico DNS è consentito in uscita da tale gruppo di sicurezza al gruppo di sicurezza del cluster associato ai nodi. I nodi eseguono i pod CoreDNS, su cui i pod hanno eseguito la ricerca del nome.

7. Da `TerminalA`, rimuovere le regole del gruppo di sicurezza che consentono la comunicazione DNS dal gruppo di sicurezza al gruppo di sicurezza del cluster. Se non hai aggiunto le regole DNS al gruppo di sicurezza del cluster in un passaggio precedente, sostituiscile *`$my_cluster_security_group_id`* con l'ID del gruppo di sicurezza in cui hai creato le regole.

```
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. Tentare di accedere nuovamente all'applicazione da `TerminalB`.

   ```
   curl my-app
   ```

   Di seguito viene riportato un output di esempio.

   ```
   curl: (6) Could not resolve host: my-app
   ```

   Il tentativo ha esito negativo perché il pod non è più in grado di accedere ai pod CoreDNS a cui è associato il gruppo di sicurezza del cluster. Il gruppo di sicurezza del cluster non dispone più delle relative regole che consentono la comunicazione DNS dal gruppo di sicurezza associato al pod.

   Se si tenta di accedere all'applicazione utilizzando gli indirizzi IP restituiti in un passaggio precedente per uno dei pod, si riceve comunque una risposta perché tutte le porte sono consentite tra i pod a cui è associato il gruppo di sicurezza e non è necessaria una ricerca del nome.

9. Dopo avere eseguito diverse prove, è possibile rimuovere la policy del gruppo di sicurezza, l'applicazione e il gruppo di sicurezza di esempio creati. Esegui i comandi seguenti da `TerminalA`.

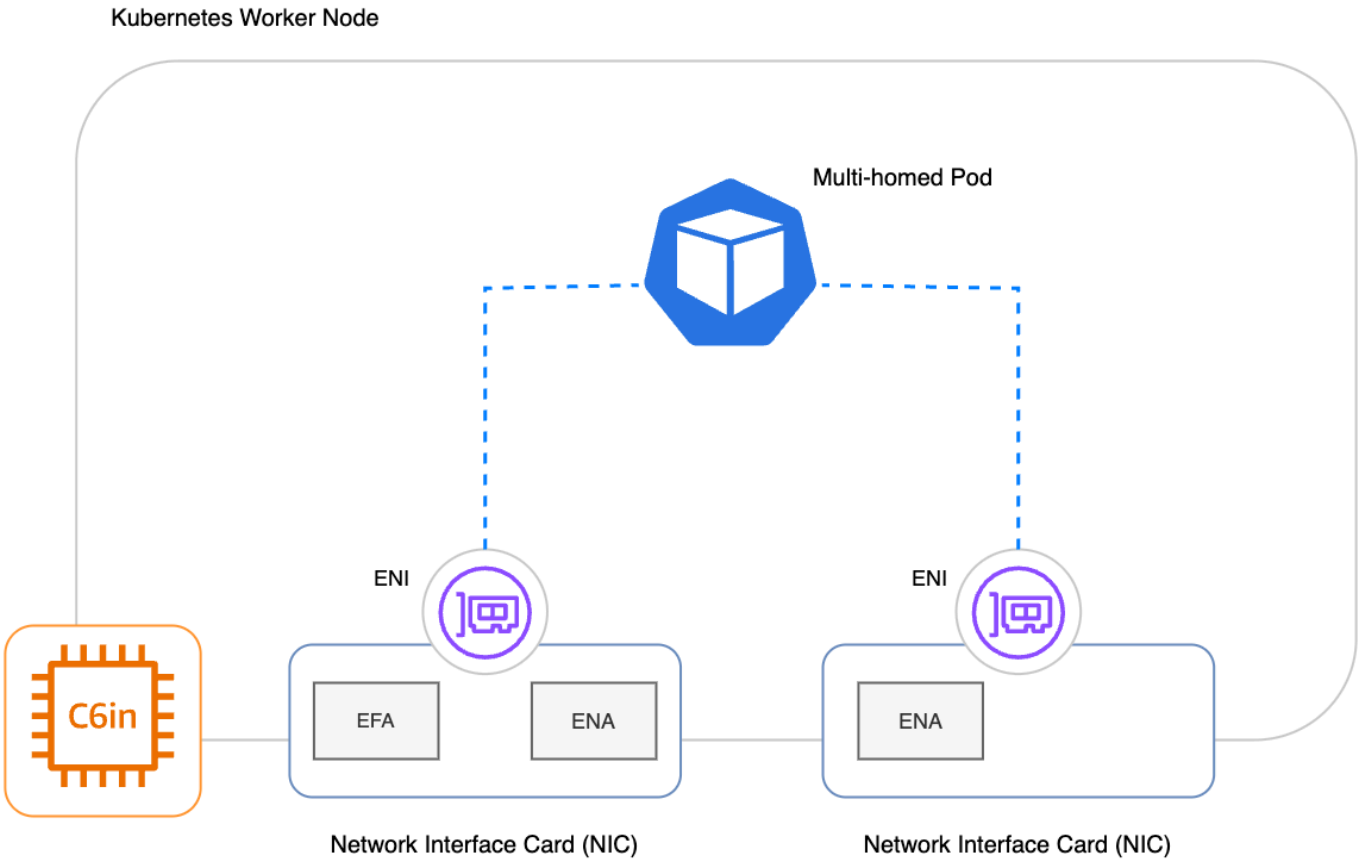
   ```
   kubectl delete namespace my-namespace
   aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
   security-group-rule-ids $my_inbound_self_rule_id
   wait
   sleep 45s
   aws ec2 delete-security-group --group-id $my_pod_security_group_id
   ```

Collegamento di interfacce di rete multiple ai pod

Per impostazione predefinita, il plugin CNI di Amazon VPC assegna un indirizzo IP a ciascun pod. Questo indirizzo IP è collegato a un'interfaccia di rete elastica che gestisce tutto il traffico in entrata e in uscita per il pod. Per aumentare le prestazioni della larghezza di banda e della velocità di pacchetti al secondo, è possibile utilizzare la funzionalità Multi-NIC di CNI di VPC per configurare un pod multi-homed. Un pod multi-homed è un singolo pod Kubernetes che utilizza più interfacce di rete (e più indirizzi IP). Eseguendo un pod multi-homed, è possibile distribuire il traffico delle applicazioni su più

interfacce di rete utilizzando connessioni simultanee. Ciò è particolarmente utile per i casi d'uso di intelligenza artificiale (IA), machine learning (ML) e calcolo ad alte prestazioni (HPC).

Il diagramma seguente mostra un pod multi-homed in esecuzione su un nodo di lavoro con più schede di interfaccia di rete () in uso. NICs



Contesto

Su Amazon EC2, un'interfaccia di rete elastica è un componente di rete logico in un VPC che rappresenta una scheda di rete virtuale. Per molti tipi di EC2 istanze, le interfacce di rete condividono una singola scheda di interfaccia di rete (NIC) nell'hardware. Questa singola NIC ha una larghezza di banda e una velocità massime di pacchetti al secondo.

Se la funzionalità Multi-NIC è abilitata, CNI di VPC non assegna gli indirizzi IP in blocco, come avviene per impostazione predefinita. Invece, CNI di VPC assegna un indirizzo IP a un'interfaccia di rete su ciascuna scheda di rete su richiesta all'avvio di un nuovo pod. Tale comportamento riduce la velocità di esaurimento degli indirizzi IP, che è aumentata usando pod multi-homed. Poiché la CNI

di VPC assegna l'indirizzo IP su richiesta, l'avvio dei pod potrebbe richiedere più tempo sulle istanze con la funzionalità multi-NIC abilitata.

Considerazioni

- Assicurati che sul tuo cluster Kubernetes sia in esecuzione la versione `1.20.0` CNI di VPC e successive. La funzionalità multi-NIC è disponibile solo nella versione `1.20.0` o successiva di CNI di VPC.
- Abilita la variabile di ambiente `ENABLE_MULTI_NIC` nel plugin CNI di VPC. È possibile eseguire il comando seguente per impostare la variabile e avviare una distribuzione di DaemonSet.
  - `kubectl set env daemonset aws-node -n kube-system ENABLE_MULTI_NIC=true`
- Assicuratevi di creare nodi di lavoro con più schede di interfaccia di rete (NICs). Per un elenco di EC2 istanze con più schede di interfaccia di rete, consulta Schede di rete nella Amazon EC2 User Guide.
- Se la funzionalità Multi-NIC è abilitata, CNI di VPC non assegna gli indirizzi IP in blocco, come avviene per impostazione predefinita. Poiché la CNI di VPC assegna l'indirizzo IP su richiesta, l'avvio dei pod potrebbe richiedere più tempo sulle istanze con la funzionalità multi-NIC abilitata. Per ulteriori informazioni, consulta la sezione the section called "Contesto" precedente.
- Con la funzionalità multi-NIC abilitata, per impostazione predefinita i pod non dispongono di più interfacce di rete. È necessario configurare ciascun carico di lavoro per utilizzare multi-NIC. Aggiungi l'annotazione `k8s.amazonaws.com/nicConfig: multi-nic-attachment` ai carichi di lavoro che dovrebbero avere più interfacce di rete.

Considerazioni su **`IPv6`**

- Policy IAM personalizzata: per i cluster `IPv6`, crea e utilizza la seguente policy IAM personalizzata per CNI di VPC. Questa policy è specifica per multi-NIC. Per informazioni più generali sull'utilizzo di CNI di VPC con i cluster `IPv6`, consulta 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/*"
        }
    ]
}
```

- IPv6 Meccanismo di transizione non disponibile: se si utilizza la funzionalità multi-NIC, CNI di VPC non assegna un indirizzo IPv4 ai pod di un cluster IPv6. Altrimenti, CNI di VPC assegna un indirizzo IPv4 host locale a ciascun pod in modo che un pod possa comunicare con risorse esterne IPv4 in un altro Amazon VPC o su Internet.

Utilizzo

Dopo aver abilitato la funzionalità multi-NIC nel CNI di VPC e riavviato i pod aws-node, è possibile configurare ciascun carico di lavoro in modo che sia multi-homed. Il seguente esempio di una configurazione YAML con l'annotazione richiesta:

```
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:
...
```

Domande frequenti

1. Cos'è una scheda di interfaccia di rete (NIC)?

Una scheda di interfaccia di rete (NIC), chiamata anche semplicemente scheda di rete, è un dispositivo fisico che consente la connettività di rete per l'hardware di cloud computing sottostante. Nei EC2 server moderni, questo si riferisce alla scheda di rete Nitro. Un'interfaccia di rete elastica (ENI) è una rappresentazione virtuale di questa scheda di rete sottostante.

Alcuni tipi di EC2 istanza ne hanno più di uno NICs per aumentare le prestazioni in termini di larghezza di banda e velocità dei pacchetti. In questi casi, è possibile assegnare schede di rete secondarie ENIs alle schede di rete aggiuntive. Ad esempio, ENI #1 può funzionare come interfaccia per la NIC collegata all'indice 0 della scheda di rete, mentre ENI #2 può funzionare come interfaccia per la NIC collegata a un indice di schede di rete separato.

2. Cos'è un pod multi-homed?

Un pod multi-homed è un singolo pod Kubernetes con più interfacce di rete (e per implicazione più indirizzi IP). Ogni interfaccia di rete pod è associata a un'interfaccia di rete elastica (ENI) e si ENIs tratta di rappresentazioni logiche di separate NICs sul nodo di lavoro sottostante. Con più interfacce di rete, un pod multi-homed ha una capacità di trasferimento dati aggiuntiva, che aumenta anche la velocità di trasferimento dei dati.

> **Important**
>
> Il VPC CNI può configurare pod multihomed solo su tipi di istanza che ne hanno più di uno. NICs

3. Qual è il vantaggio di utilizzare questa funzionalità?

Se è necessario scalare le prestazioni di rete nei carichi di lavoro basati su Kubernetes, è possibile utilizzare la funzionalità Multi-NIC per eseguire pod multihomed che si interfacciano con tutti i dispositivi sottostanti NICs a cui è collegato un dispositivo ENA. L'utilizzo di schede di rete aggiuntive aumenta la capacità di larghezza di banda e le prestazioni di velocità di pacchetti nelle applicazioni distribuendo il traffico delle applicazioni su più connessioni simultanee. Ciò è particolarmente utile per i casi d'uso di intelligenza artificiale (IA), machine learning (ML) e calcolo ad alte prestazioni (HPC).

4. Come posso utilizzare questa funzionalità?

1. Innanzitutto, è necessario assicurarsi che il proprio cluster Kubernetes utilizzi versione 1.20 o successiva di CNI di VPC. Per i passaggi per aggiornare CNI di VPC come componente aggiuntivo EKS, consulta the section called "Aggiornamento (componente aggiuntivo EKS)".
2. Quindi, è necessario abilitare il supporto multi-NIC nel CNI di VPC usando la variabile di ambiente `ENABLE_MULTI_NIC`.
3. Quindi, è necessario assicurarsi di creare e unire nodi con più schede di rete. Per un elenco di tipi di EC2 istanze con più schede di rete, consulta Schede di rete nella Amazon EC2 User Guide.
4. Infine, configura ciascun carico di lavoro per utilizzare più interfacce di rete (pod multi-homed) o utilizza un'unica interfaccia di rete.

5. Come posso configurare i miei carichi di lavoro per utilizzarne più di uno NICs su un nodo di lavoro supportato?

Per utilizzare i pod multi-homed, è necessario aggiungere la seguente annotazione: `k8s.amazonaws.com/nicConfig: multi-nic-attachment`. Ciò collegherà un ENI da ogni NIC dell'istanza sottostante al pod (mappatura uno a più tra un pod e il NICs).

Se manca questa annotazione, CNI di VPC presuppone che il pod richieda solo 1 interfaccia di rete e gli assegna un IP da un'ENI su qualsiasi NIC disponibile.

6. Quali adattatori di interfaccia di rete sono supportati da questa funzionalità?

È possibile utilizzare qualsiasi adattatore di interfaccia di rete se si dispone di almeno un ENA collegato alla scheda di rete sottostante per il traffico IP. Per ulteriori informazioni su ENA, consulta Elastic Network Adapter (ENA) nella Amazon EC2 User Guide.

Configurazioni dei dispositivi di rete supportate:

- Le interfacce ENA offrono tutte le tradizionali funzionalità di rete e routing IP necessarie per supportare la rete IP per un VPC. Per ulteriori informazioni, consulta Abilita reti avanzate con ENA sulle tue EC2 istanze.
- Le interfacce EFA (EFA con ENA) offrono sia il dispositivo ENA per le reti IP che il dispositivo EFA per le comunicazioni a bassa latenza e a throughput elevato.

> **Important**
>
> Se a una scheda di rete è collegato un adattatore solo EFA, il CNI di VPC lo ignorerà durante il provisioning della connettività di rete per un pod multi-homed. Tuttavia, se si combina un adattatore solo EFA con un adattatore ENA su una scheda di rete, il VPC CNI ENIs gestirà anche questo dispositivo. Per utilizzare interfacce solo EFA con i cluster EKS, consulta the section called "Configurazione dei cluster di formazione con EFA".

7. Posso vedere se un nodo del mio cluster supporta ENA?

Sì, puoi utilizzare la AWS CLI o l' EC2 API per recuperare informazioni di rete su un' EC2 istanza nel tuo cluster. Ciò fornisce dettagli sul fatto che l'istanza disponga o meno del supporto ENA. Nell'esempio seguente, sostituiscilo `<your-instance-id>` con l'ID di EC2 istanza di un nodo.

AWS Esempio di CLI:

```
aws ec2 describe-instances --instance-ids <your-instance-id> --query
 "Reservations[].Instances[].EnaSupport"
```

Output di esempio:

```
[ true ]
```

8. Posso vedere i diversi indirizzi IP associati a un pod?

No, non è semplice. Tuttavia, è possibile utilizzare `nsenter` dal nodo per eseguire strumenti di rete comuni come `ip route show` e visualizzare gli indirizzi IP e le interfacce aggiuntivi.

9. Posso controllare il numero di interfacce di rete per i miei pod?

No. Quando il carico di lavoro è configurato per utilizzarne più di uno NICs su un'istanza supportata, un singolo pod ha automaticamente un indirizzo IP da ogni scheda di rete dell'istanza. In alternativa, i pod single-homed avranno un'interfaccia di rete collegata a una NIC sull'istanza.

> **Important**
>
> Le schede di rete a cui è collegato solo un dispositivo solo EFA sono ignorate da CNI di VPC.

10. Posso configurare i miei pod per utilizzare una scheda NIC specifica?

No, questa funzionalità non è supportata. Se un pod ha l'annotazione pertinente, CNI di VPC lo configura automaticamente per utilizzare ogni NIC con un adattatore ENA sul nodo worker.

11. Questa funzionalità funziona con le altre funzionalità di rete di CNI di VPC?

Sì, la funzionalità multi-NIC di CNI di VPC funziona sia con reti personalizzate che con la rilevazione avanzata delle sottoreti. Tuttavia, i pod multi-homed non utilizzano sottoreti o gruppi di sicurezza personalizzati. Invece, CNI di VPC assegna indirizzi IP e interfacce di rete ai pod multi-homed con la stessa configurazione di sottorete e gruppo di sicurezza del nodo. Per ulteriori informazioni sulle reti personalizzate, consulta the section called "Rete personalizzata".

La funzionalità multi-NIC di CNI di VPC non funziona e non può essere combinata con i gruppi di sicurezza per i pod.

12. Posso utilizzare le politiche di rete con questa funzionalità?

Sì, è possibile utilizzare le policy di rete Kubernetes con multi-NIC. Le policy di rete Kubernetes limitano il traffico di rete verso e dai pod. Per ulteriori informazioni sull'applicazione delle policy di rete con CNI di VPC, consulta the section called "Policy di Kubernetes".

13. Il supporto multi-NIC è abilitato in modalità automatica EKS?

La multi-NIC non è supportata per i cluster in modalità automatica EKS.

Plug-in CNI alternativi per cluster Amazon EKS

Il plug-in CNI di Amazon VPC per Kubernetes è l'unico plug-in supportato da Amazon EKS con nodi Amazon EC2. Amazon EKS supporta le funzionalità principali di Cilium e Calico per Amazon EKS Hybrid Nodes. Amazon EKS esegue Kubernetes a monte, permettendo di installare plug-in

CNI compatibili alternativi sui nodi Amazon EC2 del cluster. Se hai nodi Fargate nel cluster, il plug-in CNI di Amazon VPC per Kubernetes è già presente sui nodi Fargate. È l'unico plug-in CNI che puoi utilizzare con i nodi Fargate. Qualsiasi tentativo di installare un plug-in CNI alternativo sui nodi Fargate avrà esito negativo.

Se prevedi di utilizzare un plug-in CNI alternativo sui nodi Amazon EC2, ti consigliamo vivamente di ottenere supporto commerciale o di disporre delle competenze interne per risolvere i problemi e contribuire alle correzioni del progetto del plug-in CNI.

Amazon EKS dispone di una rete di partner che offrono supporto per plug-in CNI compatibili alternativi. Per informazioni dettagliate sulle versioni, sulle qualifiche e sui test eseguiti, consulta la documentazione dei partner riportata di seguito.

Partner	Product	Documentazione
Tigera	Calico	Istruzioni di installazione
Isovalent	Cilium	Istruzioni di installazione
Juniper	Rete Contrail cloud native (CN2)	Istruzioni di installazione
VMware	Antrea	Istruzioni di installazione

Amazon EKS mira a dare all'utente una vasta gamma di opzioni per coprire tutti i casi d'uso.

Plug-in alternativi compatibili per le policy di rete

Calico è una soluzione ampiamente adottata per il networking e la sicurezza dei container. L'utilizzo di Calico su EKS offre un'applicazione delle policy di rete completamente conforme per i cluster EKS. Inoltre, puoi scegliere di utilizzare la rete di Calico, che conserva gli indirizzi IP del VPC sottostante. Calico Cloud migliora le funzionalità di Calico Open Source, fornendo funzionalità avanzate di sicurezza e osservabilità.

Il flusso di traffico da e verso i pod con gruppi di sicurezza associati non è soggetto all'applicazione di policy di rete ed è limitato esclusivamente all'applicazione dei gruppi di sicurezza Amazon VPC.

Se utilizzi l'applicazione della policy di rete di Calico, consigliamo di impostare la variabile di ambiente `ANNOTATE_POD_IP` su `true` per evitare un problema noto con Kubernetes. Per utilizzare questa

funzionalità, devi aggiungere l'autorizzazione `patch` per i pod a `aws-node` ClusterRole. L'aggiunta di autorizzazioni di patch a `aws-node` DaemonSet aumenta l'ambito di sicurezza del plug-in. Per ulteriori informazioni, consulta ANNOTATE_POD_IP nel repository CNI VPC su GitHub.

Considerazioni per Amazon EKS Auto Mode

Amazon EKS Auto Mode non supporta plug-in CNI o plug-in di policy di rete alternativi. Per ulteriori informazioni, consulta Modalità automatica EKS.

Collegare più interfacce di rete ai pod con Multus

Multus CNI è un plug-in CNI (Container Network Interface) per Amazon EKS che consente di allegare più interfacce di rete a un pod. Per ulteriori informazioni, consulta la documentazione Multus-CNI su GitHub.

In Amazon EKS, ogni pod ha un'interfaccia di rete assegnata dal plug-in CNI di Amazon VPC. Con Multus, è possibile creare un pod multi-homed con più interfacce. Ciò avviene grazie poiché Multus agisce come un "meta-plugin"; un plugin CNI che può chiamare altri plug-in CNI multipli. Il supporto AWS per Multus viene configurato con il plugin CNI di Amazon VPC come plugin delegato predefinito.

- Amazon EKS non costruirà e pubblicherà i plug-in CNI della virtualizzazione I/O singola root (SR-IOV) e del Data Plane Development Kit (DPDK). Tuttavia, è possibile ottenere l'accelerazione dei pacchetti collegandosi direttamente ad Amazon EC2 Elastic Network Adapters (ENA) tramite il dispositivo host gestito Multus e plug-in `ipvlan`.
- Amazon EKS supporta Multus, che fornisce un processo generico che consente un semplice concatenamento di plug-in CNI aggiuntivi. Multus e il processo di concatenamento è supportato, ma AWS non fornisce supporto per tutti i plug-in CNI compatibili che possono essere concatenati o per problemi che possono sorgere in quei plug-in CNI non correlati alla configurazione del concatenamento.
- Amazon EKS fornisce supporto e gestione del ciclo di vita per il plug-in Multus, ma non è responsabile di alcun indirizzo IP o gestione aggiuntiva associata alle interfacce di rete aggiuntive. L'indirizzo IP e la gestione dell'interfaccia di rete predefinita che utilizza il plug-in CNI di Amazon VPC rimangono invariati.
- Solo il plug-in CNI di Amazon VPC è ufficialmente supportato come plug-in delegato predefinito. Se si sceglie di non utilizzare il plug-in CNI di Amazon VPC per la rete primaria, è necessario modificare il manifesto di installazione Multus pubblicato per riconfigurare il plug-in delegato predefinito in un CNI alternativo.

- Multus è supportato solo quando si utilizza il CNI di Amazon VPC come CNI principale. Non supportiamo il CNI di Amazon VPC se utilizzato per interfacce di ordine superiore, secondarie o di altro tipo.
- Per evitare che il plug-in CNI di Amazon VPC tenti di gestire interfacce di rete aggiuntive assegnate ai pod, aggiungi il seguente tag alle interfacce di rete:

  key

  : `node.k8s.amazonaws.com/no_manage`

  value

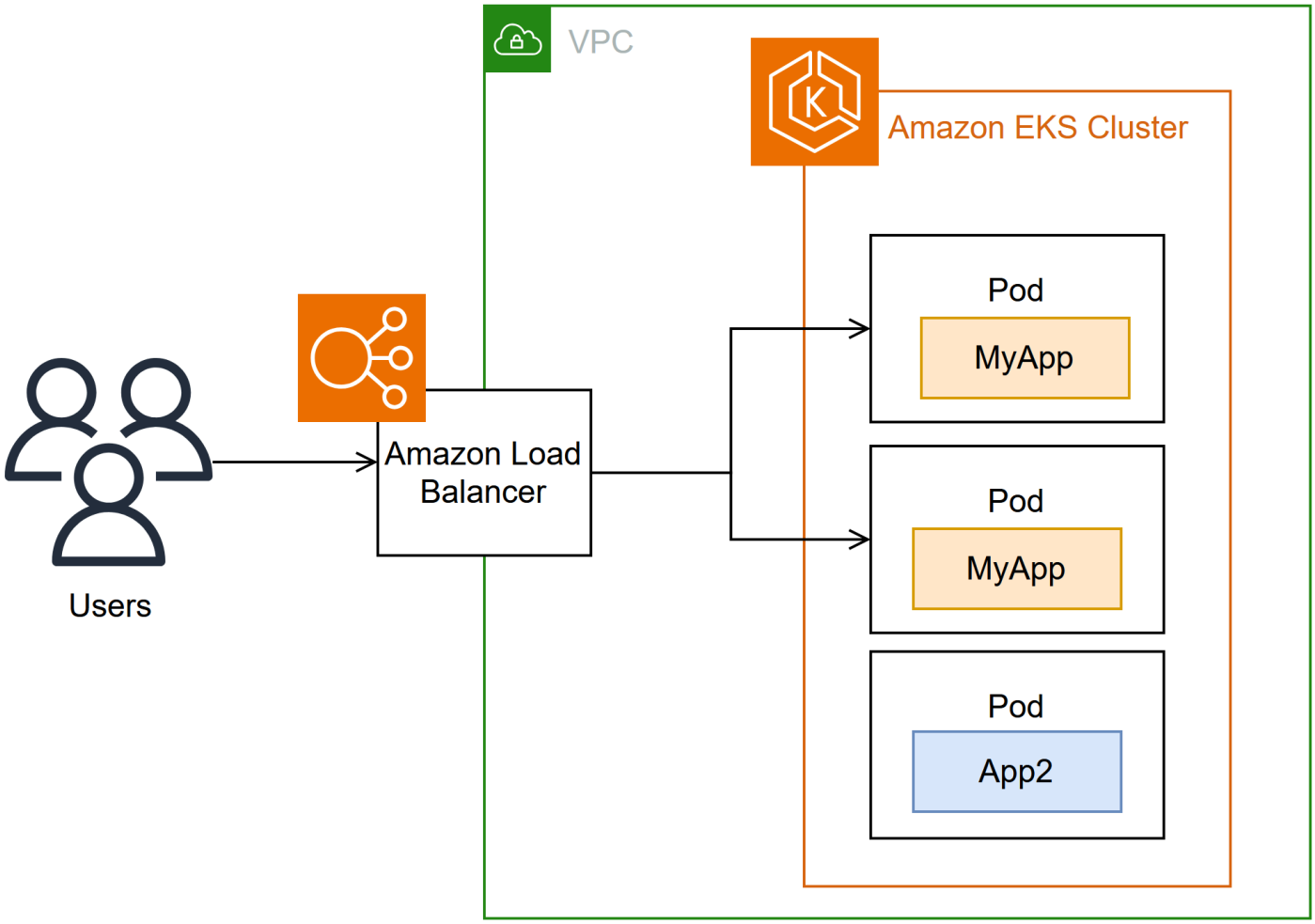
  : `true`
- Multus è compatibile con policy di rete, ma queste devono essere arricchite per includere porte e indirizzi IP che possono far parte di interfacce di rete aggiuntive allegate ai pod.

Per una procedura di implementazione, consulta Multus Setup Guide su GitHub.

Indirizza il traffico Internet con AWS Load Balancer Controller

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

Il AWS Load Balancer Controller gestisce AWS Elastic Load Balancer per un cluster Kubernetes. Puoi usare il controller per esporre le app del cluster a Internet. Il controller fornisce sistemi di AWS bilanciamento del carico che puntano alle risorse Cluster Service o Ingress. In altri termini, il controller crea un unico indirizzo IP o nome DNS che punta a più pod del cluster.



Il controller monitora le risorse di Kubernetes Ingress o Service. In risposta, crea le risorse AWS Elastic Load Balancing appropriate. Puoi configurare il comportamento specifico dei bilanciatori del carico applicando annotazioni alle risorse Kubernetes. Ad esempio, è possibile collegare gruppi di AWS sicurezza ai sistemi di bilanciamento del carico utilizzando le annotazioni.

Il controller fornisce le risorse seguenti:

Kubernetes **`Ingress`**

L'LBC crea un AWS Application Load Balancer (ALB) quando crei un `Ingress`. Rivedi le annotazioni che puoi applicare a una risorsa Ingress.

Servizio Kubernetes di tipo **`LoadBalancer`**

L'LBC crea un AWS Network Load Balancer (NLB) quando crei un servizio Kubernetes di tipo `LoadBalancer`. Rivedi le annotazioni che puoi applicare a una risorsa Service.

In passato, il Network Load Balancer (NLB) Kubernetes veniva utilizzato per le destinazioni istanza, ma per le destinazioni IP veniva utilizzato l'LBC. Con la versione AWS Load Balancer Controller `2.3.0` o successiva, puoi creare NLBs utilizzando entrambi i tipi di destinazione. Per ulteriori informazioni sui tipi di destinazioni dei NLB, consultare Target type (Tipo di destinazione) nella Guida per l'utente di Network Load Balancer.

Il controller è un progetto open source gestito su. GitHub

Prima di implementare il controller, è consigliabile esaminare i prerequisiti e le considerazioni in Route application and HTTP traffic with Application Load Balancers e the section called "Bilanciamento del carico di rete". In questi argomenti, verrà distribuita un'app di esempio che include un sistema di AWS bilanciamento del carico.

API Kubernetes **`Gateway`**

Con la versione AWS Load Balancer Controller `2.14.0` o successiva, LBC crea un Application Load AWS Balancer (ALB) quando crei un Kubernetes. `Gateway` Kubernetes Gateway standardizza più configurazioni rispetto a Ingress, che richiedeva annotazioni personalizzate per molte opzioni comuni. Rivedi la configurazione che puoi applicare a una risorsa Gateway. Per ulteriori informazioni sull'`Gateway`API, consulta Gateway API nella documentazione di Kubernetes.

Installare il controller

È possibile utilizzare una delle seguenti procedure per installare il AWS Load Balancer Controller:

- Se non conosci Amazon EKS, ti consigliamo di utilizzare Helm per l'installazione perché semplifica l'installazione del Load AWS Balancer Controller. Per ulteriori informazioni, consulta the section called "Installa con Helm".
- Per configurazioni avanzate, come cluster con accesso di rete limitato ai registri di container pubblici, utilizza i manifesti Kubernetes. Per ulteriori informazioni, consulta the section called "Installa con i manifesti".

Migrazione dalle versioni obsolete del controller

- Se sono installate versioni obsolete del Load AWS Balancer Controller, consulta. the section called "Esegui la migrazione da obsoleto"

- Le versioni obsolete non possono essere aggiornate. Devono essere rimossi e deve essere installata una versione corrente del AWS Load Balancer Controller.
- Le versioni obsolete includono:
  - AWS ALB Ingress Controller for Kubernetes («Ingress Controller»), un predecessore del Load Balancer Controller. AWS
  - Qualsiasi `0.1.x` versione del AWS Load Balancer Controller

Provider cloud legacy

Kubernetes include un provider cloud legacy per AWS. Il provider cloud legacy è in grado di effettuare il provisioning di sistemi di bilanciamento del AWS carico, in modo simile al Load Balancer Controller AWS . Il provider cloud legacy crea i Classic Load Balancer. Se non installi il AWS Load Balancer Controller, Kubernetes utilizzerà per impostazione predefinita il provider cloud legacy. È necessario installare il AWS Load Balancer Controller ed evitare di utilizzare il provider cloud legacy.

Important

Nelle versioni 2.5 e successive, il AWS Load Balancer Controller diventa il controller predefinito per le risorse del servizio Kubernetes e crea un Network Load AWS Balancer (NLB) per ogni servizio. `type: LoadBalancer` Lo fa creando un webhook mutante per i servizi, che imposta il campo `spec.loadBalancerClass` a `service.k8s.aws/nlb` per nuovi servizi di`type: LoadBalancer`. È possibile disattivare questa funzione e tornare a utilizzare il Cloud Provider obsoleto come controller predefinito, impostando il valore della tabella di comando `enableServiceMutatorWebhook` a `false`. Il cluster non eseguirà il provisioning dei nuovi Classic Load Balancer per i tuoi servizi a meno che non disattivi questa funzionalità. I sistemi Classic Load Balancer esistenti continueranno a funzionare.

Installa il AWS Load Balancer Controller con Helm

Tip

Registrati ai prossimi workshop Amazon EKS.

> **Tip**
>
> Con la modalità automatica di Amazon EKS, non devi installare o aggiornare componenti aggiuntivi di rete. Auto Mode include funzionalità di pod networking e bilanciamento del carico.
> Per ulteriori informazioni, consulta Modalità automatica EKS.

Questo argomento descrive come installare il AWS Load Balancer Controller utilizzando Helm, un gestore di pacchetti per Kubernetes, e. `eksctl` Il controller viene installato con le opzioni predefinite. Per ulteriori informazioni sul controller, inclusi i dettagli sulla configurazione con annotazioni, vedere la documentazione del controller Load AWS Balancer su. GitHub

Nei passaggi seguenti, sostituisci i valori di esempio con i tuoi valori.

Prerequisiti

Prima di iniziare questo tutorial, completa i seguenti passaggi:

- Crea un cluster Amazon EKS. Per crearne uno, consulta Nozioni di base.
- Installa Helm sul computer locale.
- Assicurati che il plug-in CNI di Amazon VPC per Kubernetes, `kube-proxy` e i componenti aggiuntivi CoreDNS siano alle versioni minime elencate in Service account tokens.
- Scopri i concetti di AWS Elastic Load Balancing. Per ulteriori informazioni, consulta la Guida per l'utente di Elastic Load Balancing.
- Scopri il servizio Kubernetes e le risorse in entrata.

Considerazioni

Prima di procedere con i passaggi per la configurazione descritti in questa pagina, considera quanto segue:

- La policy e il ruolo IAM (`AmazonEKSLoadBalancerControllerRole`) possono essere riutilizzati su più cluster EKS nello stesso account. AWS
- Se stai installando il controller sullo stesso cluster in cui il ruolo (`AmazonEKSLoadBalancerControllerRole`) è stato originariamente creato, vai a Step 2: Install Load Balancer Controller dopo aver verificato l'esistenza del ruolo.

- Se utilizzato, IRSA (IAM Roles for Service Accounts) deve essere configurato per ogni cluster e l'ARN del provider OpenID Connect (OIDC) nella policy di attendibilità del ruolo è specifico per ogni cluster EKS. Inoltre, se stai installando il controller su un nuovo cluster con un`AmazonEKSLoadBalancerControllerRole` esistente, aggiorna la policy di attendibilità del ruolo per includere il provider OIDC del nuovo cluster e crea un nuovo account di servizio con l'annotazione di ruolo appropriata. Per determinare se disponi già di un provider OIDC o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

Passaggio 1: crea un ruolo IAM utilizzando **`eksctl`**

I passaggi seguenti si riferiscono alla versione di rilascio del AWS Load Balancer Controller v2.14.1. Per ulteriori informazioni su tutte le versioni, consulta la pagina di rilascio del AWS Load Balancer Controller su. GitHub

1. Scarica una policy IAM per il AWS Load Balancer Controller che gli consenta di effettuare chiamate per tuo AWS APIs conto.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.14.1/docs/install/iam_policy.json
```

   - Se utilizzi una AWS partizione non standard, ad esempio una regione governativa o cinese, consulta le politiche GitHub e scarica la politica appropriata per la tua regione.
2. Creare una policy IAM utilizzando le policy scaricate nel passaggio precedente.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

   > **Note**
   >
   > Se visualizzi la politica in Console di gestione AWS, la console mostra gli avvisi per il servizio ELB, ma non per il servizio ELB v2. Ciò accade perché alcune azioni nella policy esistono per ELB v2, ma non per ELB. Queste avvertenze per ELB possono essere ignorate.

3. Sostituisci i valori per il nome del cluster, il codice regionale e l'ID dell'account.

```
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
```

Fase 2: Installare AWS Load Balancer Controller

1. Aggiungi l'archivio `eks-charts` cartografico Helm. AWS mantiene questo repository attivo. GitHub

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Aggiornare il repository locale per assicurarsi di avere i grafici più recenti.

```
helm repo update eks
```

3. Installa il AWS Load Balancer Controller.

   Se stai distribuendo il controller su EC2 nodi Amazon con accesso limitato al servizio di metadati di Amazon EC2 Instance (IMDS) o se esegui la distribuzione su Fargate o Amazon EKS Hybrid Nodes, aggiungi i seguenti flag `helm` al comando che segue:

   - `--set region=`*`region-code`*
   - `--set vpcId=`*`vpc-xxxxxxxx`*

   Sostituisci *`my-cluster`* con il nome del cluster. Nel comando seguente, `aws-load-balancer-controller` indica l'account del servizio Kubernetes creato in una fase precedente.

   Per ulteriori informazioni sulla configurazione del grafico helm, consulta values.yaml su. 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**
>
> Il grafico implementato non riceve automaticamente gli aggiornamenti di sicurezza. È necessario eseguire manualmente l'aggiornamento a un grafico più recente quando diventa disponibile. Durante l'aggiornamento, passate al comando precedente. *`install`* `upgrade`

Il `helm install` comando installa automaticamente le definizioni di risorse personalizzate (CRDs) per il controller. Il comando `helm upgrade` non lo fa. Se si utilizza, `helm upgrade,` è necessario installare manualmente il CRDs. Esegui il comando seguente per installare CRDs:

```
wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-controller/crds/crds.yaml
kubectl apply -f crds.yaml
```

Passaggio 3: verifica che il controller sia installato

1. Verificare che il controller sia installato.

   ```
   kubectl get deployment -n kube-system aws-load-balancer-controller
   ```

   Di seguito viene riportato un output di esempio:

   ```
   NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
   aws-load-balancer-controller   2/2     2            2           84s
   ```

   Riceverai l'output precedente se hai eseguito l'implementazione con Helm. Se hai eseguito l'implementazione utilizzando il manifesto Kubernetes, hai una sola replica.

2. Prima di utilizzare il controller per il provisioning AWS delle risorse, il cluster deve soddisfare requisiti specifici. Per ulteriori informazioni, consultare the section called "Bilanciamento del carico di applicazione" e the section called "Bilanciamento del carico di rete".

Installa AWS Load Balancer Controller con manifesti

> **Tip**
>
> Con la modalità automatica di Amazon EKS, non è necessario installare o aggiornare componenti aggiuntivi di rete. Auto Mode include funzionalità di pod networking e bilanciamento del carico.
> Per ulteriori informazioni, consulta Modalità automatica EKS.

Questo argomento descrive come installare il controller scaricando e applicando i manifesti di Kubernetes. É possibile visualizzare l'intera documentazione per il controller su GitHub.

Nei passaggi seguenti, sostituisci i valori di esempio con i tuoi valori.

Prerequisiti

Prima di iniziare questo tutorial, completa i seguenti passaggi:

- Crea un cluster Amazon EKS. Per crearne uno, consulta Nozioni di base.
- Installa Helm sul computer locale.
- Assicurati che il plug-in CNI di Amazon VPC per Kubernetes, `kube-proxy` e i componenti aggiuntivi CoreDNS siano alle versioni minime elencate in Service account tokens.
- Scopri i concetti di AWS Elastic Load Balancing. Per ulteriori informazioni, consulta la Guida per l'utente di Elastic Load Balancing.
- Scopri il servizio Kubernetes e le risorse in entrata.

Considerazioni

Prima di procedere con i passaggi per la configurazione descritti in questa pagina, considera quanto segue:

- La policy e il ruolo IAM (`AmazonEKSLoadBalancerControllerRole`) possono essere riutilizzati su più cluster EKS nello stesso account. AWS
- Se stai installando il controller sullo stesso cluster in cui il ruolo (`AmazonEKSLoadBalancerControllerRole`) è stato originariamente creato, vai a Step 2: Install cert-manager dopo aver verificato l'esistenza del ruolo.

- Se utilizzato, IRSA (IAM Roles for Service Accounts) deve essere configurato per ogni cluster e l'ARN del provider OpenID Connect (OIDC) nella policy di attendibilità del ruolo è specifico per ogni cluster EKS. Inoltre, se stai installando il controller su un nuovo cluster con un`AmazonEKSLoadBalancerControllerRole` esistente, aggiorna la policy di attendibilità del ruolo per includere il provider OIDC del nuovo cluster e crea un nuovo account di servizio con l'annotazione di ruolo appropriata. Per determinare se disponi già di un provider OIDC o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

Passaggio 1: configura IAM

I passaggi seguenti si riferiscono alla versione di rilascio del AWS Load Balancer Controller v2.14.1. Per ulteriori informazioni su tutte le versioni, consulta la pagina di rilascio del AWS Load Balancer Controller su. GitHub

1. Scarica una policy IAM per il AWS Load Balancer Controller che gli consenta di effettuare chiamate per tuo AWS APIs conto.

   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. Creare una policy IAM utilizzando le policy scaricate nel passaggio precedente.

   ```
   aws iam create-policy \
       --policy-name AWSLoadBalancerControllerIAMPolicy \
       --policy-document file://iam_policy.json
   ```

> **Note**
>
> Se visualizzi la policy in Console di gestione AWS, la console mostra gli avvisi per il servizio ELB, ma non per il servizio ELB v2. Ciò accade perché alcune azioni nella policy esistono per ELB v2, ma non per ELB. Queste avvertenze per ELB possono essere ignorate.

Example

eksctl

a. Sostituisci *my-cluster* con il nome del cluster e *111122223333* con il tuo ID account, quindi esegui il 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. Recupera l'ID del provider OIDC del cluster e memorizzalo in una variabile.

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

b. Determina se un provider OIDC IAM con l'ID del cluster è già presente nell'account. È necessario configurare OIDC sia per il cluster che per IAM.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Se viene restituito un output, devi creare un provider OIDC IAM per il tuo cluster. Se non viene restituito alcun output, devi creare un provider OIDC IAM per il cluster. Per ulteriori informazioni, consulta the section called "Gestore di identità OIDC IAM".

c. Copia i seguenti contenuti sul dispositivo. Sostituisci *111122223333* con l'ID del tuo account. *region-code*Sostituiscilo con la AWS regione in cui si trova il cluster. Sostituisci *EXAMPLED539D4633E53DE1B71EXAMPLE* con l'output restituito nella fase precedente.

```
{
    "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. Creare il ruolo IAM.

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

e. Allegare la policy IAM gestita da Amazon EKS richiesta al ruolo IAM. Sostituisci *111122223333* con l'ID del tuo account.

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

```
  --policy-arn arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --role-name AmazonEKSLoadBalancerControllerRole
```

f. Copia i seguenti contenuti sul dispositivo. Sostituisci *111122223333* con l'ID del tuo account. Dopo aver sostituito il testo, esegui il comando modificato per creare il file `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. Creare l'account del servizio Kubernetes nel cluster. L'account del servizio Kubernetes denominato `aws-load-balancer-controller` è annotato con il ruolo IAM che hai creato denominato. *AmazonEKSLoadBalancerControllerRole*

```
kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

Passaggio 2: installa **`cert-manager`**

Installa `cert-manager` utilizzando uno dei metodi seguenti per inserire la configurazione del certificato nei Webhook. Per ulteriori informazioni, consulta Getting Started nella documentazione cert-manager.

Si consiglia di utilizzare il registro dei container `quay.io` per installare `cert-manager`. Se i tuoi nodi non hanno accesso al registro dei container `quay.io`, installa `cert-manager` utilizzando Amazon ECR (vedi sotto).

Example

Quay.io

a. Se i tuoi nodi hanno accesso al registro del container `quay.io`, installa `cert-manager` per inserire la configurazione del certificato nei Webhook.

```
kubectl apply \
    --validate=false \
    -f https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-manager.yaml
```

Amazon ECR

a. Installa `cert-manager` utilizzando uno dei metodi seguenti per inserire la configurazione del certificato nei Webhook. Per ulteriori informazioni, consulta Getting Started nella documentazione cert-manager.

b. Eseguire il download del manifesto.

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-manager.yaml
```

c. Estrai le seguenti immagini e inviale a un repository a cui i tuoi nodi hanno accesso. Per ulteriori informazioni su come estrarre, taggare e inviare immagini al tuo repository, consulta la sezione the section called "Copia di un'immagine in un repository".

```
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. Nel manifesto, sostituisci `quay.io` per le tre immagini con il nome del tuo registro. Il comando seguente presuppone che il nome del repository privato sia lo stesso del repository di origine. Sostituisci *111122223333.dkr.ecr.region-code.amazonaws.com* con il tuo registro privato.

```
sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./cert-manager.yaml
```

e. Applicare il file manifesto.

```
kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

Fase 3: Installare AWS Load Balancer Controller

1. Scaricare le specifiche del controller. Per ulteriori informazioni sull'utilizzo del controller, consulta la documentazione su 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. Apporta le seguenti modifiche al file.

   a. Se hai scaricato il file `v2_14_1_full.yaml`, emetti il seguente comando per rimuovere la sezione `ServiceAccount` nel manifesto. Se non rimuovi questa sezione, l'annotazione richiesta apportata all'account del servizio in un passaggio precedente viene sovrascritta. Inoltre, se elimini il controller, conserva l'account del servizio creato in una fase precedente.

   ```
   sed -i.bak -e '764,772d' ./v2_14_1_full.yaml
   ```

   Se hai scaricato una versione diversa del file, apri il file in un editor e rimuovi le righe seguenti.

   ```
   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. Sostituisci `your-cluster-name` nella sezione `Deployment spec` del file con il nome del cluster sostituendo *`my-cluster`* con il nome del tuo cluster.

   ```
   sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_14_1_full.yaml
   ```

c. Se i tuoi nodi non hanno accesso ai repository di immagini Amazon EKS Amazon ECR, devi estrarre la seguente immagine e inviarla a un repository a cui i tuoi nodi hanno accesso. Per ulteriori informazioni su come estrarre, taggare e inviare un'immagine al tuo repository, consulta la sezione the section called "Copia di un'immagine in un repository".

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.14.1
```

Aggiungi il nome del tuo registro al manifesto. Il comando seguente presuppone che il nome del repository privato sia lo stesso del repository di origine e aggiunge il nome del tuo registro privato al file. Sostituisci *111122223333.dkr.ecr.region-code.amazonaws.com* con il tuo registro. Questa riga presuppone che il nome del repository privato sia lo stesso del repository di origine. In caso contrario, sostituisci il testo `eks/aws-load-balancer-controller` dopo il nome del registro privato con il nome del tuo repository.

```
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. (Richiesto solo per Fargate o IMDS con restrizioni)

Se stai distribuendo il controller su EC2 nodi Amazon con accesso limitato al servizio di metadati di Amazon EC2 Instance (IMDS) o se lo stai distribuendo su Fargate o su nodi ibridi Amazon EKS, aggiungi il seguente. `following parameters - args:`

```
[...]
spec:
      containers:
        - args:
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code


[...]
```

3. Applicare il file.

```
kubectl apply -f v2_14_1_full.yaml
```

4. Scarica il manifesto di `IngressClass` e `IngressClassParams` sul 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. Applica il file manifesto al cluster.

```
kubectl apply -f v2_14_1_ingclass.yaml
```

Passaggio 4: verifica che il controller sia installato

1. Verificare che il controller sia installato.

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

   Di seguito viene riportato un output di esempio:

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

   Riceverai l'output precedente se hai eseguito l'implementazione con Helm. Se hai eseguito l'implementazione utilizzando il manifesto Kubernetes, hai una sola replica.
2. Prima di utilizzare il controller per il provisioning AWS delle risorse, il cluster deve soddisfare requisiti specifici. Per ulteriori informazioni, consultare the section called "Bilanciamento del carico di applicazione" e the section called "Bilanciamento del carico di rete".

Esegui la migrazione delle app da Controller di ingresso ALB obsoleto

Questo argomento descrive come migrare da versioni obsolete del Controller. Più specificamente, descrive come rimuovere le versioni obsolete del Load Balancer AWS Controller.

- Le versioni obsolete non possono essere aggiornate. Devi prima rimuoverle e quindi installare una versione attuale.
- Le versioni obsolete includono:
  - AWS ALB Ingress Controller for Kubernetes («Ingress Controller»), un predecessore del Load Balancer Controller. AWS
  - Qualsiasi `0.1.x` versione del AWS Load Balancer Controller

Rimuovi la versione obsoleta del controller

> **Note**
>
> Potresti aver installato la versione obsoleta utilizzando Helm o manualmente con i manifesti di Kubernetes. Completare la procedura utilizzando lo strumento con cui è stata installata in origine.

1. Se il grafico Helm `incubator/aws-alb-ingress-controller` è installato, procedi alla disinstallazione.

   ```
   helm delete aws-alb-ingress-controller -n kube-system
   ```

2. Se la versione `0.1.x` del grafico `eks-charts/aws-load-balancer-controller` è installata, procedi alla disinstallazione. L'aggiornamento dalla versione `0.1.x` alla versione `1.0.0` non ha esito positivo a causa di incompatibilità con la versione dell'API webhook.

   ```
   helm delete aws-load-balancer-controller -n kube-system
   ```

3. Verificare che il controller sia già installato.

   ```
   kubectl get deployment -n kube-system alb-ingress-controller
   ```

   Questo è l'output restituito se il controller non è installato.

   ```
   Error from server (NotFound): deployments.apps "alb-ingress-controller" not found
   ```

   Questo è l'output restituito se il controller è installato.

   ```
   NAME                   READY UP-TO-DATE AVAILABLE AGE
   alb-ingress-controller 1/1   1          1         122d
   ```

4. Immettere i seguenti comandi per rimuovere il controller.

   ```
   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
   ```

Migrazione a AWS Load Balancer Controller

Per migrare dall'ALB Ingress Controller for Kubernetes al Load AWS Balancer Controller, devi:

1. Rimuovere il Controller di ingresso ALB (vedi sopra).
2. Installa il AWS Load Balancer Controller.
3. Aggiungi una policy aggiuntiva al ruolo IAM utilizzato dal AWS Load Balancer Controller. Questa policy consente a LBC la gestione delle risorse create dal Controller di ingresso ALB per Kubernetes.
4. Scaricare la policy IAM. Questa policy consente al AWS Load Balancer Controller di gestire le risorse create dall'ALB Ingress Controller per Kubernetes. É anche possibile consultare la policy.

```
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 il cluster si trova nelle regioni AWS GovCloud (Stati Uniti orientali) o AWS GovCloud (Stati Uniti occidentali), sostituiscilo con. AWS `arn:aws:` `arn:aws-us-gov:`

```
sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

6. Creare la policy IAM e annotate l'ARN restituito.

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

7. Collega la policy IAM al ruolo IAM utilizzato dal AWS Load Balancer Controller. Sostituiscila *`your-role-name`* con il nome del ruolo, ad esempio`AmazonEKSLoadBalancerControllerRole`.

   Se hai creato il ruolo utilizzando`eksctl`, allora per trovare il nome del ruolo che è stato creato, apri la AWS CloudFormation console e seleziona lo stack eksctl- ***`my-cluster`*** - addon-iamserviceaccount-kube-system - aws-load-balancer-controller. Selezionare la scheda Risorse. Il nome del ruolo è nella colonna ID fisico.

```
aws iam attach-role-policy \
  --role-name your-role-name \
  --policy-arn arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerAdditionalIAMPolicy
```

Gestisci CoredNS per DNS nei cluster Amazon EKS

> **Tip**
>
> Con la modalità automatica di Amazon EKS, non devi installare o aggiornare componenti aggiuntivi di rete. Auto Mode include funzionalità di pod networking e bilanciamento del carico.
> Per ulteriori informazioni, consulta Modalità automatica EKS.

CoreDNS è un server DNS flessibile ed estensibile che può fungere da cluster DNS Kubernetes. Quando si avvia un cluster Amazon EKS con almeno un nodo, due repliche dell'immagine CoreDNS vengono distribuite per impostazione predefinita, indipendentemente dal numero di nodi distribuiti nel cluster. I contenitori CoreDNS forniscono la risoluzione dei nomi per tutti i pod del cluster. I pod CoreDNS possono essere distribuiti ai nodi Fargate se il cluster include un profilo Fargate con un namespace che corrisponde al namespace del `deployment` CoreDNS. Per ulteriori informazioni su CoreDNS, consultare Utilizzo di CoreDNS per l'individuazione dei servizi nella documentazione Kubernetes.

Versioni di CoreDNS

La tabella seguente elenca la versione più recente del componente aggiuntivo di Amazon EKS disponibile per ogni versione di Kubernetes.

Versione di Kubernetes	Versione di CoreDNS
1.35	v1.13.2-eksbuild.1
1,34	v1.13.2-eksbuild.1
1.33	v1.13.2-eksbuild.1
1,32	v1.11.4-eksbuild.28
1.31	v1.11.4-eksbuild.28
1.30	v1.11.4-eksbuild.28
1,29	v1.11.4-eksbuild.28

> **Important**
>
> Se gestisci autonomamente questo componente aggiuntivo, le versioni nella tabella potrebbero non essere le stesse di quelle autogestite disponibili. Per ulteriori informazioni sull'aggiornamento del componente aggiuntivo del tipo autogestito, consulta the section called "Aggiornamento (autogestito)".

Considerazioni importanti sull'aggiornamento di CoreDNS

- Gli aggiornamenti CoreDNS utilizzano PodDisruptionBudget un per aiutare a mantenere la disponibilità del servizio DNS durante il processo di aggiornamento.
- Per migliorare la stabilità e la disponibilità di Implementazione di CoreDNS, le versioni `v1.9.3-eksbuild.6` e successive e `v1.10.1-eksbuild.3` vengono distribuite con un `PodDisruptionBudget`. Se hai implementato un `PodDisruptionBudget` esistente, l'aggiornamento a queste versioni potrebbe non riuscire. Se l'aggiornamento non riesce, il problema dovrebbe essere risolto eseguendo una delle seguenti operazioni:
  - Quando esegui l'aggiornamento del componente aggiuntivo Amazon EKS, scegli di sovrascrivere le impostazioni esistenti come opzione di risoluzione dei conflitti. Se hai effettuato altre impostazioni personalizzate su Implementazione, assicurati di eseguire il backup delle impostazioni prima dell'aggiornamento in modo da poter riapplicare le altre impostazioni personalizzate dopo l'aggiornamento.
  - Rimuovi `PodDisruptionBudget` esistente e riprova a eseguire l'aggiornamento.
- Nelle versioni aggiuntive EKS `v1.9.3-eksbuild.3` e successive e `v1.10.1-eksbuild.6` e successive, Implementazione di CoreDNS imposta il `readinessProbe` per utilizzare l'endpoint `/ready`. Questo endpoint è abilitato nel file di configurazione `Corefile` per CoreDNS.

  Se utilizzi un dispositivo personalizzato `Corefile`, devi aggiungere il plug-in `ready` alla configurazione, in modo che l'endpoint `/ready` sia attivo in CoreDNS e possa essere utilizzato dalla sonda.
- Nelle versioni aggiuntive EKS `v1.9.3-eksbuild.7` e successive e `v1.10.1-eksbuild.4` e successive, è possibile modificare il. `PodDisruptionBudget` È possibile modificare il componente aggiuntivo e modificare queste impostazioni nelle configurazioni facoltative utilizzando i campi dell'esempio seguente. Questo esempio mostra l'impostazione predefinita `PodDisruptionBudget`.

```
{
    "podDisruptionBudget": {
        "enabled": true,
        "maxUnavailable": 1
        }
}
```

Puoi impostare `maxUnavailable` o `minAvailable`, ma non puoi impostare entrambi sullo stesso `PodDisruptionBudget`. Per ulteriori informazioni su*`PodDisruptionBudgets`*, consulta Specificare a nella documentazione di Kubernetes PodDisruptionBudget.

Tieni presente che, se imposti `enabled` su `false`, il `PodDisruptionBudget` non viene rimosso. Dopo aver impostato questo campo su `false`, è necessario eliminare l'oggetto `PodDisruptionBudget`. Allo stesso modo, se modifichi il componente aggiuntivo per utilizzarne una versione precedente (esegui il downgrade del componente aggiuntivo) dopo l'aggiornamento a una versione con un `PodDisruptionBudget`, questo non viene rimosso `PodDisruptionBudget`. Esegui il seguente comando per eliminare il `PodDisruptionBudget`:

```
kubectl delete poddisruptionbudget coredns -n kube-system
```

- Nelle versioni dei componenti aggiuntivi di EKS `v1.10.1-eksbuild.5` e successive, modifica la tolleranza predefinita da `node-role.kubernetes.io/master:NoSchedule` a `node-role.kubernetes.io/control-plane:NoSchedule` per conformarla a KEP 2067. Per ulteriori informazioni su KEP 2067, consulta KEP-2067: Rinomina l'etichetta «master» e il taint di kubeadm in Kubernetes Enhancement Proposals () on. KEPs GitHub

  Nelle versioni dei componenti aggiuntivi di EKS `v1.8.7-eksbuild.8` e successive e `v1.9.3-eksbuild.9` e successive, entrambe le tolleranze sono impostate per essere compatibili con tutte le versioni di Kubernetes.
- Nelle versioni aggiuntive EKS `v1.9.3-eksbuild.11` e `v1.10.1-eksbuild.7` e successive, l'implementazione di CoreDNS imposta un valore predefinito per `topologySpreadConstraints`. Il valore predefinito garantisce che i pod CoreDNS siano distribuiti tra le zone di disponibilità se sono disponibili nodi in più zone di disponibilità. Puoi impostare un valore personalizzato che verrà utilizzato al posto di quello predefinito. Il valore predefinito è riportato di seguito:

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
```

```
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

Considerazioni sugli aggiornamenti **v1.11** di CoreDNS

- Nelle versioni `v1.11.1-eksbuild.4` e successive del componente aggiuntivo di EKS, l'immagine del container è basata su un'immagine di base minima gestita da Amazon EKS Distro, che contiene pacchetti minimi e non ha shell. Per ulteriori informazioni, consulta Amazon EKS Distro. L'utilizzo e la risoluzione dei problemi dell'immagine CoreDNS rimangono gli stessi.

Creazione del componente aggiuntivo CoreDNS di Amazon EKS

Creazione del componente aggiuntivo CoreDNS di Amazon EKS. Prima di creare il componente aggiuntivo, è necessario disporre di un cluster. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ".

1. Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster.

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.10.1-eksbuild.13
   ```

2. Scopri qual è il tipo di componente aggiuntivo attualmente installato sul cluster. A seconda dello strumento con cui hai creato il cluster, al momento potresti non avere il componente aggiuntivo del tipo Amazon EKS installato sul cluster. Sostituisci *my-cluster* con il nome del cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se viene restituito un numero di versione, sul cluster è installato il componente aggiuntivo del tipo Amazon EKS e non è necessario completare i passaggi rimanenti di questa procedura. Se viene restituito un errore, sul cluster non è installato il componente aggiuntivo del tipo Amazon EKS. Completa i passaggi rimanenti di questa procedura per installarlo.

3. Salva la configurazione del componente aggiuntivo attualmente installato.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

4. Creare il componente aggiuntivo utilizzando AWS CLI. Se desideri utilizzare la Console di gestione AWS o il `eksctl` per creare il componente aggiuntivo, consulta the section called "Creare un componente aggiuntivo" e indica `coredns` come nome del componente aggiuntivo. Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato.
   - Sostituisci *`my-cluster`* con il nome del tuo cluster.
   - Sostituire *`v1.11.3-eksbuild.1`* con la versione più recente indicata nella tabella delle versioni più recenti per la versione del cluster.

```
aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-version
 v1.11.3-eksbuild.1
```

   Se al componente aggiuntivo attuale sono state applicate impostazioni personalizzate in conflitto con le impostazioni predefinite del componente aggiuntivo di Amazon EKS, la creazione potrebbe non andare a buon fine. Se la creazione non riesce, riceverai un errore che può aiutare a risolvere il problema. In alternativa, puoi aggiungere `--resolve-conflicts OVERWRITE` al comando precedente. Ciò consente al componente aggiuntivo di sovrascrivere le impostazioni personalizzate esistenti. Una volta creato il componente aggiuntivo, è possibile aggiornarlo con impostazioni personalizzate.

5. Verificare che la versione più recente del componente aggiuntivo per la versione Kubernetes del cluster sia stata aggiunta al cluster. Sostituisci *`my-cluster`* con il nome del tuo cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

   La creazione del componente aggiuntivo potrebbe richiedere alcuni secondi.

   Di seguito viene riportato un output di esempio:

```
v1.11.3-eksbuild.1
```

6. Se hai creato impostazioni personalizzate per il componente aggiuntivo originale, prima di creare il componente aggiuntivo di Amazon EKS, utilizza la configurazione che hai salvato precedentemente per aggiornare il componente aggiuntivo di Amazon EKS con le tue impostazioni

personalizzate. Per istruzioni sull'aggiornamento del componente aggiuntivo, consultare la pagina the section called "Aggiornamento (componente aggiuntivo EKS)".

Aggiornamento del componente aggiuntivo CoreDNS di Amazon EKS

Aggiorna il componente aggiuntivo del tipo Amazon EKS. Se al cluster non è stato aggiunto il componente aggiuntivo Amazon EKS, aggiungerlo o consultare the section called "Aggiornamento (autogestito)".

Prima di iniziare, consultare le considerazioni sull'aggiornamento. Per ulteriori informazioni, consulta the section called "Considerazioni importanti sull'aggiornamento di CoreDNS".

1. Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster. Sostituire *my-cluster* con il nome del cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    "addon.addonVersion" --output text
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.10.1-eksbuild.13
   ```

   Se la versione restituita è la stessa della versione Kubernetes del cluster presente nella tabella delle versioni più recenti, la versione più recente è già installata nel cluster e non è necessario completare il resto di questa procedura. Se nell'output viene restituito un errore invece di un numero di versione, il componente aggiuntivo del tipo Amazon EKS non è installato sul cluster. È necessario creare il componente aggiuntivo prima di poterlo aggiornare con questa procedura.
2. Salva la configurazione del componente aggiuntivo attualmente installato.

   ```
   kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
   ```
3. Aggiornare il componente aggiuntivo utilizzando AWS CLI. Se desideri utilizzare la Console di gestione AWS o il `eksctl` per aggiornare il componente aggiuntivo, consulta the section called "Aggiornamento di un componente aggiuntivo". Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato.

   - Sostituisci *my-cluster* con il nome del tuo cluster.

- Sostituire *v1.11.3-eksbuild.1* con la versione più recente indicata nella tabella delle versioni più recenti per la versione del cluster.
- L'opzione `--resolve-conflicts`*PRESERVE* mantiene i valori di configurazione esistenti per il componente aggiuntivo. Se sono stati configurati valori personalizzati per le impostazioni del componente aggiuntivo e non si utilizza questa opzione, Amazon EKS sovrascrive i valori personalizzati con quelli predefiniti. Se utilizzi questa opzione, è preferibile testare eventuali modifiche ai campi e ai valori su un cluster non di produzione prima di aggiornare il componente aggiuntivo sul cluster di produzione. Se modifichi questo valore in `OVERWRITE`, tutte le impostazioni vengono modificate nei valori predefiniti di Amazon EKS. Se sono stati configurati valori personalizzati per un'impostazione qualunque, è possibile che vengano sovrascritti con i valori predefiniti di Amazon EKS. Se questo valore viene modificato in `none`, Amazon EKS non modifica il valore di alcuna impostazione, ma l'aggiornamento potrebbe non andare a buon fine. Se l'aggiornamento non riesce, riceverai un messaggio di errore che ti aiuterà a risolvere il conflitto.
- Se non viene aggiornata un'impostazione di configurazione, rimuovere `--configuration-values '{`*"replicaCount":3*`}'` dal comando. Se viene aggiornata un'impostazione di configurazione, sostituire *"replicaCount":3* con l'impostazione desiderata. In questo esempio, il numero di repliche di CoreDNS è impostato su `3`. Il valore specificato deve essere valido per lo schema di configurazione. Se non si conosce lo schema di configurazione, è possibile eseguire `aws eks describe-addon-configuration --addon-name coredns --addon-version` *v1.11.3-eksbuild.1* sostituendo *v1.11.3-eksbuild.1* con il numero di versione del componente aggiuntivo di cui si desidera visualizzare la configurazione. Lo schema viene restituito nell'output. Se disponi di una configurazione personalizzata, desideri rimuoverla e reimpostare i valori di tutte le impostazioni ai valori predefiniti di Amazon EKS, rimuovi *"replicaCount":3* dal comando in modo da avere delle `{}` vuote. Per ulteriori informazioni sulle impostazioni di CoreDNS, consultare la pagina Customizing DNS Service della documentazione di 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}'
```

Il completamento dell'aggiornamento potrebbe richiedere alcuni secondi.

4. Conferma che la versione del componente aggiuntivo sia stata aggiornata. Sostituisci *my-cluster* con il nome del tuo cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

Il completamento dell'aggiornamento potrebbe richiedere alcuni secondi.

Di seguito viene riportato un output di esempio:

```
{
    "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}"
    }
}
```

Aggiornamento del componente aggiuntivo CoreDNS autogestito di Amazon EKS

Important

Consigliamo di aggiungere al cluster il componente aggiuntivo del tipo Amazon EKS anziché quello del tipo autogestito. In caso di dubbi sulle differenze tra i due tipi, consultare la pagina the section called "Componenti aggiuntivi Amazon EKS". Per ulteriori informazioni sull'aggiunta di un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Se non è possibile utilizzare il componente aggiuntivo di Amazon EKS, si consiglia di segnalare il problema nel repository GitHub delle roadmap dei container.

Prima di iniziare, consultare le considerazioni sull'aggiornamento. Per ulteriori informazioni, consulta the section called "Considerazioni importanti sull'aggiornamento di CoreDNS".

1. Verifica che sul cluster sia installato il componente aggiuntivo del tipo autogestito. Sostituisci *my-cluster* con il nome del tuo cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se viene restituito un messaggio di errore, sul cluster è installato il componente aggiuntivo del tipo autogestito. Completa i passaggi rimanenti di questa procedura. Se viene restituito il numero di versione, sul cluster è installato il tipo Amazon EKS del componente aggiuntivo. Per aggiornare il componente aggiuntivo del tipo Amazon EKS, utilizzare la procedura descritta alla pagina Update the CoreDNS Amazon EKS add-on anziché quella qui indicata. In caso di dubbi sulle differenze tra i tipi di componente aggiuntivo, consultare la pagina the section called "Componenti aggiuntivi Amazon EKS".
2. Scopri qual è la versione dell'immagine di container attualmente installata sul cluster.

   ```
   kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.8.7-eksbuild.2
   ```
3. Se la versione corrente di CoreDNS è `v1.5.0` o successiva, ma precedente alla versione indicata nella tabella Versioni CoreDNS, saltare questa fase. Se la versione corrente è precedente alla `1.5.0`, è necessario modificare `ConfigMap` per CoreDNS al fine di utilizzare il componente aggiuntivo avanzato, anziché il componente aggiuntivo proxy.

   a. Aprire il file `ConfigMap` con il comando seguente:

      ```
      kubectl edit configmap coredns -n kube-system
      ```

   b. Sostituire `proxy` nella linea seguente con `forward`. Salvare il file e uscire dall'editor.

      ```
      proxy . /etc/resolv.conf
      ```
4. Se il cluster è stato originariamente implementato su Kubernetes `1.17` o versioni precedenti, potrebbe essere necessario rimuovere una riga obsoleta dal manifesto CoreDNS.

> **Important**
>
> È necessario completare questo passaggio prima di eseguire l'aggiornamento alla versione `1.7.0` di CoreDNS, ma è preferibile completarlo anche si esegue l'aggiornamento a una versione precedente.

a. Controllare se il manifesto CoreDNS presenta la riga di inserimento.

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
 grep upstream
```

Se non viene restituito alcun output, nel manifesto non è presente la riga ed è possibile passare direttamente alla fase successiva per aggiornare CoreDNS. Se viene restituito l'output, è necessario rimuovere la riga di inserimento.

b. Modificare `ConfigMap` con il seguente comando, rimuovendo la riga nel file che ha la parola `upstream` in esso. Non apportare ulteriori modifiche al file. Una volta rimossa la riga di inserimento, salvare le modifiche.

```
kubectl edit configmap coredns -n kube-system -o yaml
```

5. Recuperare la versione attuale dell'immagine CoreDNS:

```
kubectl describe deployment coredns -n kube-system | grep Image
```

Di seguito viene riportato un output di esempio:

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. Se si esegue l'aggiornamento a CoreDNS versione `1.8.3` o successive, è necessario aggiungere l'autorizzazione `endpointslices` al `system:coredns` del `clusterrole` Kubernetes.

```
kubectl edit clusterrole system:coredns -n kube-system
```

Aggiungere le seguenti righe sotto le righe di autorizzazioni esistenti nella sezione `rules` del file.

```
[...]
```

```
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
[...]
```

7. Aggiornare il componente aggiuntivo CoreDNS sostituendo*602401143452* e *region-code* con i valori dell'output restituiti in una fase precedente. Sostituire *v1.11.3-eksbuild.1* con la versione CoreDNS indicata nella tabella delle versioni più recenti per la versione 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
   ```

   Di seguito viene riportato un output di esempio:

   ```
   deployment.apps/coredns image updated
   ```

8. Controlla nuovamente la versione dell'immagine del container per accertarti che sia stata aggiornata alla versione specificata nel passaggio precedente.

   ```
   kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.11.3-eksbuild.1
   ```

Scalare i pod CoreDNS per un traffico DNS elevato

Quando si avvia un cluster Amazon EKS con almeno un nodo, viene distribuita per impostazione predefinita un'implementazione di due repliche dell'immagine CoreDNS, indipendentemente dal numero di nodi distribuiti nel cluster. I contenitori CoreDNS forniscono la risoluzione dei nomi per tutti i pod del cluster. Le applicazioni utilizzano la risoluzione dei nomi per connettersi a pod e servizi nel cluster, nonché a servizi esterni al cluster. Con l'aumentare del numero di richieste di risoluzione dei nomi (query) provenienti dai pod, i pod CoreDNS possono sovraccaricarsi, rallentare e rifiutare le richieste che i pod non sono in grado di gestire.

Per gestire il carico maggiore sui pod CoreDNS, valutare l'implementazione di un sistema di dimensionamento automatico per CoreDNS. Amazon EKS può gestire il dimensionamento automatico dell'implementazione CoreDNS nella versione del componente aggiuntivo EKS di CoreDNS. Questo autoscaler di CoreDNS monitora continuamente lo stato del cluster, incluso il numero di nodi e core CPU. Sulla base di tali informazioni, il controller adatterà dinamicamente il numero di repliche dell'implementazione CoreDNS in un cluster EKS. Questa funzionalità è disponibile per CoreDNS versione `v1.9` e successive. Per ulteriori informazioni sulle versioni compatibili con il dimensionamento automatico di CoreDNS, consultare la sezione di seguito.

Il sistema gestisce automaticamente le repliche di CoreDNS utilizzando una formula dinamica basata sia sul numero di nodi che di core CPU nel cluster, calcolato come il massimo di (numberOfNodes diviso 16) e (numberOfCPUCores diviso 256). Valuta la domanda su periodi di picco di 10 minuti e aumenta verticalmente in modo immediato quando necessario per gestire l'aumento del carico di query DNS, mentre riduce verticalmente in modo graduale tramite la riduzione delle repliche del 33% ogni 3 minuti per mantenere la stabilità del sistema ed evitare interruzioni.

Consigliamo di utilizzare questa funzionalità insieme ad altre best practice di dimensionamento automatico dei cluster EKSper migliorare la disponibilità dell'applicazione e la scalabilità del cluster in generale.

Prerequisiti

Esistono tre prerequisiti affinché Amazon EKS possa scalare l'implementazione CoreDNS:

- È necessario utilizzare la versione componente aggiuntivo EKS di CoreDNS.
- Il cluster deve eseguire almeno le versioni minime del cluster e della piattaforma.
- Il cluster deve eseguire almeno la versione minima del componente aggiuntivo EKS di CoreDNS.

Versione minima del cluster

Il dimensionamento automatica di CoreDNS viene eseguito da un nuovo componente nel piano di controllo (control-plane) del cluster, gestito da Amazon EKS. Per questo, è necessario aggiornare il cluster a una versione EKS che supporti la versione minima della piattaforma che include il nuovo componente.

Un nuovo cluster Amazon EKS. Per implementarne uno, consulta Nozioni di base. Il cluster deve eseguire una delle versioni di Kubernetes e della piattaforma elencate nella tabella seguente o versioni successive. Sono supportate anche tutte le versioni di Kubernetes e della piattaforma successive a quelle elencate. È possibile verificare la versione corrente di Kubernetes sostituendo

my-cluster nel seguente comando con il nome del cluster e quindi eseguendo il comando modificato:

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Versione di Kubernetes 1.15	Versione della piattaforma
Non elencata	Tutte le versioni della piattaforma
1.29.3	eks.7
1.28.8	eks.13
1.27.12	eks.17
1.26.15	eks.18

> **Note**
>
> Sono supportate anche tutte le versioni della piattaforma delle versioni successive di Kubernetes, ad esempio la versione Kubernetes 1.30 da eks.1 in poi.

Versione minima del componente aggiuntivo EKS

Versione di Kubernetes 1.15	1.29	1.28
	v1.11.1-eksbuild.9	v1.10.1-eksbuild.11

Configurazione del dimensionamento automatico di CoreDNS nella Console di gestione AWS

1. Assicurarsi che il cluster disponga della versione minima del cluster o successiva.

   Amazon EKS aggiorna automaticamente i cluster tra le versioni della piattaforma della stessa versione di Kubernetes e non è possibile avviare questo processo autonomamente. È invece possibile aggiornare il cluster alla versione successiva di Kubernetes e ciò comporterà l'aggiornamento del cluster alla versione K8s e alla versione della piattaforma più recente.

Le nuove versioni di Kubernetes hanno introdotto modifiche significative. Pertanto consigliamo di verificare il comportamento delle applicazioni utilizzando un cluster separato con la nuova versione di Kubernetes prima di eseguire l'aggiornamento sui cluster di produzione.

Per aggiornare un cluster a una nuova versione di Kubernetes, seguire la procedura descritta alla pagina Update existing cluster to new Kubernetes version.

2. Assicurarsi di disporre del componente aggiuntivo EKS per CoreDNS, non dell'implementazione CoreDNS autogestita.

   A seconda dello strumento con cui hai creato il cluster, al momento potresti non avere il componente aggiuntivo del tipo Amazon EKS installato sul cluster. Eseguire il comando seguente per verificare quale tipo di componente aggiuntivo è installato sul cluster. Sostituisci `my-cluster` con il nome del cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se viene restituito un numero di versione, sul cluster è installato il tipo Amazon EKS del componente aggiuntivo ed è possibile procedere al passaggio successivo. Se viene restituito un errore, sul cluster non è installato il componente aggiuntivo del tipo Amazon EKS. Completare i passaggi rimanenti della procedura descritta alla pagina Create the CoreDNS Amazon EKS add-on per sostituire la versione autogestita con il componente aggiuntivo EKS.

3. Assicurarsi che il componente aggiuntivo EKS per CoreDNS disponga di una versione uguale o successiva rispetto alla versione minima del componente aggiuntivo EKS.

   Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster. È possibile verificarla nella Console di gestione AWS o eseguendo il seguente comando:

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.10.1-eksbuild.13
   ```

   Confrontare questa versione con quella minima del componente aggiuntivo EKS riportata nella sezione precedente. Se necessario, aggiornare il componente aggiuntivo EKS a una versione

successiva seguendo la procedura descritta alla pagina Update the CoreDNS Amazon EKS add-on.

4. Aggiungere la configurazione del dimensionamento automatico alle Opzioni di configurazione facoltative del componente aggiuntivo EKS.
   a. Aprire la Console Amazon EKS.
   b. Nel riquadro di navigazione a sinistra, seleziona Cluster, quindi seleziona il nome del cluster per cui configurare il componente aggiuntivo.
   c. Seleziona la scheda Componenti aggiuntivi.
   d. Selezionare la casella nella parte superiore destra della casella del componente aggiuntivo CoreDNS e scegliere Modifica.
   e. Nella pagina Configurazione di CoreDNS:
      i. Selezionare la Versione da utilizzare. Si consiglia di mantenere la stessa versione rispetto al passaggio precedente e di aggiornare la versione e la configurazione con operazioni separate.
      ii. Scegli Impostazioni di configurazione facoltative.
      iii. Inserire la chiave JSON `"autoscaling":` e il valore di un oggetto JSON annidato con una chiave `"enabled":` e un valore `true` nei Valori di configurazione. Il testo risultante deve essere un oggetto JSON valido. Se questa chiave e questo valore sono gli unici dati nella casella di testo, racchiudi la chiave e il valore tra parentesi graffe `{ }`. Il seguente esempio mostra che il dimensionamento automatico è abilitato:

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

      iv. (Facoltativo) È possibile fornire valori minimi e massimi a cui il dimensionamento automatico può scalare il numero di pod CoreDNS.

      Il seguente esempio mostra che il dimensionamento automatico è abilitato e tutte le chiavi facoltative hanno dei valori. Si consiglia di utilizzare un numero minimo di pod CoreDNS sempre superiore a 2 per garantire la resilienza del servizio DNS nel cluster.

```
{
  "autoScaling": {
    "enabled": true,
```

```
    "minReplicas": 2,
    "maxReplicas": 10
  }
}
```

f. Scegliere Salva modifiche per applicare la nuova configurazione sostituendo i pod CoreDNS.

Amazon EKS applica le modifiche ai componenti aggiuntivi EKS utilizzando un rollout dell'implementazione Kubernetes per CoredNS. È possibile monitorare lo stato del rollout nella Cronologia degli aggiornamenti del componente aggiuntivo nella Console di gestione AWS e con `kubectl rollout status deployment/coredns --namespace kube-system`.

`kubectl rollout` dispone dei seguenti comandi:

```
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 il rollout richiede troppo tempo, Amazon EKS annullerà l'operazione e un messaggio contenente il tipo di Aggiornamento del componente aggiuntivo e lo stato Non riuscito sarà aggiunto alla Cronologia degli aggiornamenti del componente aggiuntivo. Per esaminare eventuali problemi, iniziare dalla cronologia del rollout ed eseguire `kubectl logs` su un pod CoreDNS per visualizzare i log di CoreDNS.

5. Se la nuova voce nella Cronologia degli aggiornamenti ha lo stato Operazione riuscita, il rollout è stato completato e il componente aggiuntivo utilizza la nuova configurazione in tutti i pod CoreDNS. Con la modifica del numero di nodi e core CPU dei nodi nel cluster, Amazon EKS scala il numero di repliche dell'implementazione CoreDNS.

Configurazione del dimensionamento automatico nell'Interfaccia della linea di comando AWS

1. Assicurarsi che il cluster disponga della versione minima del cluster o successiva.

Amazon EKS aggiorna automaticamente i cluster tra le versioni della piattaforma della stessa versione di Kubernetes e non è possibile avviare questo processo autonomamente. È invece

possibile aggiornare il cluster alla versione successiva di Kubernetes e ciò comporterà l'aggiornamento del cluster alla versione K8s e alla versione della piattaforma più recente.

Le nuove versioni di Kubernetes hanno introdotto modifiche significative. Pertanto consigliamo di verificare il comportamento delle applicazioni utilizzando un cluster separato con la nuova versione di Kubernetes prima di eseguire l'aggiornamento sui cluster di produzione.

Per aggiornare un cluster a una nuova versione di Kubernetes, seguire la procedura descritta alla pagina Update existing cluster to new Kubernetes version.

2. Assicurarsi di disporre del componente aggiuntivo EKS per CoreDNS, non dell'implementazione CoreDNS autogestita.

   A seconda dello strumento con cui hai creato il cluster, al momento potresti non avere il componente aggiuntivo del tipo Amazon EKS installato sul cluster. Eseguire il comando seguente per verificare quale tipo di componente aggiuntivo è installato sul cluster. Sostituisci `my-cluster` con il nome del cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Se viene restituito il numero di versione, sul cluster è installato il tipo Amazon EKS del componente aggiuntivo. Se viene restituito un errore, sul cluster non è installato il componente aggiuntivo del tipo Amazon EKS. Completare i passaggi rimanenti della procedura descritta alla pagina Create the CoreDNS Amazon EKS add-on per sostituire la versione autogestita con il componente aggiuntivo EKS.

3. Assicurarsi che il componente aggiuntivo EKS per CoreDNS disponga di una versione uguale o successiva rispetto alla versione minima del componente aggiuntivo EKS.

   Scopri qual è la versione del componente aggiuntivo attualmente installata sul cluster. È possibile verificarla nella Console di gestione AWS o eseguendo il seguente comando:

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   Di seguito viene riportato un output di esempio:

   ```
   v1.10.1-eksbuild.13
   ```

Confrontare questa versione con quella minima del componente aggiuntivo EKS riportata nella sezione precedente. Se necessario, aggiornare il componente aggiuntivo EKS a una versione successiva seguendo la procedura descritta alla pagina Update the CoreDNS Amazon EKS add-on.

4. Aggiungere la configurazione del dimensionamento automatico alle Opzioni di configurazione facoltative del componente aggiuntivo EKS.

   Esegui il seguente comando AWS CLI. Sostituisci `my-cluster` con il nome del cluster e l'ARN del ruolo IAM con il ruolo che stai utilizzando.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true}}'
   ```

   Amazon EKS applica le modifiche ai componenti aggiuntivi EKS utilizzando un rollout dell'implementazione Kubernetes per CoredNS. È possibile monitorare lo stato del rollout nella Cronologia degli aggiornamenti del componente aggiuntivo nella Console di gestione AWS e con `kubectl rollout status deployment/coredns --namespace kube-system`.

   `kubectl rollout` dispone dei seguenti comandi:

   ```
   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 il rollout richiede troppo tempo, Amazon EKS annullerà l'operazione e un messaggio contenente il tipo di Aggiornamento del componente aggiuntivo e lo stato Non riuscito sarà aggiunto alla Cronologia degli aggiornamenti del componente aggiuntivo. Per esaminare eventuali problemi, iniziare dalla cronologia del rollout ed eseguire `kubectl logs` su un pod CoreDNS per visualizzare i log di CoreDNS.

5. (Facoltativo) È possibile fornire valori minimi e massimi a cui il dimensionamento automatico può scalare il numero di pod CoreDNS.

Il seguente esempio mostra che il dimensionamento automatico è abilitato e tutte le chiavi facoltative hanno dei valori. Si consiglia di utilizzare un numero minimo di pod CoreDNS sempre superiore a 2 per garantire la resilienza del servizio DNS nel 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. Verificare lo stato dell'aggiornamento del componente aggiuntivo eseguendo il seguente comando:

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

Se viene visualizzata la riga `"status": "ACTIVE"`, il rollout è stato completato e il componente aggiuntivo utilizza la nuova configurazione in tutti i pod CoreDNS. Con la modifica del numero di nodi e core CPU dei nodi nel cluster, Amazon EKS scala il numero di repliche dell'implementazione CoreDNS.

Monitorare la risoluzione DNS di Kubernetes con parametri CoreDNS

CoreDNS come componente aggiuntivo EKS espone i parametri di CoreDNS sulla porta `9153` nel formato Prometheus nel servizio `kube-dns`. Puoi usare Prometheus, l'agente Amazon CloudWatch o qualsiasi altro sistema compatibile per eseguire lo scraping (raccogliere) questi parametri.

Per un esempio di configurazione di scraping compatibile sia con Prometheus che con l'agente CloudWatch, consulta Configurazione dell'agente CloudWatch per Prometheus nella Guida per l'utente di Amazon CloudWatch.

Gestisci **`kube-proxy`** nei cluster Amazon EKS

> **Tip**
>
> Con la modalità automatica di Amazon EKS, non è necessario installare o aggiornare componenti aggiuntivi di rete. Auto Mode include funzionalità di pod networking e bilanciamento del carico.
> Per ulteriori informazioni, consulta Modalità automatica EKS.

Consigliamo di aggiungere al cluster il componente aggiuntivo del tipo Amazon EKS anziché quello del tipo autogestito. Se non sono ben chiare le differenze tra i due tipi, consulta the section called "Componenti aggiuntivi Amazon EKS". Per ulteriori informazioni sull'aggiunta di un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Se non riesci a utilizzare il componente aggiuntivo Amazon EKS, ti consigliamo di segnalare un problema sul motivo per cui non puoi farlo all'archivio della roadmap GitHub di Containers.

Il componente aggiuntivo `kube-proxy` viene implementato su ogni nodo Amazon EC2 del cluster Amazon EKS. Mantiene le regole di rete sui nodi e consente la comunicazione di rete con i tuoi pod. Il componente aggiuntivo non viene implementato sui nodi Fargate nel cluster. Per ulteriori informazioni, consulta kube-proxy nella documentazione Kubernetes.

Installa come componente aggiuntivo di Amazon EKS

Versioni **`kube-proxy`**

La tabella seguente elenca la versione più recente del componente aggiuntivo di Amazon EKS disponibile per ogni versione di Kubernetes.

Versione di Kubernetes	Versione **`kube-proxy`**
1,35	v1.35.0-eksbuild.2
1,34	v1.34.3-eksbuild.2
1.33	v1.33.7-eksbuild.2
1,32	v1.32.11-eksbuild.2
1.31	v1.31.14-eksbuild.2
1.30	v1.30.14-eksbuild.20
1,29	v1.29.15-eksbuild.28

> **Note**
>
> Una versione precedente della documentazione non era corretta. Le versioni di `kube-proxy` `v1.28.5`, `v1.27.9` e `v1.26.12` non sono disponibili.

Se gestisci autonomamente questo componente aggiuntivo, le versioni nella tabella potrebbero non essere le stesse delle versioni autogestite disponibili.

Immagine di container **`kube-proxy`**

L'immagine del container `kube-proxy` è basata su un'immagine di base minima gestita da Amazon EKS Distro, che contiene pacchetti minimi e non ha shell. Per ulteriori informazioni, consulta Amazon EKS Distro.

La tabella seguente elenca la versione più recente disponibile dell'immagine del container `kube-proxy` auto-gestita per ciascuna versione del cluster Amazon EKS.

Versione	kube-proxy
1,35	v1.35.0-eksbuild.2
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

Quando aggiorni un tipo di componente aggiuntivo Amazon EKS, specifichi una versione del componente aggiuntivo Amazon EKS valida, che potrebbe non essere una versione elencata in questa tabella. Questo accade perché le versioni dei componenti aggiuntivi di Amazon EKS non sempre corrispondono a quelle delle immagini del container specificate durante l'aggiornamento del componente aggiuntivo del tipo autogestito. Quando si aggiorna il componente aggiuntivo del tipo autogestito, si specifica una versione valida dell'immagine di container indicata in questa tabella.

Aggiornare il componente aggiuntivo autogestito **kube-proxy** Kubernetes

> **Important**
>
> Consigliamo di aggiungere al cluster il componente aggiuntivo del tipo Amazon EKS anziché quello del tipo autogestito. Se non conosci bene le differenze tra i due tipi, consulta la pagina the section called "Componenti aggiuntivi Amazon EKS". Per ulteriori informazioni sull'aggiunta di un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Creare un componente aggiuntivo". Se non riesci a utilizzare il componente aggiuntivo di Amazon EKS, ti invitiamo a segnalare il problema in Containers roadmap GitHub repository.

Prerequisiti

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.

Considerazioni

- `Kube-proxy` su un cluster Amazon EKS ha la stessa policy di compatibilità e disallineamento di Kubernetes. Scopri Verifying Amazon EKS add-on version compatibility with a cluster.
  1. Verifica che sul cluster sia installato il componente aggiuntivo del tipo autogestito. Sostituisci *`my-cluster`* con il nome del tuo cluster.

     ```
     aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
      addon.addonVersion --output text
     ```

     Se viene restituito un messaggio di errore, sul cluster è installato il componente aggiuntivo del tipo autogestito. I passaggi rimanenti di questo argomento riguardano l'aggiornamento del componente aggiuntivo del tipo autogestito. Se viene restituito il numero di versione, sul cluster è installato il componente aggiuntivo del tipo Amazon EKS. Per aggiornarlo, utilizza la procedura descritta in Updating an Amazon EKS add-on anziché quella presente in questo argomento. Se non conosci bene le differenze tra i due tipi di componente aggiuntivo, consulta the section called "Componenti aggiuntivi Amazon EKS".
  2. Scopri qual è la versione dell'immagine di container attualmente installata sul cluster.

     ```
     kubectl describe daemonset kube-proxy -n kube-system | grep Image
     ```

Di seguito viene riportato un output di esempio:

```
Image:    602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.29.1-eksbuild.2
```

Nell'output di esempio, *v1.29.1-eksbuild.2* è la versione installata sul cluster.

3. Aggiorna il componente aggiuntivo kube-proxy sostituendo *602401143452* e *region-code* con i valori dell'output del passaggio precedente. Sostituisci *v1.30.6-eksbuild.3* con la versione kube-proxy riportata nella tabella Latest available self-managed kube-proxy container image version for each Amazon EKS cluster version.

> **Important**
>
> I manifesti per ogni tipo di immagine sono diversi e non compatibili tra i tipi di immagine predefiniti o minimi. È necessario utilizzare lo stesso tipo di immagine di quella precedente, in modo che il punto di ingresso e gli argomenti corrispondano.

```
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
```

Di seguito viene riportato un output di esempio:

```
daemonset.apps/kube-proxy image updated
```

4. Verifica che la nuova versione sia ora installata nel cluster.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

Di seguito viene riportato un output di esempio:

```
v1.30.0-eksbuild.3
```

5. Se utilizzi nodi x86 e Arm nello stesso cluster e il cluster è stato implementato il prima del 17 agosto 2020. Modificare, il manifesto kube-proxy attraverso il seguente comando, per includere un selettore di nodi per più architetture hardware. Questa è un'operazione da compiere

una sola volta. Dopo aver aggiunto il selettore al manifesto, non occorrerà aggiungerlo ogni volta che aggiorni il componente aggiuntivo. Se il cluster è stato implementato in data 17 agosto 2020 o successiva, il `kube-proxy` è già in grado di supportare una multi-architettura.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Aggiungere il seguente selettore di nodo al file nell'editor, quindi salvare il file. Per un esempio di dove includere questo testo nell'editor, vedere manifesto CNI su GitHub. Ciò consente a Kubernetes di estrarre l'immagine hardware corretta in base all'architettura hardware del nodo.

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

6. Se il cluster è stato originariamente creato con Kubernetes `1.14` o versione successiva, è possibile saltare questo passaggio perché `kube-proxy` include già `Affinity Rule`. Se in origine hai creato un cluster Amazon EKS con la versione `1.13` di Kubernetes o precedente e intendi utilizzare nodi Fargate nel tuo cluster, modifica il manifesto `kube-proxy` in modo da includere una regola `NodeAffinity` per impedire ai pod `kube-proxy` la pianificazione su nodi Fargate. Questa modifica si applica una sola volta. Dopo aver aggiunto `Affinity Rule` al manifesto, non occorrerà aggiungerlo ogni volta che aggiorni il componente aggiuntivo. Modificare il DaemonSet `kube-proxy`.

   ```
   kubectl edit -n kube-system daemonset/kube-proxy
   ```

   Aggiungi i seguenti elementi `Affinity Rule` alla sezione DaemonSet `spec` del file nell'editor e salvare il file. Per un esempio di dove includere questo testo nell'editor, vedere manifesto CNI su GitHub.

   ```
   - key: eks.amazonaws.com/compute-type
     operator: NotIn
     values:
     - fargate
   ```

Informazioni su come implementare carichi di lavoro e componenti aggiuntivi su Amazon EKS

I carichi di lavoro vengono implementati in container, a loro volta implementati in pod in Kubernetes. Un pod include uno o più container. In genere, uno o più pod che forniscono lo stesso servizio vengono implementati in un servizio Kubernetes. Dopo aver implementato più pod che forniscono lo stesso servizio, puoi:

- Visualizzare le informazioni sui carichi di lavoro in esecuzione su ogni cluster utilizzando il Console di gestione AWS.
- Dimensionare verticalmente i pod verso l'alto o verso il basso con Vertical Pod Autoscaler di Kubernetes.
- Dimensionare orizzontalmente verso l'alto o verso il basso il numero di pod necessari per soddisfare la domanda con Horizontal Pod Autoscaler di Kubernetes.
- Creare un Network Load Balancer esterno (per i pod accessibili a internet) o interno (per i pod privati) per bilanciare il traffico di rete tra i pod. Il load balancer indirizza il traffico al livello 4 del modello OSI.
- Creare un Application Load Balancer per bilanciare il traffico delle applicazioni tra i pod. Il load balancer di applicazione indirizza il traffico al livello 7 del modello OSI.
- Se sei un nuovo utente di Kubernetes, questo argomento aiuta a implementare un'applicazione di esempio.
- È possibile limitare gli indirizzi IP assegnabili a un servizio con `externalIPs`.

Implementazione di un'applicazione esemplificativa

In questa sezione, imparerai a implementare un'applicazione esemplificativa nel cluster sui nodi Linux.

Prerequisiti

- Un cluster Kubernetes esistente con almeno un nodo. Se non si dispone di un cluster Amazon EKS esistente, è possibile implementarne uno utilizzando una delle guide in Nozioni di base.
- `Kubectl` installato sul computer. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".

- `Kubectl` configurato per comunicare con il cluster. Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".
- Se si prevede di implementare il carico di lavoro di esempio in Fargate, è necessario disporre di un profilo Fargate che include lo stesso spazio dei nomi creato in questo tutorial, che è `eks-sample-app`, a meno che non cambi il nome. Se hai creato un cluster con una delle guide in Nozioni di base, dovrai creare un nuovo profilo o aggiungere il namespace al profilo esistente, poiché il profilo creato nelle guide introduttive non specifica il namespace utilizzato in questo tutorial. Il VPC deve disporre di almeno una sottorete privata.

Sebbene molte variabili siano modificabili nei passaggi seguenti, si consiglia di modificare solo i valori delle variabili, se specificato. Con una migliore conoscenza dei pod, delle implementazioni e dei servizi Kubernetes, è possibile provare a modificare altri valori.

Creare uno spazio dei nomi

Uno spazio dei nomi consente di raggruppare le risorse in Kubernetes. Per ulteriori informazioni, consultare Spazio dei nomi nella documentazione Kubernetes. Se intendi di implementare la tua applicazione esemplificativa per semplificare la gestione del calcolo con AWS Fargate, assicurati che il valore per `namespace` nel tuo Definisci quali pod utilizzano AWS Fargate al momento di un avvio sia `eks-sample-app`.

```
kubectl create namespace eks-sample-app
```

Creazione di un'implementazione Kubernetes

Creare un'implementazione Kubernetes. Questa implementazione esemplificativa estrae un'immagine container da un repository pubblico e ne implementa tre repliche (pod individuali) sul cluster. Per ulteriori informazioni, consultare la sezione Implementazioni nella documentazione Kubernetes.

1. Salva i contenuti seguenti in un file denominato `eks-sample-deployment.yaml`. I container nell'applicazione esemplificativa non utilizzano l'archiviazione di rete, ma potrebbe essere necessaria per determinate applicazioni. Per ulteriori informazioni, consulta Archiviazione di dati per app.
   - `amd64` o `arm64` `values` nella chiave `kubernetes.io/arch` significa che l'applicazione può essere implementata su entrambe le architetture hardware (se entrambe sono presenti nel cluster). Ciò è possibile perché questa immagine è un'immagine multi-architettura,

ma non tutte lo sono. È possibile determinare l'architettura hardware su cui è supportata l'immagine visualizzando i dettagli dell'immagine nel repository da cui la stai estraendo. Quando si implementano immagini che non supportano un tipo di architettura hardware o che non si desidera implementare, rimuovere quel tipo ti architettura dal manifesto. Per ulteriori informazioni, consulta Etichette, annotazioni e taint note nella documentazione Kubernetes.

- `kubernetes.io/os: linux` `nodeSelector` significa che se avessi nodi Linux e Windows, ad esempio nel cluster, l'immagine verrebbe implementata solo sui nodi Linux. Per ulteriori informazioni, consulta Etichette, annotazioni e taint note nella documentazione 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. Applicare il manifesto di implementazione al cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

Crea un servizio.

Un servizio consente di accedere a tutte le repliche tramite un unico indirizzo IP o nome. Per ulteriori informazioni, consultare la sezione Servizio nella documentazione Kubernetes. Anche se non implementato nell'applicazione esemplificativa, se si dispone di applicazioni che devono interagire con altri servizi AWS, ti consigliamo di creare account di servizio Kubernetes per i tuoi pod e di associarli agli account IAM AWS. Specificando gli account di servizio, è possibile che i pod dispongano delle autorizzazioni minime necessarie per interagire con altri servizi. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

1. Salva i contenuti seguenti in un file denominato `eks-sample-service.yaml`. Kubernetes assegna al servizio il proprio indirizzo IP accessibile solo dall'interno del cluster. Per accedere al servizio dall'esterno del cluster, implementare il controller del load balancer AWS per bilanciare il carico dell'applicazione o del traffico di rete verso il servizio.

```
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. Applicare il manifesto del servizio al cluster.

```
kubectl apply -f eks-sample-service.yaml
```

Esame delle risorse create

1. Visualizzare tutte le risorse nello spazio dei nomi `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

Di seguito viene riportato un output di esempio:

```
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
```

Nell'output, è possibile visualizzare il servizio e l'implementazione specificati nel manifesto di esempio implementato nei passaggi precedenti. Verranno mostrati anche tre pod. Questo perché 3 `replicas` sono state specificate nel manifesto di esempio. Per ulteriori informazioni sui pod, consulta Pod nella documentazione Kubernetes. Kubernetes crea automaticamente la risorsa `replicaset`, anche se non è specificato nei manifesti esemplificativi. Per ulteriori informazioni su `ReplicaSets`, consultare ReplicaSet nella documentazione Kubernetes.

> **Note**
>
> Kubernetes mantiene il numero di repliche specificate nel manifesto. Se si tratta di un'implementazione di produzione e si desidera che Kubernetes scali orizzontalmente il numero di repliche e scali verticalmente le risorse di calcolo per i pod, utilizza Scale pod deployments with Horizontal Pod Autoscaler e Adjust pod resources with Vertical Pod Autoscaler per fare ciò.

2. Visualizzare i dettagli del servizio implementato.

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

Di seguito viene riportato un output di esempio:

```
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>
```

Nell'output precedente, il valore per `IP:` è un indirizzo IP univoco che può essere raggiunto da qualsiasi nodo o pod all'interno del cluster, ma non può essere raggiunto dall'esterno del cluster. I valori per `Endpoints` sono indirizzi IP assegnati all'interno del VPC ai pod che fanno parte del servizio.

3. Visualizza i dettagli di uno dei pod elencati nell'output al momento della visualizzazione dello spazio dei nomi in un passaggio precedente. Sostituisci *`776d8f8fd8-78w66`* con il valore restituito per uno dei pod.

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

Output esemplificativo abbreviato

```
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
[...]
```

Nell'output precedente, il valore per `IP:` è un IP univoco assegnato al pod dall'intervallo CIDR assegnato alla sottorete in cui si trova il nodo. Se si preferisce che ai pod vengano assegnati indirizzi IP da blocchi CIDR diversi, è possibile modificare il comportamento di default. Per ulteriori informazioni, consulta the section called "Rete personalizzata". È anche possibile vedere come il pianificatore Kubernetes abbia pianificato il pod sul `Node` con l'indirizzo IP *`192.168.45.132`*.

> **Tip**
>
> Anziché utilizzare la riga di comando, è possibile visualizzare molti dettagli su pod, servizi, implementazioni e altre risorse Kubernetes nella Console di gestione AWS. Per ulteriori informazioni, consulta the section called "Accesso alle risorse di cluster".

Eseguire una shell su un pod

1. Esegui una shell sul pod descritto nel passaggio precedente, sostituendo *65 b 7669776-m6qxz* con l'ID di uno dei tuoi pod.

   ```
   kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app -- /
   bin/bash
   ```

2. Dalla shell del pod, visualizzare l'output dal server Web installato con l'implementazione in una fase precedente. È necessario specificare solo il nome del servizio. Ciò è risolto sull'indirizzo IP del servizio da CoreDNS, implementato con un cluster Amazon EKS, per impostazione predefinita.

   ```
   curl eks-sample-linux-service
   ```

   Di seguito viene riportato un output di esempio:

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```

3. Dalla shell del pod, visualizza il server DNS per il pod.

   ```
   cat /etc/resolv.conf
   ```

   Di seguito viene riportato un output di esempio:

   ```
   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
   ```

   Nell'output precedente, `10.100.0.10` viene assegnato automaticamente come `nameserver` per tutti i pod implementati nel cluster.

4. Disconnettersi dal pod digitando `exit`.

5. Una volta terminato l'uso dell'applicazione esemplificativa, sarà possibile rimuovere il namespace, il servizio e l'implementazione esemplificativa con il seguente comando.

```
kubectl delete namespace eks-sample-app
```

Fasi successive

Dopo aver implementato l'applicazione esemplificativa, potresti provare a svolgere alcune delle seguenti esercitazioni:

- Esegui il routing del traffico di applicazioni e HTTP con Application Load Balancer
- Esegui il routing del traffico TCP e UDP con Network Load Balancer

Implementazione di un'applicazione esemplificativa su Windows

In questa sezione, imparerai a implementare un'applicazione esemplificativa nel cluster sui nodi Windows.

Prerequisiti

- Un cluster Kubernetes esistente con almeno un nodo. Se non si dispone di un cluster Amazon EKS esistente, è possibile implementarne uno utilizzando una delle guide in Nozioni di base. Devi avere il Supporto Windows abilitato per il cluster e almeno un nodo Windows di Amazon EC2.
- `Kubectl` installato sul computer. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".
- `Kubectl` configurato per comunicare con il cluster. Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".
- Se si prevede di implementare il carico di lavoro di esempio in Fargate, è necessario disporre di un profilo Fargate che include lo stesso spazio dei nomi creato in questo tutorial, che è `eks-sample-app`, a meno che non cambi il nome. Se hai creato un cluster con una delle guide in Nozioni di base, dovrai creare un nuovo profilo o aggiungere il namespace al profilo esistente, poiché il profilo creato nelle guide introduttive non specifica il namespace utilizzato in questo tutorial. Il VPC deve disporre di almeno una sottorete privata.

Sebbene molte variabili siano modificabili nei passaggi seguenti, si consiglia di modificare solo i valori delle variabili, se specificato. Con una migliore conoscenza dei pod, delle implementazioni e dei servizi Kubernetes, è possibile provare a modificare altri valori.

Creare uno spazio dei nomi

Uno spazio dei nomi consente di raggruppare le risorse in Kubernetes. Per ulteriori informazioni, consultare Spazio dei nomi nella documentazione Kubernetes. Se intendi di implementare la tua applicazione esemplificativa per semplificare la gestione del calcolo con AWS Fargate, assicurati che il valore per `namespace` nel tuo Definisci quali pod utilizzano AWS Fargate al momento di un avvio sia `eks-sample-app`.

```
kubectl create namespace eks-sample-app
```

Creazione di un'implementazione Kubernetes

Questa implementazione esemplificativa estrae un'immagine container da un repository pubblico e ne implementa tre repliche (pod individuali) sul cluster. Per ulteriori informazioni, consultare la sezione Implementazioni nella documentazione Kubernetes.

1. Salva i contenuti seguenti in un file denominato `eks-sample-deployment.yaml`. I container nell'applicazione esemplificativa non utilizzano l'archiviazione di rete, ma potrebbe essere necessaria per determinate applicazioni. Per ulteriori informazioni, consulta Archiviazione di dati per app.
   - `kubernetes.io/os: windows` `nodeSelector` significa che se avessi nodi Windows e Linux, ad esempio nel cluster, l'immagine verrebbe implementata solo nei nodi Windows. Per ulteriori informazioni, consulta Etichette, annotazioni e taint note nella documentazione 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. Applicare il manifesto di implementazione al cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

Crea un servizio.

Un servizio consente di accedere a tutte le repliche tramite un unico indirizzo IP o nome. Per ulteriori informazioni, consultare la sezione Servizio nella documentazione Kubernetes. Anche se non implementato nell'applicazione esemplificativa, se si dispone di applicazioni che devono interagire con altri servizi AWS, ti consigliamo di creare account di servizio Kubernetes per i tuoi pod e di associarli agli account IAM AWS. Specificando gli account di servizio, è possibile che i

pod dispongano delle autorizzazioni minime necessarie per interagire con altri servizi. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

1. Salva i contenuti seguenti in un file denominato `eks-sample-service.yaml`. Kubernetes assegna al servizio il proprio indirizzo IP accessibile solo dall'interno del cluster. Per accedere al servizio dall'esterno del cluster, implementare il controller del load balancer AWS per bilanciare il carico dell'applicazione o del traffico di rete verso il servizio.

```
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. Applicare il manifesto del servizio al cluster.

```
kubectl apply -f eks-sample-service.yaml
```

Esame delle risorse create

1. Visualizzare tutte le risorse nello spazio dei nomi `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

Di seguito viene riportato un output di esempio:

```
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
```

Nell'output, è possibile visualizzare il servizio e l'implementazione specificati nel manifesto di esempio implementato nei passaggi precedenti. Verranno mostrati anche tre pod. Questo perché 3 `replicas` sono state specificate nel manifesto di esempio. Per ulteriori informazioni sui pod, consulta Pod nella documentazione Kubernetes. Kubernetes crea automaticamente la risorsa `replicaset`, anche se non è specificato nei manifesti esemplificativi. Per ulteriori informazioni su `ReplicaSets`, consultare ReplicaSet nella documentazione Kubernetes.

> **Note**
>
> Kubernetes mantiene il numero di repliche specificate nel manifesto. Se si tratta di un'implementazione di produzione e si desidera che Kubernetes scali orizzontalmente il numero di repliche e scali verticalmente le risorse di calcolo per i pod, utilizza Scale pod deployments with Horizontal Pod Autoscaler e Adjust pod resources with Vertical Pod Autoscaler per fare ciò.

2. Visualizzare i dettagli del servizio implementato.

```
kubectl -n eks-sample-app describe service eks-sample-windows-service
```

Di seguito viene riportato un output di esempio:

```
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>
```

Nell'output precedente, il valore per `IP:` è un indirizzo IP univoco che può essere raggiunto da qualsiasi nodo o pod all'interno del cluster, ma non può essere raggiunto dall'esterno del cluster. I valori per `Endpoints` sono indirizzi IP assegnati all'interno del VPC ai pod che fanno parte del servizio.

3. Visualizza i dettagli di uno dei pod elencati nell'output al momento della visualizzazione dello spazio dei nomi in un passaggio precedente. Sostituisci *776d8f8fd8-78w66* con il valore restituito per uno dei pod.

```
kubectl -n eks-sample-app describe pod eks-sample-windows-deployment-65b7669776-m6qxz
```

Output esemplificativo abbreviato

```
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
[...]
```

Nell'output precedente, il valore per `IP:` è un IP univoco assegnato al pod dall'intervallo CIDR assegnato alla sottorete in cui si trova il nodo. Se si preferisce che ai pod vengano assegnati indirizzi IP da blocchi CIDR diversi, è possibile modificare il comportamento di default. Per ulteriori informazioni, consulta the section called "Rete personalizzata". È anche possibile vedere come il pianificatore Kubernetes abbia pianificato il pod sul `Node` con l'indirizzo IP *`192.168.45.132`*.

> **Tip**
>
> Anziché utilizzare la riga di comando, è possibile visualizzare molti dettagli su pod, servizi, implementazioni e altre risorse Kubernetes nella Console di gestione AWS. Per ulteriori informazioni, consulta the section called "Accesso alle risorse di cluster".

Eseguire una shell su un pod

1. Esegui una shell sul pod descritto nel passaggio precedente, sostituendo *`65 b 7669776-m6qxz`* con l'ID di uno dei tuoi pod.

```
kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-app --
 powershell.exe
```

2. Dalla shell del pod, visualizzare l'output dal server Web installato con l'implementazione in una fase precedente. È necessario specificare solo il nome del servizio. Ciò è risolto sull'indirizzo IP del servizio da CoreDNS, implementato con un cluster Amazon EKS, per impostazione predefinita.

```
Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
```

Di seguito viene riportato un output di esempio:

```
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. Dalla shell del pod, visualizza il server DNS per il pod.

```
Get-NetIPConfiguration
```

Output abbreviato

```
InterfaceAlias       : vEthernet
[...]
IPv4Address          : 192.168.63.14
[...]
DNSServer            : 10.100.0.10
```

Nell'output precedente, `10.100.0.10` viene assegnato automaticamente come il server DNS per tutti i pod implementati nel cluster.

4. Disconnettersi dal pod digitando `exit`.
5. Una volta terminato l'uso dell'applicazione esemplificativa, sarà possibile rimuovere il namespace, il servizio e l'implementazione esemplificativa con il seguente comando.

```
kubectl delete namespace eks-sample-app
```

Fasi successive

Dopo aver implementato l'applicazione esemplificativa, potresti provare a svolgere alcune delle seguenti esercitazioni:

- Esegui il routing del traffico di applicazioni e HTTP con Application Load Balancer
- Esegui il routing del traffico TCP e UDP con Network Load Balancer

Regolazione delle risorse del pod con Vertical Pod Autoscaler

Vertical Pod Autoscaler di Kubernetes regola automaticamente le prenotazioni di CPU e memoria per i pod per aiutare a "dimensionare correttamente" le applicazioni. Questo processo di adeguamento può migliorare l'utilizzo delle risorse del cluster e liberare CPU e memoria per altri pod. Questo

argomento illustra come implementare Vertical Pod Autoscaler nel cluster e verificarne il funzionamento.

- Il cluster Amazon EKS è già esistente. In caso contrario, consulta Nozioni di base.
- Kubernetes Metrics Server è già installato. Per ulteriori informazioni, consulta the section called "Metrics Server".
- Stai utilizzando un client `kubectl` che è configurato per comunicare con il cluster Amazon EKS.
- OpenSSL `1.1.1` o versioni successive deve essere installato sul dispositivo.

Implementazione di Vertical Pod Autoscaler

In questa sezione, verrà implementato Vertical Pod Autoscaler nel cluster.

1. Aprire una finestra del terminale e passare a una directory in cui si desidera scaricare il codice sorgente di Vertical Pod Autoscaler.
2. Clona il repository GitHub kubernetes/autoscaler.

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. Passare alla directory `vertical-pod-autoscaler`.

```
cd autoscaler/vertical-pod-autoscaler/
```

4. (Facoltativo) Se è già stata implementata un'altra versione di Vertical Pod Autoscaler, rimuoverla con il comando seguente.

```
./hack/vpa-down.sh
```

5. Se i nodi non hanno accesso Internet al registro del container `registry.k8s.io`, devi estrarre le seguenti immagini e inviarle al tuo repository privato. Per ulteriori informazioni su come estrarre e inviare immagini al tuo repository privato, consulta la sezione the section called "Copia di un'immagine in un repository".

```
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 invii le immagini a un repository Amazon ECR privato, sostituisci `registry.k8s.io` nei manifesti con il tuo registro. Sostituire *111122223333* con l'ID account. Sostituisci *region-code* con la regione AWS in cui si trova il cluster. I comandi seguenti presuppongono che tu abbia dato al repository lo stesso nome del repository nel manifesto. Se hai dato un nome diverso al repository, dovrai modificare anch'esso.

```
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. Implementare Vertical Pod Autoscaler nel cluster mediante il comando seguente.

```
./hack/vpa-up.sh
```

7. Verifica che i pod di Vertical Pod Autoscaler siano stati creati correttamente.

```
kubectl get pods -n kube-system
```

Di seguito viene riportato un output di esempio:

```
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
```

Test dell'installazione di Vertical Pod Autoscaler

In questa sezione, verrà implementata un'applicazione di esempio per verificare il corretto funzionamento di Vertical Pod Autoscaler.

1. Implementare l'applicazione di esempio `hamster.yaml` per Vertical Pod Autoscaler mediante il comando seguente.

```
kubectl apply -f examples/hamster.yaml
```

2. Recupera i pod dall'applicazione esemplificativa `hamster`.

```
kubectl get pods -l app=hamster
```

Di seguito viene riportato un output di esempio:

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. Descrivi uno dei pod per visualizzare la prenotazione di `cpu` e `memory`. Sostituisci *`c7d89d6db-rglf5`* con uno degli ID restituiti nell'output del passaggio precedente.

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

Di seguito viene riportato un output di esempio:

```
[...]
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
```

```
[...]
```

Nota che il pod originale riserva 100 millicpu di CPU e 50 mebibyte di memoria. Per questa applicazione esemplificativa, 100 millicpu è un valore inferiore a quanto il pod deve eseguire, quindi risulta limitato dalla CPU. Il pod riserva inoltre una quantità di memoria nettamente inferiore rispetto alla necessaria. L'implementazione di Vertical Pod Autoscaler `vpa-recommender` analizza i pod hamster per verificare se i requisiti di CPU e memoria siano appropriati. Se sono necessari adeguamenti, `vpa-updater` rilancia i pod con valori aggiornati.

4. Attendi che `vpa-updater` avvii un nuovo pod hamster. Potrebbero essere necessari uno o due minuti. È possibile monitorare i pod con il comando seguente.

> **Note**
>
> Se non si è sicuri che sia stato avviato un nuovo pod, confronta i nomi dei pod facendo riferimento all'elenco precedente. All'avvio del nuovo pod, sarà visualizzato il relativo nome.

```
kubectl get --watch Pods -l app=hamster
```

5. Quando è avviato un nuovo pod hamster, aggiungi una descrizione e visualizza le prenotazioni di CPU e memoria aggiornate.

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

Di seguito viene riportato un output di esempio:

```
[...]
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
[...]
```

Nell'output precedente puoi osservare che la prenotazione della `cpu` è passata a 587 millicpu, ossia un valore di oltre cinque volte superiore a quello originale. La `memory` è aumentata a 262.144 KB (circa 250 mebibyte), ossia cinque volte il valore originale. Questo pod disponeva di un numero insufficiente di risorse e pertanto Vertical Pod Autoscaler ha corretto la stima con un valore più adeguato.

6. Descrivere la risorsa `hamster-vpa` per visualizzare la nuova raccomandazione.

```
kubectl describe vpa/hamster-vpa
```

Di seguito viene riportato un output di esempio:

```
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. Dopo aver provato l'applicazione di esempio, è possibile eliminarla con il comando seguente.

```
kubectl delete -f examples/hamster.yaml
```

Scalare le implementazioni dei pod con Horizontal Pod Autoscaler

Horizontal Pod Autoscaler di Kubernetes scala automaticamente il numero di pod in un'implementazione, un controller di replica o un set di repliche in base all'utilizzo della CPU di tale risorsa. Ciò consente alle applicazioni di aumentare orizzontalmente le risorse in base all'aumento della domanda, o di ridurle orizzontalmente quando non sono necessarie, liberando così i nodi per altre applicazioni. Quando imposti una percentuale di utilizzo della CPU target, Horizontal Pod Autoscaler aumenta o riduce le risorse dell'applicazione in modo conforme all'obiettivo prefissato.

Horizontal Pod Autoscaler è una risorsa API standard in Kubernetes il cui funzionamento richiede solo l'installazione di una fonte di parametri, (ad esempio Kubernetes Metrics Server), nel cluster Amazon EKS. Non è necessario implementare o installare Horizontal Pod Autoscaler sul cluster per

iniziare a dimensionare le applicazioni. Per ulteriori informazioni, consulta Horizontal Pod Autoscaler nella documentazione Kubernetes.

Utilizza questo argomento per preparare Horizontal Pod Autoscaler per il cluster Amazon EKS e per verificare che funzioni con un'applicazione di esempio.

> **Note**
>
> Questo argomento si basa su Horizontal Pod autoscaler walkthrough nella documentazione Kubernetes.

- Il cluster Amazon EKS è già esistente. Se non lo è, consulta Nozioni di base.
- Kubernetes Metrics Server è già installato. Per ulteriori informazioni, consulta the section called "Metrics Server".
- Stai utilizzando un client `kubectl` che è configurato per comunicare con il cluster Amazon EKS.

Esecuzione di un'applicazione di test di Horizontal Pod Autoscaler

In questa sezione implementerai un'applicazione di esempio per verificare il corretto funzionamento di Horizontal Pod Autoscaler.

> **Note**
>
> Questo esempio si basa su Horizontal Pod autoscaler walkthrough nella documentazione Kubernetes.

1. Implementa un'applicazione server Web Apache con il comando seguente.

   ```
   kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
   ```

   Questo pod di server Web Apache ha un limite di CPU di 500 millicpu e serve sulla porta 80.
2. Crea una risorsa Horizontal Pod Autoscaler per l'implementazione di `php-apache`.

   ```
   kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
   ```

Questo comando crea un componente di scalabilità automatica che punta a un utilizzo della CPU del 50% per l'implementazione, con un minimo di un pod e un massimo di dieci pod. Quando il carico medio della CPU è inferiore al 50%, tale componente cerca di ridurre il numero di pod nell'implementazione, fino a un minimo di uno. Quando il carico è maggiore del 50%, il componente di scalabilità automatica cerca di aumentare il numero di pod nell'implementazione, fino a un massimo di dieci. Per ulteriori informazioni, consulta How does a HorizontalPodAutoscaler work? nella documentazione Kubernetes.

3. Descrivere il componente di scalabilità automatica con il comando seguente per visualizzarne i dettagli.

```
kubectl get hpa
```

Di seguito viene riportato un output di esempio:

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   0%/50%    1         10        1          51s
```

Come si può vedere, il carico attuale della CPU è `0%` perché non è ancora presente il carico sul server. Il conteggio dei pod è già al limite più basso (uno), quindi non può essere ridotto.

4. Crea un carico per il server Web eseguendo un container.

```
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. Per osservare l'incremento dell'implementazione, esegui periodicamente il seguente comando in un terminale separato dal terminale in cui hai eseguito il passaggio precedente.

```
kubectl get hpa php-apache
```

Di seguito viene riportato un output di esempio:

```
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   250%/50%   1         10        5          4m44s
```

Potrebbe essere necessario più di un minuto per aumentare il numero di repliche. Finché la percentuale effettiva della CPU è superiore alla percentuale di destinazione, il conteggio delle repliche aumenta fino a 10. In questo caso, è `250%`, quindi il numero di `REPLICAS` continua ad aumentare.

> **Note**
>
> Potrebbero essere necessari alcuni minuti prima che il numero di repliche raggiunga il limite massimo. Se, ad esempio, sono necessarie solo 6 repliche affinché il carico della CPU rimanga pari o inferiore al 50%, il carico non verrà dimensionato oltre 6 repliche.

6. Arresta il carico. Nella finestra del terminale in cui si sta generando il carico, arrestare quest'ultimo tenendo premuti i tasti `Ctrl+C`. È possibile guardare il dimensionamento delle repliche nuovamente a 1 eseguendo di nuovo il seguente comando nel terminale corrispondente.

```
kubectl get hpa
```

Di seguito viene riportato un output di esempio:

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    0%/50%    1         10        1          25m
```

> **Note**
>
> Il periodo di tempo di default per il ridimensionamento è di cinque minuti, quindi ci vorrà un po' di tempo prima di vedere il conteggio delle repliche raggiungere di nuovo 1, anche quando la percentuale corrente della CPU è 0. L'intervallo di tempo è modificabile. Per ulteriori informazioni, consulta Horizontal Pod Autoscaler nella documentazione Kubernetes.

7. Al termine dell'esercitazione con l'applicazione di esempio, elimina le risorse `php-apache`.

```
kubectl delete deployment.apps/php-apache service/php-apache
 horizontalpodautoscaler.autoscaling/php-apache
```

Esecuzione del routing del traffico TCP e UDP con Network Load Balancer

> **Note**
>
> Novità: la modalità automatica di Amazon EKS automatizza le attività di routine per il bilanciamento del carico. Per ulteriori informazioni, consulta:
>
> - the section called "Implementa il bilanciatore del carico"
> - the section called "Creare servizio"

Il traffico di rete è bilanciato al carico L4 del modello OSI. Per bilanciare il carico del traffico delle applicazioniL7, distribuisci un Kubernetes`ingress`, che fornisce un Application Load AWS Balancer. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di applicazione". Per ulteriori informazioni sulle differenze tra i due tipi di bilanciamento del carico, consulta le funzionalità di Elastic Load Balancing sul AWS sito Web.

Quando crei un Kubernetes `Service` di tipo Kubernetes`LoadBalancer`, il controller di bilanciamento del carico del provider AWS cloud crea AWSClassic Load Balancer per impostazione predefinita, ma può anche creare Network Load Balancer. AWS Questo controller in futuro riceverà solo correzioni di bug critici. Per ulteriori informazioni sull'utilizzo del load balancer del provider AWS cloud, consulta il controller di bilanciamento del carico del provider AWS cloud nella documentazione di Kubernetes. Il suo utilizzo non è descritto in questo argomento.

Ti consigliamo di utilizzare una versione `2.7.2` o successiva del AWS Load Balancer Controller anziché il controller di bilanciamento del carico del provider AWS cloud. Il AWS Load Balancer Controller crea AWS Network Load Balancer, ma non crea AWS Classic Load Balancer. Il resto di questo argomento riguarda l'utilizzo del AWS Load Balancer Controller.

Un AWS Network Load Balancer può bilanciare il carico del traffico di rete su Pods distribuiti su destinazioni IP e istanze EC2 Amazon, su destinazioni IP AWS Fargate o su nodi ibridi Amazon EKS come destinazioni IP. Per ulteriori informazioni, vedere AWS Load Balancer Controller su. GitHub

Prerequisiti

Prima di poter bilanciare il carico del traffico di rete utilizzando il AWS Load Balancer Controller, è necessario soddisfare i seguenti requisiti.

- Avere un cluster esistente. Se non si dispone di un cluster esistente, consultare Nozioni di base. Se è necessario aggiornare la versione di un cluster esistente, vedere the section called "Aggiornamento della versione di Kubernetes".
- Implementa il AWS Load Balancer Controller sul tuo cluster. Per ulteriori informazioni, consulta the section called " AWS Controller Load Balancer". Consigliamo la versione `2.7.2` o successiva.
- Disporre di almeno una sottorete. Se si trovano più sottoreti con tag in una zona di disponibilità, il controller sceglie la sottorete il cui ID sottorete viene prima in ordine lessicografico. La sottorete deve avere a disposizione almeno otto indirizzi IP.
- Se utilizzi la versione AWS Load Balancer Controller `2.1.1` o precedente, le sottoreti devono essere etichettate come segue. Se si utilizza la versione `2.1.2` o successiva, questo tag è facoltativo. Potresti voler etichettare una sottorete se hai più cluster in esecuzione nello stesso VPC o più AWS servizi che condividono sottoreti in un VPC e desideri un maggiore controllo su dove vengono forniti i load balancer per ogni cluster. Se specifichi esplicitamente la sottorete IDs come annotazione su un oggetto servizio, Kubernetes e il Load AWS Balancer Controller utilizzano tali sottoreti direttamente per creare il load balancer. L'assegnazione di tag delle sottoreti non è obbligatoria se si sceglie di utilizzare questo metodo per il provisioning dei bilanciatori del carico ed è possibile ignorare i seguenti requisiti di assegnazione di tag delle sottoreti pubbliche e private. Sostituisci *`my-cluster`* con il nome del cluster.
  - Chiave: `kubernetes.io/cluster/<my-cluster>`
  - Valore: `shared` o `owned`
- Le sottoreti pubbliche e private devono soddisfare i seguenti requisiti, a meno che non specifichi esplicitamente subnet come annotazione su un servizio o un oggetto di ingresso. IDs Se esegui il provisioning dei load balancer specificando esplicitamente la sottorete IDs come annotazione su un servizio o un oggetto di ingresso, Kubernetes e il Load Balancer Controller utilizzano tali sottoreti direttamente per creare il AWS load balancer e i seguenti tag non sono necessari.
  - Sottoreti private: deve essere taggato nel seguente formato. In questo modo Kubernetes e Load Balancer Controller sanno che le sottoreti possono essere utilizzate per bilanciatori di AWS carico interni. Se utilizzi `eksctl` o un AWS AWS CloudFormation modello Amazon EKS per creare il tuo VPC dopo il 26 marzo 2020, le sottoreti vengono etichettate in modo appropriato al momento della creazione. Per ulteriori informazioni sui modelli AWS AWS CloudFormation VPC Amazon EKS, vedere the section called "Crea un VPC".
    - Chiave: `kubernetes.io/role/internal-elb`
    - Valore: 1

- Sottoreti pubbliche: deve essere taggato nel seguente formato. In questo modo Kubernetes sa come utilizzare solo quelle sottoreti per bilanciatori di carico esterni invece di scegliere una sottorete pubblica in ogni zona di disponibilità (in base all'ordine lessicografico della sottorete). IDs Se utilizzi `eksctl` o un AWS CloudFormation modello Amazon EKS per creare il tuo VPC dopo il 26 marzo 2020, le sottoreti vengono etichettate in modo appropriato al momento della creazione. Per ulteriori informazioni sui modelli AWS CloudFormation VPC di Amazon EKS, consulta. the section called "Crea un VPC"
  - Chiave: `kubernetes.io/role/elb`
  - Valore: 1

Se i tag del ruolo della sottorete non vengono aggiunti in modo esplicito, il controller del servizio Kubernetes esamina la tabella di routing delle sottoreti VPC del cluster per determinare se la sottorete è privata o pubblica. Si consiglia di non fare affidamento su questo comportamento e di aggiungere esplicitamente i tag di ruolo pubblici o privati. Il AWS Load Balancer Controller non esamina le tabelle delle rotte e richiede la presenza dei tag privati e pubblici per una corretta individuazione automatica.

Considerazioni

- La configurazione del load balancer è controllata da annotazioni che vengono aggiunte al manifest del servizio. Le annotazioni del servizio sono diverse quando si utilizza il AWS Load Balancer Controller rispetto a quando si utilizza AWS il controller di bilanciamento del carico del provider cloud. Assicurati di controllare le annotazioni per il AWS Load Balancer Controller prima di distribuire i servizi.
- Quando si utilizza il plug-in Amazon VPC CNI per Kubernetes, il Load AWS Balancer Controller può bilanciare il carico su destinazioni IP o istanze EC2 Amazon e destinazioni IP Fargate. Quando utilizzi plug-in CNI compatibili alternativi, il controller può bilanciare il carico solo per le destinazioni di istanza, a meno che non si stia bilanciando il carico su Amazon EKS Hybrid Nodes. Per i nodi ibridi, il controller può bilanciare il carico delle destinazioni IP. Per ulteriori informazioni sui tipi di destinazioni dei Network Load Balancer, consultare Target type (Tipo di destinazione) nella Guida per l'utente di Network Load Balancer.
- Se si desidera aggiungere tag al bilanciatore del carico quando o dopo la sua creazione, aggiungere la seguente annotazione nella specifica del servizio. Per ulteriori informazioni, consulta AWS Resource Tags nella documentazione del AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
```

- È possibile assegnare Indirizzi IP elastici al Network Load Balancer aggiungendo la seguente annotazione. Sostituisci i valori `Allocation IDs` di esempio con i tuoi indirizzi IP elastici. Il numero di `Allocation IDs` deve corrispondere al numero di sottoreti utilizzate per il load balancer. Per ulteriori informazioni, consultare la documentazione sul Controller del load balancer AWS.

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations: eipalloc-
xxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyy
```

- Amazon EKS aggiunge una regola in entrata al gruppo di sicurezza del nodo per il traffico client e una regola per ogni sottorete del bilanciatore del carico nel VPC per i controlli dell'integrità per ogni Network Load Balancer creato. L'implementazione di un servizio di tipo `LoadBalancer` può non riuscire se Amazon EKS tenta di creare regole che superano la quota per il numero massimo di regole consentite per un gruppo di sicurezza. Per ulteriori informazioni, consultare Gruppi di sicurezza in Quote di Amazon VPC nella Guida per l'utente di Amazon VPC. Considerare le opzioni seguenti per ridurre al minimo le possibilità di superare il numero massimo di regole per un gruppo di sicurezza:
  - Richiedere un aumento delle regole per ogni quota di gruppo di sicurezza. Per ulteriori informazioni, consultare Richiesta di un aumento di quota nella Guida per l'utente di Service Quotas.
  - Al posto di destinazioni di istanza, utilizzare destinazioni IP. Con le destinazioni IP, è possibile condividere le regole per le stesse porte di destinazione. È possibile specificare manualmente le sottoreti del load balancer con un'annotazione. Per ulteriori informazioni, consulta Annotations on GitHub.
  - Utilizzare un ingresso invece di un servizio di tipo `LoadBalancer` per inviare traffico al tuo servizio. L' AWS Application Load Balancer richiede meno regole rispetto ai Network Load Balancer. È possibile condividere un ALB su più ingressi. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di applicazione". Non è possibile condividere un Network Load Balancer su più servizi.
  - Implementa i cluster su più account.
- Se i pod vengono eseguiti su Windows in un cluster Amazon EKS, un singolo servizio con un bilanciatore del carico può supportare fino a 1024 pod back-end. Ogni pod ha il proprio indirizzo IP univoco.

- Consigliamo di creare nuovi Network Load Balancer solo con il Load AWS Balancer Controller. Il tentativo di sostituire i Network Load Balancer esistenti creati con il controller di bilanciamento del carico del provider di servizi AWS cloud può comportare la creazione di più Network Load Balancer che potrebbero causare tempi di inattività delle applicazioni.

Creazione di un Network Load Balancer

È possibile creare un Network Load Balancer con destinazioni IP o istanza.

Creare un Network Load Balancer: destinazioni IP

- Puoi utilizzare destinazioni IP con pod distribuiti su EC2 nodi Amazon, Fargate o nodi ibridi Amazon EKS. Il servizio Kubernetes deve essere creato come tipo `LoadBalancer`. Per ulteriori informazioni, consulta Type LoadBalancer nella documentazione di Kubernetes.

  Per creare un load balancer che utilizzi destinazioni IP, aggiungere le seguenti annotazioni a un manifesto del servizio e implementare il servizio. Il `external` valore per `aws-load-balancer-type` è ciò che fa sì che il AWS Load Balancer Controller, anziché il controller del load balancer del provider AWS cloud, crei il Network Load Balancer. É possibile visualizzare un manifesto del servizio di esempio con le annotazioni.

  ```
  service.beta.kubernetes.io/aws-load-balancer-type: "external"
  service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
  ```

  > **Note**
  >
  > Se si sta eseguendo il bilanciamento del carico sui pod `IPv6`, aggiungere la seguente annotazione. È possibile bilanciare il carico solo su destinazioni da `IPv6` a IP, non su destinazioni di istanza. Senza questa annotazione, il bilanciamento del carico è su `IPv4`.

  ```
  service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
  ```

  Per impostazione predefinita, i Network Load Balancer vengono creati con `internal` `aws-load-balancer-scheme`. È possibile avviare i Network Load Balancer in qualsiasi sottorete del VPC del cluster, incluse le sottoreti non specificate al momento della creazione del cluster.

Kubernetes esamina la tabella di routing delle sottoreti per stabilire se sono pubbliche o private. Le sottoreti pubbliche hanno una route direttamente a Internet tramite un gateway Internet; non vale altrettanto per le sottoreti private.

Se desideri creare un Network Load Balancer in una sottorete pubblica per bilanciare il carico sui nodi Amazon EC2 (Fargate può essere solo privato), `internet-facing` specificalo con la seguente annotazione:

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Note**
>
> Al fine di garantire la compatibilità con le versioni precedenti, l'annotazione `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` è ancora supportata. Consigliamo, tuttavia, di utilizzare le annotazioni precedenti per i load balancer anziché `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"`.

> **Important**
>
> Non modificare le annotazioni dopo aver creato il servizio. Se è necessario modificarlo, eliminare l'oggetto di servizio e crearlo nuovamente con il valore desiderato per questa annotazione.

Creare un Network Load Balancer: destinazioni di istanza

- Il controller di bilanciamento del carico del provider AWS cloud crea Network Load Balancer solo con destinazioni di istanza. La versione `2.2.0` e le successive del AWS Load Balancer Controller creano anche Network Load Balancer con destinazioni di istanza. Ti consigliamo di utilizzarlo, anziché il controller di bilanciamento del carico del provider AWS cloud, per creare nuovi Network Load Balancer. Puoi utilizzare gli obiettivi delle istanze Network Load Balancer con i Pod distribuiti sui nodi EC2 Amazon, ma non su Fargate. Per bilanciare il carico del traffico di rete tra i pod implementati su Fargate, è necessario utilizzare destinazioni IP.

  Per implementare un Network Load Balancer in una sottorete privata, la specifica del servizio deve avere le seguenti annotazioni. É possibile visualizzare un manifesto del servizio di esempio con le

annotazioni. Il `external` valore per `aws-load-balancer-type` è ciò che fa sì che il AWS Load Balancer Controller, anziché il controller del load balancer del provider AWS cloud, crei il Network Load Balancer.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
```

Per impostazione predefinita, i Network Load Balancer vengono creati con `internal` `aws-load-balancer-scheme`. Per i Network Load Balancer interni, il cluster Amazon EKS deve essere configurato per l'utilizzo di almeno una sottorete privata nel VPC. Kubernetes esamina la tabella di routing delle sottoreti per stabilire se sono pubbliche o private. Le sottoreti pubbliche hanno una route direttamente a Internet tramite un gateway Internet; non vale altrettanto per le sottoreti private.

Se desideri creare un Network Load Balancer in una sottorete pubblica per bilanciare il carico EC2 sui nodi Amazon, specificalo `internet-facing` con la seguente annotazione:

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Important**
>
> Non modificare le annotazioni dopo aver creato il servizio. Se è necessario modificarlo, eliminare l'oggetto di servizio e crearlo nuovamente con il valore desiderato per questa annotazione.

(Facoltativo) Implementare un'applicazione di esempio

- Almeno una sottorete pubblica o privata nel VPC del cluster.
- Implementa il AWS Load Balancer Controller sul tuo cluster. Per ulteriori informazioni, consulta the section called " AWS Controller Load Balancer". Consigliamo la versione `2.7.2` o successiva.
  1. Se si sta eseguendo l'implementazione su Fargate, assicurarsi di avere una sottorete privata disponibile nel VPC e di creare un profilo Fargate. Se non si sta eseguendo l'implementazione su Fargate, salta questo passaggio. Puoi creare il profilo eseguendo il seguente comando o nella Console di gestione AWS utilizzando gli stessi valori per `name` e `namespace` che si trovano nel comando. Sostituire i valori di esempio con i propri valori.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name nlb-sample-app \
    --namespace nlb-sample-app
```

2. Implementa un'applicazione di esempio.

   a. Crea un namespace per l'applicazione.

   ```
   kubectl create namespace nlb-sample-app
   ```

   b. Salva nel computer i seguenti contenuti in un file denominato `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. Applica il file manifesto al cluster.

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

3. Crea un servizio con un Network Load Balancer connesso a Internet che bilanci il carico sulle destinazioni IP.

a. Salva sul computer i seguenti contenuti in un file denominato `sample-service.yaml`. Se si sta eseguendo l'implementazione sui nodi Fargate, rimuovere la riga `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. Applica il file manifesto al cluster.

```
kubectl apply -f sample-service.yaml
```

4. Verifica che il servizio sia stato distribuito.

```
kubectl get svc nlb-sample-service -n nlb-sample-app
```

Di seguito viene riportato un output di esempio.

```
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**
>
> I valori di *10.100.240.137* e *xxxxxxxxxx* - *xxxxxxxxxxxxxxxx* saranno diversi dall'output di esempio (saranno univoci per il sistema di bilanciamento del carico) e *us-*

west-2 potrebbero essere diversi per te, a seconda della AWS regione in cui si trova il cluster.

5. Apri Amazon EC2 Console di gestione AWS. Seleziona Target Groups (Gruppi di destinazioni) in Load Balancing (Bilanciamento del carico) nel pannello di navigazione a sinistra. Nella colonna Nome, seleziona il nome del gruppo di destinazione in cui il valore nella colonna bilanciatore del carico corrisponde al nome nella colonna `EXTERNAL-IP` dell'output nel passaggio precedente. Ad esempio, se l'output era uguale all'output indicato sopra sarà necessario selezionare il gruppo di destinazione denominato `k8s-default-samplese-`*`xxxxxxxxxx`* . Target type (Tipo di destinazione) è `IP` perché è stato specificato nel manifesto del servizio di esempio.
6. Selezionare il proprio Gruppo di destinazione e poi scegliere la scheda Destinazioni. In Destinazioni registrate, verranno visualizzati tre indirizzi IP delle tre repliche implementate in un passaggio precedente. Prima di continuare, attendere fino a quando lo stato di tutti gli obiettivi è Integro. Potrebbero passare diversi minuti prima che tutti gli obiettivi siano `healthy`. Gli obiettivi potrebbero essere in uno stato `unhealthy` prima di passare a uno stato `healthy`.
7. Invia il traffico al servizio sostituendo *xxxxxxxxxx-xxxxxxxxxxxxxxxx* e *us-west-2* con i valori restituiti nell'output di un operazione precedente per `EXTERNAL-IP`. Se hai eseguito l'implementazione a una sottorete privata, devi visualizzare la pagina da un dispositivo all'interno del VPC, ad esempio un host bastione. Per ulteriori informazioni, consulta Bastion host Linux in AWS.

   ```
   curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
   ```

   Di seguito viene riportato un output di esempio.

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```
8. Al termine dell'implementazione, del servizio e del namespace esemplificativi, rimuoverli.

   ```
   kubectl delete namespace nlb-sample-app
   ```

Instradare il traffico di applicazioni e HTTP con Application Load Balancer

> **Note**
>
> Novità: la modalità automatica di Amazon EKS automatizza le attività di routine per il bilanciamento del carico. Per ulteriori informazioni, consulta:
>
> - the section called "Implementa il bilanciatore del carico"
> - the section called "Creare una classe in ingresso"

Quando crei un Kubernetes`ingress`, viene fornito un Application Load AWS Balancer (ALB) che bilancia il carico del traffico delle applicazioni. Per ulteriori informazioni, consulta Cos'è un Application Load Balancer? nella Guida per l'utente di Application Load Balancers e Ingress nella documentazione di Kubernetes. ALBs può essere utilizzato con i Pod distribuiti sui nodi o su AWS Fargate. È possibile implementare un ALB a sottoreti pubbliche o private.

Il traffico delle applicazioni è bilanciato a `L7` del modello OSI. Per bilanciare il carico del traffico di rete a `L4`, implementare un `service` Kubernetes del tipo `LoadBalancer`. Questo tipo fornisce un AWS Network Load Balancer. Per ulteriori informazioni, consulta the section called "Bilanciamento del carico di rete". Per ulteriori informazioni sulle differenze tra i due tipi di bilanciamento del carico, consulta le funzionalità di Elastic Load Balancing sul AWS sito Web.

Prerequisiti

Prima di poter bilanciare il carico del traffico dell'applicazione in un'applicazione, è necessario soddisfare i seguenti requisiti.

- Avere un cluster esistente. Se non si dispone di un cluster esistente, consultare Nozioni di base. Se è necessario aggiornare la versione di un cluster esistente, vedere the section called "Aggiornamento della versione di Kubernetes".
- Implementa il AWS Load Balancer Controller sul tuo cluster. Per ulteriori informazioni, consulta the section called " AWS Controller Load Balancer". Consigliamo la versione `2.7.2` o successiva.
- Almeno due sottoreti devono trovarsi in diverse zone di disponibilità. Il AWS Load Balancer Controller sceglie una sottorete da ogni zona di disponibilità. Quando si trovano più sottoreti

taggate in una zona di disponibilità, il controller sceglie la sottorete il cui ID sottorete viene prima in ordine lessicografico. Ogni sottorete deve avere a disposizione almeno otto indirizzi IP.

Se utilizzi più gruppi di sicurezza collegati al nodo (worker), esattamente un gruppo di sicurezza deve essere taggato come segue. Sostituisci *my-cluster* con il nome del cluster.

- Chiave: `kubernetes.io/cluster/<my-cluster>`
- Valore: `shared` o `owned`

- Se utilizzi la versione AWS Load Balancer Controller `2.1.1` o precedente, le sottoreti devono essere etichettate nel formato seguente. Se utilizzi la versione `2.1.2` o successiva, l'assegnazione di tag è facoltativa. Tuttavia, si consiglia di taggare una sottorete nel caso si verifichi una delle seguenti situazioni. Hai più cluster in esecuzione nello stesso VPC o AWS più servizi che condividono sottoreti in un VPC. Oppure, si desidera un maggiore controllo sulla posizione in cui viene eseguito il provisioning dei bilanciatori di carico per ciascun cluster. Sostituisci *my-cluster* con il nome del cluster.
  - Chiave: `kubernetes.io/cluster/<my-cluster>`
  - Valore – `shared` o `owned`
- Le sottoreti pubbliche e private devono soddisfare i seguenti requisiti. Questo a meno che non si specifichi esplicitamente subnet IDs come annotazione su un servizio o un oggetto in ingresso. Si supponga di effettuare il provisioning dei sistemi di bilanciamento del carico specificando esplicitamente la sottorete IDs come annotazione su un servizio o un oggetto in ingresso. In questa situazione, Kubernetes e il controller del AWS bilanciamento del carico utilizzano direttamente tali sottoreti per creare il sistema di bilanciamento del carico e i seguenti tag non sono necessari.
  - Sottoreti private: deve essere taggato nel seguente formato. In questo modo Kubernetes e il controller del bilanciamento del carico sanno che le sottoreti possono essere utilizzate per bilanciatori di AWS carico interni. Se utilizzi `eksctl` o un AWS CloudFormation modello Amazon EKS per creare il tuo VPC dopo il 26 marzo 2020, le sottoreti vengono etichettate in modo appropriato al momento della creazione. Per ulteriori informazioni sui modelli AWS CloudFormation VPC di Amazon EKS, consulta. the section called "Crea un VPC"
    - Chiave: `kubernetes.io/role/internal-elb`
    - Valore: 1
  - Sottoreti pubbliche – Deve essere taggato nel seguente formato. In questo modo Kubernetes sa di utilizzare solo le sottoreti specificate per i load balancer esterni. In questo modo, Kubernetes non sceglie una sottorete pubblica in ogni zona di disponibilità (in base all'ordine lessicografico degli ID sottorete). Se utilizzi `eksctl` o un AWS CloudFormation modello Amazon EKS per creare il tuo VPC dopo il 26 marzo 2020, le sottoreti vengono etichettate in modo appropriato

al momento della creazione. Per ulteriori informazioni sui modelli AWS CloudFormation VPC di Amazon EKS, consulta. the section called "Crea un VPC"

- Chiave: `kubernetes.io/role/elb`
- Valore: 1

Se i tag del ruolo della sottorete non vengono aggiunti in modo esplicito, il controller del servizio Kubernetes esamina la tabella di routing delle sottoreti VPC del cluster. Ciò consente di determinare se la sottorete è privata o pubblica. Si consiglia di non fare affidamento su questo comportamento. È preferibile, invece, aggiungere esplicitamente i tag del ruolo privato o pubblico. Il AWS Load Balancer Controller non esamina le tabelle delle rotte. Richiede inoltre la presenza di tag privati e pubblici per un efficiente rilevamento automatico.

- Il AWS Load Balancer Controller crea ALBs le AWS risorse di supporto necessarie ogni volta che una risorsa di ingresso Kubernetes viene creata sul cluster con l'annotazione. `kubernetes.io/ingress.class: alb` La risorsa di ingresso configura l'ALB per instradare il traffico HTTP o HTTPS verso pod differenti all'interno del cluster. Per assicurarti che i tuoi oggetti in ingresso utilizzino il AWS Load Balancer Controller, aggiungi la seguente annotazione alla specifica di ingresso di Kubernetes. Per ulteriori informazioni, consulta Specifiche di ingresso su GitHub.

```
annotations:
    kubernetes.io/ingress.class: alb
```

> **Note**
>
> Se bilanci il carico sui pod `IPv6`, aggiungi la seguente annotazione alla specifica di ingresso. È possibile bilanciare il carico solo su destinazioni da `IPv6` a IP, non su destinazioni di istanza. Senza questa annotazione, il bilanciamento del carico è su `IPv4`.

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

- Il AWS Load Balancer Controller supporta le seguenti modalità di traffico:
  - Istanza: registra i nodi all'interno del cluster come destinazioni per l'ALB. Il traffico che raggiunge l'ALB viene indirizzato a `NodePort` per il servizio e quindi inoltrato ai pod. Questa è la modalità di traffico predefinita. È anche possibile specificarla esplicitamente con l'annotazione `alb.ingress.kubernetes.io/target-type: instance`.

> **Note**
>
> Il servizio Kubernetes deve specificare il tipo `NodePort` o `LoadBalancer` per utilizzare questa modalità di traffico.

  - IP – Registra i pod come destinazioni per l'ALB. Il traffico che raggiunge l'ALB viene indirizzato direttamente ai pod per il servizio. È necessario specificare l'annotazione `alb.ingress.kubernetes.io/target-type: ip` per utilizzare questa modalità di traffico. Il tipo di destinazione IP è richiesto quando i pod di destinazione sono in esecuzione su Fargate o Amazon EKS Hybrid Nodes.
- Per etichettare ALBs creato dal controller, aggiungi la seguente annotazione al controller:. `alb.ingress.kubernetes.io/tags` Per un elenco di tutte le annotazioni disponibili supportate dal Load AWS Balancer Controller, consulta Ingress annotations on. GitHub
- L'aggiornamento o il downgrade della versione del controller ALB può introdurre modifiche importanti alle funzionalità che si basano su di essa. Per ulteriori informazioni sulle modifiche che vengono introdotte in ogni versione, consulta le note di rilascio del controller ALB su GitHub.

Riutilizzo con Ingress Groups ALBs

Puoi condividere un Application Load Balancer su più risorse di servizio tramite `IngressGroups`.

Per aggiungere un ingresso a un gruppo, aggiungere la seguente annotazione a una specifica di risorsa di ingresso Kubernetes.

```
alb.ingress.kubernetes.io/group.name: my-group
```

Il nome del gruppo deve:

- Avere una lunghezza pari o inferiore a 63 caratteri.
- Essere formato da lettere minuscole, numeri, `-`, e `.`
- Iniziare e finire con una lettera o un numero.

Il controller unisce automaticamente le regole di ingresso per tutti gli ingressi nello stesso gruppo di ingresso. Le supporta con un singolo ALB. La maggior parte delle annotazioni definite in un ingresso si applica solo ai percorsi definiti da tale ingresso. Per impostazione predefinita, le risorse in ingresso non appartengono ad alcun gruppo di ingresso.

> **Warning**
>
> Potenziale rischio per la sicurezza
> Specifica un gruppo di ingressi per un ingresso solo quando tutti gli utenti Kubernetes che dispongono dell'autorizzazione RBAC per creare o modificare le risorse di ingresso rientrano nello stesso limite di attendibilità. Se si aggiunge l'annotazione con un nome di gruppo, altri utenti Kubernetes potrebbero creare o modificare i propri ingressi così da appartenere allo stesso gruppo di ingressi. Ciò può causare comportamenti indesiderati, ad esempio la sovrascrittura delle regole esistenti con regole con priorità più alta.

Puoi aggiungere un numero d'ordine della risorsa di ingresso.

```
alb.ingress.kubernetes.io/group.order: '10'
```

Il numero può variare tra 1 e 1.000. Viene considerato prima il numero più basso per tutti gli ingressi nello stesso gruppo di ingressi. Tutti gli ingressi senza questa annotazione vengono valutati con un valore pari a zero. La duplicazione delle regole con un numero più alto può causare la sovrascrittura delle regole con un numero più basso. Per impostazione predefinita, l'ordine delle regole tra ingressi all'interno dello stesso gruppo di ingressi viene determinato in base all'ordine lessicografico del namespace e del nome.

> **Important**
>
> Assicurarsi che ogni ingresso nello stesso gruppo di ingressi disponga di un numero di priorità univoco. Non puoi avere numeri d'ordine duplicati tra gli ingressi.

(Facoltativo) Implementare un'applicazione di esempio

- Almeno una sottorete pubblica o privata nel VPC del cluster.
- Implementa il AWS Load Balancer Controller sul tuo cluster. Per ulteriori informazioni, consulta the section called " AWS Controller Load Balancer". Consigliamo la versione `2.7.2` o successiva.

Puoi eseguire l'applicazione di esempio su un cluster con EC2 nodi Amazon, Fargate Pods o entrambi.

1. Se non stai eseguendo l'implementazione su Fargate, salta questa fase. Se stai eseguendo l'implementazione su Fargate, crea un profilo Fargate. Puoi creare il profilo eseguendo il seguente comando o nella Console di gestione AWS utilizzando gli stessi valori per `name` e `namespace` che si trovano nel comando. Sostituire i valori di esempio con i propri valori.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name alb-sample-app \
    --namespace game-2048
```

2. Implementa il gioco 2048 come applicazione di esempio per verificare che il Load AWS Balancer Controller crei AWS un ALB come risultato dell'oggetto di ingresso. Completa i passaggi per il tipo di sottorete verso la quale stai eseguendo l'implementazione.

   a. Se stai eseguendo l'implementazione su pod in un cluster creato con la famiglia `IPv6`, passa alla fase successiva.

   - Pubblica:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
```

   - Privata:

     A. Eseguire il download del manifesto.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
```

     B. Modificare il file e individuare la riga `alb.ingress.kubernetes.io/scheme: internet-facing`.

     C. Modificare *`internet-facing`* in `internal` e salvare il file.

     D. Applicare il file manifesto al cluster.

```
kubectl apply -f 2048_full.yaml
```

   b. Se lo stai distribuendo su Pods in un cluster creato con la famiglia, completa i IPv6 seguenti passaggi.

      i. Eseguire il download del manifesto.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.14.1/docs/examples/2048/2048_full.yaml
```

ii. Aprire il file in un editor e aggiungere la seguente riga alle annotazioni nella specifica di ingresso.

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

iii. Se stai bilanciando il carico sui pod interni piuttosto che sui pod rivolti a Internet, modifica la riga che indica `alb.ingress.kubernetes.io/scheme: internet-facing` in `alb.ingress.kubernetes.io/scheme: internal`

iv. Salvare il file.

v. Applica il file manifesto al cluster.

```
kubectl apply -f 2048_full.yaml
```

3. Dopo pochi minuti, verificare che la risorsa di ingresso sia stata creata con il comando seguente.

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

Di seguito viene riportato un output di esempio:

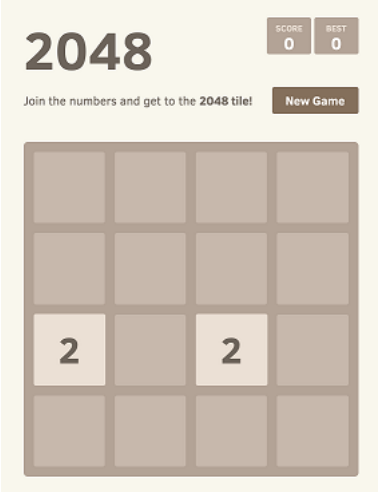
```
NAME           CLASS    HOSTS   ADDRESS
                     PORTS   AGE
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-
code.elb.amazonaws.com   80      2m32s
```

> **Note**
>
> Se hai creato il load balancer in una sottorete privata, il valore in `ADDRESS` nell'output precedente è preceduto da `internal-`.

Se l'ingresso non è stato creato correttamente dopo alcuni minuti, esegui il comando seguente per visualizzare i log del AWS Load Balancer Controller. Questi registri possono contenere messaggi di errore che consentono di diagnosticare eventuali problemi relativi all'implementazione.

```
kubectl logs -f -n kube-system -l app.kubernetes.io/instance=aws-load-balancer-
controller
```

1. Se è stata eseguita l'implementazione a una sottorete pubblica, aprire un browser e accedere all'URL ADDRESS dall'output del comando precedente per visualizzare l'applicazione di esempio. Se non visualizzi nulla, aggiorna il browser e riprova. Se hai eseguito l'implementazione a una sottorete privata, devi visualizzare la pagina da un dispositivo all'interno del VPC, ad esempio un host bastione. Per ulteriori informazioni, consulta Bastion host Linux in AWS.



2. Dopo aver provato l'applicazione di esempio, eliminarla utilizzando uno dei comandi seguenti.
   - Se è stato applicato il manifesto anziché una copia scaricata, utilizzare il comando seguente.

     ```
     kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-
     balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
     ```

   - Se è stato il manifesto scaricato e modificato, utilizzare il seguente comando.

     ```
     kubectl delete -f 2048_full.yaml
     ```

Limitazione degli indirizzi IP esterni assegnabili ai servizi

I servizi Kubernetes possono essere raggiunti dall'interno di un cluster tramite:

- Un indirizzo IP del cluster assegnato automaticamente da Kubernetes
- Qualsiasi indirizzo IP specificato per la proprietà `externalIPs` in una specifica di servizio. Gli indirizzi IP esterni non sono gestiti da Kubernetes e sono responsabilità dell'amministratore del

cluster. Gli Indirizzi IP esterni specificati con `externalIPs` sono diversi dall'indirizzo IP esterno assegnato a un servizio di tipo `LoadBalancer` da un provider cloud.

Per ulteriori informazioni sui servizi Kubernetes, consultare Servizio nella documentazione Kubernetes. È possibile limitare gli indirizzi IP che possono essere specificati per `externalIPs` in una specifica di servizio.

1. Implementazione di `cert-manager` per gestire i certificati webhook. Per ulteriori informazioni, consultare la documentazione Cert-Manager.

```
kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/
cert-manager.yaml
```

2. Verifica che i pod `cert-manager` siano in esecuzione.

```
kubectl get pods -n cert-manager
```

   Di seguito viene riportato un output di esempio:

```
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. Esamina i servizi esistenti per assicurarsi che a nessuno di essi siano assegnati indirizzi IP esterni non contenuti nell'intervallo CIDR per cui si desidera limitare gli indirizzi.

```
kubectl get services -A
```

   Di seguito viene riportato un output di esempio:

```
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 uno qualsiasi dei valori costituiscono indirizzi IP che non si trovano all'interno del blocco per cui si desidera limitare l'accesso, sarà necessario modificare gli indirizzi in modo che si trovino all'interno del blocco e implementare nuovamente i servizi. Ad esempio, al servizio `my-service` nell'output precedente è assegnato un indirizzo IP esterno che non rientra nell'intervallo CIDR esemplificativo del passaggio 5.

4. Eseguire il download del manifesto del webhook IP esterno. È anche possibile visualizzare il codice sorgente per il webhook su GitHub.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
```

5. Specifica i blocchi CIDR. Aprire il file scaricato nell'editor e rimuovere il \# all'inizio delle righe seguenti.

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

Sostituire `10.0.0.0/8` con il proprio blocco CIDR. É possibile specificare tutti i blocchi che si desiderano. Se si specificano blocchi multipli, aggiungere una virgola tra i blocchi.

6. Se il cluster non è nella regione AWS `us-west-2`, sostituisci `us-west-2`, `602401143452` e `amazonaws.com` nel file con i seguenti comandi. Prima di eseguire i comandi, sostituisci *`codice regione`* e *`111122223333`* con il valore per la tua regione AWS dall'elenco in Visualizza i registri delle immagini dei container Amazon per i componenti aggiuntivi 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. Applica il manifesto al cluster.

```
kubectl apply -f externalip-webhook.yaml
```

Qualsiasi tentativo di implementare un servizio nel cluster con un indirizzo IP specificato per `externalIPs` che non è contenuto nei blocchi specificati nella fase Specifica i blocchi CIDR avrà esito negativo.

Copia di un'immagine di container da un repository a un altro

In questo argomento viene descritto come estrarre un'immagine di container da un repository a cui i nodi non hanno accesso e inviarla a un repository a cui i nodi hanno accesso. Puoi inviare l'immagine ad Amazon ECR o a un repository alternativo al quale i tuoi nodi hanno accesso.

- Il motore Docker deve essere installato e configurato sul computer. Per ulteriori informazioni, consulta la sezione Installazione del motore Docker nella documentazione di Docker.
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Un endpoint VPC di interfaccia per Amazon ECR se si desidera che i nodi estraggano o trasferiscano le immagini di container in un repository Amazon ECR privato sulla rete di Amazon. Per ulteriori informazioni, consulta la sezione Creazione degli endpoint VPC per Amazon ECR nella Guida per l'utente di Amazon Elastic Container Registry.

Completa i passaggi seguenti per estrarre un'immagine del container da un repository e inviarla al tuo repository. Negli esempi riportati in questo argomento, viene estratta l'immagine per l'helper di parametri del plugin CNI di Amazon VPC per Kubernetes. Quando segui questi passaggi, assicurati di sostituire i valori di esempio con i tuoi valori.

1. Se non si dispone ancora di un repository Amazon ECR o di un altro repository, crearne uno a cui i nodi abbiano accesso. Il seguente comando crea un repository privato Amazon ECR. Il nome di un repository privato Amazon ECR deve iniziare con una lettera. Può contenere soltanto lettere minuscole, numeri, trattini (-), trattini bassi (_) e barre (/). Per ulteriori informazioni, consulta la

sezione Creazione di un repository privato nella Guida per l'utente di Amazon Elastic Container Registry.

Puoi sostituire *cni-metrics-helper* con un qualsiasi valore a tua scelta. Come best practice, crea un repository separato per ogni immagine. Lo consigliamo perché i tag immagine devono essere univoci all'interno di un repository. Sostituisci *region-code* con una AWS regione supportata da Amazon ECR.

```
aws ecr create-repository --region region-code --repository-name cni-metrics-helper
```

2. Determina il registro, il repository e il tag (facoltativo) dell'immagine che i nodi devono estrarre. Questa informazione è nel formato `registry/repository[:tag]`.

   Molti degli argomenti di Amazon EKS sull'installazione delle immagini richiedono l'applicazione di un file manifesto o l'installazione dell'immagine utilizzando un grafico Helm. Tuttavia, prima di applicare un file manifesto o installare un grafico Helm, visualizzare il contenuto del manifesto o del file `values.yaml` del grafico. In questo modo, puoi determinare il registro, il repository e il tag da estrarre.

   Ad esempio, è possibile trovare la seguente riga nel file manifesto dell'helper di parametri del plugin CNI di Amazon VPC per Kubernetes. Il registro è `602401143452.dkr.ecr.us-west-2.amazonaws.com`, che è un registro privato Amazon ECR. Il repository è `cni-metrics-helper`.

   ```
   image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.12.6"
   ```

   Potresti visualizzate le seguenti varianti per la posizione dell'immagine:

   - Solo `repository-name:tag`. In questo caso, `docker.io` di solito è il registro, ma non è specificato poiché di default Kubernetes gli antepone un nome di repository se non viene specificato alcun registro.
   - `repository-name/repository-namespace/repository:tag`. il namespace del repository è facoltativo, ma talvolta viene specificato dal proprietario del repository per la classificazione delle immagini. Ad esempio, tutte le EC2 immagini Amazon nella Amazon ECR Public Gallery utilizzano lo `aws-ec2` spazio dei nomi.

     Prima di installare un'immagine con Helm, visualizza il file `values.yaml` Helm per determinare la posizione dell'immagine. Ad esempio, il file values.yaml dell'helper di parametri del plugin CNI di Amazon VPC per Kubernetes include le seguenti righe.

```
image:
  region: us-west-2
  tag: v1.12.6
  account: "602401143452"
  domain: "amazonaws.com"
```

3. Estrai l'immagine del container specificata nel file manifesto.

   a. Se si esegue l'estrazione da un registro pubblico, come la Galleria pubblica di Amazon ECR, è possibile passare alla fase secondaria successiva poiché l'autenticazione non è richiesta. In questo esempio, esegui l'autenticazione a un registro privato di Amazon ECR contenente il repository per l'immagine dell'helper per i parametri CNI. Amazon EKS conserva l'immagine in ogni registro elencato nella pagina View Amazon container image registries for Amazon EKS add-ons. Puoi eseguire l'autenticazione a uno qualsiasi dei registri sostituendo *602401143452* e *region-code* con le informazioni di un registro differente. Esiste un registro separato per ciascuna Regione AWS in cui è supportato Amazon EKS.

   ```
   aws ecr get-login-password --region region-code | docker login --username AWS --
   password-stdin 602401143452.dkr.ecr.region-code.amazonaws.com
   ```

   b. Estrai l'immagine. In questo esempio, esegui l'estrazione dal registro a cui ti sei autenticato nel passaggio secondario precedente. Sostituisci *602401143452* e *region-code* con le informazioni che hai fornito nella fase secondaria precedente.

   ```
   docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-
   helper:v1.12.6
   ```

4. Tagga l'immagine che hai estratto con il registro, il repository e il tag. L'esempio seguente presuppone che tu abbia estratto l'immagine dal file manifesto e la invierai al repository privato Amazon ECR creato nel primo passaggio. Sostituisci *111122223333* con l'ID del tuo account. Sostituisci *region-code* con la AWS regione in cui hai creato il tuo repository privato Amazon ECR.

   ```
   docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-code.amazonaws.com/
   cni-metrics-helper:v1.12.6
   ```

5. Esegui l'autenticazione al tuo registro. In questo esempio, esegui l'autenticazione al registro privato Amazon ECR creato nel primo passaggio. Per ulteriori informazioni, consulta la sezione Autenticazione del registro nella Guida per l'utente di 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. Invia l'immagine al tuo repository. In questo esempio, invii l'immagine al registro privato Amazon ECR creato nel primo passaggio. Per ulteriori informazioni, consulta la sezione Invio di un'immagine Docker nella Guida per l'utente di Amazon Elastic Container Registry.

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
```

7. Aggiorna il file manifesto utilizzato per determinare l'immagine in un passaggio precedente con il `registry/repository:tag` per l'immagine che hai inviato. Se si esegue l'installazione con un grafico Helm, spesso è possibile specificare il `registry/repository:tag`. Quando installi il grafico, specifica il `registry/repository:tag` dell'immagine che hai inviato al repository.

Visualizza i registri delle immagini del container per i componenti aggiuntivi di Amazon EKS

Quando implementi AWS Componenti aggiuntivi Amazon EKS nel tuo cluster, i nodi estraggono le immagini dei container necessarie dal registro specificato nel meccanismo di installazione del componente aggiuntivo, ad esempio un manifesto di installazione o un file `values.yaml` Helm. Le immagini vengono estratte da un repository privato Amazon ECR di Amazon EKS. Amazon EKS replica le immagini in un repository in ogni regione supportata AWS da Amazon EKS. I nodi possono estrarre l'immagine del container su Internet da uno dei seguenti registri. In alternativa, i tuoi nodi possono trasferire l'immagine sulla rete di Amazon se hai creato un endpoint VPC di interfaccia per Amazon ECR ()AWS PrivateLink nel tuo VPC. I registri richiedono l'autenticazione con un account IAM. AWS I tuoi nodi si autenticano utilizzando il ruolo IAM del nodo Amazon EKS, a cui sono associate le autorizzazioni nella policy IAM EC2 ContainerRegistryReadOnly gestita di Amazon.

AWS Regione	Registro
af-south-1	877085696533.dkr.ecr.af-south-1.amazonaws.com
ap-east-1	800184023465.dkr.ecr.ap-east-1.amazonaws.com

AWS Regione	Registro
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-5	151610086707.dkr. ecr.ap-southeast-5.amazonaws.com
ap-southeast-6	333609536671.dkr. ecr.ap-southeast-6.amazonaws.com
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

AWS Regione	Registro
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
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

AWS Regione	Registro
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
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-1amazonaws.com
us-gov-west-1	013241004608.dkr.ecr. us-gov-west-1amazonaws.com
eusc-de-east-1	877088126301.dkr.ecr. eusc-de-east-1.amazonaws.eu

Componenti aggiuntivi Amazon EKS

Un componente aggiuntivo è un software che fornisce funzionalità operative di supporto alle applicazioni Kubernetes, ma non è specifico per l'applicazione. Ciò include software come agenti

di osservabilità o driver Kubernetes che consentono al cluster di interagire con le risorse sottostanti AWS per il networking, l'elaborazione e lo storage. Il software aggiuntivo viene in genere creato e gestito dalla community Kubernetes, dai provider di servizi cloud o da fornitori di terze parti. AWS Per ogni cluster, Amazon EKS installa automaticamente componenti aggiuntivi autogestiti come il plugin CNI di Amazon VPC per Kubernetes, `kube-proxy`, e CoreDNS. Tenere presente che il componente aggiuntivo CNI VPC non è compatibile con Amazon EKS Hybrid Nodes e non viene implementato su nodi ibridi. È possibile modificare la configurazione predefinita dei componenti aggiuntivi e aggiornarli quando lo desideri.

I componenti aggiuntivi di Amazon EKS forniscono l'installazione e la gestione di un set curato di componenti aggiuntivi per i cluster Amazon EKS. Tutti i componenti aggiuntivi di Amazon EKS includono le patch di sicurezza e le correzioni di bug più recenti e sono convalidati per AWS funzionare con Amazon EKS. I componenti aggiuntivi Amazon EKS permettono di garantire costantemente sicurezza e stabilità ai cluster Amazon EKS e di ridurre la quantità di lavoro necessaria per installare, configurare e aggiornare i componenti aggiuntivi. Se un componente aggiuntivo autogestito, ad esempio `kube-proxy` è già in esecuzione nel cluster ed è disponibile come componente aggiuntivo di Amazon EKS, è possibile installare il componente aggiuntivo `kube-proxy` Amazon EKS per iniziare a beneficiare delle funzionalità dei componenti aggiuntivi di Amazon EKS.

Tramite l'API Amazon EKS. è possibile aggiornare specifici campi di configurazione gestiti da Amazon EKS per i componenti aggiuntivi Amazon EKS. É anche possibile modificare i campi di configurazione non gestiti da Amazon EKS direttamente all'interno del cluster Kubernetes una volta avviato il componente aggiuntivo. Ciò include, ove applicabile, la definizione di campi di configurazione specifici per un componente aggiuntivo. Una volta apportate, queste modifiche non vengono sovrascritte da Amazon EKS. Ciò è reso possibile utilizzando la caratteristica applicazione lato server Kubernetes. Per ulteriori informazioni, consulta the section called “Campi che puoi personalizzare”.

È possibile utilizzare i componenti aggiuntivi di Amazon EKS con qualsiasi tipo di nodo Amazon EKS. Per ulteriori informazioni, consulta Gestione del calcolo.

Puoi aggiungere, aggiornare o eliminare componenti aggiuntivi Amazon EKS utilizzando l'API Amazon EKS Console di gestione AWS, la AWS CLI e. `eksctl` La creazione di componenti aggiuntivi di Amazon EKS è possibile anche utilizzando AWS CloudFormation.

Considerazioni

Tenere in considerazione quanto segue quando si utilizzano i componenti aggiuntivi Amazon EKS:

- Per configurare i componenti aggiuntivi per il cluster, il principale IAM deve disporre delle autorizzazioni IAM per l'utilizzo dei componenti aggiuntivi. Per ulteriori informazioni, consulta le operazioni con Addon nel loro nome in Operazioni definite da Amazon Elastic Kubernetes Service.
- I componenti aggiuntivi di Amazon EKS vengono eseguiti sui nodi con provisioning o configurazione per il cluster. I tipi di nodi includono EC2 istanze Amazon, Fargate e nodi ibridi.
- È possibile modificare i campi non gestiti da Amazon EKS per personalizzare l'installazione di un componente aggiuntivo Amazon EKS. Per ulteriori informazioni, consulta the section called "Campi che puoi personalizzare".
- Se crei un cluster con Console di gestione AWS, i componenti aggiuntivi Amazon EKS`kube-proxy`, Amazon VPC CNI per Kubernetes e CoredNS Amazon EKS vengono aggiunti automaticamente al cluster. Se utilizzi `eksctl` per creare il cluster con un file `config`, `eksctl` può anche creare il cluster con i componenti aggiuntivi di Amazon EKS. Se si crea il cluster utilizzando `eksctl` senza un file `config` o con qualsiasi altro strumento, `kube-proxy` autogestito, il plugin CNI di Amazon VPC per Kubernetes e i componenti aggiuntivi CoreDNS vengono installati in alternativa ai componenti aggiuntivi Amazon EKS. Puoi gestirli in autonomia o aggiungere manualmente i componenti aggiuntivi di Amazon EKS dopo la creazione del cluster. Indipendentemente dal metodo utilizzato per creare il cluster, il componente aggiuntivo CNI VPC non viene installato sui nodi ibridi.
- `eks:addon-cluster-admin` `ClusterRoleBinding` associa `cluster-admin` `ClusterRole` all'identità Kubernetes `eks:addon-manager`. Il ruolo dispone delle autorizzazioni necessarie all'identità `eks:addon-manager` per creare namespace Kubernetes e per installare componenti aggiuntivi nei namespace. Se il `ClusterRoleBinding` `eks:addon-cluster-admin` viene rimosso, il cluster Amazon EKS continua a funzionare, ma Amazon EKS non è più in grado di gestire i componenti aggiuntivi. Tutti i cluster a partire dalle seguenti versioni della piattaforma utilizzano il nuovo `ClusterRoleBinding`.
- Un sottoinsieme di componenti aggiuntivi EKS di è stato convalidato per verificarne la compatibilità con Amazon EKS Hybrid Nodes. AWS Per ulteriori informazioni, consultare la tabella di compatibilità alla pagina the section called " AWS componenti aggiuntivi".

Namespace personalizzato per componenti aggiuntivi

Per la community e AWS i componenti aggiuntivi, puoi facoltativamente specificare uno spazio dei nomi personalizzato durante la creazione del componente aggiuntivo. Una volta installato un componente aggiuntivo in un namespace specifico, è necessario rimuovere e creare nuovamente il componente per modificarne il namespace.

Se non viene specificato un namespace, verrà utilizzato il namespace predefinito del componente aggiuntivo.

Utilizzare namespace personalizzati per una migliore organizzazione e un miglior isolamento degli oggetti del componente aggiuntivo all'interno del cluster EKS. Questa flessibilità consente di conformare i componenti aggiuntivi alle esigenze operative e alla strategia di namespace esistente.

È possibile impostare un namespace personalizzato durante la creazione di un componente aggiuntivo. Per ulteriori informazioni, consulta the section called "Creare un componente aggiuntivo".

Ottenere un namespace predefinito di un componente aggiuntivo

Il namespace predefinito di un componente aggiuntivo è il namespace in cui sarà installato se non ne viene specificato uno.

Utilizzare il seguente comando per ottenere il namespace predefinito di un componente aggiuntivo:

```
aws eks describe-addon-versions --addon-name <addon-name> --query
 "addons[].defaultNamespace"
```

Output di esempio:

```
[
    "kube-system"
]
```

Considerazioni per la modalità automatica di Amazon EKS

La modalità automatica di Amazon EKS include funzionalità che garantiscono l'operatività essenziale del cluster, tra cui:

- Reti pod
- Reti di servizio
- DNS del cluster
- Dimensionamento automatico
- Storage a blocchi
- Controller del bilanciatore del carico

- Agente Pod Identity
- Agente di monitoraggio del nodo

Con il calcolo in modalità automatica, molti dei componenti aggiuntivi EKS più utilizzati diventano ridondanti, come:

- CNI di Amazon VPC
- kube-proxy
- CoreDNS
- Driver CSI per Amazon EBS
- EKS Pod Identity Agent

Tuttavia, se il cluster combina la modalità Auto con altre opzioni di elaborazione come EC2 istanze autogestite, Gruppi di nodi gestiti o AWS Fargate, questi componenti aggiuntivi rimangono necessari. AWS ha migliorato i componenti aggiuntivi EKS con regole anti-affinità che garantiscono automaticamente la pianificazione dei pod aggiuntivi solo sui tipi di elaborazione supportati. Inoltre, ora gli utenti possono ora utilizzare l'API `DescribeAddonVersions` dei componenti aggiuntivi EKS per verificare i tipi di calcolo supportati (computeTypes) per ciascun componente aggiuntivo e le relative versioni specifiche. Inoltre, con la modalità EKS Auto, i controller sopra elencati funzionano su un'infrastruttura di proprietà. AWS Di conseguenza, potrebbero non essere nemmeno visibili negli account, a meno che non si utilizzi la modalità automatica EKS con altri tipi di calcolo. In tal caso, i controller installati sul cluster saranno visibili.

Se si intende abilitare la modalità automatica EKS su un cluster esistente, potrebbe essere necessario aggiornare la versione di alcuni componenti aggiuntivi. Per ulteriori informazioni, consultare the section called "Versioni richieste del componente aggiuntivo" per la modalità automatica EKS.

Supporto

AWS pubblica diversi tipi di componenti aggiuntivi con diversi livelli di supporto.

- Componenti aggiuntivi AWS : questi componenti aggiuntivi sono creati e completamente supportati da AWS.
  - Usa un AWS componente aggiuntivo per lavorare con altri AWS servizi, come Amazon EFS.
  - Per ulteriori informazioni, consulta the section called " AWS componenti aggiuntivi".

- AWS Componenti aggiuntivi del Marketplace: questi componenti aggiuntivi vengono analizzati AWS e supportati da un partner indipendente. AWS
  - Utilizzare un componente aggiuntivo del marketplace per aggiungere funzionalità preziose e sofisticate al cluster, come il monitoraggio con Splunk.
  - Per ulteriori informazioni, consulta the section called "Componenti aggiuntivi di Marketplace".
- Componenti aggiuntivi della community: questi componenti aggiuntivi vengono analizzati AWS ma supportati dalla comunità open source.
  - Utilizzare un componente aggiuntivo della community per ridurre la complessità dell'installazione di software open source comuni, come Kubernetes Metrics Server.
  - I componenti aggiuntivi della community sono forniti dal codice sorgente di. AWS AWS convalida i componenti aggiuntivi della community solo per la compatibilità delle versioni.
  - Per ulteriori informazioni, consulta the section called "Componenti aggiuntivi della community".

La tabella seguente illustra la portata del supporto per ogni tipo di componente aggiuntivo:

Categoria	Funzionalità	AWS componenti aggiuntivi	AWS Componenti aggiuntivi del Marketplace	Componenti aggiuntivi della community
Sviluppo	Creato da AWS	Sì	No	Sì
Sviluppo	Convalidato da AWS	Sì	No	Sì*
Sviluppo	Convalidato dal partner AWS	No	Sì	No
Maintenance (Manutenzione)	Scansionato da AWS	Sì	Sì	Sì
Maintenance (Manutenzione)	Patchato da AWS	Sì	No	Sì
Maintenance (Manutenzione)	Patchato dal partner AWS	No	Sì	No

Categoria	Funzionalità	AWS componenti aggiuntivi	AWS Componenti aggiuntivi del Marketplace	Componenti aggiuntivi della community
Distribuzione	Pubblicato da AWS	Sì	No	Sì
Distribuzione	Pubblicato da AWS Partner	No	Sì	No
Supporto	Supporto di installazione di base fornito da AWS	Sì	Sì	Sì
Supporto	AWS Supporto completo	Sì	No	No
Supporto	Supporto completo AWS per i partner	No	Sì	No

*: la convalida dei componenti aggiuntivi della community include solo la compatibilità delle versioni di Kubernetes. Ad esempio, se installi un componente aggiuntivo della community su un cluster, AWS verifica se è compatibile con la versione Kubernetes del cluster.

AWS I componenti aggiuntivi di Marketplace possono scaricare dipendenze software aggiuntive da fonti esterne esterne a. AWS Queste dipendenze esterne non vengono analizzate o convalidate da. AWS Considera i tuoi requisiti di sicurezza quando distribuisci componenti aggiuntivi di AWS Marketplace che recuperano dipendenze esterne.

AWS componenti aggiuntivi

Nel tuo cluster puoi creare i seguenti componenti aggiuntivi di Amazon EKS. È possibile visualizzare l'elenco più aggiornato dei componenti aggiuntivi disponibili utilizzando `eksctl` Console di gestione AWS, o la AWS CLI. Per visualizzare tutti i componenti aggiuntivi disponibili o per installare un componente aggiuntivo, consulta la pagina the section called “Creare un componente aggiuntivo”. Se

un componente aggiuntivo richiede autorizzazioni IAM, è necessario disporre di un provider OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". Puoi creare o eliminare un componente aggiuntivo dopo averlo installato. Per ulteriori informazioni, consulta the section called "Aggiornamento di un componente aggiuntivo" o the section called "Rimozione di un componente aggiuntivo". Per ulteriori informazioni su considerazioni specifiche in merito all'esecuzione di componenti aggiuntivi EKS con Amazon EKS Hybrid Nodes, consulta the section called "Configurare i componenti aggiuntivi".

È possibile utilizzare uno dei seguenti componenti aggiuntivi di Amazon EKS.

Description	Ulteriori informazioni	Tipi di elaborazione compatibili
Fornire una rete VPC nativa per il tuo cluster	the section called "Plug-in della CNI di Amazon VPC per Kubernetes"	EC2
Un server DNS flessibile ed estensibile che può fungere da cluster DNS Kubernetes	the section called "CoreDNS"	EC2, Fargate, modalità automatica EKS, nodi ibridi EKS
Mantenere le regole di rete su ciascun nodo Amazon EC2	the section called "`Kube-proxy`"	Nodi ibridi EC2, EKS
Fornisci l'archiviazione Amazon EBS per il cluster	the section called "Driver CSI per Amazon EBS"	EC2
Fornire l'archiviazione Amazon EBS per il cluster	the section called "Driver CSI per Amazon EFS"	EC2, modalità automatica EKS
Fornisci lo storage Amazon FSx for Lustre per il tuo cluster	the section called "Driver Amazon FSx CSI"	EC2, modalità automatica EKS
Fornire l'archiviazione Amazon S3 per il cluster	the section called "Driver CSI di Mountpoint per Amazon S3"	EC2, modalità automatica EKS
Rilevare ulteriori problemi di integrità dei nodi	the section called "Agente di monitoraggio del nodo"	Nodi ibridi EC2, EKS

Description	Ulteriori informazioni	Tipi di elaborazione compatibili
Abilitare l'utilizzo della funzionalità snapshot nei driver CSI compatibili, come il driver CSI per Amazon EBS.	the section called "Controller di snapshot CSI"	EC2, Fargate, modalità automatica EKS, nodi ibridi EKS
SageMaker HyperPod la governance delle attività ottimizza l'allocazione e l'utilizzo delle risorse di calcolo tra i team nei cluster Amazon EKS, risolvendo le inefficienze nella prioritizzazione delle attività e nella condivisione delle risorse.	the section called "Gestione delle SageMaker HyperPod attività di Amazon"	EC2, modalità automatica EKS,
Amazon SageMaker HyperPod Observability AddOn offre funzionalità complete di monitoraggio e osservabilità per i HyperPod cluster.	the section called "Componente aggiuntivo Amazon SageMaker HyperPod Observability"	EC2, modalità automatica EKS,
L'operatore di SageMaker HyperPod formazione di Amazon consente una formazione distribuita efficiente sui cluster Amazon EKS con funzionalità avanzate di pianificazione e gestione delle risorse.	the section called "Operatore SageMaker HyperPod di formazione Amazon"	EC2, modalità automatica EKS

Description	Ulteriori informazioni	Tipi di elaborazione compatibili
Amazon SageMaker HyperPod Inference Operator consente l'implementazione e la gestione di carichi di lavoro di inferenza AI ad alte prestazioni con utilizzo ottimizzato delle risorse ed efficienza dei costi.	the section called "Operatore di SageMaker HyperPod inferenza Amazon"	EC2, modalità automatica EKS
Un agente Kubernetes che raccoglie e riporta i dati del flusso di rete ad Amazon CloudWatch, abilitando il monitoraggio completo delle connessioni TCP tra i nodi del cluster.	the section called "AWS Network Flow Monitor Agent"	EC2, modalità automatica EKS
Distribuzione del progetto sicura, pronta per la produzione e supportata AWS OpenTelemetry	the section called "AWS Distro per OpenTelemetry"	EC2, Fargate, modalità automatica EKS, nodi ibridi EKS
Servizio di monitoraggio della sicurezza che analizza ed elabora le fonti di dati fondamentali, tra cui eventi di AWS CloudTrail gestione e log di flusso di Amazon VPC. Amazon elabora GuardDuty anche funzionalità, come i log di controllo di Kubernetes e il monitoraggio del runtime	the section called "GuardDuty Agente Amazon"	EC2, modalità automatica EKS

Description	Ulteriori informazioni	Tipi di elaborazione compatibili
Servizio di monitoraggio e osservabilità fornito da AWS. Questo componente aggiuntivo installa l' CloudWatch agente e abilita sia CloudWatch Application Signals che CloudWatch Container Insights con una migliore osservabilità per Amazon EKS	the section called "Agente Amazon CloudWatch Observability"	EC2, modalità automatica EKS, nodi ibridi EKS
Possibilità di gestire le credenziali per le applicazioni, analogamente al modo in cui i profili dell'istanza EC2 forniscono le credenziali alle istanze EC2	the section called "EKS Pod Identity Agent"	Nodi ibridi EC2, EKS
Abilita cert-manager per emettere certificati X.509 da Private CA. AWS Richiede cert-manager.	the section called "AWS Connettore CA privato per Kubernetes"	EC2, Fargate, modalità automatica EKS, nodi ibridi EKS
Genera metriche Prometheus sulle prestazioni dei dispositivi di rete SR-IOV	the section called "Esportatore di metriche di rete SR-IOV"	EC2
Recupera i AWS segreti da Secrets Manager e i parametri da AWS Systems Manager Parameter Store e montali come file nei pod Kubernetes.	the section called "AWS Fornitore di driver CSI Secrets Store"	EC2, modalità automatica EKS, nodi ibridi EKS
Con Spaces, puoi creare JupyterLab e gestire applicazioni Code Editor per eseguire carichi di lavoro ML interattivi.	the section called "SageMaker Spazi Amazon"	Hyperpod

Plug-in della CNI di Amazon VPC per Kubernetes

Il componente aggiuntivo Amazon EKS del plugin CNI di Amazon VPC per Kubernetes è un plugin di interfaccia di rete container (CNI) Kubernetes che fornisce una rete VPC nativa per il cluster. Il tipo autogestito o gestito di questo componente aggiuntivo è installato su ogni nodo Amazon EC2 per impostazione predefinita. Per ulteriori informazioni, consulta Plugin di interfaccia di rete container (CNI) Kubernetes.

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Il nome del componente aggiuntivo di Amazon EKS è `vpc-cni`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo utilizza i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

Se il cluster utilizza la famiglia `IPv4`, sono necessarie le autorizzazioni in AmazonEKS_CNI_Policy. Se il cluster utilizza la `IPv6` famiglia, è necessario creare una policy IAM con le autorizzazioni in IPv6 modalità. È possibile creare un ruolo IAM, collegare una delle policy e annotare l'account del servizio Kubernetes utilizzato dal componente aggiuntivo con il comando seguente.

Sostituisci *`my-cluster`* con il nome del cluster e *`AmazonEKSVPCCNIRole`* con il nome del ruolo. Se il tuo cluster utilizza la famiglia `IPv6`, sostituisci *`AmazonEKS_CNI_Policy`* con il nome della policy che hai creato. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo, collegarvi la policy e annotare l'account del servizio Kubernetes, consulta the section called "Assegnazione di un ruolo 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
```

Aggiornare le informazioni

Puoi aggiornare solo una versione secondaria alla volta. Ad esempio, se la versione corrente è `1.28.x-eksbuild.y` e desideri aggiornarla a `1.30.x-eksbuild.y`, devi prima eseguire l'aggiornamento a `1.29.x-eksbuild.y` e quindi l'aggiornamento a `1.30.x-eksbuild.y`. Per ulteriori informazioni sull'aggiornamento del componente aggiuntivo, consulta the section called "Aggiornamento (componente aggiuntivo EKS)".

CoreDNS

Il componente aggiuntivo CoreDNS per Amazon EKS è un server DNS flessibile ed estensibile che può fungere da DNS del cluster di Kubernetes. Il tipo autogestito o gestito di questo componente aggiuntivo è stato installato, per impostazione predefinita, al momento della creazione del cluster. Quando si avvia un cluster Amazon EKS con almeno un nodo, due repliche dell'immagine CoreDNS vengono distribuite per impostazione predefinita, indipendentemente dal numero di nodi distribuiti nel cluster. I contenitori CoreDNS forniscono la risoluzione dei nomi per tutti i pod del cluster. È possibile implementare i pod CoreDNS ai nodi Fargate se il cluster include un profilo Fargate con uno namespace che corrisponde al namespace per l'implementazione di CoreDNS. Per ulteriori informazioni, consulta the section called "Definisci i profili"

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Il nome del componente aggiuntivo di Amazon EKS è `coredns`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione.

Informazioni aggiuntive

Per ulteriori informazioni su CoreDNS, consulta Utilizzo di CoreDNS per il rilevamento servizi e Personalizzazione del servizio DNS nella documentazione di Kubernetes.

`Kube-proxy`

Il componente aggiuntivo di Amazon EKS `Kube-proxy` mantiene le regole di rete su ciascun nodo Amazon EC2. Consente la comunicazione di rete con i tuoi pod. Il tipo autogestito o gestito di questo componente aggiuntivo è installato su ogni nodo Amazon EC2 nel cluster per impostazione predefinita.

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Il nome del componente aggiuntivo di Amazon EKS è `kube-proxy`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione.

Aggiornare le informazioni

Prima di aggiornare la versione corrente, considera i seguenti requisiti:

- `Kube-proxy` su un cluster Amazon EKS ha la stessa policy di compatibilità e disallineamento di Kubernetes.

Informazioni aggiuntive

Per ulteriori informazioni su `kube-proxy`, consulta kube-proxy nella documentazione Kubernetes.

Driver CSI per Amazon EBS

Il componente aggiuntivo di Amazon EKS per il driver CSI per Amazon EBS è un plugin di interfaccia di archiviazione del container (CSI) Kubernetes che fornisce archiviazione Amazon EBS per il cluster.

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. La modalità automatica include una funzionalità di archiviazione a blocchi.

Per ulteriori informazioni, consulta the section called "Implementazione di un carico di lavoro stateful".

Il nome del componente aggiuntivo di Amazon EKS è `aws-ebs-csi-driver`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo usa i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA". Le autorizzazioni nella EBSCSIDriverpolitica AWS gestita da Amazon Policy sono obbligatorie. Puoi creare un ruolo IAM e collegarvi la policy gestita con il seguente comando. Sostituisci *`my-cluster`* con il nome del cluster e *`AmazonEKS_EBS_CSI_DriverRole`* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se hai la necessità di utilizzare uno strumento diverso o una chiave KMS personalizzata per la crittografia, consulta la sezione the section called "Passaggio 1: Creazione di un ruolo 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
```

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Amazon EBS".

Driver CSI per Amazon EFS

Il componente aggiuntivo di Amazon EKS per il driver EFS per Amazon EBS è un plugin di interfaccia di archiviazione del container (CSI) Kubernetes che fornisce archiviazione Amazon EFS per il cluster.

Il nome del componente aggiuntivo di Amazon EKS è `aws-efs-csi-driver`.

Autorizzazioni IAM richieste

Autorizzazioni IAM richieste: questo componente aggiuntivo usa la funzionalità Ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali

con IRSA". Le autorizzazioni nella EFSCSIDriverpolitica AWS gestita da Amazon Policy sono obbligatorie. Puoi creare un ruolo IAM e collegarvi la policy gestita con i seguenti comandi. Sostituisci *my-cluster* con il nome del cluster e *AmazonEKS_EFS_CSI_DriverRole* con il nome del ruolo. Questi comandi richiedono l'installazione di eksctl sul tuo dispositivo. Se hai bisogno di usare uno strumento diverso, vedi the section called "Passaggio 1: Creazione di un ruolo 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"
```

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Amazon EFS".

Driver Amazon FSx CSI

Il driver Amazon FSx CSI (componente aggiuntivo Amazon EKS) è un plug-in Kubernetes Container Storage Interface (CSI) che fornisce lo storage Amazon for Lustre FSx per il tuo cluster.

Il nome del componente aggiuntivo di Amazon EKS è `aws-fsx-csi-driver`.

> **Note**
>
> - Le installazioni preesistenti di driver Amazon FSx CSI nel cluster possono causare errori di installazione dei componenti aggiuntivi. Quando tenti di installare la versione del componente aggiuntivo Amazon EKS mentre esiste un driver FSx CSI non EKS, l'installazione avrà esito negativo a causa di conflitti di risorse. Usa il flag `OVERWRITE` durante l'installazione per risolvere il problema:

```
aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
 --resolve-conflicts OVERWRITE
```

- Il componente aggiuntivo Amazon FSx CSI Driver EKS richiede l'agente EKS Pod Identity per l'autenticazione. Senza questo componente, il componente aggiuntivo fallirà con l'errore `Amazon EKS Pod Identity agent is not installed in the cluster`, impedendo le operazioni di volume. Installa l'agente Pod Identity prima o dopo la distribuzione del componente aggiuntivo FSx CSI Driver. Per ulteriori informazioni, consulta the section called "Configurazione dell'agente".

Autorizzazioni IAM richieste

Questo componente aggiuntivo usa i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA". Le autorizzazioni nella policy FSx FullAccess AWS gestita da Amazon sono obbligatorie. Puoi creare un ruolo IAM e collegarvi la policy gestita con il seguente comando. Sostituisci *`my-cluster`* con il nome del cluster e *`AmazonEKS_FSx_CSI_DriverRole`* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo 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
```

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Amazon FSx per Lustre".

Driver CSI di Mountpoint per Amazon S3

Il componente aggiuntivo Mountpoint per Amazon S3 di Amazon EKS per il driver CSI è un plugin di interfaccia di archiviazione del container (CSI) Kubernetes che fornisce archiviazione Amazon S3 per il cluster.

Il nome del componente aggiuntivo di Amazon EKS è `aws-mountpoint-s3-csi-driver`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo utilizza i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA".

Il ruolo IAM creato richiederà una policy che fornisca l'accesso a S3. Segui i Suggerimenti sulle autorizzazioni IAM di Mountpoint durante la creazione della policy. In alternativa, puoi utilizzare la politica AWS gestita AmazonS3 FullAccess, ma questa politica gestita concede più autorizzazioni di quelle necessarie per Mountpoint.

Puoi creare un ruolo IAM e collegarvi la policy gestita con i seguenti comandi. Sostituisci *`my-cluster`* con il nome del cluster, *`region-code`* con il codice AWS regionale corretto, *`AmazonEKS_S3_CSI_DriverRole`* con il nome del tuo ruolo e *`AmazonEKS_S3_CSI_DriverRole_ARN`* con il ruolo ARN. Questi comandi richiedono l'installazione di eksctl sul tuo dispositivo. Per istruzioni sull'uso della console IAM o della AWS CLI, consulta. the section called "Fase 2: creazione di un ruolo 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
```

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Mountpoint per Amazon S3".

Controller di snapshot CSI

Il controller di snapshot dell'interfaccia di archiviazione di container (CSI) consente l'uso della funzionalità snapshot nei driver CSI compatibili, come il driver CSI di Amazon EBS.

Il nome del componente aggiuntivo di Amazon EKS è `snapshot-controller`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione.

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta the section called "Controller snapshot CSI".

Gestione delle SageMaker HyperPod attività di Amazon

SageMaker HyperPod la governance delle attività è un robusto sistema di gestione progettato per semplificare l'allocazione delle risorse e garantire un utilizzo efficiente delle risorse di elaborazione tra team e progetti per i cluster Amazon EKS. Questa funzionalità offre agli amministratori la possibilità di impostare:

- Livelli di priorità per varie attività
- Allocazione delle risorse di calcolo per ogni team
- In che modo ogni team presta e prende in prestito risorse di calcolo inattive
- Se un team rende prerilasciabili le proprie attività

HyperPod la governance delle attività fornisce anche l'osservabilità del cluster Amazon EKS, offrendo visibilità in tempo reale sulla capacità del cluster. Ciò include la disponibilità e l'utilizzo delle risorse di calcolo, l'allocazione e l'utilizzo del team e le informazioni sull'esecuzione delle attività e sui tempi di attesa, per consentirti di prendere decisioni informate e gestire in modo proattivo le risorse.

Il nome del componente aggiuntivo di Amazon EKS è `amazon-sagemaker-hyperpod-taskgovernance`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione.

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta SageMaker HyperPod la sezione Task Governance

Componente aggiuntivo Amazon SageMaker HyperPod Observability

L'Amazon SageMaker HyperPod Observability Add-on offre funzionalità complete di monitoraggio e osservabilità per i cluster. HyperPod Questo componente aggiuntivo distribuisce e gestisce automaticamente i componenti di monitoraggio essenziali, tra cui Node Exporter, DCGM Exporter ed EFA Exporter. kube-state-metrics Raccoglie e inoltra i parametri a un'istanza Prometheus gestita da Amazon (AMP) designata dal cliente ed espone un endpoint OTLP per metriche personalizzate e acquisizione di eventi dai job di addestramento dei clienti.

Il componente aggiuntivo si integra con l' HyperPod ecosistema più ampio estrapolando le metriche da vari componenti, tra cui il componente aggiuntivo Task Governance, Training Operator, Kubeflow e KEDA. HyperPod HyperPod Tutte le metriche raccolte sono centralizzate in Prometheus gestito da Amazon, che consente ai clienti di ottenere una visione unificata dell'osservabilità tramite i dashboard di Grafana gestito da Amazon. Ciò fornisce end-to-end visibilità sullo stato del cluster, sull'utilizzo delle risorse e sulle prestazioni lavorative di formazione in tutto l'ambiente. HyperPod

Il nome del componente aggiuntivo di Amazon EKS è `amazon-sagemaker-hyperpod-observability`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo utilizza i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA". Di seguito sono elencate le policy gestite richieste:

- `AmazonPrometheusRemoteWriteAccess`: per la scrittura remota delle metriche a partire dal cluster fino ad AMP
- `CloudWatchAgentServerPolicy`- per la scrittura remota dei log dal cluster a CloudWatch

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo e sulle sue funzionalità, consulta SageMaker HyperPod Observability.

Operatore SageMaker HyperPod di formazione Amazon

L'operatore di SageMaker HyperPod formazione di Amazon ti aiuta ad accelerare lo sviluppo di modelli di intelligenza artificiale generativi gestendo in modo efficiente la formazione distribuita

su cluster di GPU di grandi dimensioni. Introduce funzionalità intelligenti di ripristino dai guasti, rilevamento delle interruzioni e gestione a livello di processo che riducono al minimo le interruzioni dell'addestramento e riducono i costi. A differenza dell'infrastruttura di addestramento tradizionale che richiede il riavvio completo del processo in caso di guasto, questo operatore implementa un ripristino chirurgico del processo per garantire il corretto funzionamento dei job di addestramento.

L'operatore collabora anche con le funzioni di monitoraggio HyperPod dello stato di salute e osservabilità, fornendo visibilità in tempo reale sull'esecuzione della formazione e il monitoraggio automatico di parametri critici come i picchi di perdita e il degrado della produttività. Puoi definire le policy di ripristino tramite semplici configurazioni YAML senza modifiche al codice, che consentono di rispondere rapidamente e ripristinare gli stati irreversibili dell'addestramento. Queste funzionalità di monitoraggio e ripristino interagiscono per garantire prestazioni di addestramento ottimali riducendo al minimo il sovraccarico operativo.

Il nome del componente aggiuntivo di Amazon EKS è `amazon-sagemaker-hyperpod-training-operator`.

Per ulteriori informazioni, consulta Using the HyperPod training operator nella Amazon SageMaker Developer Guide.

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede le autorizzazioni IAM e utilizza EKS Pod Identity.

AWS suggerisce la politica `AmazonSageMakerHyperPodTrainingOperatorAccess` gestita.

Per ulteriori informazioni, consulta Installazione dell'operatore di formazione nella Amazon SageMaker Developer Guide.

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta SageMaker HyperPod Training Operator.

Operatore di SageMaker HyperPod inferenza Amazon

Amazon SageMaker HyperPod offre un' end-to-endesperienza che supporta l'intero ciclo di vita dello sviluppo dell'intelligenza artificiale, dalla sperimentazione e formazione interattive ai flussi di lavoro di inferenza e post-formazione. Ora fornisce una piattaforma di inferenza completa che combina la flessibilità di Kubernetes con l'eccellenza operativa di un'esperienza gestita. Implementa,

ridimensiona e ottimizza i tuoi modelli GenAI con affidabilità di livello aziendale utilizzando lo stesso calcolo per l'intero ciclo di vita del modello. HyperPod

Amazon SageMaker HyperPod offre interfacce di distribuzione flessibili che consentono di distribuire modelli tramite diversi metodi, tra cui kubectl, Python SDK, Amazon SageMaker Studio UI o CLI. HyperPod La funzionalità offre funzionalità di scalabilità automatica avanzate con allocazione dinamica delle risorse che si adatta automaticamente in base alla domanda. Inoltre, include funzionalità complete di osservabilità e monitoraggio che tengono traccia di metriche critiche come la latenza e l'utilizzo della time-to-first-token GPU per aiutarti a ottimizzare le prestazioni.

Il nome del componente aggiuntivo di Amazon EKS è `amazon-sagemaker-hyperpod-inference`.

Metodi di installazione

È possibile installare questo componente aggiuntivo utilizzando uno dei seguenti metodi:

- SageMaker Console (consigliata): offre un'esperienza di installazione semplificata con configurazione guidata.
- Console dei componenti aggiuntivi EKS o CLI: richiede l'installazione manuale dei componenti aggiuntivi di dipendenza prima di installare l'operatore di inferenza. Per informazioni sulle dipendenze richieste, consulta la sezione relativa ai prerequisiti di seguito.

Prerequisiti

Prima di installare il componente aggiuntivo dell'operatore di inferenza tramite la console EKS Add-ons o la CLI, assicurati che siano installate le seguenti dipendenze.

Componenti aggiuntivi EKS richiesti:

- Driver CSI Amazon S3 Mountpoint (versione minima: v1.14.1-eksbuild.1)
- Metrics Server (versione minima: v0.7.2-eksbuild.4)
- Amazon FSx CSI Driver (versione minima: v1.6.0-eksbuild.1)
- Cert Manager (versione minima: v1.18.2-eksbuild.2)

Per istruzioni di installazione dettagliate per ciascuna dipendenza, vedere Installazione dell'operatore di inferenza.

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede le autorizzazioni IAM e utilizza OIDC/IRSA.

Le seguenti politiche gestite sono consigliate in quanto forniscono le autorizzazioni con ambito minimo:

- `AmazonSageMakerHyperPodInferenceAccess`- fornisce i privilegi di amministratore necessari per configurare l'operatore di inferenza
- `AmazonSageMakerHyperPodGatedModelAccess`- fornisce SageMaker HyperPod l'accesso a modelli chiusi in SageMaker Jumpstart (ad esempio, Meta Llama, GPT-Neo)

Per ulteriori informazioni, vedere Installazione dell'operatore di inferenza.

Informazioni aggiuntive

Per ulteriori informazioni sull'operatore di SageMaker HyperPod inferenza di Amazon, consulta operatore di SageMaker HyperPod inferenza.

Per informazioni sulla risoluzione dei problemi, consulta Risoluzione dei problemi di distribuzione SageMaker HyperPod del modello.

AWS Network Flow Monitor Agent

Amazon CloudWatch Network Flow Monitor Agent è un'applicazione Kubernetes che raccoglie le statistiche di connessione TCP da tutti i nodi di un cluster e pubblica report sul flusso di rete su Amazon Network Flow Monitor Ingestion. CloudWatch APIs

Il nome del componente aggiuntivo di Amazon EKS è `aws-network-flow-monitoring-agent`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede autorizzazioni IAM.

Quindi, è necessario aggiungere la policy gestita da `CloudWatchNetworkFlowMonitorAgentPublishPolicy` a tale componente aggiuntivo.

Per ulteriori informazioni sulla configurazione IAM richiesta, consulta IAM Policy nel GitHub repository Amazon CloudWatch Network Flow Monitor Agent.

Per ulteriori informazioni sulla policy gestita, consulta CloudWatchNetworkFlowMonitorAgentPublishPolicyla Amazon CloudWatch User Guide.

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo, consulta il repository Amazon CloudWatch Network Flow Monitor Agent GitHub .

Agente di monitoraggio del nodo

Il componente aggiuntivo Amazon EKS dell'agente di monitoraggio dei nodi è in grado di rilevare ulteriori problemi di integrità dei nodi. Questi segnali di integrità aggiuntivi possono essere sfruttati anche dalla funzione opzionale di riparazione automatica dei nodi per sostituire automaticamente i nodi secondo necessità.

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Il nome del componente aggiuntivo di Amazon EKS è `eks-node-monitoring-agent`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione supplementare.

Informazioni aggiuntive

Per ulteriori informazioni, consulta the section called "Integrità del nodo".

AWS Distro per OpenTelemetry

Il componente aggiuntivo AWS Distro for OpenTelemetry Amazon EKS è una distribuzione del progetto sicura, pronta per la produzione e AWS supportata. OpenTelemetry Per ulteriori informazioni, consulta AWS Distro for on. OpenTelemetry GitHub

Il nome del componente aggiuntivo di Amazon EKS è `adot`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede le autorizzazioni IAM solo se si utilizza una delle risorse personalizzate preconfigurate a cui è possibile accedere tramite la configurazione avanzata.

Informazioni aggiuntive

Per ulteriori informazioni, consulta Getting Started with AWS Distro for OpenTelemetry use EKS Add-Ons nella AWS Distro per la documentazione. OpenTelemetry

ADOT richiede che il `cert-manager` componente aggiuntivo sia distribuito nel cluster come prerequisito, altrimenti questo componente aggiuntivo non funzionerà se distribuito direttamente tramite https://registry.terraform. io/modules/terraform-aws-modules/eks/aws/ latest`cluster_addons`proprietà. Per ulteriori requisiti, consulta Requisiti per iniziare a AWS OpenTelemetry usare i componenti aggiuntivi EKS nella AWS distribuzione per la documentazione. OpenTelemetry

GuardDuty Agente Amazon

Il componente aggiuntivo Amazon EKS per l' GuardDuty agente Amazon raccoglie gli eventi di runtime (accesso ai file, esecuzione dei processi, connessioni di rete) dai nodi del cluster EKS per l'analisi tramite GuardDuty Runtime Monitoring. GuardDuty stesso (non l'agente) è il servizio di monitoraggio della sicurezza che analizza ed elabora le fonti di dati fondamentali, tra cui gli eventi di AWS CloudTrail gestione e i log di flusso di Amazon VPC, oltre a funzionalità come i log di controllo di Kubernetes e il monitoraggio del runtime.

Il nome del componente aggiuntivo di Amazon EKS è `aws-guardduty-agent`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo non richiede alcuna autorizzazione.

Informazioni aggiuntive

Per ulteriori informazioni, consulta Runtime Monitoring per i cluster Amazon EKS in Amazon GuardDuty.

- Per rilevare potenziali minacce alla sicurezza nei cluster Amazon EKS, abilita il monitoraggio del GuardDuty runtime di Amazon e distribuisci l'agente GuardDuty di sicurezza nei cluster Amazon EKS.

Agente Amazon CloudWatch Observability

L'agente Amazon CloudWatch Observability Amazon EKS è il componente aggiuntivo del servizio di monitoraggio e osservabilità fornito da. AWS Questo componente aggiuntivo installa l' CloudWatch

agente e abilita sia CloudWatch Application Signals che CloudWatch Container Insights con una migliore osservabilità per Amazon EKS. Per ulteriori informazioni, consulta Amazon CloudWatch Agent.

Il nome del componente aggiuntivo di Amazon EKS è `amazon-cloudwatch-observability`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo utilizza i ruoli IAM per gli account di servizio di Amazon EKS. Per ulteriori informazioni, consulta the section called "Credenziali con IRSA". Le autorizzazioni incluse nelle politiche AWSXrayWriteOnlyAccessCloudWatchAgentServerPolicy AWS gestite sono obbligatorie. È possibile creare un ruolo IAM, collegarvi una delle policy gestite e annotare l'account del servizio Kubernetes utilizzato dal componente aggiuntivo con il comando seguente. Sostituisci *`my-cluster`* con il nome del cluster e *`AmazonEKS_Observability_role`* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo, collegarvi la policy e annotare l'account del servizio Kubernetes, consulta the section called "Assegnazione di un ruolo 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
```

Informazioni aggiuntive

Per ulteriori informazioni, consulta Installare l' CloudWatch agente.

AWS Connettore CA privato per Kubernetes

Il AWS Private CA Connector for Kubernetes è un componente aggiuntivo per cert-manager che consente agli utenti di ottenere certificati da Private Certificate Authority (Private CA). AWS AWS

- Il nome del componente aggiuntivo di Amazon EKS è `aws-privateca-connector-for-kubernetes`.
- Il namespace del componente aggiuntivo è `aws-privateca-issuer`.

Questo componente aggiuntivo richiede `cert-manager`. `cert-manager` è disponibile su Amazon EKS come componente aggiuntivo della community. Per ulteriori informazioni su questo componente aggiuntivo, consulta the section called "Gestione certificati". Per ulteriori informazioni sui componenti aggiuntivi di installazione, consulta the section called "Creare un componente aggiuntivo".

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede autorizzazioni IAM.

Usa EKS Pod Identity per collegare la policy IAM `AWSPrivateCAConnectorForKubernetesPolicy` all'account del servizio Kubernetes `aws-privateca-issuer`. Per ulteriori informazioni, consulta the section called "Utilizzo di Pod Identities".

Per informazioni sulle autorizzazioni richieste, consulta il Managed Policy Reference. AWSPrivateCAConnectorForKubernetesPolicy AWS

Informazioni aggiuntive

Per ulteriori informazioni, consulta l'archivio AWS Private CA Issuer for GitHub Kubernetes.

Per ulteriori informazioni sulla configurazione del componente aggiuntivo, consulta values.yaml nel repository. `aws-privateca-issuer` GitHub Verifica che la versione di values.yaml corrisponda alla versione del componente aggiuntivo installato sul cluster.

Questo componente aggiuntivo tollera la contaminazione utilizzata dalla modalità automatica EKS. `CriticalAddonsOnly` `system` NodePool Per ulteriori informazioni, consulta the section called "Eseguire componenti aggiuntivi critici".

EKS Pod Identity Agent

Il componente aggiuntivo di Amazon EKS dell'agente Pod Identity fornisce la possibilità di gestire le credenziali per le applicazioni, analogamente al modo in cui i profili dell'istanza Amazon EC2 forniscono le credenziali alle istanze Amazon EC2.

> **Note**
>
> Non è necessario installare questo componente aggiuntivo sui cluster in modalità automatica EKS di Amazon. La modalità automatica EKS di Amazon si integra con Pod Identity EKS. Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Il nome del componente aggiuntivo di Amazon EKS è `eks-pod-identity-agent`.

Autorizzazioni IAM richieste

Il componente aggiuntivo dell'agente Pod Identity di per sé non richiede un ruolo IAM. Utilizza le autorizzazioni del ruolo IAM del nodo Amazon EKS per funzionare, ma non necessita di un ruolo IAM dedicato per il componente aggiuntivo.

Aggiornare le informazioni

Puoi aggiornare solo una versione secondaria alla volta. Ad esempio, se la versione corrente è `1.28.x-eksbuild.y` e desideri aggiornarla a `1.30.x-eksbuild.y`, devi prima eseguire l'aggiornamento a `1.29.x-eksbuild.y` e quindi l'aggiornamento a `1.30.x-eksbuild.y`. Per ulteriori informazioni sull'aggiornamento del componente aggiuntivo, consulta the section called "Aggiornamento di un componente aggiuntivo".

Esportatore di metriche di rete SR-IOV

Il componente aggiuntivo di Amazon EKS dell'esportazione di metriche di rete SR-IOV raccoglie ed espone le metriche sui dispositivi di rete SR-IOV in formato Prometheus. Consente il monitoraggio delle prestazioni della rete SR-IOV su nodi bare metal EKS. L'esportatore funziona DaemonSet su nodi con interfacce di SR-IOV-capable rete ed esporta metriche che possono essere eliminate da Prometheus.

> **Note**
>
> Questo componente aggiuntivo richiede nodi con interfacce di rete. SR-IOV-capable

Proprietà	Valore
Nome del componente aggiuntivo	`sriov-network-metrics-exporter`
Namespace	`monitoring`
Documentazione	Repo SR-IOV Network Metrics Exporter GitHub
Nome account del servizio	Nessuno
Policy gestite da IAM	Nessuno

Proprietà	Valore
Autorizzazioni IAM personalizzate	Nessuno

AWS Fornitore di driver CSI Secrets Store

Il AWS provider del driver CSI Secrets Store è un componente aggiuntivo che consente di recuperare segreti da Secrets Manager AWS e parametri da AWS Systems Manager Parameter Store e montarli come file nei pod Kubernetes.

Autorizzazioni IAM richieste

Il componente aggiuntivo non richiede autorizzazioni IAM. Tuttavia, i pod delle applicazioni richiederanno le autorizzazioni IAM per recuperare i segreti da Secrets Manager e AWS i parametri da AWS Systems Manager Parameter Store. Dopo aver installato il componente aggiuntivo, l'accesso deve essere configurato tramite IAM Roles for Service Accounts (IRSA) o EKS Pod Identity. Per utilizzare IRSA, consulta la documentazione di configurazione IRSA di Secrets Manager. Per utilizzare EKS Pod Identity, consulta la documentazione di configurazione di Secrets Manager Pod Identity.

AWS suggerisce la politica `AWSSecretsManagerClientReadOnlyAccess` gestita.

Per ulteriori informazioni sulle autorizzazioni richieste, consulta `AWSSecretsManagerClientReadOnlyAccess` il AWS Managed Policy Reference.

Informazioni aggiuntive

Per ulteriori informazioni, consulta il repository secrets-store-csi-driver -provider-awsGitHub .

Per ulteriori informazioni sul componente aggiuntivo, consulta la documentazione di AWS Secrets Manager relativa al componente aggiuntivo.

SageMaker Spazi Amazon

Il componente aggiuntivo Amazon SageMaker Spaces offre la possibilità di eseguire notebook IDEs e notebook su cluster EKS o -EKS. HyperPod Gli amministratori possono utilizzare la console EKS per installare il componente aggiuntivo sul proprio cluster e definire configurazioni di spazio predefinite come immagini, risorse di elaborazione, archiviazione locale per le impostazioni dei notebook (storage aggiuntivo da allegare ai rispettivi spazi), file system e script di inizializzazione.

Gli sviluppatori di intelligenza artificiale possono utilizzare kubectl per creare, aggiornare ed eliminare spazi. Hanno la flessibilità di utilizzare le configurazioni predefinite fornite dagli amministratori o personalizzare le impostazioni. Gli sviluppatori di intelligenza artificiale possono accedere ai propri spazi su EKS o HyperPod -EKS utilizzando il codice VS locale IDEs, and/or il browser Web che ospita il proprio JupyterLab CodeEditor IDE su un dominio DNS personalizzato configurato dai propri amministratori. Possono anche utilizzare la funzionalità di port forwarding di Kubernetes per accedere agli spazi nei loro browser web.

Il nome del componente aggiuntivo di Amazon EKS è `amazon-sagemaker-spaces`.

Autorizzazioni IAM richieste

Questo componente aggiuntivo richiede autorizzazioni IAM. Per ulteriori informazioni sulla configurazione IAM richiesta, consulta IAM Permissions Setup nella Amazon SageMaker Developer Guide.

Informazioni aggiuntive

Per ulteriori informazioni sul componente aggiuntivo e sulle sue funzionalità, consulta SageMaker AI Notebooks on nella Amazon Developer HyperPod Guide. SageMaker

Componenti aggiuntivi della community

Puoi utilizzarlo AWS APIs per installare componenti aggiuntivi della community, come Kubernetes Metrics Server. Puoi scegliere di installare componenti aggiuntivi della community come componenti aggiuntivi di Amazon EKS per ridurre la complessità della manutenzione del software su più cluster.

Ad esempio, puoi utilizzare l' AWS API, la CLI o la console di gestione per installare componenti aggiuntivi della community. Puoi installare un componente aggiuntivo della community durante la creazione del cluster.

I componenti aggiuntivi della community si gestiscono proprio come i componenti aggiuntivi di Amazon EKS esistenti. I componenti aggiuntivi della community sono diversi dai componenti aggiuntivi esistenti in quanto hanno un ambito di supporto unico.

> **Note**
>
> L'utilizzo dei componenti aggiuntivi della community è a discrezione dell'utente. Nell'ambito del modello di responsabilità condivisa tra te e te AWS, ci si aspetta che tu comprenda cosa stai installando nel tuo cluster per questi plug-in di terze parti. Sei anche responsabile

dei componenti aggiuntivi della community che soddisfano le sue esigenze di sicurezza del cluster. Per ulteriori informazioni, consulta the section called "Supporto per il software distribuito su EKS".

I componenti aggiuntivi della community non sono creati da. AWS AWS convalida i componenti aggiuntivi della community solo per la compatibilità delle versioni. Ad esempio, se installi un componente aggiuntivo della community su un cluster, AWS verifica se è compatibile con la versione Kubernetes del cluster.

È importante sottolineare AWS che non fornisce il supporto completo per i componenti aggiuntivi della community. AWS supporta solo le operazioni del ciclo di vita eseguite utilizzando AWS APIs, ad esempio l'installazione o l'eliminazione di componenti aggiuntivi.

Se è necessario supporto per un componente aggiuntivo della community, utilizzare le risorse di progetto esistenti. Ad esempio, è possibile creare un GitHub problema nel repository del progetto.

Determinare il tipo di componente aggiuntivo

Puoi utilizzare la AWS CLI per determinare il tipo di componente aggiuntivo Amazon EKS.

Utilizza il seguente comando della CLI per recuperare le informazioni su un componente aggiuntivo. Puoi sostituire `metrics-server` con il nome di qualsiasi componente aggiuntivo.

```
aws eks describe-addon-versions --addon-name metrics-server
```

Esamina l'output della CLI per il campo `owner`.

```
{
    "addons": [
        {
            "addonName": "metrics-server",
            "type": "observability",
            "owner": "community",
            "addonVersions": [
```

Se il valore di `owner` è`community`, allora il componente aggiuntivo è un componente aggiuntivo della community. AWS fornisce solo supporto per l'installazione, l'aggiornamento e la rimozione del componente aggiuntivo. Se hai domande sulla funzionalità e sul funzionamento del componente aggiuntivo stesso, utilizza le risorse della community come GitHub Issues.

Installare o aggiornare il componente aggiuntivo della community

Puoi installare o aggiornare i componenti aggiuntivi della community come fai con gli altri componenti aggiuntivi di Amazon EKS.

- the section called "Creare un componente aggiuntivo"
- the section called "Aggiornamento di un componente aggiuntivo"
- the section called "Rimozione di un componente aggiuntivo"

Componenti aggiuntivi della community disponibili

I seguenti componenti aggiuntivi della community sono disponibili presso Amazon EKS.

- the section called "Kubernetes Metrics Server"
- the section called "kube-state-metrics"
- the section called "Prometheus Node Exporter"
- the section called "Gestione certificati"
- the section called "DNS esterno"
- the section called "Fluent Bit"

Kubernetes Metrics Server

Il Kubernetes Metrics Server è una fonte scalabile ed efficiente di metriche delle risorse dei container per le pipeline di scalabilità automatica integrate in Kubernetes. Raccoglie le metriche delle risorse da Kubelets e le espone nell'apiserver Kubernetes tramite l'API Metriche per essere utilizzate da Horizontal Pod Autoscaler e Vertical Pod Autoscaler.

Proprietà	Valore
Nome del componente aggiuntivo	`metrics-server`
Namespace	`kube-system`
Documentazione	GitHub Readme
Nome account del servizio	Nessuno

Proprietà	Valore
Policy gestite da IAM	Nessuno
Autorizzazioni IAM personalizzate	Nessuno

kube-state-metrics

Agente del componente aggiuntivo per generare ed esporre metriche a livello di cluster.

Lo stato degli oggetti di Kubernetes nell'API Kubernetes può essere esposto sotto forma di metriche. Un agente aggiuntivo chiamato kube-state-metrics può connettersi al server dell'API Kubernetes ed esporre un endpoint HTTP con metriche generate dallo stato dei singoli oggetti nel cluster. Espone varie informazioni sullo stato degli oggetti come etichette e annotazioni, tempi di startup e terminazione, stato o Passaggio in cui si trova attualmente l'oggetto.

Proprietà	Valore
Nome del componente aggiuntivo	`kube-state-metrics`
Namespace	`kube-state-metrics`
Documentazione	Metrics for Kubernetes Object States nella documentazione di Kubernetes
Nome account del servizio	Nessuno
Policy gestite da IAM	Nessuno
Autorizzazioni IAM personalizzate	Nessuno

Prometheus Node Exporter

Esportatore Prometheus per i parametri hardware e del sistema operativo esposti dai kernel *NIX, scritto in Go con collettori di metriche collegabili. Prometheus Node Exporter espone un'ampia varietà di metriche relative all'hardware e al kernel.

Proprietà	Valore
Nome del componente aggiuntivo	`prometheus-node-exporter`
Namespace	`prometheus-node-exporter`
Documentazione	Monitoring Linux host metrics with the Node Exporter nella documentazione di Prometheus
Nome account del servizio	Nessuno
Policy gestite da IAM	Nessuno
Autorizzazioni IAM personalizzate	Nessuno

Gestione certificati

Gestione certificati può essere utilizzato per gestire la creazione e il rinnovo dei certificati.

Proprietà	Valore
Nome del componente aggiuntivo	`cert-manager`
Namespace	`cert-manager`
Documentazione	Cert Manager Docs
Nome account del servizio	Nessuno
Policy gestite da IAM	Nessuno
Autorizzazioni IAM personalizzate	Nessuno

DNS esterno

Il componente aggiuntivo del DNS esterno di EKS può essere utilizzato per gestire i record DNS di Route53 tramite le risorse di Kubernetes.

Le autorizzazioni del DNS esterno possono essere ridotte a `route53:ChangeResourceRecordSets`, `route53:ListHostedZones` e `route53:ListResourceRecordSets` sulle zone ospitate che si desidera gestire.

Proprietà	Valore
Nome del componente aggiuntivo	`external-dns`
Namespace	`external-dns`
Documentazione	GitHub Readme
Nome account del servizio	`external-dns`
Policy IAM gestite	`arn:aws: iam::aws:policy/AmazonRoute53FullAccess`
Autorizzazioni IAM personalizzate	Nessuno

Fluent Bit

Fluent Bit è un processore e server d'inoltro di log leggero e ad alte prestazioni. Consente di raccogliere dati data/logs da diverse fonti, unificarli e inviarli a più destinazioni, tra cui Amazon CloudWatch Logs, Amazon S3 e Amazon Kinesis Data Firehose. Fluent Bit è progettato pensando alle prestazioni e all'efficienza delle risorse, il che lo rende ideale per gli ambienti Kubernetes.

Questo componente aggiuntivo non richiede autorizzazioni IAM nella configurazione predefinita. Tuttavia, potrebbe essere necessario concedere a questo componente aggiuntivo le autorizzazioni IAM se configuri una posizione di output. AWS Per ulteriori informazioni, consulta the section called "Utilizzo di Pod Identities".

Proprietà	Valore
Nome del componente aggiuntivo	`fluent-bit`
Namespace	`fluent-bit`
Documentazione	Fluent Bit Documentation

Proprietà	Valore
Nome account del servizio	`fluent-bit`
Policy IAM gestite	Nessuno
Autorizzazioni IAM personalizzate	Nessuno

Visualizzare le attribuzioni

Puoi scaricare le attribuzioni open source e le informazioni sulle licenze per i componenti aggiuntivi della community.

1. Determina il nome e la versione del componente aggiuntivo per il quale desideri scaricare le attribuzioni.
2. Aggiorna il seguente comando con il nome e la versione:

   ```
   curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/<add-on-name>/<add-on-version>/attributions.zip
   ```

   Esempio:

   ```
   curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/kube-state-metrics/v2.14.0-eksbuild.1/attributions.zip
   ```
3. Utilizza il comando per scaricare il file.

Utilizza questo file zip per visualizzare informazioni sulle attribuzioni delle licenze.

AWS Componenti aggiuntivi del Marketplace

Oltre all'elenco precedente di componenti aggiuntivi di Amazon EKS, puoi anche aggiungere un'ampia selezione di componenti software aggiuntivi operativi per Amazon EKS offerti da fornitori di software indipendenti. Scegli un componente aggiuntivo per avere maggiori informazioni sul componente e sui relativi requisiti di installazione.

Accuknox

Il nome del componente aggiuntivo è `accuknox_kubearmor` e il namespace è `kubearmor`. Accuknox pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta la sezione Guida introduttiva KubeArmor nella KubeArmor documentazione.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Akuity

Il nome del componente aggiuntivo è `akuity_agent` e il namespace è `akuity`. Akuity pubblica il componente aggiuntivo.

Per informazioni su come funziona il componente aggiuntivo, consulta Installazione dell'agente Akuity su Amazon EKS con il componente aggiuntivo Akuity EKS nella documentazione della piattaforma Akuity.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Calyptia

Il nome del componente aggiuntivo è `calyptia_fluent-bit` e il namespace è `calytia-fluentbit`. Calyptia pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Guida introduttiva all'agente Calyptia Core sul sito Web della documentazione di Calyptia.

Nome account del servizio

Il nome dell'account del servizio è `clyptia-fluentbit`.

AWS politica IAM gestita

Questo componente aggiuntivo utilizza la policy gestita da `AWSMarketplaceMeteringRegisterUsage`. Per ulteriori informazioni, consulta AWSMarketplaceMeteringRegisterUsagela AWS Managed Policy Reference Guide.

Comando per creare il ruolo IAM richiesto

Il comando seguente richiede la presenza di un fornitore OpenID Connect (OIDC) di IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". Sostituisci *`my-cluster`* con il nome del cluster e *`my-calyptia-role`* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo e prendere nota dell'account di servizio Kubernetes, consulta the section called "Assegnazione di un ruolo 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
```

Observability Collector di Cisco

Il nome del componente aggiuntivo è `cisco_cisco-cloud-observability-collectors` e il namespace è `appdynamics`. Cisco pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Use the Cisco Cloud Observability AWS Marketplace Add-Ons nella documentazione Cisco. AppDynamics

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Cisco Observability Operator

Il nome del componente aggiuntivo è `cisco_cisco-cloud-observability-operators` e il namespace è `appdynamics`. Cisco pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Use the Cisco Cloud Observability AWS Marketplace Add-Ons nella documentazione Cisco. AppDynamics

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

CLOUDSOFT

Il nome del componente aggiuntivo è `cloudsoft_cloudsoft-amp` e il namespace è `cloudsoft-amp`. CLOUDSOFT pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Componente aggiuntivo di Amazon EKS nella documentazione CLOUDSOFT.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Cribl

Il nome del componente aggiuntivo è `cribl_cribledge` e il namespace è `cribledge`. Cribl pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione del componente aggiuntivo Cribl di Amazon EKS per Edge nella documentazione di Cribl

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Dynatrace

Il nome del componente aggiuntivo è `dynatrace_dynatrace-operator` e il namespace è `dynatrace`. Dynatrace pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Monitoraggio di Kubernetes nella documentazione di Dynatrace.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Datree

Il nome del componente aggiuntivo è `datree_engine-pro` e il namespace è `datree`. Datree pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Integrazione di Amazon EKS nella documentazione Datree.

Nome account del servizio

Il nome dell'account del servizio è datree-webhook-server-awsmp.

AWS politica IAM gestita

La politica gestita è AWSLicenseManagerConsumptionPolicy. Per ulteriori informazioni, consulta AWSLicenseManagerConsumptionPolicyla AWS Managed Policy Reference Guide.

Comando per creare il ruolo IAM richiesto

Il comando seguente richiede la presenza di un fornitore OpenID Connect (OIDC) di IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". Sostituisci *my-cluster* con il nome del cluster e *my-datree-role* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo e prendere nota dell'account di servizio Kubernetes, consulta the section called "Assegnazione di un ruolo 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
```

Autorizzazioni personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Datadog

Il nome del componente aggiuntivo è `datadog_operator` e il namespace è `datadog-agent`. Datadog pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione dell'agente Datadog su Amazon EKS con il componente aggiuntivo Datadog Operator nella documentazione di Datadog.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Groundcover

Il nome del componente aggiuntivo è `groundcover_agent` e il namespace è`groundcover`. Groundcover pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione del componente aggiuntivo Groundcover di Amazon EKS nella documentazione di Groundcover.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

IBM Instana

Il nome del componente aggiuntivo è `instana-agent` e il namespace è `instana-agent`. IBM pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Implementazione dell'osservabilità per i carichi di lavoro Amazon EKS utilizzando il componente aggiuntivo Amazon EKS di Instana e Monitoraggio e ottimizzazione dei costi di Amazon EKS con IBM Instana e Kubecost nel blog. AWS

Instana Observability (Instana) offre un componente aggiuntivo Amazon EKS che implementa agenti Instana nei cluster Amazon EKS. I clienti possono utilizzare questo componente aggiuntivo per raccogliere e analizzare dati sulle prestazioni in tempo reale per ottenere approfondimenti sulle loro applicazioni containerizzate. Il componente aggiuntivo Instana Amazon EKS offre visibilità su tutti

gli ambienti Kubernetes. Una volta distribuito, l'agente Instana rileva automaticamente i componenti all'interno dei cluster Amazon EKS, inclusi nodi, namespace, implementazioni, servizi e pod.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Grafana Labs

Il nome del componente aggiuntivo è `grafana-labs_kubernetes-monitoring` e il namespace è `monitoring`. Grafana Labs pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Configura il monitoraggio di Kubernetes come componente aggiuntivo con Amazon EKS nella documentazione di Grafana Labs.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Guance

- Publisher: GUANCE
- Nome: `guance_datakit`
- Namespace: `datakit`
- Nome account di servizio: con questo componente aggiuntivo non è utilizzato un account di servizio.

- Policy IAM gestita da AWS : con questo componente aggiuntivo non è utilizzata una policy gestita.
- Autorizzazioni IAM personalizzate: con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.
- Istruzioni di configurazione e utilizzo: consulta Using Amazon EKS add-on nella documentazione di Guance.

Proxy HA

Il nome è `haproxy-technologies_kubernetes-ingress-ee` e il namespace è `haproxy-controller`. HA Proxy pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Integrazione di Amazon EKS nella documentazione Datree.

Nome account del servizio

Il nome dell'account del servizio è `customer defined`.

AWS politica IAM gestita

La politica gestita è AWSLicenseManagerConsumptionPolicy. Per ulteriori informazioni, consulta AWSLicenseManagerConsumptionPolicyla AWS Managed Policy Reference Guide.

Comando per creare il ruolo IAM richiesto

Il comando seguente richiede la presenza di un fornitore OpenID Connect (OIDC) di IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". Sostituisci *`my-cluster`* con il nome del cluster e *`my-haproxy-role`* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo e prendere nota dell'account di servizio Kubernetes, consulta the section called "Assegnazione di un ruolo 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
```

Autorizzazioni personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Kpow

Il nome del componente aggiuntivo è `factorhouse_kpow` e il namespace è `factorhouse`. Factorhouse pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta AWS Marketplace LM nella documentazione di Kpow.

Nome account del servizio

Il nome dell'account del servizio è `kpow`.

AWS politica IAM gestita

La politica gestita è AWSLicenseManagerConsumptionPolicy. Per ulteriori informazioni, consulta AWSLicenseManagerConsumptionPolicyla AWS Managed Policy Reference Guide.

Comando per creare il ruolo IAM richiesto

Il comando seguente richiede la presenza di un fornitore OpenID Connect (OIDC) di IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". Sostituisci *`my-cluster`* con il nome del cluster e *`my-kpow-role`* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo e prendere nota dell'account di servizio Kubernetes, consulta the section called "Assegnazione di un ruolo 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
```

Autorizzazioni personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Kubecost

Il nome del componente aggiuntivo è `kubecost_kubecost` e il namespace è `kubecost`. Kubecost pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Integrazione di AWS Cloud Billing nella documentazione di Kubecost.

Nel tuo cluster deve essere installato Archivia volumi Kubernetes con Amazon EBS, altrimenti comparirà un messaggio di errore.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Kasten

Il nome del componente aggiuntivo è `kasten_k10` e il namespace è `kasten-io`. Kasten by Veeam pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione di K10 sull'utilizzo del componente aggiuntivo AWS Amazon EKS nella documentazione di Kasten.

È necessario che il driver CSI di Amazon EBS sia installato nel cluster `StorageClass` predefinito.

Nome account del servizio

Il nome dell'account del servizio è `k10-k10`.

AWS politica IAM gestita

La politica gestita è AWSLicenseManagerConsumptionPolicy. Per ulteriori informazioni, consulta AWSLicenseManagerConsumptionPolicyla AWS Managed Policy Reference Guide.

Comando per creare il ruolo IAM richiesto

Il comando seguente richiede la presenza di un fornitore OpenID Connect (OIDC) di IAM per il cluster. Per stabilire se ne possiedi uno o per crearne uno, consulta the section called "Gestore di identità OIDC IAM". Sostituisci *`my-cluster`* con il nome del cluster e *`my-kasten-role`* con il nome del ruolo. Questo comando richiede che eksctl sia installato sul tuo dispositivo. Se devi utilizzare uno strumento diverso per creare il ruolo e prendere nota dell'account di servizio Kubernetes, consulta the section called "Assegnazione di un ruolo 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
```

Autorizzazioni personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Kong

Il nome del componente aggiuntivo è `kong_konnect-ri` e il namespace è `kong`. Kong pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione del componente aggiuntivo Kong Gateway di EKS nella documentazione di Kong.

Nel tuo cluster deve essere installato Archivia volumi Kubernetes con Amazon EBS, altrimenti comparirà un messaggio di errore.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

LeakSignal

Il nome del componente aggiuntivo è `leaksignal_leakagent` e il namespace è. `leakagent` LeakSignal pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, vedere https://www.leaksignal.com/docs/ LeakAgent / Deployment/AWS%20Eks%20Addon/ [Installare il componente aggiuntivo] nella documentazione LeakAgent LeakSignal

Nel tuo cluster deve essere installato Archivia volumi Kubernetes con Amazon EBS, altrimenti comparirà un messaggio di errore.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

NetApp

Il nome del componente aggiuntivo è `netapp_trident-operator` e il namespace è. `trident` NetApp pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Configurare il componente aggiuntivo Trident EKS nella documentazione. NetApp

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

New Relic

Il nome del componente aggiuntivo è `new-relic_kubernetes-operator` e il namespace è `newrelic`. New Relic pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione del componente aggiuntivo New Relic per EKS nella documentazione di New Relic.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Rafay

Il nome del componente aggiuntivo è `rafay-systems_rafay-operator` e il namespace è `rafay-system`. Rafay pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione del componente aggiuntivo Rafay di Amazon EKS nella documentazione di Rafay.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Rad Security

- Editore: RAD SECURITY
- Nome: `rad-security_rad-security`
- Namespace: `ksoc`
- Nome account di servizio: con questo componente aggiuntivo non è utilizzato un account di servizio.
- Policy IAM gestita da AWS : con questo componente aggiuntivo non è utilizzata una policy gestita.
- Autorizzazioni IAM personalizzate: con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.
- Istruzioni di configurazione e utilizzo: vedi Installazione di Rad tramite The AWS Marketplace nella documentazione di Rad Security.

SolarWinds

- Publisher: SOLARWINDS
- Nome: `solarwinds_swo-k8s-collector-addon`
- Namespace: `solarwinds`
- Nome account di servizio: con questo componente aggiuntivo non è utilizzato un account di servizio.
- Policy IAM gestita da AWS : con questo componente aggiuntivo non è utilizzata una policy gestita.
- Autorizzazioni IAM personalizzate: con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.
- Istruzioni di configurazione e utilizzo: consulta Monitorare un cluster Amazon EKS nella SolarWinds documentazione.

Solo

Il nome del componente aggiuntivo è `solo-io_istio-distro` e il namespace è `istio-system`. Solo pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione di Istio nella documentazione di Solo.io.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Snyk

- Publisher: SNYK
- Nome: `snyk_runtime-sensor`
- Namespace: `snyk_runtime-sensor`

- Nome account di servizio: con questo componente aggiuntivo non è utilizzato un account di servizio.
- Policy IAM gestita da AWS : con questo componente aggiuntivo non è utilizzata una policy gestita.
- Autorizzazioni IAM personalizzate: con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.
- Istruzioni di configurazione e utilizzo: consulta Sensore di runtime Snyk nei documenti per gli utenti di Snyk.

Stormforge

Il nome del componente aggiuntivo è `stormforge_optimize-Live` e il namespace è `stormforge-system`. Stormforge pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installazione dell' StormForge agente nella StormForge documentazione.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

SUSE

- Publisher: SUSE
- Nome: `suse_observability-agent`
- Namespace: `suse-observability`
- Nome account di servizio: con questo componente aggiuntivo non è utilizzato un account di servizio.
- Policy IAM gestita da AWS : con questo componente aggiuntivo non è utilizzata una policy gestita.
- Autorizzazioni IAM personalizzate: con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

- Istruzioni per la configurazione e l'uso: consulta Avvio rapido nella documentazione di SUSE.

Splunk

Il nome del componente aggiuntivo è `splunk_splunk-otel-collector-chart` e il namespace è `splunk-monitoring`. Splunk pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Installa il componente aggiuntivo di Splunk per Amazon EKS nella documentazione di Splunk.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Teleport

Il nome del componente aggiuntivo è `teleport_teleport` e il namespace è `teleport`. Teleport pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Come funziona Teleport nella documentazione di Teleport.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Tetrate

Il nome del componente aggiuntivo è `tetrate-io_istio-distro` e il namespace è `istio-system`. Tetrate lo pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta il sito Web Tetrate Istio Distro.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Upbound Universal Crossplane

Il nome del componente aggiuntivo è `upbound_universal-crossplane` e il namespace è `upbound-system`. Upbound pubblica il componente aggiuntivo.

Per informazioni sul componente aggiuntivo, consulta Upbound Universal Crossplane (UXP) nella documentazione di Upbound.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Upwind

Il nome del componente aggiuntivo è `upwind` e il namespace è `upwind`. Upwind pubblica il componente aggiuntivo.

Per ulteriori informazioni sul componente aggiuntivo, consulta Documentazione di Upwind.

Nome account del servizio

Con questo componente aggiuntivo non è utilizzato un account di servizio.

AWS politica IAM gestita

Con questo componente aggiuntivo non è utilizzata una policy gestita.

Autorizzazioni IAM personalizzate

Con questo componente aggiuntivo non sono utilizzate autorizzazioni personalizzate.

Creare un componente aggiuntivo Amazon EKS

I componenti aggiuntivi Amazon EKS sono software aggiuntivi per cluster Amazon EKS. Tutti i componenti aggiuntivi di Amazon EKS:

- Includono le patch di sicurezza e le correzioni dei bug più recenti.
- Sono convalidati AWS per funzionare con Amazon EKS.
- Riducono la quantità di lavoro necessaria per gestire il software del componente aggiuntivo.

Puoi creare un componente aggiuntivo Amazon EKS utilizzando `eksctl` Console di gestione AWS, o la AWS CLI. Se il componente aggiuntivo richiede un ruolo IAM, consultare i dettagli del componente aggiuntivo specifico alla pagina Amazon EKS add-ons per conoscere i dettagli sulla creazione del ruolo.

Prerequisiti

Completare le seguenti operazioni prima di creare un componente aggiuntivo:

- Il cluster deve esistere prima di poter creare un componente aggiuntivo. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ".
- Verificare se il componente aggiuntivo richiede un ruolo IAM. Per ulteriori informazioni, consulta the section called "Verifica la compatibilità".
- Verificare che la versione del componente aggiuntivo Amazon EKS sia compatibile con il cluster. Per ulteriori informazioni, consulta the section called "Verifica la compatibilità".
- Verifica che la versione 0.190.0 o successiva dello strumento da riga di `eksctl` comando sia installata sul tuo computer o. AWS CloudShell Per ulteriori informazioni, vedere Installazione sul sito Web `eksctl`.

Procedura

Puoi creare un componente aggiuntivo Amazon EKS utilizzando `eksctl` Console di gestione AWS, o la AWS CLI. Se il componente aggiuntivo richiede un ruolo IAM, consultare i dettagli del componente aggiuntivo specifico alla pagina Available Amazon EKS add-ons from AWS per conoscere i dettagli sulla creazione del ruolo.

Creare un componente aggiuntivo (eksctl)

1. Visualizza i nomi dei componenti aggiuntivi disponibili per una versione del cluster. Sostituisci *1.33* con la versione del cluster.

   ```
   eksctl utils describe-addon-versions --kubernetes-version 1.33 | grep AddonName
   ```

   Di seguito viene riportato un output di esempio:

   ```
   "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. Visualizza le versioni disponibili per il componente aggiuntivo da creare. Sostituisci *1.33* con la versione del cluster. Sostituisci *name-of-addon* con il nome del componente aggiuntivo di cui visualizzare le versioni. Il nome deve essere uno di quelli restituiti al passaggio precedente.

   ```
   eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-addon |
    grep AddonVersion
   ```

   L'output seguente è un esempio di ciò che viene restituito per il componente aggiuntivo denominato `vpc-cni`. Come vedi, il componente aggiuntivo ha diverse versioni disponibili.

   ```
   "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. Determina se il componente aggiuntivo che desideri creare è un componente aggiuntivo Amazon EKS o AWS Marketplace. Il AWS Marketplace offre componenti aggiuntivi di terze parti che richiedono il completamento di passaggi aggiuntivi per creare il componente aggiuntivo.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-addon | grep ProductUrl
```

Se non viene restituito alcun output, il componente aggiuntivo è Amazon EKS. Se viene restituito un output, il componente aggiuntivo è un componente aggiuntivo AWS Marketplace. Il seguente output viene fornito per un componente aggiuntivo denominato `teleport_teleport`.

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?sku=3bda70bb-566f-4976-806c-f96faef18b26"
```

Puoi trovare ulteriori informazioni sul componente aggiuntivo nel AWS Marketplace con l'URL restituito. Se il componente aggiuntivo richiede un abbonamento, puoi abbonarti al componente aggiuntivo tramite Marketplace AWS . Se intendi creare un componente aggiuntivo dal AWS Marketplace, il responsabile IAM che stai utilizzando per creare il componente aggiuntivo deve avere l'autorizzazione per creare il ruolo collegato al AWSServiceRoleForAWSLicenseManagerRoleservizio. Per ulteriori informazioni sull'assegnazione di autorizzazioni a un'entità IAM, consulta Aggiunta e rimozione di autorizzazioni per identità IAM nella Guida per l'utente di IAM.

3. Crea un componente aggiuntivo di Amazon EKS. Copia il comando e sostituiscilo come segue: *`user-data`*
   - Sostituisci *`my-cluster`* con il nome del cluster.
   - Sostituisci *`name-of-addon`* con il nome del componente aggiuntivo da creare.
   - Se desideri una versione del componente aggiuntivo precedente all'ultima versione, sostituiscila *`latest`* con il numero di versione restituito nell'output di un passaggio precedente che desideri utilizzare.
   - Se il componente aggiuntivo utilizza un ruolo dell'account di servizio, sostituisci *`111122223333`* con l'ID dell'account e sostituisci *`role-name`* con il nome del ruolo. Per istruzioni sulla creazione di un ruolo per l'account di servizio, consultare la documentazione del componente aggiuntivo da

creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per specificare un ruolo per l'account di servizio è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

Se il componente aggiuntivo non utilizza un ruolo per l'account di servizio, eliminare `--service-account-role-arn arn:aws: iam::111122223333:role/role-name`.

- Questo comando di esempio sovrascrive la configurazione di qualunque versione autogestita esistente del componente aggiuntivo, se presente. Se non desideri sovrascrivere la configurazione di un componente aggiuntivo autogestito esistente, rimuovi l'opzione. *`--force`* Se rimuovi l'opzione e il componente aggiuntivo di Amazon EKS deve sovrascrivere la configurazione di un componente aggiuntivo autogestito esistente, la creazione del componente aggiuntivo di Amazon EKS non riesce e viene visualizzato un messaggio di errore per semplificare la soluzione del conflitto. Prima di specificare questa opzione, assicurarsi che il componente aggiuntivo Amazon EKS non gestisca le impostazioni da gestire autonomamente, perché questa opzione le sovrascrive.

```
eksctl create addon --cluster my-cluster --name name-of-addon --version latest \
    --service-account-role-arn arn:aws: iam::111122223333:role/role-name --force
```

Puoi vedere un elenco completo di tutte opzioni disponibili per il comando.

```
eksctl create addon --help
```

Per ulteriori informazioni sulle opzioni disponibili, consulta Componenti aggiuntivi nella documentazione di `eksctl`.

Crea componente aggiuntivo (console)AWS

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Scegliere il nome del cluster per cui si desidera creare il componente aggiuntivo.
4. Selezionare la scheda Componenti aggiuntivi.
5. Scegli Ottieni altri componenti aggiuntivi.

6. Nella pagina Select add-ons (Seleziona componenti aggiuntivi), scegli i componenti aggiuntivi da aggiungere al tuo cluster. Puoi aggiungere tutti i componenti aggiuntivi Amazon EKS e i componenti aggiuntivi AWS Marketplace di cui hai bisogno.

   Per i componenti aggiuntivi di AWS Marketplace, il responsabile IAM che stai utilizzando per creare il componente aggiuntivo deve disporre delle autorizzazioni per leggere i diritti per il componente aggiuntivo da. AWS LicenseManager AWS LicenseManager richiede un ruolo AWSServiceRoleForAWSLicenseManagerRolecollegato al servizio (SLR) che consente alle risorse di gestire le licenze per tuo conto. AWS L'SLR è un requisito una tantum per singolo account e non sarà necessario creare SLR separati per ciascun componente aggiuntivo o per ciascun cluster. Per ulteriori informazioni sull'assegnazione di autorizzazioni a un principale IAM, consulta Aggiunta e rimozione di autorizzazioni per identità IAM nella Guida per l'utente di IAM.

   Se i componenti aggiuntivi del AWS Marketplace che desideri installare non sono elencati, puoi fare clic sulla numerazione delle pagine per visualizzare altri risultati di pagina o effettuare una ricerca nella casella di ricerca. Nelle Opzioni di filtro, puoi eseguire una ricerca anche per categoria, fornitore o modello di prezzi e quindi scegliere i componenti aggiuntivi dai risultati della ricerca. Dopo aver selezionato i componenti aggiuntivi da installare, scegliere Avanti.

7. Nella pagina Configura le impostazioni dei componenti aggiuntivi selezionati, procedi come segue:

   a. Scegli Visualizza le opzioni di abbonamento per aprire il modulo delle opzioni di abbonamento. Consulta le sezioni Dettagli sui prezzi e Note legali, quindi scegli il pulsante Abbonati per continuare.

   b. Per Versione, selezionare la versione da installare. È preferibile la versione contrassegnata come latest, a meno che il singolo componente aggiuntivo da creare non consigli una versione differente. Per determinare se un componente aggiuntivo ha una versione consigliata, consultare la documentazione del componente aggiuntivo da creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi".

   c. Sono disponibili due opzioni per configurare i ruoli per i componenti aggiuntivi: il ruolo IAM di EKS Pod Identity e i ruoli IAM per gli account di servizio (IRSA). Seguire la procedura adeguata di seguito per l'opzione preferita. Se tutti i componenti aggiuntivi selezionati includono Richiede un abbonamento in Stato, scegliere Avanti. Non è possibile procedere con la configurazione di questi componenti aggiuntivi finché non sarà stato sottoscritto l'abbonamento dopo la creazione del cluster. Per i componenti aggiuntivi che non includono Richiede un abbonamento in Stato:

      i. Per il ruolo IAM di Pod Identity per l'account di servizio, è possibile utilizzare un ruolo IAM di EKS Pod Identity esistente o crearne uno con il pulsante Crea ruolo consigliato. Questo campo indicherà solo le opzioni con la policy di attendibilità adeguata. Se non esiste alcun

ruolo da selezionare, non si dispone di un ruolo esistente con una policy di attendibilità corrispondente. Per configurare un ruolo IAM di EKS Pod Identity per gli account di servizio del componente aggiuntivo selezionato, scegliere Crea ruolo consigliato. La procedura guidata per la creazione dei ruoli si apre in una finestra separata. La procedura guidata compilerà automaticamente le informazioni sul ruolo come indicato di seguito. Per ogni componente aggiuntivo in cui si desidera creare il ruolo IAM di EKS Pod Identity, completare i passaggi della procedura guidata IAM come indicato di seguito.

- Nel passaggio Seleziona un'entità affidabile, l'opzione di AWS servizio per EKS e lo use case per EKS - Pod Identity sono preselezionate e la politica di attendibilità appropriata verrà compilata automaticamente per il componente aggiuntivo. Ad esempio, il ruolo verrà creato con la policy di attendibilità adeguata contenente il principale IAM pods.eks.amazonaws.com, come descritto nella pagina the section called "Vantaggi delle associazioni EKS Pod Identity". Scegli Next (Successivo).
- Al passaggio Aggiungi autorizzazioni, viene preselezionata la policy gestita adeguata per la policy del ruolo per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo CNI di Amazon VPC, il ruolo verrà creato con la policy gestita `AmazonEKS_CNI_Policy`, come descritto nella pagina the section called "Plug-in della CNI di Amazon VPC per Kubernetes". Scegli Next (Successivo).
- Al passaggio Nomina, verifica e crea, in Nome del ruolo, viene inserito automaticamente il nome del ruolo predefinito per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo Amazon VPC CNI, il ruolo verrà creato con il nome Amazon. EKSPod IdentityAmazon VPCCNIRole In Descrizione, la descrizione predefinita viene compilata automaticamente con la descrizione adeguata per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo Amazon VPC CNI, il ruolo verrà creato con la descrizione Permette ai pod in esecuzione nel cluster Amazon EKS di accedere alle risorse. AWS In Policy di attendibilità, viene visualizzata la policy di attendibilità compilata per il componente aggiuntivo. Scegli Crea ruolo.

  NOTA: mantenere il nome del ruolo predefinito consente a EKS di preselezionare il ruolo dei componenti aggiuntivi nei nuovi cluster o quando si aggiungono componenti aggiuntivi ai cluster esistenti. È comunque possibile sostituire tale nome e il ruolo sarà disponibile per il componente aggiuntivo in tutti i cluster, ma dovrà essere selezionato manualmente dal menu a discesa.

ii. Per i componenti aggiuntivi che non includono Richiede un abbonamento in Stato e per i quali si desidera configurare i ruoli utilizzando IRSA, consultare la documentazione relativa al componente aggiuntivo da creare per definire una policy IAM e collegarla a un ruolo.

Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per selezionare un ruolo IAM è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

iii. Scegli Optional configuration settings (Impostazioni di configurazione facoltative).

iv. Se il componente aggiuntivo richiede una configurazione, inseriscila nella casella Configuration values (Valori di configurazione). Per determinare se il componente aggiuntivo richiede informazioni di configurazione, consultare la documentazione del componente aggiuntivo da creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi".

v. Selezionare una delle opzioni disponibili per Metodo di risoluzione dei conflitti. Se si sceglie Sostituisci come Metodo di risoluzione dei conflitti, una o più impostazioni per il componente aggiuntivo esistente possono essere sovrascritte con le impostazioni del componente aggiuntivo Amazon EKS. Se non si abilita questa opzione e c'è un conflitto con le impostazioni esistenti, l'operazione non va a buon fine e viene visualizzato un messaggio di errore per aiutarti a risolvere il conflitto. Prima di scegliere questa opzione, assicurarsi che il componente aggiuntivo Amazon EKS non gestisca le impostazioni che devono essere autogestite.

vi. Se si desidera installare il componente aggiuntivo in un namespace specifico, inserirlo nel campo Namespace. Per AWS i componenti aggiuntivi della community, puoi definire uno spazio dei nomi Kubernetes personalizzato in cui installare il componente aggiuntivo. Per ulteriori informazioni, consulta the section called "Namespace personalizzato per componenti aggiuntivi".

vii. Scegli Next (Successivo).

8. Nella pagina Rivedi e aggiungi, scegli Crea. Una volta completata l'installazione del componente aggiuntivo, vengono visualizzati i componenti aggiuntivi installati.
9. Se uno dei componenti aggiuntivi installati richiede una sottoscrizione, completa la procedura seguente:
   a. scegli il pulsante Subscribe (Sottoscrivi) nell'angolo inferiore destro del componente aggiuntivo. Verrai indirizzato alla pagina del componente aggiuntivo nel AWS Marketplace. Leggi le informazioni sul componente aggiuntivo, ad esempio Product Overview (Panoramica del prodotto) e Pricing information (Informazioni sui prezzi).
   b. Seleziona il pulsante Continue to Subscribe (Continua la sottoscrizione) nella parte superiore destra della pagina del componente aggiuntivo.

c. Leggi i Terms and Conditions (Termini e condizioni). Se li accetti, scegli Accept Terms (Accetta i termini). L'elaborazione della sottoscrizione può richiedere vari minuti. Durante l'elaborazione della sottoscrizione, il pulsante Return to Amazon EKS Console (Torna alla console Amazon EKS) è disattivato.

d. Al termine dell'elaborazione dell'abbonamento, il pulsante Return to Amazon EKS Console (Torna alla console Amazon EKS) non è più disattivato. Scegli il pulsante per tornare alla scheda Add-ons (Componenti aggiuntivi) della console Amazon EKS per il tuo cluster.

e. Per il componente aggiuntivo che hai sottoscritto, scegli Remove and reinstall (Rimuovi e reinstalla), quindi scegli Reinstall add-on (Reinstalla componente aggiuntivo). L'installazione del componente aggiuntivo può richiedere vari minuti. Una volta completata l'installazione, puoi configurare il componente aggiuntivo.

Crea componente aggiuntivo (AWS CLI)

1. È necessaria la versione `2.12.3` o una versione successiva `1.27.160` o una versione o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
2. Determina quali componenti aggiuntivi sono disponibili. Puoi vedere tutti i componenti aggiuntivi disponibili, incluso il tipo e l'autore. Puoi anche visualizzare l'URL dei componenti aggiuntivi disponibili tramite AWS Marketplace. Sostituisci *`1.33`* con la versione del 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
```

Di seguito viene riportato un output di esempio:

```
-----------------------------------------------------------------------------------------
```

```
|
 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         |
|  [...]                                                        |  [...]
          |  [...]           |  [...]       |  [...]              |
+-------------------------------------------------------------
+------------------------------+------------------+-------------
+---------------------+
```

Il tuo output potrebbe essere diverso. In questo output di esempio, sono disponibili tre diversi componenti aggiuntivi di tipo `networking` e cinque componenti aggiuntivi con un autore di tipo `eks`. I componenti aggiuntivi con `aws-marketplace` nella colonna `0wner` potrebbero richiedere un abbonamento per poter essere installati. Per ulteriori informazioni sul componente aggiuntivo e per sottoscriverlo, puoi visitare l'URL.

3. Puoi vedere quali versioni sono disponibili per ogni componente aggiuntivo. Sostituisci *1.33* con la versione del tuo cluster, quindi sostituisci *vpc-cni* con il nome del componente aggiuntivo restituito nella fase precedente.

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

Di seguito viene riportato un output di esempio:

```
-----------------------------------------
|          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   |
+-----------------+----------------------+
```

Per impostazione predefinita, la versione con `True` nella colonna `Defaultversion` è la versione con cui viene creato il componente aggiuntivo.

4. (Facoltativo) Trova le opzioni di configurazione del componente aggiuntivo scelto utilizzando il comando seguente:

```
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\"}}}"
}
```

L'output sarà uno schema JSON standard.

Di seguito è riportato un esempio di valori di configurazione validi, in formato JSON, che funzionano con lo schema precedente.

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
}
```

Di seguito è riportato un esempio di valori di configurazione validi, in formato YAML, che funzionano con lo schema precedente.

```
  resources:
    limits:
```

```
        cpu: 100m
```

5. Stabilire se il componente aggiuntivo richiede autorizzazioni IAM. In tal caso, è necessario (1) stabilire se si desidera utilizzare EKS Pod Identity o ruoli IAM per gli account di servizio (IRSA), (2) stabilire l'ARN del ruolo IAM da utilizzare con il componente aggiuntivo e (3) stabilire il nome dell'account di servizio Kubernetes usato dal componente aggiuntivo. Per ulteriori informazioni, consulta the section called "Recupero delle informazioni IAM".

   - Amazon EKS suggerisce di utilizzare EKS Pod Identity, se supportato dal componente aggiuntivo. Ciò richiede che l'agente Pod Identity sia installato nel cluster. Per ulteriori informazioni sull'utilizzo di Pod Identity con i componenti aggiuntivi, consultare la pagina the section called "Ruoli IAM".
   - Se il componente aggiuntivo o il cluster non sono configurati per EKS Pod Identity, utilizzare IRSA. Confermare che IRSA sia configurato sul cluster.
   - Consultare la documentazione dei componenti aggiuntivi Amazon EKS per stabilire se il componente aggiuntivo richiede le autorizzazioni IAM e il nome dell'account di servizio Kubernetes associato.

     a. Crea un componente aggiuntivo di Amazon EKS. Copia il comando seguente sul tuo dispositivo. Apportare le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato:

   - Sostituisci *my-cluster* con il nome del cluster.
   - Sostituisci *vpc-cni* con il nome del componente aggiuntivo da creare restituito nell'output del passaggio precedente.
   - Sostituisci *version-number* con il nome del componente aggiuntivo da utilizzare restituito nell'output della fase precedente.
   - Per installare il componente aggiuntivo in un namespace Kubernetes personalizzato, aggiungere l'opzione `--namespace-config 'namespace=<my-namespace>`. Questa opzione è disponibile solo per i AWS componenti aggiuntivi della community. Per ulteriori informazioni, consulta the section called "Namespace personalizzato per componenti aggiuntivi"
   - Se il componente aggiuntivo non richiede le autorizzazioni IAM, elimina. *<service-account-configuration>*
   - Esegui una delle seguenti operazioni:
     - Se il componente aggiuntivo (1) richiede le autorizzazioni IAM e (2) il cluster utilizza EKS Pod Identities, sostituiscilo *<service-account-configuration>* con la seguente associazione di identità del pod. Sostituisci *<service-account-name>* con il nome dell'account di servizio utilizzato dal componente aggiuntivo. Sostituisci *<role-arn>* con

l'ARN di un ruolo IAM. Il ruolo deve disporre della policy di attendibilità richiesta da EKS Pod Identity.

```
--pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
```

- Se il componente aggiuntivo (1) richiede autorizzazioni IAM e (2) il cluster utilizza IRSA, sostituiscilo *<service-account-configuration>* con la seguente configurazione IRSA. *111122223333*Sostituiscilo con l'ID del tuo account e *role-name* con il nome di un ruolo IAM esistente che hai creato. Per istruzioni sulla creazione del ruolo, consultare la documentazione del componente aggiuntivo da creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per specificare un ruolo per l'account di servizio è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

```
--service-account-role-arn arn:aws::iam::111122223333:role/role-name
```

- Questo comando di esempio sovrascrive l'opzione `--configuration-values` di qualunque versione autogestita esistente del componente aggiuntivo, se presente. Sostituiscilo con i valori di configurazione desiderati, come una stringa o un file di input. Se non si desidera fornire valori di configurazione, eliminare l'opzione `--configuration-values`. Se non desideri che la AWS CLI sovrascriva la configurazione di un componente aggiuntivo autogestito esistente, rimuovi l'opzione. *--resolve-conflicts OVERWRITE* Se rimuovi l'opzione e il componente aggiuntivo di Amazon EKS deve sovrascrivere la configurazione di un componente aggiuntivo autogestito esistente, la creazione del componente aggiuntivo di Amazon EKS non riesce e viene visualizzato un messaggio di errore per semplificare la soluzione del conflitto. Prima di specificare questa opzione, assicurarsi che il componente aggiuntivo Amazon EKS non gestisca le impostazioni da gestire autonomamente, perché questa opzione le sovrascrive.

```
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
```

Per un elenco completo delle opzioni disponibili, consulta `create-addon` nella Documentazione di riferimento della riga di comando Amazon EKS. Se il componente aggiuntivo creato include `aws-marketplace` elencato nella colonna `Owner` di un passaggio precedente, la creazione potrebbe non riuscire e si potrebbe ricevere un messaggio di errore simile al seguente.

```
{
    "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 ricevi un errore simile a quello nell'output precedente, visita l'URL nell'output di un passaggio precedente per sottoscrivere il componente aggiuntivo. Dopo la sottoscrizione, esegui nuovamente il comando `create-addon`.

Aggiornamento di un componente aggiuntivo di Amazon EKS

Amazon EKS non aggiorna automaticamente un componente aggiuntivo quando sono rilasciate nuove versioni o dopo l'aggiornamento del cluster a una nuova versione secondaria di Kubernetes. Per aggiornare un componente aggiuntivo per un cluster esistente, devi avviare l'aggiornamento. Dopo l'avvio dell'aggiornamento, Amazon EKS aggiorna automaticamente il componente aggiuntivo. Prima di aggiornare un componente aggiuntivo, consulta la documentazione attuale del componente

aggiuntivo. Per un elenco dei componenti aggiuntivi disponibili, consulta the section called " AWS componenti aggiuntivi". Se il componente aggiuntivo richiede un ruolo IAM, consulta i dettagli del componente aggiuntivo specifico in Componenti aggiuntivi di Available Amazon EKS da AWS per conoscere i dettagli sulla creazione del ruolo.

Prerequisiti

Completa le seguenti operazioni prima di creare un componente aggiuntivo:

- Verifica se il tuo componente aggiuntivo richiede un ruolo IAM. Per ulteriori informazioni, consulta the section called "Componenti aggiuntivi Amazon EKS".
- Verifica che la versione del componente aggiuntivo Amazon EKS sia compatibile con il cluster. Per ulteriori informazioni, consulta the section called "Verifica la compatibilità".

Procedura

Puoi aggiornare un componente aggiuntivo Amazon EKS utilizzando `eksctl` Console di gestione AWS, o la AWS CLI.

Aggiornamento del componente aggiuntivo (eksctl)

1. Determina i componenti aggiuntivi e le versioni dei componenti aggiuntivi installati sul tuo cluster. Sostituisci *`my-cluster`* con il nome del cluster.

   ```
   eksctl get addon --cluster my-cluster
   ```

   Di seguito viene riportato un output di esempio:

   ```
   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
   ```

   L'output potrebbe avere un aspetto diverso in base ai componenti aggiuntivi e alle versioni presenti nel tuo cluster. Nell'output di esempio precedente, due componenti aggiuntivi esistenti nel cluster hanno versioni più recenti disponibili nella colonna `UPDATE AVAILABLE`.

2. Aggiorna il componente aggiuntivo.

a. Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando in base alla necessità:

- Sostituisci *my-cluster* con il nome del cluster.
- *region-code*Sostituiscilo con la AWS regione in cui si trova il cluster.
- Sostituisci *vpc-cni* con il nome del componente aggiuntivo da aggiornare restituito nell'output del passaggio precedente.
- Se desideri aggiornare una versione del componente aggiuntivo precedente all'ultima versione, sostituisci *latest* con il numero di versione che vuoi utilizzare e che è stato restituito nell'output di un passaggio precedente. Alcuni componenti aggiuntivi includono versioni consigliate. Per ulteriori informazioni, consulta la documentazione del componente aggiuntivo che stai aggiornando. Per un elenco dei componenti aggiuntivi, consultathe section called " AWS componenti aggiuntivi". * Se il componente aggiuntivo utilizza un account di servizio Kubernetes e un ruolo IAM, sostituiscilo *111122223333* con l'ID dell'account e *role-name* con il nome di un ruolo IAM esistente che hai creato. Per istruzioni sulla creazione del ruolo, consultare la documentazione del componente aggiuntivo da creare. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per specificare un ruolo per l'account di servizio è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

  Se il componente aggiuntivo non utilizza un account di servizio Kubernetes e un ruolo IAM, elimina la riga `serviceAccountRoleARN: arn:aws:iam::`*111122223333*`:role/`*role-name* .
- L'opzione *preserve* mantiene i valori esistenti per il componente aggiuntivo. Se hai impostato valori personalizzati per le impostazioni del componente aggiuntivo e non utilizzi questa opzione, Amazon EKS sovrascrive i tuoi valori con quelli predefiniti. Se utilizzi questa opzione, è preferibile testare eventuali modifiche ai campi e ai valori su un cluster non di produzione prima di aggiornare il componente aggiuntivo sul cluster di produzione. Se modifichi questo valore in `overwrite`, tutte le impostazioni vengono modificate nei valori predefiniti di Amazon EKS. Se hai impostato valori personalizzati per un'impostazione qualsiasi, è possibile che siano sovrascritti con i valori predefiniti di Amazon EKS. Se modifichi questo valore in `none`, Amazon EKS non modifica il valore di alcuna impostazione, ma l'aggiornamento potrebbe non riuscire. Se l'aggiornamento non riesce, riceverai un messaggio di errore che ti aiuterà a risolvere il conflitto.

```
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. Esegui il comando modificato per creare il file `update-addon.yaml`.

c. Applica il file di configurazione al tuo cluster.

```
eksctl update addon -f update-addon.yaml
```

Per ulteriori informazioni sull'aggiornamento dei componenti aggiuntivi, consultaAggiornamento dei componenti aggiuntivi nella documentazione di `eksctl`.

Componente aggiuntivo di aggiornamento (console)AWS

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Scegli il nome del cluster per cui aggiornare il componente aggiuntivo.
4. Selezionare la scheda Componenti aggiuntivi.
5. Scegli il componente aggiuntivo da aggiornare.
6. Scegli Modifica.
7. Nella *`name of addon`* pagina Configura, procedi come segue:
   a. Scegli Versione da utilizzare. Il componente aggiuntivo potrebbe avere una versione raccomandata. Per ulteriori informazioni, consulta la documentazione del componente aggiuntivo che stai aggiornando. Per un elenco dei componenti aggiuntivi, consulta the section called " AWS componenti aggiuntivi".
   b. Sono disponibili due opzioni per configurare i ruoli per i componenti aggiuntivi: il ruolo IAM di EKS Pod Identity e i ruoli IAM per gli account di servizio (IRSA). Seguire la procedura adeguata di seguito per l'opzione preferita. Se tutti i componenti aggiuntivi selezionati includono Richiede

un abbonamento sotto Stato, seleziona Avanti. Per i componenti aggiuntivi che non includono Richiede un abbonamento in Stato, procedi secondo quanto segue:

i. Per il ruolo IAM di Pod Identity per l'account di servizio, è possibile utilizzare un ruolo IAM di EKS Pod Identity esistente o crearne uno con il pulsante Crea ruolo consigliato. Questo campo indicherà solo le opzioni con la policy di attendibilità adeguata. Se non esiste alcun ruolo da selezionare, non si dispone di un ruolo esistente con una policy di attendibilità corrispondente. Per configurare un ruolo IAM di EKS Pod Identity per gli account di servizio del componente aggiuntivo selezionato, scegliere Crea ruolo consigliato. La procedura guidata per la creazione dei ruoli si apre in una finestra separata. La procedura guidata compilerà automaticamente le informazioni sul ruolo come indicato di seguito. Per ogni componente aggiuntivo in cui si desidera creare il ruolo IAM di EKS Pod Identity, completare i passaggi della procedura guidata IAM come indicato di seguito.

   - Nel passaggio Seleziona entità affidabile, l'opzione di AWS servizio per EKS e lo use case per EKS - Pod Identity sono preselezionati e la politica di attendibilità appropriata verrà compilata automaticamente per il componente aggiuntivo. Ad esempio, il ruolo verrà creato con la policy di attendibilità adeguata contenente il principale IAM pods.eks.amazonaws.com, come descritto nella pagina the section called "Vantaggi delle associazioni EKS Pod Identity". Scegli Next (Successivo).
   - Al passaggio Aggiungi autorizzazioni, viene preselezionata la policy gestita adeguata per la policy del ruolo per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo CNI di Amazon VPC, il ruolo verrà creato con la policy gestita `AmazonEKS_CNI_Policy`, come descritto nella pagina the section called "Plug-in della CNI di Amazon VPC per Kubernetes". Scegli Next (Successivo).
   - Al passaggio Nomina, verifica e crea, in Nome del ruolo, viene inserito automaticamente il nome del ruolo predefinito per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo Amazon VPC CNI, il ruolo verrà creato con il nome Amazon. EKSPod IdentityAmazon VPCCNIRole In Descrizione, la descrizione predefinita viene compilata automaticamente con la descrizione adeguata per il componente aggiuntivo. Ad esempio, per il componente aggiuntivo Amazon VPC CNI, il ruolo verrà creato con la descrizione Permette ai pod in esecuzione nel cluster Amazon EKS di accedere alle risorse. AWS In Policy di attendibilità, viene visualizzata la policy di attendibilità compilata per il componente aggiuntivo. Scegli Crea ruolo.

> **Note**
>
> Mantenere il nome del ruolo predefinito consente a EKS di preselezionare il ruolo dei componenti aggiuntivi nei nuovi cluster o quando si aggiungono componenti aggiuntivi ai cluster esistenti. È comunque possibile sostituire tale nome e il ruolo sarà disponibile per il componente aggiuntivo in tutti i cluster, ma dovrà essere selezionato manualmente dal menu a discesa.

ii. Per i componenti aggiuntivi che non includono Richiede un abbonamento in Stato e per i quali si desidera configurare i ruoli utilizzando IRSA, consultare la documentazione relativa al componente aggiuntivo da creare per definire una policy IAM e collegarla a un ruolo. Per un elenco dei componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per selezionare un ruolo IAM è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

c. Scegliere Impostazioni di configurazione facoltative.

d. In Valori di configurazione, inserisci qualsiasi informazione di configurazione specifica del componente aggiuntivo. Per ulteriori informazioni, consulta la documentazione del componente aggiuntivo che stai aggiornando. Per un elenco di componenti aggiuntivi, consulta the section called " AWS componenti aggiuntivi"... per Metodo di risoluzione dei conflitti, seleziona una delle opzioni disponibili. Se hai impostato valori personalizzati per le impostazioni del componente aggiuntivo, è preferibile l'opzione Preserve (Conserva). Se non scegli questa opzione, Amazon EKS sovrascrive i tuoi valori con quelli predefiniti. Se utilizzi questa opzione, è preferibile testare eventuali modifiche ai campi e ai valori su un cluster non di produzione prima di aggiornare il componente aggiuntivo sul cluster di produzione. Se modifichi questo valore in sovrascrivi, tutte le impostazioni sono modificate nei valori predefiniti di Amazon EKS. Se hai impostato valori personalizzati per un'impostazione qualsiasi, è possibile che siano sovrascritti con i valori predefiniti di Amazon EKS. Se modifichi questo valore in nessuno, Amazon EKS non modifica il valore di alcuna impostazione, ma l'aggiornamento potrebbe non riuscire. Se l'aggiornamento non riesce, riceverai un messaggio di errore che ti aiuterà a risolvere il conflitto.

8. Scegli Save changes (Salva modifiche).

Componente aggiuntivo di aggiornamento (AWS CLI)

1. È necessaria la versione `2.12.3` o una versione successiva `1.27.160` o una versione o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
2. Visualizza l'elenco dei componenti aggiuntivi installati. Sostituisci *`my-cluster`* con il nome del cluster.

   ```
   aws eks list-addons --cluster-name my-cluster
   ```

   Di seguito viene riportato un output di esempio:

   ```
   {
       "addons": [
           "coredns",
           "kube-proxy",
           "vpc-cni"
       ]
   }
   ```

3. Visualizza la versione corrente del componente aggiuntivo da aggiornare. Sostituisci *`my-cluster`* con il nome del tuo cluster e *`vpc-cni`* con il nome del componente aggiuntivo da aggiornare.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    "addon.addonVersion" --output text
   ```

   Di seguito viene riportato un output di esempio.

   ```
   v1.10.4-eksbuild.1
   ```

4. Determina le versioni del componente aggiuntivo disponibili per la versione del cluster. Sostituiscilo *1.33* con la versione del cluster e *vpc-cni* con il nome del componente aggiuntivo che desideri aggiornare.

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

Di seguito viene riportato un output di esempio.

```
-----------------------------------------
|          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   |
+-----------------+----------------------+
```

Per impostazione predefinita, la versione con `True` nella colonna `Defaultversion` è la versione con cui viene creato il componente aggiuntivo.

5. Aggiorna il componente aggiuntivo. Copia il comando seguente sul tuo dispositivo. Apporta le seguenti modifiche al comando, se necessario, quindi esegui il comando modificato. Per ulteriori informazioni su questo comando, consulta update-addon nel Riferimento della riga di comando Amazon EKS.
   - Sostituisci *my-cluster* con il nome del cluster.
   - Sostituisci *vpc-cni* con il nome del componente aggiuntivo da aggiornare che è stato restituito nell'output di un passaggio precedente.
   - Sostituisci *version-number* con la versione a cui eseguire l'aggiornamento restituita nell'output del passaggio precedente. Alcuni componenti aggiuntivi includono versioni consigliate. Per ulteriori informazioni, consulta la documentazione del componente aggiuntivo che stai aggiornando. Per un elenco dei componenti aggiuntivi, consulta. the section called " AWS componenti aggiuntivi" * Se il componente aggiuntivo utilizza un account di servizio Kubernetes e un ruolo IAM, sostituiscilo *111122223333* con l'ID dell'account e *role-name* con il nome di un ruolo IAM esistente che hai creato. Per istruzioni sulla creazione del ruolo, consultare la documentazione del componente aggiuntivo da creare. Per un elenco dei

componenti aggiuntivi, consultare la pagina the section called " AWS componenti aggiuntivi". Per specificare un ruolo per l'account di servizio è necessario disporre di un gestore OpenID Connect (OIDC) IAM per il cluster. Per stabilire se ne possiedi uno per il tuo cluster o per crearne uno, consulta the section called "Gestore di identità OIDC IAM".

Se il componente aggiuntivo non utilizza un account di servizio Kubernetes e un ruolo IAM, elimina la riga `serviceAccountRoleARN: arn:aws: iam::111122223333:role/role-name` .

- L'opzione `--resolve-conflicts PRESERVE` mantiene i valori esistenti per il componente aggiuntivo. Se hai impostato valori personalizzati per le impostazioni del componente aggiuntivo e non utilizzi questa opzione, Amazon EKS sovrascrive i tuoi valori con quelli predefiniti. Se utilizzi questa opzione, è preferibile testare eventuali modifiche ai campi e ai valori su un cluster non di produzione prima di aggiornare il componente aggiuntivo sul cluster di produzione. Se modifichi questo valore in `OVERWRITE`, tutte le impostazioni vengono modificate nei valori predefiniti di Amazon EKS. Se hai impostato valori personalizzati per un'impostazione qualsiasi, è possibile che siano sovrascritti con i valori predefiniti di Amazon EKS. Se modifichi questo valore in `NONE`, Amazon EKS non modifica il valore di alcuna impostazione, ma l'aggiornamento potrebbe non riuscire. Se l'aggiornamento non riesce, riceverai un messaggio di errore che ti aiuterà a risolvere il conflitto.
- Se desideri rimuovere tutte le configurazioni personalizzate, esegui l'aggiornamento utilizzando l'opzione *`--configuration-values '{}'`*. In tal modo, verranno ripristinati i valori predefiniti di tutte le configurazioni personalizzate. Se non desideri modificare la configurazione personalizzata, non fornire il flag. *`--configuration-values`* Per modificare una configurazione personalizzata, sostituisci *`{}`* con i nuovi parametri.

```
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. Controlla lo stato dell'aggiornamento. Sostituiscilo *`my-cluster`* con il nome del cluster e *`vpc-cni`* con il nome del componente aggiuntivo che stai aggiornando.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

Di seguito viene riportato un output di esempio.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "UPDATING",
    }
}
```

L'aggiornamento è completo quando lo stato è `ACTIVE`.

Verifica la compatibilità delle versioni aggiuntive di Amazon EKS con un cluster

Prima di creare un componente aggiuntivo di Amazon EKS, devi verificare che la versione del componente aggiuntivo di Amazon EKS sia compatibile con il cluster.

Utilizza describe-addon-versions API per elencare le versioni disponibili dei componenti aggiuntivi di EKS e quali versioni di Kubernetes supporta ciascuna versione del componente aggiuntivo.

1. Verifica che AWS CLI sia installata e funzionante con `aws sts get-caller-identity`. Se questo comando non funziona, consulta Get started with the AWS CLI.
2. Determina il nome del componente aggiuntivo per il quale recuperare le informazioni sulla compatibilità della versione, ad esempio `amazon-cloudwatch-observability`.
3. Determina la versione di Kubernetes per il cluster, ad esempio `1.33`.
4. Usa AWS CLI per recuperare le versioni del componente aggiuntivo compatibili con la versione di Kubernetes del cluster.

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --
kubernetes-version 1.33
```

Di seguito viene riportato un output di esempio.

```
{
    "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
                        }
                    ],
                }
            ]
        }
    ]
}
```

Questo output mostra che la versione `vX.X.X-eksbuild.X` del componente aggiuntivo è compatibile con quella `1.33` del cluster di Kubernetes.

Compatibilità del componente aggiuntivo con i tipi di elaborazione

Il campo `computeTypes` nell'output `describe-addon-versions` indica la compatibilità di un componente aggiuntivo con i nodi gestiti da EKS Auto Mode o i nodi ibridi. I componenti aggiuntivi contrassegnati `auto` funzionano con l'infrastruttura gestita da AWS basata sul cloud di EKS Auto Mode, mentre quelli contrassegnati `hybrid` possono essere eseguiti su nodi on-premises collegati al piano di controllo del cloud EKS.

Per ulteriori informazioni, consulta the section called "Considerazioni per la modalità automatica di Amazon EKS".

Rimozione di un componente aggiuntivo Amazon EKS da un cluster

Puoi rimuovere un componente aggiuntivo Amazon EKS dal tuo cluster utilizzando `eksctl` Console di gestione AWS, o la AWS CLI.

Quando rimuovi un componente aggiuntivo Amazon EKS da un cluster:

- non sono previste interruzioni per la funzionalità fornita dal componente aggiuntivo.
- Se utilizzi ruoli IAM per account di servizio (IRSA) e al componente aggiuntivo è associato un ruolo IAM, il ruolo IAM non è rimosso.
- Se utilizzi Pod Identity, tutte le associazioni Pod Identity di proprietà del componente aggiuntivo sono rimosse. Se si specifica l'`--preserve`opzione nella AWS CLI, le associazioni vengono mantenute.
- Amazon EKS interrompe la gestione delle impostazioni per il componente aggiuntivo.
- La console smette di segnalarti la disponibilità di nuove versioni.
- Non è possibile aggiornare il componente aggiuntivo utilizzando AWS strumenti o. APIs
- Puoi scegliere di lasciare il software del componente aggiuntivo nel tuo cluster in modo da poterlo gestire autonomamente, oppure puoi rimuoverlo. È opportuno rimuovere il software del componente aggiuntivo dal tuo cluster se non esistono risorse sul cluster che dipendono dalla funzionalità fornita dal componente aggiuntivo.

Prerequisiti

Completa le seguenti operazioni prima di creare un componente aggiuntivo:

- Un cluster Amazon EKS esistente. Per implementarne uno, consulta Nozioni di base.
- Verifica se il tuo componente aggiuntivo richiede un ruolo IAM. Per ulteriori informazioni, consultare la pagina
- La versione `0.215.0` o quelle successive dello strumento a riga di comando `eksctl` deve essere installata sul dispositivo o nella AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta Installazione nella documentazione di `eksctl`.

Procedura

Sono possibili due opzioni per rimuovere un componente aggiuntivo Amazon EKS:

- Mantenere il software aggiuntivo sul cluster – Questa opzione rimuove la gestione di Amazon EKS su qualsiasi impostazione. Inoltre, rimuove la possibilità per Amazon EKS di notificare gli aggiornamenti e aggiornare automaticamente il componente aggiuntivo Amazon EKS dopo l'avvio di un aggiornamento. Tuttavia, mantiene il software aggiuntivo sul cluster. Questa opzione rende il componente aggiuntivo come autogestito anziché come un componente aggiuntivo Amazon EKS. Con questa opzione, non ci sono tempi di inattività per il componente aggiuntivo.
- Rimuovere completamente il software aggiuntivo dal cluster – Consigliamo di rimuovere il componente aggiuntivo Amazon EKS dal cluster solo se non ci sono risorse nel cluster che dipendono dallo stesso.

Puoi rimuovere un componente aggiuntivo Amazon EKS utilizzando `eksctl` Console di gestione AWS, o la AWS CLI.

Rimuovere un componente aggiuntivo (eksctl)

1. Determina i componenti aggiuntivi installati sul tuo cluster. Sostituisci *`my-cluster`* con il nome del cluster.

   ```
   eksctl get addon --cluster my-cluster
   ```

   Di seguito viene riportato un output di esempio:

   ```
   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
   [...]
   ```

   L'output potrebbe avere un aspetto diverso in base ai componenti aggiuntivi e alle versioni presenti nel tuo cluster.

2. Rimuovi il componente aggiuntivo. Sostituisci *`my-cluster`* con il nome del cluster e *`name-of-add-on`* con il nome del componente aggiuntivo da rimuovere restituito nell'output del passaggio precedente. Se rimuovi l'*`--preserve`*opzione, oltre al fatto che Amazon EKS non gestisce più il componente aggiuntivo, il software aggiuntivo viene eliminato dal cluster.

   ```
   eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
   ```

Per ulteriori informazioni sulla rimozione dei componenti aggiuntivi, consulta Eliminare componenti aggiuntivi nella documentazione di `eksctl`.

Rimuovi componente aggiuntivo (console)AWS

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Scegli il nome del cluster per cui desideri rimuovere il componente aggiuntivo di Amazon EBS.
4. Selezionare la scheda Componenti aggiuntivi.
5. Scegli il componente aggiuntivo che vuoi rimuovere.
6. Scegli Rimuovi.
7. Nella finestra di dialogo Rimuovi: ***name of addon*** conferma, effettuate le seguenti operazioni:
   a. Se desideri che Amazon EKS interrompa la gestione delle impostazioni per il componente aggiuntivo, seleziona Mantieni sul cluster. Effettuare questa operazione se si desidera conservare il software aggiuntivo sul cluster. In questo modo sarà possibile gestire autonomamente tutte le impostazioni del componente aggiuntivo.
   b. Inserisci il nome del componente aggiuntivo.
   c. Scegli Rimuovi.

Rimuovi componente aggiuntivo (AWS CLI)

1. È necessaria la versione `0.215.0` o una versione successiva dello strumento da riga di `eksctl` comando installata sul dispositivo o. AWS CloudShell Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Visualizza l'elenco dei componenti aggiuntivi installati. Sostituisci *my-cluster* con il nome del cluster.

```
aws eks list-addons --cluster-name my-cluster
```

Di seguito viene riportato un output di esempio.

```
{
    "addons": [
        "coredns",
```

```
        "kube-proxy",
        "vpc-cni",
        "name-of-addon"
    ]
}
```

3. Rimuovi il componente aggiuntivo installato. Sostituisci *my-cluster* con il nome del tuo cluster e *name-of-add-on* con il nome del componente aggiuntivo da rimuovere. La rimozione *--preserve* comporta l'eliminazione del software aggiuntivo dal cluster.

   ```
   aws eks delete-addon --cluster-name my-cluster --addon-name name-of-addon --preserve
   ```

   Di seguito è riportato un output di esempio abbreviato.

   ```
   {
       "addon": {
           "addonName": "name-of-add-on",
           "clusterName": "my-cluster",
           "status": "DELETING",
       }
   }
   ```

4. Controlla lo stato della rimozione. Sostituiscilo *my-cluster* con il nome del cluster e *name-of-addon* con il nome del componente aggiuntivo che stai rimuovendo.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
   ```

   Di seguito è riportato l'output esemplificativo dopo la rimozione del componente aggiuntivo.

   ```
   An error occurred (ResourceNotFoundException) when calling the DescribeAddon
    operation: No addon: name-of-addon found in cluster: my-cluster
   ```

Ruoli IAM per i componenti aggiuntivi di Amazon EKS

Alcuni componenti aggiuntivi Amazon EKS richiedono ruoli IAM e autorizzazioni per chiamare le API AWS. Ad esempio, il componente aggiuntivo CNI di Amazon VPC richiama determinate API AWS per configurare le risorse di rete nell'account. A questi componenti aggiuntivi deve essere concessa l'autorizzazione tramite IAM. Più specificamente, l'account di servizio del pod che esegue il componente aggiuntivo deve essere associato a un ruolo IAM con una policy IAM specifica.

Il modo consigliato per concedere autorizzazioni AWS ai carichi di lavoro del cluster è utilizzare la funzionalità di Pod Identity di Amazon EKS. Puoi utilizzare un'associazione di Pod Identity per mappare l'account di servizio di un componente aggiuntivo a un ruolo IAM. Se un pod utilizza un account di servizio con un'associazione, Amazon EKS imposta le variabili di ambiente nei container del pod. Le variabili di ambiente configurano AWS SDK, inclusa AWS CLI, per utilizzare le credenziali di EKS Pod Identity. Per ulteriori informazioni, consulta the section called "Pod Identity"

I componenti aggiuntivi di Amazon EKS possono aiutare a gestire il ciclo di vita delle associazioni di Pod Identity corrispondenti al componente aggiuntivo. Ad esempio, puoi creare o aggiornare un componente aggiuntivo di Amazon EKS e l'associazione di Pod Identity necessaria in una singola chiamata API. Amazon EKS fornisce anche un'API per il recupero delle policy IAM suggerite.

1. Conferma che l'agente Pod Identity di Amazon EKS sia configurato sul cluster.
2. Determina se il componente aggiuntivo che si desidera installare richiede le autorizzazioni IAM utilizzando l'operazione `describe-addon-versions` su AWS CLI. Se il flag `requiresIamPermissions` è `true`, allora dovresti usare l'operazione `describe-addon-configurations` per determinare le autorizzazioni necessarie al componente aggiuntivo. La risposta include un elenco di policy IAM gestite suggerite.
3. Recupera il nome dell'account di servizio Kubernetes e la policy IAM utilizzando l'operazione `describe-addon-configuration` CLI. Valuta l'ambito della policy suggerita rispetto ai requisiti di sicurezza.
4. Crea un ruolo IAM utilizzando la policy di autorizzazioni suggerita e la policy di attendibilità richiesta da Pod Identity. Per ulteriori informazioni, consulta the section called "Crea un'associazione Pod Identity (AWS Console)".
5. Crea o aggiorna un componente aggiuntivo Amazon EKS utilizzando la CLI. Specifica almeno un'associazione di Pod Identity. Un'associazione di Pod Identity è il nome di un account di servizio Kubernetes e l'ARN del ruolo IAM.
   - Le associazioni di Pod Identity create utilizzando le API del componente aggiuntivo sono di proprietà del rispettivo componente aggiuntivo. Se elimini il componente aggiuntivo, viene eliminata anche l'associazione di Pod Identity. Puoi impedire questa eliminazione a cascata utilizzando l'opzione `preserve` quando si elimina un componente aggiuntivo utilizzando la AWS CLI o l'API. Se necessario, puoi anche aggiornare o eliminare direttamente l'associazione di Pod Identity. I componenti aggiuntivi non possono assumere la proprietà delle associazioni di Pod Identity esistenti. Devi eliminare l'associazione esistente e ricrearla usando un'operazione di creazione o aggiornamento di un componente aggiuntivo.

- Amazon EKS consiglia di utilizzare le associazioni di Pod Identity per gestire le autorizzazioni IAM per i componenti aggiuntivi. Il metodo precedente, ruoli IAM per account di servizio (IRSA), è ancora supportato. Puoi specificare sia un `serviceAccountRoleArn` IRSA che un'associazione di Pod Identity per un componente aggiuntivo. Se l'agente Pod Identity EKS è installato nel cluster, l'`serviceAccountRoleArn` verrà ignorato ed EKS utilizzerà l'associazione di Pod Identity fornita. Se Pod Identity non è abilitato, verrà utilizzato l'`serviceAccountRoleArn`.
- Se aggiorni le associazioni di Pod Identity per un componente aggiuntivo esistente, Amazon EKS avvia un riavvio in sequenza dei pod del componente aggiuntivo.

Recupero delle informazioni IAM su un componente aggiuntivo di Amazon EKS

Prima di creare un componente aggiuntivo, utilizza AWS CLI per determinare:

- Se il componente aggiuntivo richiede le autorizzazioni IAM
- La policy IAM consigliata da utilizzare

Procedura

1. Determina il nome del componente aggiuntivo che desideri installare e la versione Kubernetes del tuo cluster. Per ulteriori informazioni sui componenti aggiuntivi, consulta the section called "Componenti aggiuntivi Amazon EKS".
2. Utilizza AWS CLI per stabilire se il componente aggiuntivo richiede autorizzazioni IAM.

   ```
   aws eks describe-addon-versions \
   --addon-name <addon-name> \
   --kubernetes-version <kubernetes-version>
   ```

   Per esempio:

   ```
   aws eks describe-addon-versions \
   --addon-name aws-ebs-csi-driver \
   --kubernetes-version 1.30
   ```

   Esamina il seguente output esemplificativo. Tieni presente che `requiresIamPermissions` è `true` e la versione del componente aggiuntivo predefinita. È necessario specificare la versione del componente aggiuntivo quando si recupera la policy IAM consigliata.

```
{
    "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 il componente aggiuntivo richiede autorizzazioni IAM, utilizza AWS CLI per recuperare una policy IAM consigliata.

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name <addon-name> \
--addon-version <addon-version>
```

Per esempio:

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name aws-ebs-csi-driver \
--addon-version v1.31.0-eksbuild.1
```

Esamina l'output successivo. Prendi nota di `recommendedManagedPolicies`.

```
[
    {
        "serviceAccount": "ebs-csi-controller-sa",
        "recommendedManagedPolicies": [
            "arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"
        ]
    }
]
```

4. Crea un ruolo IAM e collega la policy gestita consigliata. In alternativa, riconsulta la policy gestita e definisci le autorizzazioni in modo appropriato. Per ulteriori informazioni, consulta the section called "Crea un'associazione Pod Identity (AWS Console)".

Riferimento di supporto Pod Identity

La tabella seguente indica se determinati componenti aggiuntivi di Amazon EKS supportano EKS Pod Identity.

Nome del componente aggiuntivo	Supporto di Pod Identity	Versione minima richiesta
Driver CSI per Amazon EBS	Sì	v1.26.0-eksbuild.1
CNI di Amazon VPC	Sì	v1.15.5-eksbuild.1
Driver CSI per Amazon EFS	Sì	v2.0.5-eksbuild.1
AWS Distro per OpenTelemetry	Sì	v0.94.1-eksbuild.1
Driver CSI di Mountpoint per Amazon S3	No	N/D
Agente Osservabilità di Amazon CloudWatch	Sì	v3.1.0-eksbuild.1

L'ultimo aggiornamento di questa tabella risale al 28 ottobre 2024.

Utilizzo di Pod Identity per assegnare un ruolo IAM a un componente aggiuntivo Amazon EKS

Alcuni componenti aggiuntivi Amazon EKS richiedono ruoli IAM e autorizzazioni. Prima di aggiungere o aggiornare un componente aggiuntivo Amazon EKS per utilizzare un'associazione Pod Identity, verifica il ruolo e la policy da utilizzare. Per ulteriori informazioni, consulta the section called "Recupero delle informazioni IAM".

1. Determina:
   - `cluster-name`: il nome del cluster su cui installare il componente aggiuntivo.
   - `addon-name`: il nome del componente aggiuntivo da installare.
   - `service-account-name`: il nome dell'account di servizio Kubernetes utilizzato dal componente aggiuntivo.
   - `iam-role-arn`: l'ARN di un ruolo IAM con le autorizzazioni sufficienti per il componente aggiuntivo. Il ruolo deve disporre della policy di attendibilità richiesta per EKS Pod Identity. Per ulteriori informazioni, consulta the section called "Crea un'associazione Pod Identity (AWS Console)".
2. Aggiorna il componente aggiuntivo utilizzando AWS CLI. È anche possibile specificare le associazioni Pod Identity durante la creazione di un componente aggiuntivo, utilizzando la stessa sintassi `--pod-identity-assocations`. Nota che quando specifichi le associazioni di Pod Identity durante l'aggiornamento di un componente aggiuntivo, tutte le associazioni di Pod Identity precedenti sono sovrascritte.

   ```
   aws eks update-addon --cluster-name <cluster-name> \
   --addon-name <addon-name> \
   --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-
   arn>'
   ```

   Per esempio:

   ```
   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. Convalida che l'associazione Pod Identity sia stata creata:

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

A operazione riuscita, viene visualizzato un output simile al seguente. Nota OwnerARN del componente aggiuntivo 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"
        }
    ]
}
```

Rimozione delle associazioni Pod Identity da un componente aggiuntivo Amazon EKS

Rimuovi le associazioni Pod Identity da un componente aggiuntivo Amazon EKS.

1. Determina:
   - `cluster-name`: il nome del cluster EKS su cui installare il componente aggiuntivo.
   - `addon-name`: Il nome del componente aggiuntivo Amazon EKS da installare.
2. Aggiorna il componente aggiuntivo per specificare un array vuoto di associazioni di Pod Identity.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations "[]"
```

Risoluzione dei problemi in Pod Identity per i componenti aggiuntivi di EKS

Se i componenti aggiuntivi riscontrano errori durante il tentativo di eseguire operazioni su AWS API, SDK o CLI, conferma quanto segue:

- L'agente Pod Identity è installato nel cluster.
  - Per informazioni su come installare l'agente Pod Identity, consulta the section called "Configurazione dell'agente".
- Il componente aggiuntivo ha un'associazione di Pod Identity valida.
  - Utilizza AWS CLI per recuperare le associazioni per il nome dell'account di servizio utilizzato dal componente aggiuntivo.

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

- Il ruolo IAM dispone della policy di attendibilità richiesta per Pod Identity.
  - Utilizza AWS CLI per recuperare la policy di attendibilità per un componente aggiuntivo.

```
aws iam get-role --role-name <role-name> --query Role.AssumeRolePolicyDocument
```

- Il ruolo IAM dispone delle autorizzazioni necessarie per il componente aggiuntivo.
  - Usa AWS CloudTrail per rivedere gli eventi `AccessDenied` o `UnauthorizedOperation`.
- Il nome dell'account di servizio nell'associazione di Pod Identity corrisponde al nome dell'account di servizio utilizzato dal componente aggiuntivo.
  - Per informazioni sui componenti aggiuntivi disponibili, consulta the section called " AWS componenti aggiuntivi".
- Verifica la configurazione di MutatingWebhookConfiguration denominata `pod-identity-webhook`
  - `admissionReviewVersions` del webhook deve essere `v1beta1` e non funziona con `v1`.

Determina i campi che puoi personalizzare per i componenti aggiuntivi Amazon EKS

I componenti aggiuntivi Amazon EKS vengono installati nel cluster utilizzando configurazioni standard e best practice. Per ulteriori informazioni sull'aggiunta di un componente aggiuntivo di Amazon EKS al cluster, consulta the section called "Componenti aggiuntivi Amazon EKS".

Potresti voler personalizzare la configurazione di un componente aggiuntivo Amazon EKS per abilitare funzionalità avanzate. Amazon EKS utilizza la funzionalità applicazione lato server Kubernetes per abilitare la gestione di un componente aggiuntivo da parte di Amazon EKS, senza sovrascrivere la configurazione per le impostazioni non gestite da Amazon EKS. Per ulteriori

informazioni, consulta Applicazione lato server nella documentazione di Kubernetes. Per raggiungere questo obiettivo, Amazon EKS gestisce un insieme minimo di campi per ogni componente aggiuntivo che installa. Puoi modificare senza problemi tutti i campi non gestiti da Amazon EKS o da un altro processo del piano di controllo Kubernetes, come `kube-controller-manager`.

> **Important**
>
> La modifica di un campo gestito da Amazon EKS impedisce ad Amazon EKS di gestire il componente aggiuntivo e può comportare la sovrascrittura delle modifiche quando un componente aggiuntivo viene aggiornato.

Sintassi della gestione dei campi

Quando si visualizzano i dettagli di un oggetto Kubernetes, nell'output vengono restituiti sia i campi gestiti che quelli non gestiti. I campi gestiti possono essere dei seguenti tipi:

- Completamente gestito – Tutte le chiavi del campo sono gestite da Amazon EKS. Le modifiche a qualsiasi valore causano un conflitto.
- Parzialmente gestiti – Alcune chiavi per il campo sono gestite da Amazon EKS. Solo le modifiche alle chiavi gestite esplicitamente da Amazon EKS causano un conflitto.

Entrambi i tipi di campi sono taggati con `manager: eks`.

Ogni chiave è un `.` che rappresenta il campo stesso, che viene sempre mappato a un insieme vuoto o a una stringa che rappresenta un sottocampo o un elemento. L'output per la gestione del campo è costituito dai seguenti tipi di dichiarazioni:

- `f:`*`name`* , dove *`name`* è il nome di un campo in un elenco.
- `k:`*`keys`* , dove *`keys`* è una mappa dei campi di una voce di elenco.
- `v:`*`value`* , dove *`value`* è il valore formattato JSON esatto di una elemento in elenco.
- `i:`*`index`* , dove *`index`* è la posizione di un elemento nell'elenco.

Le seguenti porzioni di output per il componente aggiuntivo CoreDNS illustrano le dichiarazioni precedenti:

- Campi completamente gestiti – Se un campo gestito ha un `f:` (campo) specificato, ma non `k:` (chiave), l'intero campo è gestito. Le modifiche apportate a qualsiasi valore in questo campo causano un conflitto.

  Nel seguente output, è possibile vedere che il container denominato `coredns` è gestito da `eks`. La `args`, `image`, e i sottocampi `imagePullPolicy` sono gestiti anche da `eks`. Le modifiche a qualsiasi valore in questi campi causano un conflitto.

  ```
  [...]
  f:containers:
    k:{"name":"coredns"}:
    .: {}
    f:args: {}
    f:image: {}
    f:imagePullPolicy: {}
  [...]
  manager: eks
  [...]
  ```

- Campi parzialmente gestiti – Se una chiave gestita ha un valore specificato, le chiavi dichiarate vengono gestite per quel campo. La modifica delle chiavi specificate causa un conflitto.

  Nel seguente output, è possibile vedere che `eks` gestisce il `config-volume` e volumi `tmp` impostati con la chiave `name`.

  ```
  [...]
  f:volumes:
    k:{"name":"config-volume"}:
      .: {}
      f:configMap:
        f:items: {}
        f:name: {}
      f:name: {}
    k:{"name":"tmp"}:
      .: {}
      f:name: {}
  [...]
  manager: eks
  [...]
  ```

- Aggiunta di chiavi a campi parzialmente gestiti – Se viene gestita solo una chiave-valore specifica, è possibile aggiungere ulteriori chiavi, ad esempio, argomenti, a un campo senza causare conflitti.

Se aggiungi altre chiavi, assicurati per prima cosa che il campo non sia gestito. L'aggiunta o la modifica di qualsiasi valore gestito causa un conflitto.

Nel seguente output, è possibile vedere come sia la chiave `name` che il campo `name` siano gestiti. L'aggiunta o la modifica di qualsiasi nome container causa un conflitto con questa chiave gestita.

```
[...]
f:containers:
  k:{"name":"coredns"}:
[...]
    f:name: {}
[...]
manager: eks
[...]
```

Procedura

È possibile utilizzare `kubectl` per vedere quali campi sono gestiti da Amazon EKS per qualsiasi componente aggiuntivo Amazon EKS.

Puoi modificare senza problemi tutti i campi non gestiti da Amazon EKS o da un altro processo del piano di controllo Kubernetes, come `kube-controller-manager`.

1. Determinare quale componente aggiuntivo si desidera esaminare. Per visualizzare tutti i file `deployments` e DaemonSets distribuiti nel cluster, consultathe section called "Accesso alle risorse di cluster".
2. Visualizzazione dei campi gestiti per un componente aggiuntivo eseguendo il comando seguente:

   ```
   kubectl get type/add-on-name -n add-on-namespace -o yaml
   ```

   Ad esempio, è possibile visualizzare i campi gestiti per il componente aggiuntivo CoreDNS con il comando seguente.

   ```
   kubectl get deployment/coredns -n kube-system -o yaml
   ```

   La gestione dei campi è elencata nella sezione seguente dell'output restituito.

   ```
   [...]
   managedFields:
   ```

```
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
[...]
```

> **Note**
>
> Se non visualizzi `managedFields` nell'output, aggiungi `--show-managed-fields` al comando ed eseguilo di nuovo. La versione di `kubectl` che stai utilizzando determina se i campi gestiti vengono restituiti per impostazione predefinita.

Fasi successive

Personalizza i campi non di proprietà del componente aggiuntivo AWS for you.

Convalida le firme delle immagini dei container durante l'implementazione

Se utilizzi AWS Signer e desideri verificare le immagini dei container firmate al momento dell'implementazione, puoi utilizzare una delle seguenti soluzioni:

- Gatekeeper e Ratify: usa Gatekeeper come controller di ammissione e Ratify configurato con un plug-in AWS Signer come Web hook per la convalida delle firme.
- Kyverno: un motore di policy di Kubernetes configurato con un plug-in AWS Signer per la convalida delle firme.

> **Note**
>
> Prima di verificare le firme delle immagini del container, configura l'archivio di attendibilità e la policy di attendibilità di Notation, come richiesto dal controller di ammissione selezionato.

Funzionalità EKS

> **Tip**
>
> Inizia: Crea funzionalità ACK | Crea funzionalitàArgo CD | Crea funzionalità kro

Amazon EKS Capabilities è un set stratificato di funzionalità di cluster completamente gestite che aiutano ad accelerare la velocità degli sviluppatori e a ridurre la complessità della creazione e della scalabilità con Kubernetes. EKS Capabilities sono funzionalità native di Kubernetes per la distribuzione continua dichiarativa, la gestione delle risorse e la creazione e l'orchestrazione AWS delle risorse Kubernetes, tutte completamente gestite da. AWS Con EKS Capabilities, puoi concentrarti maggiormente sulla creazione e sulla scalabilità dei tuoi carichi di lavoro, alleggerendo il carico operativo di questi servizi di piattaforma fondamentali. AWS Queste funzionalità vengono eseguite all'interno di EKS anziché nei cluster, eliminando la necessità di installare, mantenere e scalare i componenti critici della piattaforma sui nodi di lavoro.

Per iniziare, puoi creare una o più funzionalità EKS su un cluster EKS nuovo o esistente. Per fare ciò, puoi usare la AWS CLI, EKS Console di gestione AWS APIs, eksctl o i tuoi strumenti preferiti. infrastructure-as-code Sebbene le funzionalità EKS siano progettate per funzionare insieme, sono risorse cloud indipendenti che puoi scegliere in base al tuo caso d'uso e ai tuoi requisiti.

Tutte le versioni di Kubernetes supportate da EKS sono supportate per EKS Capabilities.

> **Note**
>
> Le funzionalità EKS sono disponibili in tutte le regioni AWS commerciali in cui è disponibile Amazon EKS. Per un elenco delle regioni supportate, consulta gli endpoint e le quote di Amazon EKS nella Guida AWS generale.

Funzionalità disponibili

AWS Controller per Kubernetes (ACK)

ACK consente la gestione delle AWS risorse utilizzando Kubernetes APIs, consentendoti di creare e gestire bucket S3, database RDS, ruoli IAM e altre risorse utilizzando risorse personalizzate

Kubernetes. AWS ACK riconcilia continuamente lo stato desiderato con lo stato effettivo in cui si trova AWS, correggendo eventuali variazioni nel tempo per mantenere il sistema integro e le risorse configurate come specificato. Puoi gestire AWS le risorse insieme ai carichi di lavoro Kubernetes utilizzando gli stessi strumenti e flussi di lavoro, con supporto per più di 50 AWS servizi tra cui S3, RDS, DynamoDB e Lambda. ACK supporta la gestione delle risorse tra account e regioni, abilitando complesse architetture di gestione di sistemi multi-account e multi-cluster. ACK supporta risorse di sola lettura e adozione in sola lettura, facilitando la migrazione da altre infrastrutture come strumenti di codice ai sistemi basati su Kubernetes.

Scopri di più su ACK →

Argo CD

Argo CD GitOps implementa una distribuzione continua basata sulle applicazioni, utilizzando i repository Git come fonte di verità per i carichi di lavoro e lo stato del sistema. Argo CD sincronizza automaticamente le risorse delle applicazioni con i cluster dai repository Git, rilevando e correggendo le deviazioni per garantire che le applicazioni distribuite corrispondano allo stato desiderato. È possibile distribuire e gestire applicazioni su più cluster da una singola istanza Argo CD, con la distribuzione automatizzata dai repository Git ogni volta che vengono apportate modifiche. L'uso combinato di Argo CD e ACK fornisce un GitOps sistema di base, che semplifica la gestione delle dipendenze dei carichi di lavoro e supporta la progettazione di interi sistemi, inclusa la gestione di cluster e infrastrutture su larga scala. Argo CD si integra con AWS Identity Center per l'autenticazione e l'autorizzazione e fornisce un'interfaccia utente Argo ospitata per visualizzare lo stato di integrità e di implementazione delle applicazioni.

Scopri di più su Argo CD →

kro (Kube Resource Orchestrator)

kro ti consente di creare Kubernetes personalizzati APIs che compongono più risorse in astrazioni di livello superiore, permettendo ai team della piattaforma di definire modelli riutilizzabili per combinazioni di risorse comuni: elementi costitutivi del cloud. Con kro, puoi comporre sia Kubernetes che AWS le risorse in astrazioni unificate, utilizzando una sintassi semplice per abilitare configurazioni dinamiche e logica condizionata. kro consente ai team della piattaforma di fornire funzionalità self-service con guardrail appropriati, consentendo agli sviluppatori di fornire infrastrutture complesse utilizzando infrastrutture semplici e appositamente create APIs mantenendo gli standard organizzativi e le migliori pratiche. le risorse kro sono semplicemente risorse Kubernetes e sono specificate in Kubernetes Manifest test che possono essere archiviati in Git o inviati a OCI- registri compatibili come Amazon ECR per un'ampia distribuzione organizzativa.

Scopri di più su kro →

Vantaggi delle funzionalità EKS

Le funzionalità EKS sono completamente gestite da AWS, eliminando la necessità di installazione, manutenzione e scalabilità dei servizi cluster di base. AWS gestisce le patch di sicurezza, gli aggiornamenti e la gestione operativa, permettendo ai team di concentrarsi sulla creazione AWS anziché sulle operazioni del cluster. A differenza dei tradizionali componenti aggiuntivi di Kubernetes che consumano risorse del cluster, le funzionalità vengono eseguite in EKS anziché sui nodi di lavoro. Ciò consente di liberare capacità e risorse del cluster per i carichi di lavoro, riducendo al minimo l'onere operativo della gestione dei controller interni al cluster e di altri componenti della piattaforma.

Con EKS Capabilities, puoi gestire distribuzioni, AWS risorse Kubernetes personalizzate e composizioni utilizzando Kubernetes nativi e strumenti come. APIs `kubectl` Tutte le funzionalità operano nel contesto dei cluster, rilevando e correggendo automaticamente le variazioni di configurazione nelle risorse applicative e dell'infrastruttura cloud. È possibile distribuire e gestire le risorse su più cluster, AWS account e regioni da un unico punto di controllo, semplificando le operazioni in ambienti complessi e distribuiti.

Le funzionalità EKS sono progettate per i GitOps flussi di lavoro e forniscono infrastrutture dichiarative e con controllo della versione e una gestione delle applicazioni. Le modifiche fluiscono da Git attraverso il sistema, fornendo audit trail, funzionalità di rollback e flussi di lavoro di collaborazione che si integrano con le pratiche di sviluppo esistenti. Questo approccio nativo di Kubernetes significa che non è necessario utilizzare più strumenti o gestire infrastructure-as-code sistemi esterni ai cluster e che esiste un'unica fonte di verità a cui fare riferimento. Lo stato desiderato, definito nella configurazione dichiarativa di Kubernetes con controllo della versione, viene applicato continuamente in tutto l'ambiente.

Prezzi

Con EKS Capabilities, non sono previsti impegni anticipati o commissioni minime. Ti viene addebitato un costo per ogni risorsa di capacità per ogni ora in cui è attiva sul tuo cluster Amazon EKS. Le risorse Kubernetes specifiche gestite da EKS Capabilities vengono inoltre fatturate su base oraria.

Per informazioni aggiornate sui prezzi, consulta la pagina dei prezzi di Amazon EKS.

> **Tip**
>
> È possibile utilizzare AWS Cost Explorer e Cost and Usage Reports per tenere traccia dei costi delle funzionalità separatamente dagli altri addebiti EKS. È possibile etichettare le funzionalità con il nome del cluster, il tipo di funzionalità e altri dettagli ai fini dell'allocazione dei costi.

Come funzionano le funzionalità EKS

Ogni funzionalità è una AWS risorsa che crei sul tuo cluster EKS. Una volta creata, la funzionalità viene eseguita in EKS ed è completamente gestita da AWS.

> **Note**
>
> È possibile creare una risorsa di funzionalità di ogni tipo (Argo CD, ACK e kro) per un determinato cluster. Non è possibile creare risorse con funzionalità multiple dello stesso tipo sullo stesso cluster.

Interagisci con le funzionalità del tuo cluster utilizzando Kubernetes APIs e strumenti standard:

- Utilizzalo per `kubectl` applicare risorse personalizzate Kubernetes
- Usa gli archivi Git come fonte di verità per GitOps i flussi di lavoro

Alcune funzionalità supportano strumenti aggiuntivi. Esempio:

- Usa la CLI di Argo CD per configurare e gestire repository e cluster con le funzionalità di Argo CD
- Utilizzate l'interfaccia utente di Argo CD per visualizzare e gestire le applicazioni gestite dalla funzionalità Argo CD

Le funzionalità sono progettate per funzionare insieme, ma sono indipendenti e completamente attivabili. È possibile abilitare una, due o tutte e tre le funzionalità in base alle proprie esigenze e aggiornare la configurazione man mano che i requisiti evolvono.

Tutti i tipi di EKS Compute sono supportati per l'uso con EKS Capabilities. Per ulteriori informazioni, consulta Gestione del calcolo.

Per la configurazione di sicurezza e i dettagli sui ruoli IAM, consultathe section called "Considerazioni sulle funzionalità EKS". Per i modelli di architettura multi-cluster, vedithe section called "Considerazioni".

Casi di utilizzo comune

GitOps per applicazioni e infrastruttura

Usa Argo CD per distribuire applicazioni e componenti operativi e ACK per gestire le configurazioni dei cluster e fornire l'infrastruttura, entrambi dai repository Git. L'intero stack (applicazioni, database, storage e rete) è definito come codice e distribuito automaticamente.

Esempio: un team di sviluppo invia modifiche a Git. Argo CD distribuisce l'applicazione aggiornata e ACK fornisce un nuovo database RDS con la configurazione corretta. Tutte le modifiche sono verificabili, reversibili e coerenti in tutti gli ambienti.

Progettazione della piattaforma con self-service

Usa kro per creare risorse personalizzate APIs che compongono risorse ACK e Kubernetes. I team della piattaforma definiscono i modelli approvati con guardrail. I team addetti alle applicazioni utilizzano soluzioni semplici e di alto livello APIs per fornire stack completi.

Esempio: un team della piattaforma crea un'API "WebApplication" che fornisce un bucket Deployment, Service, Ingress e S3. Gli sviluppatori utilizzano questa API senza dover comprendere la complessità o le autorizzazioni sottostanti. AWS

Gestione delle applicazioni multi-cluster

Usa Argo CD per distribuire applicazioni su più cluster EKS in diverse regioni o account. Gestisci tutte le implementazioni da una singola istanza Argo CD con policy e flussi di lavoro coerenti.

Esempio: distribuisci la stessa applicazione in cluster di sviluppo, gestione temporanea e produzione in più regioni. Argo CD assicura che ogni ambiente rimanga sincronizzato con il ramo Git corrispondente.

Gestione multi-cluster

Usa ACK per definire e fornire i cluster EKS, kro per personalizzare le configurazioni dei cluster con standard organizzativi e Argo CD per gestire il ciclo di vita e la configurazione dei cluster. Ciò consente la gestione dei end-to-end cluster dalla creazione fino alle operazioni in corso.

Esempio: definire i cluster EKS utilizzando ACK e kro per fornire e gestire l'infrastruttura dei cluster, definire gli standard organizzativi per il networking, le politiche di sicurezza, i componenti aggiuntivi e altre configurazioni. Usa Argo CD per creare e gestire continuamente cluster, configurazioni e aggiornamenti delle versioni di Kubernetes in tutta la tua flotta sfruttando standard coerenti e la gestione automatizzata del ciclo di vita.

Migrazioni e modernizzazione

Semplifica la migrazione a EKS con il provisioning delle risorse cloud e i flussi di lavoro nativi. GitOps Usa ACK per adottare AWS le risorse esistenti senza ricrearle e Argo CD per rendere operative le distribuzioni dei carichi di lavoro da Git.

Esempio: un team che esegue la migrazione EC2 da EKS adotta i database RDS e i bucket S3 esistenti utilizzando ACK, quindi utilizza Argo CD per distribuire applicazioni containerizzate da Git. Il percorso di migrazione è chiaro e le operazioni sono standardizzate sin dal primo giorno.

Account e Bootstrapping regionale

Automatizza l'implementazione dell'infrastruttura tra account e regioni utilizzando insieme Argo CD e ACK. Definisci la tua infrastruttura come codice in Git e lascia che siano le funzionalità a gestire l'implementazione e la gestione.

Esempio: un team della piattaforma gestisce repository Git che definiscono configurazioni di account standard, ruoli IAMVPCs, istanze RDS e stack di monitoraggio. Argo CD implementa automaticamente queste configurazioni su nuovi account e regioni, garantendo la coerenza e riducendo i tempi di configurazione manuale da giorni a minuti.

Lavorare con le risorse di capacità

Questo argomento descrive le operazioni comuni per la gestione delle risorse funzionali per tutti i tipi di capacità.

Risorse di funzionalità EKS

Le funzionalità EKS sono AWS risorse che abilitano funzionalità gestite sul tuo cluster Amazon EKS. Le funzionalità vengono eseguite in EKS, eliminando la necessità di installare e gestire controller e altri componenti operativi sui nodi di lavoro. Le funzionalità vengono create per uno specifico cluster EKS e rimangono associate a quel cluster per l'intero ciclo di vita.

Ogni risorsa funzionale ha:

- Un nome univoco all'interno del cluster
- Un tipo di funzionalità (ACK, ARGOCD o KRO)
- Un Amazon Resource Name (ARN), che specifica sia il nome che il tipo
- Un ruolo IAM basato sulla capacità
- Uno stato che indica il suo stato attuale
- Configurazione, generica e specifica per il tipo di funzionalità

Comprensione dello stato delle capacità

Le risorse di capacità hanno uno stato che indica il loro stato attuale. È possibile visualizzare lo stato e l'integrità delle funzionalità nella console EKS o utilizzando la AWS CLI.

Console:

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Capacità per visualizzare lo stato di tutte le funzionalità.
4. Per informazioni dettagliate sullo stato di salute, scegli la scheda Osservabilità, quindi Monitor cluster, quindi la scheda Capacità.

AWS CLI:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability-name
```

Stati delle capacità

CREAZIONE: La funzionalità è in fase di configurazione. Puoi allontanarti dalla console: la funzionalità continuerà a creare in background.

ATTIVA: La funzionalità è attiva e pronta per l'uso. Se le risorse non funzionano come previsto, controlla lo stato delle risorse e le autorizzazioni IAM. Per ulteriori informazioni, consulta the section called “Risoluzione dei problemi relativi alle funzionalità”.

AGGIORNAMENTO: vengono applicate le modifiche alla configurazione. Attendi che lo stato ritorni a`ACTIVE`.

ELIMINAZIONE: la funzionalità viene rimossa dal cluster.

CREATE_FAILED: l'installazione ha rilevato un errore. Le cause più comuni includono:

- La politica di fiducia dei ruoli IAM è errata o mancante
- Il ruolo IAM non esiste o non è accessibile
- Problemi di accesso al cluster
- Parametri di configurazione non validi

Consulta la sezione relativa allo stato delle funzionalità per dettagli specifici sugli errori.

UPDATE_FAILED: aggiornamento della configurazione non riuscito. Controlla la sezione sullo stato delle funzionalità per i dettagli e verifica le autorizzazioni IAM.

> **Tip**
>
> Per una guida dettagliata alla risoluzione dei problemi, consulta:
>
> - the section called "Risoluzione dei problemi relativi alle funzionalità"- Risoluzione dei problemi generali relativi alle funzionalità
> - the section called "Risoluzione dei problemi"- Problemi specifici dell'ACK
> - the section called "Risoluzione dei problemi"- Problemi specifici di Argo CD
> - the section called "Risoluzione dei problemi"- problemi specifici per kro

Crea funzionalità

Per creare una funzionalità sul tuo cluster, consulta i seguenti argomenti:

- the section called "Crea una funzionalità ACK"— Crea una funzionalità ACK per gestire AWS le risorse utilizzando Kubernetes APIs
- the section called "Crea una funzionalità Argo CD"— Crea una funzionalità Argo CD per la distribuzione continua GitOps
- the section called "Crea una capacità kro"— Crea una funzionalità kro per la composizione e l'orchestrazione delle risorse

Elenca le funzionalità

È possibile elencare tutte le risorse funzionali presenti in un cluster.

Console

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Funzionalità.
4. Visualizza le risorse relative alle funzionalità in Funzionalità gestite.

AWS CLI

Usa il `list-capabilities` comando per visualizzare tutte le funzionalità del tuo cluster. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster e sostituiscilo *`my-cluster`* con il nome del 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"
        }
    ]
}
```

Descrivi una funzionalità

Ottieni informazioni dettagliate su una funzionalità specifica, inclusi la configurazione e lo stato.

Console

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Funzionalità.
4. Scegli la funzionalità che desideri visualizzare in Funzionalità gestite.
5. Visualizza i dettagli delle funzionalità, tra cui lo stato, la configurazione e l'ora di creazione.

AWS CLI

Utilizzate il `describe-capability` comando per visualizzare informazioni dettagliate. Sostituisci *`region-code`* con la AWS regione in cui si trova il cluster, sostituisci *`my-cluster`* con il nome del cluster e sostituisci *`capability-name`* con il nome della funzionalità (ack, argocd o kro).

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name
```

Output di esempio:

```
{
  "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"
  }
}
```

Aggiorna la configurazione di una funzionalità

È possibile aggiornare alcuni aspetti della configurazione di una funzionalità dopo la creazione. Le opzioni di configurazione specifiche variano in base al tipo di funzionalità.

> **Note**
>
> Le risorse di funzionalità EKS sono completamente gestite, comprese le patch e gli aggiornamenti delle versioni. L'aggiornamento di una funzionalità aggiornerà la configurazione delle risorse e non comporterà aggiornamenti di versione dei componenti delle funzionalità gestite.

AWS CLI

Usa il `update-capability` comando per modificare una funzionalità:

```
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**
>
> Non tutte le proprietà delle funzionalità possono essere aggiornate dopo la creazione. Fate riferimento alla documentazione specifica sulle funzionalità per i dettagli su cosa è possibile modificare.

Eliminare una funzionalità

Quando non è più necessaria una funzionalità nel cluster, è possibile eliminare la risorsa relativa alla capacità.

> **Important**
>
> Eliminare le risorse del cluster prima di eliminare la funzionalità.
> L'eliminazione di una risorsa di capacità non elimina automaticamente le risorse create tramite tale funzionalità:
>
> - Tutte le Kubernetes Custom Resource Definitions (CRDs) rimangono installate nel cluster.
> - Le risorse ACK rimangono nel cluster e le AWS risorse corrispondenti rimangono nel tuo account
> - Le applicazioni Argo CD e le relative risorse Kubernetes rimangono nel cluster
> - kro ResourceGraphDefinitions e le istanze rimangono nel cluster
>
> È necessario eliminare queste risorse prima di eliminare la funzionalità per evitare risorse orfane.
> Facoltativamente, puoi scegliere di conservare AWS le risorse associate alle risorse ACK Kubernetes. Consulta le considerazioni su ACK

Console

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.

2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Funzionalità.
4. Seleziona la funzionalità che desideri eliminare dall'elenco delle funzionalità gestite.
5. Scegli la funzionalità Elimina.
6. Nella finestra di dialogo di conferma, digita il nome della funzionalità per confermare l'eliminazione.
7. Scegli Elimina.

AWS CLI

Usa il `delete-capability` comando per eliminare una risorsa di funzionalità:

Sostituisci *`region-code`* con la AWS regione in cui si trova il cluster, sostituisci *`my-cluster`* con il nome del cluster e sostituisci *`capability-name`* con il nome della funzionalità da eliminare.

```
aws eks delete-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name
```

Fasi successive

- the section called "risorse Kubernetes"— Scopri le risorse Kubernetes fornite da ciascun tipo di funzionalità
- the section called "Concetti ACK"— Comprendi i concetti e il ciclo di vita delle risorse di ACK
- the section called "Lavorare con Argo CD"— Utilizzo delle funzionalità di Argo CD per i flussi di lavoro GitOps
- the section called "concetti kro"— Comprendi i concetti kro e la composizione delle risorse

Capability: risorse Kubernetes

Dopo aver abilitato una funzionalità sul cluster, molto spesso interagirai con essa creando e gestendo risorse personalizzate Kubernetes nel cluster. Ogni funzionalità fornisce il proprio set di definizioni di risorse personalizzate (CRDs) che estendono l'API Kubernetes con funzionalità specifiche per funzionalità.

Risorse Argo CD

Quando abiliti la funzionalità Argo CD, puoi creare e gestire le seguenti risorse Kubernetes:

Applicazione

Definisce una distribuzione da un repository Git a un cluster di destinazione. `Application`le risorse specificano il repository di origine, lo spazio dei nomi di destinazione e la politica di sincronizzazione. È possibile creare fino a 1000 `Application` risorse per istanza con funzionalità Argo CD.

ApplicationSet

Genera più `Application` risorse a partire da modelli, abilitando implementazioni multi-cluster e multiambiente. `ApplicationSet`le risorse utilizzano i generatori per creare `Application` risorse in modo dinamico sulla base di elenchi di cluster, directory Git o altre fonti.

AppProject

Fornisce il raggruppamento logico e il controllo degli accessi alle risorse. `Application` `AppProject`le risorse definiscono quali repository, cluster e namespace possono essere utilizzati dalle `Application` risorse, abilitando limiti di multi-tenancy e sicurezza.

`Application`Risorsa di esempio:

```
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
```

Per ulteriori informazioni sulle risorse e sui concetti di Argo CD, vederethe section called "Concetti di Argo CD".

risorse kro

Quando abiliti la funzionalità kro, puoi creare e gestire le seguenti risorse Kubernetes:

ResourceGraphDefinition (RGD)

Definisce un'API personalizzata che compone più Kubernetes e AWS risorse in un'astrazione di livello superiore. I team della piattaforma creano `ResourceGraphDefinition` risorse per fornire modelli riutilizzabili con guardrail.

Istanze di risorse personalizzate

Dopo aver creato una `ResourceGraphDefinition` risorsa, puoi creare istanze dell'API personalizzata definita da`ResourceGraphDefinition`. kro crea e gestisce automaticamente le risorse specificate in. `ResourceGraphDefinition`

Risorsa di esempio`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
```

Istanza di `WebApplication` esempio:

```
apiVersion: v1alpha1
kind: WebApplication
metadata:
  name: my-web-app
  namespace: default
spec:
  name: my-web-app
  replicas: 3
```

Quando applichi questa istanza, kro crea automaticamente `Service` le risorse `Deployment` e definite in. `ResourceGraphDefinition`

Per ulteriori informazioni sulle risorse e sui concetti di kro, consulta. the section called "concetti kro"

Risorse ACK

Quando abiliti la funzionalità ACK, puoi creare e gestire AWS risorse utilizzando risorse personalizzate Kubernetes. ACK ne fornisce oltre 200 CRDs per più di 50 AWS servizi, consentendoti di definire AWS le risorse insieme ai carichi di lavoro Kubernetes e di gestire risorse di infrastruttura dedicate con Kubernetes. AWS

Esempi di risorse ACK:

Bucket S3

: `Bucket`le risorse creano e gestiscono bucket Amazon S3 con politiche di controllo delle versioni, crittografia e ciclo di vita.

RDS DBInstance

: `DBInstance`fornitura di risorse e gestione di istanze di database Amazon RDS con backup automatici e finestre di manutenzione.

Tabella DynamoDB

: `Table`le risorse creano e gestiscono tabelle DynamoDB con capacità fornita o su richiesta.

Ruolo IAM

: `Role`le risorse definiscono i ruoli IAM con policy di fiducia e policy di autorizzazione per l'accesso ai servizi. AWS

Funzione Lambda

`Function`le risorse creano e gestiscono funzioni Lambda con configurazione di codice, runtime e ruolo di esecuzione.

Esempio di specificazione di una `Bucket` risorsa:

```
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
```

Per ulteriori informazioni sulle risorse e sui concetti di ACK, vederethe section called "Concetti ACK".

Limiti delle risorse

Le funzionalità EKS hanno i seguenti limiti di risorse:

Limiti di utilizzo di Argo CD:

- Massimo 1000 `Application` risorse per istanza con funzionalità Argo CD
- Massimo 100 cluster remoti configurati per istanza con funzionalità Argo CD

Limiti di configurazione delle risorse:

- Massimo 150 risorse Kubernetes per `Application` risorsa in Argo CD
- Massimo 64 risorse Kubernetes per mese `ResourceGraphDefinition`

> **Note**
>
> Questi limiti si applicano al numero di risorse gestite da ciascuna istanza di funzionalità. Se hai bisogno di limiti più elevati, puoi distribuire funzionalità su più cluster.

Fasi successive

Per le attività specifiche relative alle funzionalità e la configurazione avanzata, consulta i seguenti argomenti:

- the section called "Concetti ACK"— Comprendi i concetti e il ciclo di vita delle risorse di ACK
- the section called "Lavorare con Argo CD"— Utilizzo delle funzionalità di Argo CD per i flussi di lavoro GitOps
- the section called "concetti kro"— Comprendi i concetti kro e la composizione delle risorse

Considerazioni sulle funzionalità EKS

Questo argomento tratta importanti considerazioni sull'utilizzo di EKS Capabilities, tra cui la progettazione del controllo degli accessi, la scelta tra EKS Capabilities e soluzioni autogestite, i modelli architettonici per le implementazioni multi-cluster e le migliori pratiche operative.

Capability, ruoli IAM e Kubernetes RBAC

Ogni risorsa di funzionalità EKS ha un ruolo IAM con funzionalità configurate. Il ruolo di capacità viene utilizzato per concedere le autorizzazioni di AWS servizio alle funzionalità EKS di agire per conto dell'utente. Ad esempio, per utilizzare EKS Capability for ACK per gestire i bucket Amazon S3, concederai le autorizzazioni amministrative di S3 Bucket alla funzionalità, consentendole di creare e gestire i bucket.

Una volta configurata la funzionalità, le risorse S3 in AWS possono essere create e gestite con risorse personalizzate Kubernetes nel cluster. Kubernetes RBAC è il meccanismo di controllo degli accessi all'interno del cluster che consente di determinare quali utenti e gruppi possono creare e gestire tali risorse personalizzate. Ad esempio, concedi a utenti e gruppi Kubernetes RBAC specifici le autorizzazioni per creare e gestire risorse nei namespace di tua scelta. `Bucket`

In questo modo, IAM e Kubernetes RBAC sono due parti del sistema di controllo degli accessi che regola le autorizzazioni relative alle funzionalità e alle risorse EKS. end-to-end È importante

progettare la giusta combinazione di autorizzazioni IAM e policy di accesso RBAC per il tuo caso d'uso.

Per ulteriori informazioni sulla capacità, i ruoli IAM e le autorizzazioni Kubernetes, consulta. the section called "Considerazioni sulle funzionalità EKS"

Modelli di architettura multicluster

Quando distribuisci funzionalità su più cluster, considera questi modelli architettonici comuni:

Hub and Spoke con gestione centralizzata

Esegui tutte e tre le funzionalità in un cluster gestito centralmente per orchestrare i carichi di lavoro e gestire l'infrastruttura cloud su più cluster di carichi di lavoro.

- Argo CD sul cluster di gestione distribuisce applicazioni su cluster di carico di lavoro in diverse regioni o account
- ACK sul cluster di gestione fornisce AWS risorse (RDS, S3, IAM) per tutti i cluster
- kro sul cluster di gestione crea astrazioni di piattaforma portatili che funzionano su tutti i cluster

Questo modello centralizza il carico di lavoro e la gestione dell'infrastruttura cloud e può semplificare le operazioni per le organizzazioni che gestiscono molti cluster.

Decentralizzato GitOps

I carichi di lavoro e l'infrastruttura cloud sono gestiti dalle funzionalità dello stesso cluster in cui vengono eseguiti i carichi di lavoro.

- Argo CD gestisce le risorse delle applicazioni sul cluster locale.
- Le risorse ACK vengono utilizzate per esigenze di cluster e carichi di lavoro.
- Le astrazioni della piattaforma kro vengono installate e orchestrano le risorse locali.

Questo modello decentralizza le operazioni, con i team che gestiscono i propri servizi di piattaforma dedicati in uno o più cluster.

Hub and Spoke con implementazione ACK ibrida

Combina modelli centralizzati e decentralizzati, con implementazioni di applicazioni centralizzate e gestione delle risorse in base all'ambito e alla proprietà.

- Cluster di hub:
  - Argo CD gestisce le GitOps implementazioni nel cluster locale e in tutti i cluster di carichi di lavoro remoti
  - ACK viene utilizzato nel cluster di gestione per le risorse con ambito amministrativo (database di produzione, ruoli IAM,) VPCs
  - kro viene utilizzato nel cluster di gestione per le astrazioni riutilizzabili della piattaforma
- Cluster Spoke:
  - I carichi di lavoro sono gestiti tramite Argo CD sul cluster di hub centralizzato
  - ACK viene utilizzato localmente per le risorse relative ai carichi di lavoro (bucket S3, istanze, code SQS) ElastiCache
  - kro viene utilizzato localmente per la composizione delle risorse e i modelli di blocchi costitutivi

Questo modello separa le preoccupazioni: i team di piattaforma gestiscono l'infrastruttura critica centralmente sui cluster di gestione, includendo facoltativamente i cluster di carichi di lavoro, mentre i team applicativi specificano e gestiscono le risorse cloud insieme ai carichi di lavoro.

Scelta di un pattern

Considerate questi fattori nella scelta di un'architettura:

- Struttura organizzativa: i team di piattaforma centralizzati preferiscono modelli di hub; i team decentralizzati possono preferire funzionalità per cluster
- Ambito delle risorse: le risorse con ambito amministrativo (database, IAM) spesso traggono vantaggio dalla gestione centralizzata; le risorse dei carichi di lavoro (bucket, code) possono essere gestite localmente
- Self-service: i team di piattaforma centralizzati possono creare e distribuire risorse prescrittive personalizzate per consentire un self-service sicuro delle risorse cloud per le esigenze di carichi di lavoro comuni
- Gestione della flotta di cluster: i cluster di gestione centralizzati forniscono un piano di controllo di proprietà del cliente per la gestione della flotta dei cluster EKS, insieme ad altre risorse destinate all'amministrazione
- Requisiti di conformità: alcune organizzazioni richiedono un controllo centralizzato per l'audit e la governance
- Complessità operativa: un minor numero di istanze di funzionalità semplifica le operazioni ma può creare colli di bottiglia

> **Note**
>
> Puoi iniziare con uno schema ed evolvere verso un altro man mano che la tua piattaforma matura. Le funzionalità sono indipendenti: puoi distribuirle in modo diverso tra i cluster in base alle tue esigenze.

Confronto tra le funzionalità EKS e le soluzioni autogestite

EKS Capabilities offre esperienze completamente gestite per i più diffusi strumenti e controller Kubernetes eseguiti in EKS. Ciò si differenzia dalle soluzioni autogestite, che vengono installate e utilizzate nel cluster.

Principali differenze

Implementazione e gestione

AWS gestisce completamente EKS Capabilities senza che sia richiesta l'installazione, la configurazione o la manutenzione dei componenti software. AWS installa e gestisce automaticamente tutte le Kubernetes Custom Resource Definitions (CRDs) richieste nel cluster.

Con le soluzioni autogestite, puoi installare e configurare il software del cluster utilizzando Helm charts, kubectl o altri operatori. Hai il pieno controllo sul ciclo di vita del software e sulla configurazione di runtime delle tue soluzioni autogestite, fornendo personalizzazioni a qualsiasi livello della soluzione.

Operazioni e manutenzione

AWS gestisce l'applicazione di patch e altre operazioni del ciclo di vita del software per EKS Capabilities, con aggiornamenti automatici e patch di sicurezza. EKS Capabilities è integrato con AWS funzionalità per configurazioni semplificate, offre elevata disponibilità e tolleranza ai guasti integrate ed elimina la risoluzione dei problemi all'interno del cluster dei carichi di lavoro dei controller.

Le soluzioni autogestite richiedono il monitoraggio dello stato e dei registri dei componenti, l'applicazione di patch di sicurezza e aggiornamenti delle versioni, la configurazione dell'alta disponibilità con più repliche e budget per le interruzioni dei pod, la risoluzione dei problemi e la risoluzione dei problemi relativi al carico di lavoro dei controller e la gestione di release e versioni.

Avete il pieno controllo sulle vostre implementazioni, ma ciò richiede spesso soluzioni su misura per l'accesso privato ai cluster e altre integrazioni che devono essere in linea con gli standard organizzativi e i requisiti di conformità in materia di sicurezza.

Consumo di risorse

EKS Capabilities viene eseguito in EKS e all'esterno dei cluster, liberando risorse dei nodi e delle risorse del cluster. Le funzionalità non utilizzano risorse del carico di lavoro del cluster, non consumano CPU o memoria sui nodi di lavoro, scalano automaticamente e hanno un impatto minimo sulla pianificazione della capacità del cluster.

Le soluzioni autogestite eseguono controller e altri componenti sui nodi di lavoro, consumando direttamente le risorse dei nodi di lavoro, il cluster IPs e altre risorse del cluster. La gestione dei servizi cluster richiede la pianificazione della capacità per i relativi carichi di lavoro e richiede la pianificazione e la configurazione delle richieste e dei limiti delle risorse per gestire i requisiti di scalabilità e alta disponibilità.

Supporto funzionalità

Essendo funzionalità di servizio completamente gestite, EKS Capabilities sono per loro natura convincenti rispetto alle soluzioni autogestite. Sebbene le funzionalità supportino la maggior parte delle funzionalità e dei casi d'uso, vi sarà una differenza nella copertura rispetto alle soluzioni autogestite.

Con le soluzioni autogestite, puoi controllare completamente la configurazione, le funzionalità opzionali e altri aspetti della funzionalità del tuo software. È possibile scegliere di eseguire immagini personalizzate, personalizzare tutti gli aspetti della configurazione e controllare completamente la funzionalità della soluzione autogestita.

Considerazioni sui costi

Ogni risorsa di funzionalità EKS ha un costo orario correlato, che varia in base al tipo di funzionalità. Le risorse del cluster gestite dalla funzionalità hanno inoltre un costo orario associato ai rispettivi prezzi. Per ulteriori informazioni, consultare Prezzi di Amazon EKS.

Le soluzioni autogestite non hanno costi diretti legati agli AWS addebiti, ma pagano per le risorse di elaborazione del cluster utilizzate dai controller e dai relativi carichi di lavoro. Oltre al consumo di risorse di nodi e cluster, il costo totale di proprietà delle soluzioni autogestite include i costi operativi e le spese di manutenzione, risoluzione dei problemi e supporto.

Scelta tra EKS Capabilities e soluzioni autogestite

EKS Capabilities Prendete in considerazione questa scelta quando desiderate ridurre il sovraccarico operativo e concentrarvi sul valore differenziato del software e dei sistemi, piuttosto che sulle operazioni della piattaforma in cluster per requisiti fondamentali. Utilizzate EKS Capabilities quando desiderate ridurre al minimo l'onere operativo delle patch di sicurezza e della gestione del ciclo di vita del software, liberare risorse di nodi e cluster per i carichi di lavoro delle applicazioni, semplificare la configurazione e la gestione della sicurezza e beneficiare della copertura di supporto. AWS Le funzionalità EKS sono ideali per la maggior parte dei casi d'uso in produzione e rappresentano l'approccio consigliato per le nuove implementazioni.

Soluzioni autogestite Prendi in considerazione questa scelta quando hai bisogno di versioni specifiche dell'API di risorsa Kubernetes, build di controller personalizzate, se disponi di automazione e strumenti esistenti basati su implementazioni autogestite o hai bisogno di una personalizzazione approfondita delle configurazioni di runtime dei controller. Le soluzioni autogestite offrono flessibilità per casi d'uso specializzati e hai il controllo completo sulla configurazione di implementazione e runtime.

> **Note**
>
> EKS Capabilities può coesistere nel cluster con soluzioni autogestite ed è possibile effettuare migrazioni graduali.

Confronti specifici per funzionalità

Per confronti dettagliati, tra cui funzionalità specifiche delle funzionalità, differenze a monte e percorsi di migrazione, consulta:

- the section called “Confronto con l'autogestione”
- the section called “Confronto con l'autogestione”
- the section called “Confronto con quello autogestito”

Implementa AWS risorse da Kubernetes con AWS Controllers for Kubernetes (ACK)

AWS Controllers for Kubernetes (ACK) ti consente di definire e gestire le risorse di servizio direttamente da Kubernetes. AWS Con AWS Controllers for Kubernetes (ACK), puoi gestire le risorse dei carichi di lavoro e l'infrastruttura cloud utilizzando le risorse personalizzate di Kubernetes, insieme ai carichi di lavoro delle applicazioni utilizzando Kubernetes e strumenti familiari. APIs

Con EKS Capabilities, ACK è completamente gestito da, eliminando la necessità di installare AWS, mantenere e scalare i controller ACK sui cluster.

Come funziona ACK

ACK traduce le specifiche delle risorse personalizzate di Kubernetes in chiamate API. AWS Quando crei, aggiorni o elimini una risorsa personalizzata Kubernetes che rappresenta una risorsa di AWS servizio, ACK effettua le chiamate AWS API necessarie per creare, aggiornare o eliminare la risorsa. AWS

Ogni AWS risorsa supportata da ACK ha una propria definizione di risorsa personalizzata (CRD) che definisce lo schema dell'API Kubernetes per specificarne la configurazione. Ad esempio, ACK fornisce CRDs S3 tra cui bucket, bucket policy e altre risorse S3.

ACK riconcilia continuamente lo stato delle AWS risorse con lo stato desiderato definito nelle risorse personalizzate Kubernetes. Se una risorsa si discosta dallo stato desiderato, ACK lo rileva e intraprende azioni correttive per riportarla all'allineamento. Le modifiche alle risorse Kubernetes si riflettono immediatamente sullo stato della AWS risorsa, mentre il rilevamento passivo della deriva e la correzione delle modifiche alle AWS risorse a monte possono richiedere fino a 10 ore (il periodo di risincronizzazione), ma in genere avvengono molto prima.

Esempio di manifesto delle risorse di S3 Bucket

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-ack-bucket
spec:
  name: my-unique-bucket-name
```

Quando applichi questa risorsa personalizzata al tuo cluster, ACK crea un bucket Amazon S3 nel tuo account se non esiste ancora. Le modifiche successive a questa risorsa, ad esempio la specificazione di un livello di storage non predefinito o l'aggiunta di una policy, verranno applicate alla risorsa S3 in. AWS Quando questa risorsa viene eliminata dal cluster, il bucket S3 in AWS viene eliminato per impostazione predefinita.

Vantaggi di ACK

ACK offre la gestione AWS delle risorse nativa di Kubernetes, che consente di gestire AWS le risorse utilizzando gli stessi Kubernetes APIs e gli stessi strumenti utilizzati per le applicazioni. Questo approccio unificato semplifica il flusso di lavoro di gestione dell'infrastruttura eliminando la necessità di passare da uno strumento all'altro o di apprendere sistemi separati. infrastructure-as-code Definisci AWS le tue risorse in modo dichiarativo nei manifesti di Kubernetes, abilitando i GitOps flussi di lavoro e l'infrastruttura come pratiche di codice che si integrano perfettamente con i processi di sviluppo esistenti.

ACK riconcilia continuamente lo stato desiderato delle AWS risorse con il loro stato effettivo, correggendo le deviazioni e garantendo la coerenza nell'intera infrastruttura. Questa riconciliazione continua significa che le out-of-band modifiche imperative alle AWS risorse vengono ripristinate automaticamente in modo che corrispondano alla configurazione dichiarata, mantenendo l'integrità dell'infrastruttura sotto forma di codice. È possibile configurare ACK per gestire le risorse su più AWS account e regioni, abilitando architetture multi-account complesse senza strumenti aggiuntivi.

Per le organizzazioni che migrano da altri strumenti di gestione dell'infrastruttura, ACK supporta l'adozione delle risorse, permettendo di affidare AWS le risorse esistenti alla gestione ACK senza doverle ricreare. ACK fornisce anche risorse di sola lettura per l'osservazione AWS delle risorse senza accesso alle modifiche e annotazioni per conservare facoltativamente le AWS risorse anche quando la risorsa Kubernetes viene eliminata dal cluster.

Per saperne di più e iniziare a usare EKS Capability for ACK, consulta e. the section called “Concetti ACK” the section called “Considerazioni”

AWS Servizi supportati

ACK supporta un'ampia gamma di AWS servizi, tra cui, a titolo esemplificativo ma non esaustivo:

- Amazon EC2
- Simple Storage Service (Amazon S3)

- Amazon RDS
- Amazon DynamoDB
- Amazon ElastiCache
- Amazon EKS
- Amazon SQS
- Amazon SNS
- AWS Lambda
- AWS IAM

Tutti i AWS servizi elencati come Generally Available upstream sono supportati da EKS Capability for ACK. Per ulteriori informazioni, consulta l'elenco completo dei AWS servizi supportati.

Integrazione con altre funzionalità gestite da EKS

ACK si integra con altre funzionalità gestite da EKS.

- Argo CD: utilizza Argo CD per gestire l'implementazione delle risorse ACK su più cluster, abilitando i flussi di GitOps lavoro per la tua infrastruttura. AWS
  - ACK estende i vantaggi dell' GitOps abbinamento con ArgoCD, ma ACK non richiede l'integrazione con git.
- kro (Kube Resource Orchestrator): Usa kro per comporre risorse complesse a partire da risorse ACK, creando astrazioni di livello superiore che semplificano la gestione delle risorse.
  - Puoi creare risorse composite personalizzate con kro che definiscono sia le risorse che le risorse Kubernetes. AWS I membri del team possono utilizzare queste risorse personalizzate per distribuire rapidamente applicazioni complesse.

Guida introduttiva a ACK

Per iniziare a usare EKS Capability for ACK:

1. Crea e configura un ruolo di capacità IAM con le autorizzazioni necessarie affinché ACK gestisca AWS le risorse per tuo conto.
2. Crea una risorsa con funzionalità ACK sul tuo cluster EKS tramite la AWS console, la AWS CLI o la tua infrastruttura preferita come strumento di codice.

3. Applica le risorse personalizzate Kubernetes al tuo cluster per iniziare a gestire le tue risorse in Kubernetes. AWS

Crea una funzionalità ACK

Questo capitolo spiega come creare una funzionalità ACK sul tuo cluster Amazon EKS.

Prerequisiti

Prima di creare una funzionalità ACK, assicurati di avere:

- Cluster Amazon EKS
- Un ruolo di capacità IAM con autorizzazioni per ACK per gestire le AWS risorse
- Autorizzazioni IAM sufficienti per creare risorse di funzionalità sui cluster EKS
- Lo strumento CLI appropriato installato e configurato o l'accesso alla console EKS

Per istruzioni sulla creazione dello IAM Capability Role, consultathe section called “Capacità, ruolo IAM”.

> **Important**
>
> ACK è una funzionalità di gestione dell'infrastruttura che consente di creare, modificare ed eliminare AWS risorse. Si tratta di una funzionalità amministrativa che deve essere controllata con attenzione. Chiunque sia autorizzato a creare risorse Kubernetes nel tuo cluster può creare risorse in modo efficace tramite ACK, fatte salve AWS le autorizzazioni IAM Capability Role. Il ruolo IAM Capability che fornisci determina quali AWS risorse ACK può creare e gestire. Per indicazioni sulla creazione di un ruolo appropriato con autorizzazioni con privilegi minimi, consulta e. the section called “Capacità, ruolo IAM” the section called “Considerazioni sulle funzionalità EKS”

Scegli il tuo strumento

È possibile creare una funzionalità ACK utilizzando Console di gestione AWS, AWS CLI o eksctl:

- the section called “Console”- Usa la console per un'esperienza guidata
- the section called “ AWS CLI”- Usa la AWS CLI per lo scripting e l'automazione

- the section called "eksctl"- Usa eksctl per un'esperienza nativa di Kubernetes

Cosa succede quando si crea una funzionalità ACK

Quando crei una funzionalità ACK:

1. EKS crea il servizio di funzionalità ACK e lo configura per monitorare e gestire le risorse nel cluster
2. Le definizioni di risorse personalizzate (CRDs) sono installate nel cluster
3. Viene creata automaticamente una voce di accesso per il tuo IAM Capability Role con policy di accesso specifiche per ciascuna funzionalità che concedono le autorizzazioni Kubernetes di base (vedi) the section called "Considerazioni sulle funzionalità EKS"
4. La funzionalità presuppone lo IAM Capability Role fornito dall'utente
5. ACK inizia a controllare le proprie risorse personalizzate nel cluster
6. Lo stato della capacità cambia da `CREATING` a `ACTIVE`

Una volta attivo, è possibile creare risorse ACK personalizzate nel cluster per gestire AWS le risorse.

> **Note**
>
> La voce di accesso creata automaticamente include la voce `AmazonEKSACKPolicy` che concede le autorizzazioni ACK per la gestione AWS delle risorse. Alcune risorse ACK che fanno riferimento ai segreti di Kubernetes (come i database RDS con password) richiedono politiche di accesso aggiuntive. Per ulteriori informazioni sulle voci di accesso e su come configurare autorizzazioni aggiuntive, consulta. the section called "Considerazioni sulle funzionalità EKS"

Fasi successive

Dopo aver creato la funzionalità ACK:

- the section called "Concetti ACK"- Comprendi i concetti di ACK e inizia a usare le AWS risorse
- the section called "Concetti ACK"- Scopri la riconciliazione, le esportazioni sul campo e i modelli di adozione delle risorse
- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM e i modelli multi-account

Creare una funzionalità ACK utilizzando la console

Questo argomento descrive come creare una funzionalità AWS Controllers for Kubernetes (ACK) utilizzando. Console di gestione AWS

Crea la funzionalità ACK

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Funzionalità.
4. Nella barra di navigazione a sinistra, scegli AWS Controllers for Kubernetes (ACK).
5. Scegli la funzionalità Crea AWS controller per Kubernetes.
6. Per IAM Capability Role:
   - Se disponi già di un IAM Capability Role, selezionalo dal menu a discesa
   - Se devi creare un ruolo, scegli Crea ruolo di amministratore

     Questo apre la console IAM in una nuova scheda con la policy di fiducia precompilata e la policy `AdministratorAccess` gestita. Se preferisci, puoi deselezionare questa policy e aggiungere altre autorizzazioni.

     Dopo aver creato il ruolo, torna alla console EKS e il ruolo verrà selezionato automaticamente.

     > **Important**
     >
     > La `AdministratorAccess` politica suggerita concede ampie autorizzazioni e ha lo scopo di semplificare l'avvio. Per uso in produzione, sostituiscila con una politica personalizzata che conceda solo le autorizzazioni necessarie per i AWS servizi specifici che intendi gestire con ACK. Per indicazioni sulla creazione di politiche con privilegi minimi, consulta e. the section called “Configurazione delle autorizzazioni ” the section called “Considerazioni sulle funzionalità EKS”
7. Scegli Create (Crea).

Inizia il processo di creazione delle capacità.

Verifica che la funzionalità sia attiva

1. Nella scheda Funzionalità, visualizza lo stato della capacità ACK.

2. Attendi che lo stato cambi da `CREATING` a`ACTIVE`.
3. Una volta attiva, la funzionalità è pronta per l'uso.

Per informazioni sullo stato delle funzionalità e sulla risoluzione dei problemi, vederethe section called "Lavorare con le funzionalità".

Verifica che siano disponibili risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate ACK siano disponibili nel cluster.

Utilizzo della console

1. Accedi al tuo cluster nella console Amazon EKS
2. Scegli la scheda Risorse
3. Scegli Estensioni
4. Scegliere CustomResourceDefinitions

Dovresti vedere una serie di AWS risorse CRDs elencate.

Usare kubectl

```
kubectl api-resources | grep services.k8s.aws
```

Dovresti vedere un certo numero di risorse APIs elencate. AWS

> **Note**
>
> La funzionalità di AWS Controllers for Kubernetes installerà una serie di risorse CRDs per una varietà di risorse. AWS

Fasi successive

- the section called "Concetti ACK"- Comprendi i concetti di ACK e inizia
- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM per altri servizi AWS
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa di funzionalità ACK

Crea una funzionalità ACK utilizzando la AWS CLI

Questo argomento descrive come creare una funzionalità AWS Controllers for Kubernetes (ACK) utilizzando la CLI. AWS

Prerequisiti

- AWS CLI: versione `2.12.3` o successiva. Per verificare la tua versione, `aws --version` esegui. Per ulteriori informazioni, consulta la Guida per l'utente all'installazione nell'interfaccia a riga di AWS comando.
- **`kubectl`**— Uno strumento da riga di comando per lavorare con i cluster Kubernetes. Per ulteriori informazioni, consulta the section called “Configura kubectl ed eksctl”.

Fase 1: creazione di un ruolo IAM Capability

Crea un file di policy di fiducia:

```
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
```

Crea il ruolo IAM:

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

Allega la policy `AdministratorAccess` gestita al ruolo:

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

> **Important**
>
> La `AdministratorAccess` politica suggerita concede ampie autorizzazioni e ha lo scopo di semplificare l'avvio. Per uso in produzione, sostituiscila con una politica personalizzata che conceda solo le autorizzazioni necessarie per i AWS servizi specifici che intendi gestire con ACK. Per indicazioni sulla creazione di politiche con privilegi minimi, consulta e. the section called "Configurazione delle autorizzazioni " the section called "Considerazioni sulle funzionalità EKS"

Fase 2: Creare la funzionalità ACK

Crea la risorsa di funzionalità ACK sul tuo cluster. Sostituiscila *`region-code`* con la AWS regione in cui si trova il cluster e sostituiscila *`my-cluster`* con il nome del 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
```

Il comando viene restituito immediatamente, ma la funzionalità impiega del tempo per diventare attiva poiché EKS crea l'infrastruttura e i componenti di funzionalità richiesti. EKS installerà le Kubernetes Custom Resource Definitions relative a questa funzionalità nel cluster non appena viene creata.

> **Note**
>
> Se ricevi un errore che indica che il cluster non esiste o non disponi delle autorizzazioni, verifica:
>
> - Il nome del cluster è corretto

- La tua AWS CLI è configurata per la regione corretta
- Disponi delle autorizzazioni IAM richieste

Fase 3: Verifica che la funzionalità sia attiva

Attendi che la funzionalità diventi attiva. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster e sostituiscilo *my-cluster* con il nome del cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack \
  --query 'capability.status' \
  --output text
```

La funzionalità è pronta quando viene visualizzato lo statoACTIVE. Non continuare con il passaggio successivo fino a quando lo stato non sarà raggiuntoACTIVE.

Puoi anche visualizzare i dettagli completi delle funzionalità:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack
```

Fase 4: Verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate ACK siano disponibili nel cluster:

```
kubectl api-resources | grep services.k8s.aws
```

Dovresti vedere un certo numero di AWS risorse APIs elencate.

> **Note**
>
> La funzionalità di AWS Controllers for Kubernetes installerà una serie di risorse CRDs per una varietà di risorse. AWS

Fasi successive

- the section called "Concetti ACK"- Comprendi i concetti di ACK e inizia
- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM per altri servizi AWS
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa di funzionalità ACK

Crea una funzionalità ACK usando eksctl

Questo argomento descrive come creare una funzionalità AWS Controllers for Kubernetes (ACK) usando eksctl.

> **Note**
>
> I passaggi seguenti richiedono la versione eksctl o successiva. `0.220.0` Per verificare la tua versione, esegui. `eksctl version`

Fase 1: Creare un ruolo di capacità IAM

Crea un file di policy di fiducia:

```
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
```

Crea il ruolo IAM:

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

Allega la policy `AdministratorAccess` gestita al ruolo:

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

Important

La `AdministratorAccess` politica suggerita concede ampie autorizzazioni e ha lo scopo di semplificare l'avvio. Per uso in produzione, sostituiscila con una politica personalizzata che conceda solo le autorizzazioni necessarie per i AWS servizi specifici che intendi gestire con ACK. Per indicazioni sulla creazione di politiche con privilegi minimi, consulta e. the section called “Configurazione delle autorizzazioni ” the section called “Considerazioni sulle funzionalità EKS”

Important

Questa politica concede le autorizzazioni per la gestione dei bucket S3 con`"Resource": "*"`, che consente le operazioni su tutti i bucket S3.
Per uso in produzione: * Limita il `Resource` campo a bucket ARNs o modelli di nomi specifici * Usa le chiavi di condizione IAM per limitare l'accesso tramite tag di risorsa * Concedi solo le autorizzazioni minime necessarie per il tuo caso d'uso
Per altri AWS servizi, consulta. the section called “Configurazione delle autorizzazioni ”

Collegare la policy al ruolo:

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):policy/ACKS3Policy
```

Fase 2: Creare la funzionalità ACK

Crea la funzionalità ACK usando eksctl. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster e *my-cluster* sostituiscilo con il nome del 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**
>
> La `--ack-service-controllers` bandiera è facoltativa. Se omesso, ACK abilita tutti i controller disponibili. Per prestazioni e sicurezza migliori, valuta la possibilità di abilitare solo i controller di cui hai bisogno. Puoi specificare più controller: `--ack-service-controllers s3,rds,dynamodb`

Il comando viene restituito immediatamente, ma la funzionalità impiega del tempo per diventare attiva.

Fase 3: Verificare che la funzionalità sia attiva

Verifica lo stato della capacità:

```
eksctl get capability \
  --cluster [.replaceable]`my-cluster` \
  --region [.replaceable]`region-code` \
  --name ack
```

La funzionalità è pronta quando viene visualizzato lo stato`ACTIVE`.

Fase 4: Verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate ACK siano disponibili nel cluster:

```
kubectl api-resources | grep services.k8s.aws
```

Dovresti vedere un certo numero di AWS risorse APIs elencate.

> **Note**
>
> La funzionalità di AWS Controllers for Kubernetes installerà una serie di risorse CRDs per una varietà di risorse. AWS

Fasi successive

- the section called "Concetti ACK"- Comprendi i concetti di ACK e inizia
- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM per altri servizi AWS
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa di funzionalità ACK

Concetti ACK

ACK gestisce AWS le risorse tramite Kubernetes APIs riconciliando continuamente lo stato desiderato nei manifesti con lo stato effettivo in cui si trovano. AWS Quando crei o aggiorni una risorsa personalizzata Kubernetes, ACK effettua le chiamate AWS API necessarie per creare o modificare la AWS risorsa corrispondente, quindi la monitora per rilevare eventuali variazioni e aggiorna lo stato di Kubernetes in modo che rifletta lo stato corrente. Questo approccio consente di gestire l'infrastruttura utilizzando strumenti e flussi di lavoro Kubernetes familiari, mantenendo al contempo la coerenza tra il cluster e. AWS

Questo argomento spiega i concetti fondamentali alla base del modo in cui ACK gestisce AWS le risorse tramite Kubernetes. APIs

Guida introduttiva a ACK

Dopo aver creato la funzionalità ACK (vedithe section called "Crea una funzionalità ACK"), puoi iniziare a gestire AWS le risorse utilizzando i manifest Kubernetes nel tuo cluster.

Ad esempio, crea questo manifesto del bucket S3 in, scegliendo il tuo nome di bucket `bucket.yaml` univoco.

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-test-bucket
  namespace: default
spec:
  name: my-unique-bucket-name-12345
```

Applica il manifesto:

```
kubectl apply -f bucket.yaml
```

Controlla lo stato:

```
kubectl get bucket my-test-bucket
kubectl describe bucket my-test-bucket
```

Verifica che il bucket sia stato creato in AWS:

```
aws s3 ls | grep my-unique-bucket-name-12345
```

Elimina la risorsa Kubernetes:

```
kubectl delete bucket my-test-bucket
```

Verifica che il bucket sia stato eliminato da: AWS

```
aws s3 ls | grep my-unique-bucket-name-12345
```

Il bucket non dovrebbe più apparire nell'elenco, a dimostrazione del fatto che ACK gestisce l'intero ciclo di vita delle risorse. AWS

Per ulteriori informazioni su come iniziare a usare ACK, consulta Getting Started with ACK.

Ciclo di vita delle risorse e riconciliazione

ACK utilizza un ciclo di riconciliazione continuo per garantire che le AWS risorse corrispondano allo stato desiderato definito nei manifesti di Kubernetes.

Come funziona la riconciliazione:

1. Crei o aggiorni una risorsa personalizzata Kubernetes (ad esempio, un bucket S3)
2. ACK rileva la modifica e confronta lo stato desiderato con lo stato effettivo in AWS
3. Se sono diversi, ACK effettua chiamate AWS API per riconciliare la differenza
4. ACK aggiorna lo stato delle risorse in Kubernetes in modo che rifletta lo stato corrente
5. Il ciclo si ripete continuamente, in genere ogni poche ore

La riconciliazione viene attivata quando si crea una nuova risorsa Kubernetes, si aggiorna una risorsa esistente o quando ACK rileva una deviazione `spec` dovuta a modifiche manuali apportate all'esterno di ACK. AWS Inoltre, ACK esegue una riconciliazione periodica con un periodo di risincronizzazione di 10 ore. Le modifiche alle risorse Kubernetes innescano la riconciliazione immediata, mentre il rilevamento passivo della deriva delle modifiche alle risorse upstream avviene durante la risincronizzazione periodica. AWS

Quando si esegue l'esempio introduttivo riportato sopra, ACK esegue questi passaggi:

1. Controlla se il bucket esiste in AWS
2. In caso contrario, chiama `s3:CreateBucket`
3. Aggiorna lo stato di Kubernetes con l'ARN e lo stato del bucket
4. Continua il monitoraggio della deriva

Per saperne di più su come funziona ACK, consulta ACK Reconciliation.

Condizioni di status

Le risorse ACK utilizzano le condizioni di stato per comunicare il proprio stato. La comprensione di queste condizioni consente di risolvere i problemi e comprendere lo stato delle risorse.

- Pronta: indica che la risorsa è pronta per essere consumata (condizione Kubernetes standardizzata).

- ACK. ResourceSynced: Indica che le specifiche della risorsa corrispondono allo stato della AWS risorsa.
- ack.Terminal: indica che si è verificato un errore irreversibile.
- ACK.Adopted: indica che la risorsa è stata adottata da una risorsa esistente anziché creata nuova. AWS
- ACK.Recoverable: indica un errore recuperabile che può essere risolto senza aggiornare le specifiche.
- ACK.advisory: fornisce informazioni consultive sulla risorsa.
- ACK. LateInitialized: Indica se l'inizializzazione tardiva dei campi è completa.
- ACK. ReferencesResolved: Indica se tutti `AWSResourceReference` i campi sono stati risolti.
- ACK. IAMRoleSelezionato: indica se è stato selezionato un IAMRole selettore per gestire questa risorsa.

Verifica lo stato della risorsa:

```
# 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")]}'
```

Esempio di stato:

```
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
```

Per ulteriori informazioni sullo stato e le condizioni ACK, consulta le Condizioni ACK.

Politiche di cancellazione

La politica di eliminazione di ACK controlla cosa succede alle AWS risorse quando elimini la risorsa Kubernetes.

Elimina (impostazione predefinita)

La AWS risorsa viene eliminata quando elimini la risorsa Kubernetes: questo è il comportamento predefinito.

```
# No annotation needed - this is the default
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: temp-bucket
spec:
  name: temporary-bucket
```

L'eliminazione di questa risorsa elimina il bucket S3. AWS

Mantenimento

La AWS risorsa viene conservata quando elimini la risorsa 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
```

L'eliminazione di questa risorsa la rimuove da Kubernetes ma lascia il bucket S3 attivo. AWS

La `retain` policy è utile per i database di produzione che dovrebbero durare più a lungo della risorsa Kubernetes, per le risorse condivise utilizzate da più applicazioni, per le risorse con dati importanti

che non devono essere eliminati accidentalmente o per la gestione temporanea di ACK in cui si adotta una risorsa, la si configura e poi la si rimette alla gestione manuale.

Per ulteriori informazioni sulla politica di eliminazione ACK, consulta ACK Deletion Policy.

Adozione delle risorse

L'adozione consente di affidare AWS le risorse esistenti alla gestione ACK senza doverle ricreare.

Quando usare l'adozione:

- Migrazione dell'infrastruttura esistente alla gestione ACK
- Ripristino di risorse orfane in caso di eliminazione accidentale di AWS risorse in Kubernetes
- Importazione di risorse create da altri strumenti (, Terraform) CloudFormation

Come funziona l'adozione:

```
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
```

Quando crei questa risorsa:

1. ACK verifica se esiste un bucket con quel nome in AWS
2. Se trovato, ACK lo adotta (nessuna chiamata API da creare)
3. ACK legge la configurazione corrente da AWS
4. ACK aggiorna lo stato di Kubernetes in modo che rifletta lo stato effettivo
5. Gli aggiornamenti futuri riconciliano normalmente la risorsa

Una volta adottate, le risorse vengono gestite come qualsiasi altra risorsa ACK e l'eliminazione della risorsa Kubernetes comporterà l'eliminazione della risorsa, a meno che non si utilizzi la politica AWS di eliminazione. `retain`

Quando si adottano risorse, la AWS risorsa deve già esistere e ACK necessita delle autorizzazioni di lettura per scoprirla. La `adopt-or-create` politica adotta la risorsa se esiste o la crea se non esiste. Ciò è utile quando si desidera un flusso di lavoro dichiarativo che funzioni indipendentemente dall'esistenza o meno della risorsa.

Per ulteriori informazioni sull'adozione delle risorse ACK, consulta ACK Resource Adoption.

Risorse tra account e regioni

ACK può gestire le risorse in diversi AWS account e regioni da un singolo cluster.

Annotazioni di risorse interregionali

È possibile specificare la regione di una AWS risorsa utilizzando un'annotazione:

```
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
```

Puoi anche specificare la regione di tutte le AWS risorse create in un determinato namespace:

Annotazioni dello spazio dei nomi

Imposta una regione predefinita per tutte le risorse in un namespace:

```
apiVersion: v1
kind: Namespace
metadata:
  name: production
  annotations:
    services.k8s.aws/default-region: us-west-2
```

Le risorse create in questo namespace utilizzano questa regione a meno che non vengano sovrascritte da un'annotazione a livello di risorsa.

Più account

Usa IAM Role Selectors per mappare ruoli IAM specifici ai namespace:

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
  name: target-account-config
spec:
  arn: arn:aws:iam::444455556666:role/ACKTargetAccountRole
  namespaceSelector:
    names:
      - production
```

Le risorse create nello spazio dei nomi mappato utilizzano automaticamente il ruolo specificato.

Per ulteriori informazioni sugli IAM Role Selectors, consulta ACK Cross-Account Resource Management. Per i dettagli sulla configurazione tra account, consulta. the section called "Configurazione delle autorizzazioni "

Gestione degli errori e comportamento dei nuovi tentativi

ACK gestisce automaticamente gli errori transitori e riprova le operazioni non riuscite.

Strategia di riprova:

- Gli errori transitori (limitazione della velocità, problemi temporanei di servizio, autorizzazioni insufficienti) attivano nuovi tentativi automatici
- Il backoff esponenziale impedisce il sovraccarico AWS APIs
- Il numero massimo di tentativi di riprova varia in base al tipo di errore
- Gli errori permanenti (parametri non validi, conflitti tra nomi di risorse) non riprovano

Controlla lo stato delle risorse per i dettagli degli errori utilizzando: `kubectl describe`

```
kubectl describe bucket my-bucket
```

Cerca le condizioni di stato con messaggi di errore, gli eventi che mostrano i recenti tentativi di riconciliazione e il `message` campo nelle condizioni di stato che spiegano gli errori. Gli errori più comuni includono autorizzazioni IAM insufficienti, conflitti tra i nomi delle risorse, valori di configurazione non validi nelle AWS quote di servizio e il `spec` superamento delle quote di servizio. AWS

Per la risoluzione degli errori più comuni, consulta. the section called "Risoluzione dei problemi"

Composizione delle risorse con kro

Per comporre e connettere più risorse ACK insieme, usa EKS Capability for kro (Kube Resource Orchestrator). kro fornisce un modo dichiarativo per definire gruppi di risorse, passando la configurazione tra le risorse per gestire semplicemente modelli di infrastruttura complessi.

Per esempi dettagliati di creazione di composizioni di risorse personalizzate con risorse ACK, vedi the section called "concetti kro"

Fasi successive

- the section called "Considerazioni"- Modelli e strategie di integrazione specifici per EKS

Configurare le autorizzazioni ACK

ACK richiede le autorizzazioni IAM per creare e gestire AWS risorse per tuo conto. Questo argomento spiega come IAM collabora con ACK e fornisce indicazioni sulla configurazione delle autorizzazioni per diversi casi d'uso.

Come funziona IAM con ACK

ACK utilizza i ruoli IAM per autenticarsi con le tue risorse AWS ed eseguire azioni sulle tue risorse. Esistono due modi per fornire le autorizzazioni a ACK:

Ruolo di capacità: il ruolo IAM fornito durante la creazione della funzionalità ACK. Questo ruolo viene utilizzato di default per tutte le operazioni ACK.

Selettori di ruolo IAM: ruoli IAM aggiuntivi che possono essere mappati su namespace o risorse specifici. Questi ruoli hanno la precedenza sul ruolo di capacità per le risorse che rientrano nel loro ambito.

Quando ACK deve creare o gestire una risorsa, determina quale ruolo IAM utilizzare:

1. Controlla se un IAMRole selettore corrisponde allo spazio dei nomi della risorsa
2. Se viene trovata una corrispondenza, assumi quel ruolo IAM
3. Altrimenti, usa il ruolo Capability

Questo approccio consente una gestione flessibile delle autorizzazioni, da semplici configurazioni a ruolo singolo a configurazioni complesse con più account e più team.

Guida introduttiva: configurazione semplice delle autorizzazioni

Per casi di sviluppo, test o semplici casi d'uso, puoi aggiungere tutte le autorizzazioni di servizio necessarie direttamente al Capability Role.

Questo approccio funziona bene quando:

- Stai iniziando con ACK
- Tutte le risorse si trovano nello stesso AWS account
- Un unico team gestisce tutte le risorse ACK
- Ti fidi che tutti gli utenti ACK abbiano le stesse autorizzazioni

Best practice di produzione: IAM Role Selectors

Per gli ambienti di produzione, utilizza IAM Role Selectors per implementare l'accesso con privilegi minimi e l'isolamento a livello di namespace.

Quando si utilizza IAM Role Selectors, il Capability Role necessita solo delle autorizzazioni per assumere i ruoli specifici del servizio. `sts:AssumeRole` `sts:TagSession` Non è necessario aggiungere alcuna autorizzazione di AWS servizio (come S3 o RDS) al Capability Role stesso: tali autorizzazioni vengono concesse ai singoli ruoli IAM assunti dal Capability Role.

Scelta tra modelli di autorizzazione:

Utilizza le autorizzazioni dirette (aggiungendo le autorizzazioni di servizio al ruolo Capability) quando:

- Stai iniziando e desideri la configurazione più semplice
- Tutte le risorse si trovano nello stesso account del cluster
- Hai requisiti di autorizzazione amministrativi a livello di cluster
- Tutti i team possono condividere le stesse autorizzazioni

Usa IAM Role Selectors quando:

- Gestione delle risorse su più account AWS
- Team o namespace diversi richiedono autorizzazioni diverse
- È necessario un controllo granulare degli accessi per namespace
- Vuoi seguire le pratiche di sicurezza con privilegi minimi

Puoi iniziare con autorizzazioni dirette e migrare a IAM Role Selectors in un secondo momento, man mano che le tue esigenze crescono.

Perché utilizzare IAM Role Selectors in produzione:

- Privilegio minimo: ogni namespace ottiene solo le autorizzazioni di cui ha bisogno
- Isolamento del team: il team A non può utilizzare accidentalmente le autorizzazioni del team B
- Controllo più semplice: mappatura chiara di quale namespace utilizza quale ruolo
- Supporto per più account: necessario per la gestione delle risorse in più account
- Separazione delle preoccupazioni: servizi o ambienti diversi utilizzano ruoli diversi

Configurazione di base di IAM Role Selector

Fase 1: Creare un ruolo IAM specifico per il servizio

Crea un ruolo IAM con autorizzazioni per servizi specifici: AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Configura la policy di fiducia per consentire al Capability Role di assumerla:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ACKCapabilityRole"
      },
      "Action": ["sts:AssumeRole", "sts:TagSession"]
```

```
    }
  ]
}
```

Fase 2: concedere AssumeRole l'autorizzazione a Capability Role

Aggiungi l'autorizzazione al ruolo di capacità per assumere il ruolo specifico del servizio:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole", "sts:TagSession"],
      "Resource": "arn:aws:iam::111122223333:role/ACK-S3-Role"
    }
  ]
}
```

Fase 3: Creare un selettore IAMRole

Mappa il ruolo IAM su un 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
```

Fase 4: Creare risorse nello spazio dei nomi mappato

Le risorse nello spazio dei `s3-resources` nomi utilizzano automaticamente il ruolo specificato:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-bucket
  namespace: s3-resources
spec:
```

```
  name: my-production-bucket
```

Gestione di più account

Usa IAM Role Selectors per gestire le risorse su più AWS account.

Fase 1: Creare un ruolo IAM per più account

Nell'account di destinazione (444455556666), crea un ruolo che si fidi del ruolo di capacità dell'account di origine:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ACKCapabilityRole"
      },
      "Action": ["sts:AssumeRole", "sts:TagSession"]
    }
  ]
}
```

Assegna autorizzazioni specifiche del servizio a questo ruolo.

Passaggio 2: concedere l'autorizzazione AssumeRole

Nell'account di origine (111122223333), consenti al ruolo Capability di assumere il ruolo dell'account di destinazione:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole", "sts:TagSession"],
      "Resource": "arn:aws:iam::444455556666:role/ACKTargetAccountRole"
    }
  ]
}
```

Fase 3: Creare un selettore IAMRole

Mappa il ruolo tra account diversi su un namespace:

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
  name: production-account-config
spec:
  arn: arn:aws:iam::444455556666:role/ACKTargetAccountRole
  namespaceSelector:
    names:
      - production
```

Fase 4: Creare risorse

Le risorse nel `production` namespace vengono create nell'account di destinazione:

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-bucket
  namespace: production
spec:
  name: my-cross-account-bucket
```

Tag di sessione

La funzionalità EKS ACK imposta automaticamente i tag di sessione su tutte le richieste AWS API. Questi tag consentono il controllo e la verifica granulari degli accessi identificando l'origine di ogni richiesta.

Tag di sessione disponibili

I seguenti tag di sessione sono inclusi in ogni chiamata AWS API effettuata da ACK:

Chiave del tag	Description
`eks:eks-capability-arn`	L'ARN della capacità EKS che effettua la richiesta
`eks:kubernetes-namespace`	Lo spazio dei nomi Kubernetes della risorsa gestita

Chiave del tag	Description
`eks:kubernetes-api-group`	Il gruppo di API Kubernetes della risorsa (ad esempio,) `s3.services.k8s.aws`

Utilizzo dei tag di sessione per il controllo degli accessi

Puoi utilizzare questi tag di sessione nelle condizioni delle policy IAM per limitare le risorse che ACK può gestire. Ciò fornisce un ulteriore livello di sicurezza oltre ai selettori di ruolo IAM basati su namespace.

Esempio: limita in base allo spazio dei nomi

Consenti a ACK di creare bucket S3 solo quando la richiesta proviene dal namespace: `production`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:CreateBucket",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/eks:kubernetes-namespace": "production"
        }
      }
    }
  ]
}
```

Esempio: limita per capacità

Consenti azioni solo da una funzionalità ACK specifica:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
```

```
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/eks:eks-capability-arn": "arn:aws:eks:us-
west-2:111122223333:capability/my-cluster/ack/my-ack"
        }
      }
    }
  ]
}
```

> **Note**
>
> I tag di sessione sono diversi da ACK autogestiti, che non imposta questi tag per impostazione predefinita. Ciò consente un controllo degli accessi più granulare con la funzionalità gestita.

Modelli avanzati di IAM Role Selector

Per una configurazione avanzata che include selettori di etichette, mappatura dei ruoli specifici delle risorse ed esempi aggiuntivi, consulta la documentazione ACK IRSA.

Fasi successive

- the section called "Concetti ACK"- Comprendi i concetti e il ciclo di vita delle risorse di ACK
- the section called "Concetti ACK"- Scopri le politiche di adozione ed eliminazione delle risorse
- the section called "Considerazioni sulle funzionalità EKS"- Comprendi le migliori pratiche di sicurezza per quanto riguarda le funzionalità

Considerazioni ACK per EKS

Questo argomento tratta importanti considerazioni sull'utilizzo di EKS Capability for ACK, tra cui la configurazione IAM, i modelli multi-account e l'integrazione con altre funzionalità EKS.

Modelli di configurazione IAM

La funzionalità ACK utilizza un IAM Capability Role con AWS cui eseguire l'autenticazione. Scegli il modello IAM giusto in base alle tue esigenze.

Semplice: ruolo a capacità singola

Per casi di sviluppo, test o semplici casi d'uso, concedi tutte le autorizzazioni necessarie direttamente al Capability Role.

Quando usare:

- Guida introduttiva a ACK
- Implementazioni con un solo account
- Tutte le risorse sono gestite da un unico team
- Ambienti di sviluppo e test

Esempio: aggiungi le autorizzazioni S3 e RDS al tuo Capability Role con condizioni di etichettatura delle risorse:

```
{
  "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"]
        },
      }
    }
  ]
}
```

Questo esempio limita le operazioni S3 e RDS a regioni specifiche e richiede che le risorse RDS abbiano un tag. `ManagedBy: ACK`

Produzione: IAM Role Selectors

Per gli ambienti di produzione, utilizza IAM Role Selectors per implementare l'accesso con privilegi minimi e l'isolamento a livello di namespace.

Quando usare:

- Ambienti di produzione
- Cluster con più team
- Gestione delle risorse su più account
- Requisiti di sicurezza con privilegi minimi
- Servizi diversi richiedono autorizzazioni diverse

Vantaggi:

- Ogni namespace ottiene solo le autorizzazioni necessarie
- Isolamento del team: il team A non può utilizzare le autorizzazioni del team B
- Controllo e conformità più semplici
- Necessario per la gestione delle risorse tra account

Per una configurazione dettagliata di IAM Role Selector, consulta. the section called “Configurazione delle autorizzazioni ”

Integrazione con altre funzionalità EKS

GitOps con Argo CD

Usa il CD EKS Capability for Argo per distribuire risorse ACK dai repository Git, abilitando GitOps flussi di lavoro per la gestione dell'infrastruttura.

Considerazioni:

- Archivia le risorse ACK insieme ai manifesti delle applicazioni per end-to-end GitOps
- Organizza per ambiente, servizio o tipo di risorsa in base alla struttura del tuo team
- Usa la sincronizzazione automatica di Argo CD per una riconciliazione continua

- Abilita la potatura per rimuovere automaticamente le risorse eliminate
- Prendi in considerazione hub-and-spoke i modelli per la gestione dell'infrastruttura multicluster

GitOps fornisce audit trail, funzionalità di rollback e gestione dichiarativa dell'infrastruttura. Per ulteriori informazioni su Argo CD, vedere. the section called "Lavorare con Argo CD"

Composizione delle risorse con kro

Usa EKS Capability for kro (Kube Resource Orchestrator) per comporre più risorse ACK in astrazioni di livello superiore e personalizzate. APIs

Quando usare kro con ACK:

- Crea modelli riutilizzabili per stack di infrastruttura comuni (database + backup + monitoraggio)
- Crea piattaforme self-service semplificate per i team applicativi APIs
- Gestisci le dipendenze delle risorse e passa i valori tra le risorse (funzione ARN a Lambda del bucket S3)
- Standardizza le configurazioni dell'infrastruttura tra i team
- Riduci la complessità nascondendo i dettagli di implementazione dietro risorse personalizzate

Modelli di esempio:

- Stack di applicazioni: bucket S3+coda SQS+configurazione delle notifiche
- Configurazione del database: istanza RDS + gruppo di parametri + gruppo di sicurezza + segreti
- Rete: VPC + sottoreti + tabelle di routing + gruppi di sicurezza

kro gestisce l'ordinamento delle dipendenze, la propagazione dello stato e la gestione del ciclo di vita per le risorse composte. Per ulteriori informazioni su kro, consulta. the section called "concetti kro"

Organizzazione delle risorse

Organizza le risorse ACK utilizzando i namespace e i tag delle AWS risorse Kubernetes per migliorare la gestione, il controllo degli accessi e il monitoraggio dei costi.

Organizzazione dei namespace

Usa gli spazi dei nomi Kubernetes per separare logicamente le risorse ACK per ambiente (produzione, gestione temporanea, sviluppo), team (piattaforma, dati, ML) o applicazione.

Vantaggi:

- RBAC con ambito namespace per il controllo degli accessi
- Imposta le regioni predefinite per namespace utilizzando le annotazioni
- Gestione e pulizia delle risorse più semplici
- Separazione logica in linea con la struttura organizzativa

Aggiunta di tag alle risorse

La funzionalità EKS ACK applica automaticamente i tag predefiniti a tutte le AWS risorse che crea. Questi tag differiscono dagli ACK autogestiti e offrono una tracciabilità avanzata.

Tag predefiniti applicati dalla funzionalità:

Tag Key	Description
`eks:controller-version`	La versione del controller ACK
`eks:kubernetes-namespace`	Lo spazio dei nomi Kubernetes della risorsa ACK
`eks:kubernetes-resource-name`	Il nome della risorsa Kubernetes
`eks:kubernetes-api-group`	Il gruppo di API Kubernetes (ad esempio,) `s3.services.k8s.aws`
`eks:eks-capability-arn`	L'ARN della funzionalità EKS ACK

> **Note**
>
> ACK autogestito utilizza diversi tag predefiniti: `services.k8s.aws/controller-version` e. `services.k8s.aws/namespace` I tag della funzionalità utilizzano il `eks:` prefisso per garantire la coerenza con altre funzionalità EKS.

Tag aggiuntivi consigliati:

Aggiungi tag personalizzati per l'allocazione dei costi, il monitoraggio della proprietà e per scopi organizzativi:

- Ambiente (produzione, allestimento, sviluppo)
- Proprietà del team o del reparto
- Centro di costo per l'allocazione della fatturazione
- Nome dell'applicazione o del servizio

Migrazione da altri Infrastructure-as-code strumenti

Molte organizzazioni stanno trovando valore nella standardizzazione su Kubernetes oltre all'orchestrazione dei carichi di lavoro. La migrazione dell'infrastruttura e della gestione AWS delle risorse verso ACK consente di standardizzare la gestione dell'infrastruttura utilizzando Kubernetes insieme ai carichi di lavoro delle applicazioni. APIs

Vantaggi della standardizzazione su Kubernetes per l'infrastruttura:

- Un'unica fonte di verità: gestisci sia le applicazioni che l'infrastruttura in Kubernetes, abilitando una pratica end-to-end GitOps
- Strumenti unificati: i team utilizzano le risorse e gli strumenti Kubernetes anziché apprendere più strumenti e framework
- Riconciliazione coerente: ACK riconcilia continuamente le AWS risorse come fa Kubernetes per i carichi di lavoro, rilevando e correggendo le deviazioni rispetto agli strumenti imperativi
- Composizioni native: con kro e ACK insieme, puoi fare riferimento alle risorse direttamente nelle applicazioni e nei manifesti delle risorse, passando stringhe di connessione e tra AWS le risorse ARNs
- Operazioni semplificate: un unico piano di controllo per implementazioni, rollback e osservabilità sull'intero sistema

ACK supporta l'adozione AWS delle risorse esistenti senza ricrearle, abilitando la migrazione senza tempi di inattività da Terraform o da CloudFormation risorse esterne al cluster.

Adotta una risorsa esistente:

```
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
```

Una volta adottata, la risorsa viene gestita da ACK e può essere aggiornata tramite i manifesti di Kubernetes. È possibile migrare in modo incrementale, adottando le risorse necessarie e mantenendo gli strumenti IaC esistenti per altre risorse.

ACK supporta anche risorse di sola lettura. Per le risorse gestite da altri team o strumenti a cui desideri fare riferimento ma non modificare, combina l'adozione con la politica di `retain` eliminazione e concedi solo le autorizzazioni IAM di lettura. Ciò consente alle applicazioni di scoprire l'infrastruttura condivisa (ruoli IAMVPCs, chiavi KMS) tramite APIs Kubernetes senza rischiare modifiche.

Per ulteriori informazioni sull'adozione delle risorse, consulta. the section called “Concetti ACK”

Politiche di eliminazione

Le politiche di eliminazione controllano cosa succede alle AWS risorse quando elimini la risorsa Kubernetes corrispondente. Scegli la politica giusta in base al ciclo di vita delle risorse e ai tuoi requisiti operativi.

Elimina (impostazione predefinita)

La AWS risorsa viene eliminata quando elimini la risorsa Kubernetes. Ciò mantiene la coerenza tra il cluster e AWS garantisce che le risorse non si accumulino.

Quando usare delete:

- Ambienti di sviluppo e test in cui la pulizia è importante
- Risorse temporanee legate al ciclo di vita delle applicazioni (database di test, bucket temporanei)
- Risorse che non dovrebbero durare più a lungo dell'applicazione (code SQS, cluster) ElastiCache
- Ottimizzazione dei costi: ripulisci automaticamente le risorse inutilizzate
- Ambienti gestiti GitOps in cui la rimozione delle risorse da Git dovrebbe eliminare l'infrastruttura

La politica di eliminazione predefinita è in linea con il modello dichiarativo di Kubernetes: ciò che è nel cluster corrisponde a ciò che esiste in. AWS

Mantenimento

La AWS risorsa viene conservata quando elimini la risorsa Kubernetes. Ciò protegge i dati critici e consente alle risorse di sopravvivere più a lungo della loro rappresentazione in Kubernetes.

Quando usare Retain:

- Database di produzione con dati critici che devono sopravvivere alle modifiche dei cluster
- Bucket di storage a lungo termine con requisiti di conformità o di controllo
- Risorse condivise utilizzate da più applicazioni o team
- Risorse in fase di migrazione verso diversi strumenti di gestione
- Scenari di disaster recovery in cui si desidera preservare l'infrastruttura
- Risorse con dipendenze complesse che richiedono uno smantellamento accurato

```
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

Le risorse conservate continuano a comportare AWS costi e devono essere eliminate manualmente quando non sono più necessarie. AWS Utilizza l'etichettatura delle risorse per tenere traccia delle risorse conservate per la pulizia.

Per ulteriori informazioni sulle politiche di eliminazione, vedere. the section called “Concetti ACK”

Documentazione upstream

Per informazioni dettagliate sull'uso di ACK:

- Guida all'uso di ACK - Creazione e gestione delle risorse
- Riferimento all'API ACK: documentazione API completa per tutti i servizi
- Documentazione ACK - Documentazione completa per l'utente

Fasi successive

- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM e i modelli multiaccount
- the section called "Concetti ACK"- Comprendi i concetti di ACK e il ciclo di vita delle risorse
- the section called "Risoluzione dei problemi"- Risolvi i problemi ACK
- the section called "Lavorare con Argo CD"- Implementa le risorse ACK con GitOps
- the section called "concetti kro"- Componi le risorse ACK in astrazioni di livello superiore

Risolvi i problemi relativi alle funzionalità ACK

Questo argomento fornisce linee guida per la risoluzione dei problemi relativi a EKS Capability for ACK, compresi i controlli dello stato delle funzionalità, la verifica dello stato delle risorse e i problemi di autorizzazione IAM.

> **Note**
>
> Le funzionalità EKS sono completamente gestite e eseguite all'esterno del cluster. Non hai accesso ai log dei controller o ai namespace dei controller. La risoluzione dei problemi si concentra sullo stato delle funzionalità, sullo stato delle risorse e sulla configurazione IAM.

Le funzionalità sono ATTIVE ma le risorse non vengono create

Se la tua funzionalità ACK mostra `ACTIVE` lo stato ma le risorse non vengono create AWS, controlla lo stato della capacità, lo stato delle risorse e le autorizzazioni IAM.

Verifica lo stato delle funzionalità:

È possibile visualizzare i problemi relativi allo stato e allo stato delle funzionalità nella console EKS o utilizzando la AWS CLI.

Console:

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Osservabilità.
4. Scegli Monitora cluster.
5. Scegli la scheda Capacità per visualizzare lo stato e lo stato di tutte le funzionalità.

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
```

Cause comuni:

- Autorizzazioni IAM mancanti: al Capability Role mancano le autorizzazioni per il servizio AWS
- Namespace errato: risorse create nello spazio dei nomi senza un selettore appropriato IAMRole
- Specifiche della risorsa non valide: controlla le condizioni di stato delle risorse per verificare la presenza di errori di convalida
- Limitazione delle API: AWS vengono raggiunti i limiti di velocità delle API
- Webhook di ammissione: webhook di ammissione che impediscono al controller di correggere lo stato delle risorse

Controlla lo stato delle risorse:

```
# 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
```

Verifica le autorizzazioni 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
```

Risorse create in AWS Kubernetes ma non visualizzate

ACK tiene traccia solo delle risorse create tramite i manifesti di Kubernetes. Per gestire le AWS risorse esistenti con ACK, utilizza la funzionalità di adozione.

```
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
```

Per ulteriori informazioni sull'adozione delle risorse, consultathe section called “Concetti ACK”.

Risorse per più account non create

Se non vengono create risorse in un AWS account di destinazione quando utilizzi IAM Role Selectors, verifica la relazione di fiducia e la configurazione di IAMRole Selector.

Verifica la relazione di fiducia:

```
# Check the trust policy in the target account role
aws iam get-role --role-name cross-account-ack-role --query
 'Role.AssumeRolePolicyDocument'
```

La politica di fiducia deve consentire al ruolo di capacità dell'account di origine di assumerla.

Conferma la configurazione IAMRole del selettore:

```
# List IAMRoleSelectors (cluster-scoped)
kubectl get iamroleselector

# Describe specific selector
kubectl describe iamroleselector my-selector
```

Verifica l'allineamento dello spazio dei nomi:

IAMRoleI selettori sono risorse con ambito cluster ma hanno come target namespace specifici. Assicurati che le tue risorse ACK si trovino in uno spazio dei nomi che corrisponda al selettore dello spazio dei nomi di Selector: IAMRole

```
# 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}'
```

IAMRoleControlla la condizione selezionata:

Verifica che il IAMRole selettore sia stato abbinato correttamente alla tua risorsa controllando la `ACK.IAMRoleSelected` condizione:

```
# Check if IAMRoleSelector was matched
kubectl get bucket my-cross-account-bucket -n production -o
 jsonpath='{.status.conditions[?(@.type=="ACK.IAMRoleSelected")]}'
```

Se la condizione è presente `False` o manca, il IAMRole selettore dello spazio dei nomi del selettore non corrisponde allo spazio dei nomi della risorsa. Verifica che il selettore corrisponda alle etichette dello spazio dei nomi della risorsa. `namespaceSelector`

Verifica le autorizzazioni di Capability Role:

I requisiti `sts:AssumeRole` e le `sts:TagSession` autorizzazioni del ruolo Capability Role per il ruolo dell'account di destinazione:

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole", "sts:TagSession"],
      "Resource": "arn:aws:iam::[.replaceable]`444455556666`:role/[.replaceable]`cross-
account-ack-role`"
    }
  ]
}
```

Per una configurazione dettagliata tra account, consulta. the section called “Configurazione delle autorizzazioni ”

Fasi successive

- the section called “Considerazioni”- Considerazioni e best practice relative a ACK
- the section called “Configurazione delle autorizzazioni ”- Configura le autorizzazioni IAM e i modelli multiaccount
- the section called “Concetti ACK”- Comprendi i concetti di ACK e il ciclo di vita delle risorse
- the section called “Risoluzione dei problemi relativi alle funzionalità”- Guida generale alla risoluzione dei problemi relativi alle funzionalità

Confronto tra la funzionalità EKS per ACK e l'ACK autogestito

EKS Capability for ACK offre le stesse funzionalità dei controller ACK autogestiti, ma con vantaggi operativi significativi. Per un confronto generale tra EKS Capabilities e soluzioni autogestite, vedi. the section called “Considerazioni” Questo argomento si concentra sulle differenze specifiche di ACK.

Differenze rispetto all'ACK upstream

EKS Capability for ACK si basa sui controller ACK upstream ma si differenzia nell'integrazione IAM.

IAM Capability Role: la funzionalità utilizza un ruolo IAM dedicato con una policy di fiducia che consente il `capabilities.eks.amazonaws.com` service principal, non l'IRSA (IAM Roles for Service Accounts). Puoi collegare le policy IAM direttamente al Capability Role senza la necessità di creare o annotare gli account di servizio Kubernetes o configurare i provider OIDC. Una best practice per i casi d'uso in produzione consiste nel configurare le autorizzazioni di servizio utilizzando. `IAMRoleSelector` Per ulteriori dettagli, consulta the section called “Configurazione delle autorizzazioni ”.

Tag di sessione: la funzionalità gestita imposta automaticamente i tag di sessione su tutte le richieste AWS API, abilitando il controllo e il controllo granulari degli accessi. I tag includonoeks:eks-capability-arn, e. eks:kubernetes-namespace eks:kubernetes-api-group Ciò differisce dall'ACK autogestito, che non imposta questi tag per impostazione predefinita. the section called “Configurazione delle autorizzazioni ”Per ulteriori informazioni sull'utilizzo dei tag di sessione nelle policy IAM, consulta la sezione.

Tag delle risorse: la funzionalità applica tag predefiniti diversi alle AWS risorse rispetto a ACK autogestiti. La funzionalità utilizza tag con eks: prefisso (ad esempioeks:kubernetes-namespace,eks:eks-capability-arn) anziché i services.k8s.aws/ tag utilizzati dall'ACK autogestito. Consulta the section called “Considerazioni” l'elenco completo dei tag di risorsa predefiniti.

Compatibilità delle risorse: le risorse personalizzate ACK funzionano in modo identico a quelle di ACK upstream senza modifiche ai file YAML delle risorse ACK. La funzionalità utilizza lo stesso Kubernetes APIs e CRDs quindi gli strumenti funzionano allo stesso modo. kubectl Sono supportati tutti i controller e le risorse GA dell'UPstream ACK.

Per la documentazione completa di ACK e le guide specifiche per i servizi, consulta la documentazione ACK.

Percorso di migrazione

È possibile migrare da ACK autogestito alla funzionalità gestita senza tempi di inattività:

1. Aggiorna il controller ACK autogestito kube-system per utilizzarlo per i contratti di locazione elettorali per i leader, ad esempio:

   ```
   helm upgrade --install ack-s3-controller \
     oci://public.ecr.aws/aws-controllers-k8s/s3-chart \
     --namespace ack-system \
     --set leaderElection.namespace=kube-system
   ```

   In questo modo il contratto di locazione del controllore viene spostato in avantikube-system, permettendo alla capacità gestita di coordinarsi con esso.
2. Crea la funzionalità ACK sul tuo cluster (vedithe section called “Crea una funzionalità ACK”)
3. La funzionalità gestita riconosce le AWS risorse esistenti gestite da ACK e si occupa della riconciliazione
4. Ridimensiona o rimuovi gradualmente le implementazioni di controller autogestiti:

```
helm uninstall ack-s3-controller --namespace ack-system
```

Questo approccio consente a entrambi i controller di coesistere in modo sicuro durante la migrazione. La funzionalità gestita adotta automaticamente le risorse precedentemente gestite da controller autogestiti, garantendo una riconciliazione continua senza conflitti.

Fasi successive

- the section called "Crea una funzionalità ACK"- Creazione di una risorsa con funzionalità ACK
- the section called "Concetti ACK"- Comprendi i concetti e il ciclo di vita delle risorse ACK
- the section called "Configurazione delle autorizzazioni "- Configura IAM e autorizzazioni

Implementazione continua con Argo CD

Argo CD è uno strumento dichiarativo e di distribuzione GitOps continua per Kubernetes. Con Argo CD, puoi automatizzare l'implementazione e la gestione del ciclo di vita delle tue applicazioni su più cluster e ambienti. Argo CD supporta diversi tipi di sorgenti tra cui repository Git, registri Helm (HTTP e OCI) e immagini OCI, offrendo flessibilità alle organizzazioni con diversi requisiti di sicurezza e conformità.

Con EKS Capabilities, Argo CD è completamente gestito da AWS, eliminando la necessità di installare, mantenere e scalare i controller Argo CD e le loro dipendenze dai cluster.

Come funziona Argo CD

Argo CD segue GitOps lo schema, in cui la fonte dell'applicazione (repository Git, registro Helm o immagine OCI) è la fonte di verità per definire lo stato dell'applicazione desiderato. Quando si crea una `Application` risorsa Argo CD, si specifica l'origine contenente i manifesti dell'applicazione e il cluster e lo spazio dei nomi Kubernetes di destinazione. Argo CD monitora continuamente sia la sorgente che lo stato live nel cluster, sincronizzando automaticamente qualsiasi modifica per garantire che lo stato del cluster corrisponda allo stato desiderato.

> **Note**
>
> Con EKS Capability for Argo CD, il software Argo CD viene eseguito nel piano di AWS controllo, non sui nodi di lavoro. Ciò significa che i nodi di lavoro non hanno bisogno

dell'accesso diretto ai repository Git o ai registri Helm: la funzionalità gestisce l'accesso all'origine dall'account. AWS

Argo CD fornisce tre tipi di risorse principali:

- Applicazione: definisce una distribuzione da un repository Git a un cluster di destinazione
- ApplicationSet: genera più applicazioni da modelli per distribuzioni multi-cluster
- AppProject: Fornisce il raggruppamento logico e il controllo degli accessi per le applicazioni

Esempio: creazione di un'applicazione Argo CD

L'esempio seguente mostra come creare una risorsa Argo CD`Application`:

```
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
```

> **Note**
>
> Usalo `destination.name` con il nome del cluster che hai usato durante la registrazione del cluster (come `in-cluster` per il cluster locale). Il `destination.server` campo funziona anche con il cluster EKS ARNs, ma si consiglia di utilizzare i nomi dei cluster per una migliore leggibilità.

Vantaggi di Argo CD

Argo CD implementa un GitOps flusso di lavoro in cui si definiscono le configurazioni delle applicazioni nei repository Git e Argo CD sincronizza automaticamente le applicazioni in modo che corrispondano allo stato desiderato. Questo approccio incentrato su Git fornisce un audit trail completo di tutte le modifiche, consente facili rollback e si integra naturalmente con i processi di revisione e approvazione del codice esistenti. Argo CD rileva e riconcilia automaticamente la deriva tra lo stato desiderato in Git e lo stato effettivo nei cluster, assicurando che le distribuzioni rimangano coerenti con la configurazione dichiarata.

Con Argo CD, è possibile distribuire e gestire applicazioni su più cluster da una singola istanza Argo CD, semplificando le operazioni in ambienti multi-cluster e multiregione. L'interfaccia utente di Argo CD offre funzionalità di visualizzazione e monitoraggio, che consentono di visualizzare lo stato di implementazione, l'integrità e la cronologia delle applicazioni. L'interfaccia utente si integra con AWS Identity Center (precedentemente AWS SSO) per un'autenticazione e un'autorizzazione senza interruzioni, consentendoti di controllare l'accesso utilizzando l'infrastruttura di gestione delle identità esistente.

Come parte di EKS Managed Capabilities, Argo CD è completamente gestito da AWS, eliminando la necessità di installare, configurare e mantenere l'infrastruttura Argo CD. AWS gestisce la scalabilità, l'applicazione di patch e la gestione operativa, permettendo ai team di concentrarsi sulla distribuzione delle applicazioni piuttosto che sulla manutenzione degli strumenti.

Integrazione con Identity Center AWS

EKS Managed Capabilities fornisce l'integrazione diretta tra Argo CD e AWS Identity Center, consentendo l'autenticazione e l'autorizzazione senza interruzioni per gli utenti. Quando abiliti la funzionalità Argo CD, puoi configurare l'integrazione di AWS Identity Center per mappare i gruppi e gli utenti di Identity Center ai ruoli RBAC di Argo CD, consentendoti di controllare chi può accedere e gestire le applicazioni in Argo CD.

Integrazione con altre funzionalità gestite da EKS

Argo CD si integra con altre funzionalità gestite da EKS.

- AWS Controller for Kubernetes (ACK): utilizza Argo CD per gestire l'implementazione delle risorse ACK su più cluster, abilitando i flussi di lavoro per la tua infrastruttura. GitOps AWS
- kro (Kube Resource Orchestrator): usa Argo CD per distribuire composizioni kro su più cluster, abilitando una composizione coerente delle risorse in tutto il tuo ambiente Kubernetes.

Guida introduttiva ad Argo CD

Per iniziare a usare il CD EKS Capability for Argo:

1. Crea e configura un IAM Capability Role con le autorizzazioni necessarie per consentire ad Argo CD di accedere ai tuoi sorgenti e gestire le applicazioni.
2. Crea una risorsa con funzionalità Argo CD sul tuo cluster EKS tramite la AWS console, la AWS CLI o la tua infrastruttura preferita come strumento di codice.
3. Configura l'accesso al repository e registra i cluster per la distribuzione delle applicazioni.
4. Crea risorse applicative per distribuire le tue applicazioni dalle tue fonti dichiarative.

Crea una funzionalità Argo CD

Questo argomento spiega come creare una funzionalità Argo CD sul tuo cluster Amazon EKS.

Prerequisiti

Prima di creare una funzionalità Argo CD, assicurati di avere:

- Un cluster Amazon EKS esistente che esegue una versione di Kubernetes supportata (sono supportate tutte le versioni con supporto standard ed esteso)
- AWS Identity Center configurato: necessario per l'autenticazione Argo CD (gli utenti locali non sono supportati)
- Un ruolo di capacità IAM con autorizzazioni per Argo CD
- Autorizzazioni IAM sufficienti per creare risorse di funzionalità sui cluster EKS
- `kubectl`configurato per comunicare con il cluster
- (Opzionale) L'Argo CD CLI è stata installata per una più semplice gestione di cluster e repository
- (Per CLI/EksCtl) Lo strumento CLI appropriato installato e configurato

Per istruzioni sulla creazione dello IAM Capability Role, consulta. the section called "Capacità, ruolo IAM" Per la configurazione di Identity Center, consulta Guida introduttiva a AWS Identity Center.

> **Important**
>
> Il ruolo di capacità IAM fornito determina a quali AWS risorse Argo CD può accedere. Ciò include l'accesso al repository Git tramite CodeConnections e i segreti in Secrets Manager.

Per indicazioni sulla creazione di un ruolo appropriato con autorizzazioni con privilegi minimi, consulta e. the section called "Capacità, ruolo IAM" the section called "Considerazioni sulle funzionalità EKS"

Scegli il tuo strumento

È possibile creare una funzionalità Argo CD utilizzando la Console di gestione AWS AWS CLI o eksctl:

- the section called "Console"- Usa la console per un'esperienza guidata
- the section called " AWS CLI"- Usa la AWS CLI per lo scripting e l'automazione
- the section called "eksctl"- Usa eksctl per un'esperienza nativa di Kubernetes

Cosa succede quando si crea una funzionalità Argo CD

Quando create una funzionalità Argo CD:

1. EKS crea il servizio di funzionalità Argo CD nel piano di controllo AWS
2. Le definizioni di risorse personalizzate (CRDs) sono installate nel cluster
3. Viene creata automaticamente una voce di accesso per il tuo IAM Capability Role con policy di accesso specifiche per ciascuna funzionalità che concedono le autorizzazioni Kubernetes di base (vedi) the section called "Considerazioni sulle funzionalità EKS"
4. Argo CD inizia a cercare le sue risorse personalizzate (Applicazioni,,) ApplicationSets AppProjects
5. Lo stato della funzionalità cambia da a `CREATING` `ACTIVE`
6. L'interfaccia utente di Argo CD diventa accessibile tramite il relativo URL

Una volta attiva, potete creare applicazioni Argo CD nel cluster da distribuire dalle vostre fonti dichiarative.

> **Note**
>
> La voce di accesso creata automaticamente non concede le autorizzazioni per distribuire applicazioni nei cluster. Per distribuire le applicazioni, è necessario configurare autorizzazioni Kubernetes RBAC aggiuntive per ogni cluster di destinazione. Vedi the section called

"Registra i cluster" per i dettagli sulla registrazione dei cluster e sulla configurazione dell'accesso.

Fasi successive

Dopo aver creato la funzionalità Argo CD:

- the section called "Concetti di Argo CD"- Scopri GitOps i principi, le politiche di sincronizzazione e i modelli multicluster
- the section called "Lavorare con Argo CD"- Configura l'accesso al repository, registra i cluster di destinazione e crea applicazioni
- the section called "Considerazioni su Argo CD"- Esplora i modelli di architettura multicluster e la configurazione avanzata

Crea una funzionalità Argo CD utilizzando la console

Questo argomento descrive come creare una funzionalità Argo CD utilizzando. Console di gestione AWS

Prerequisiti

- AWS Identity Center configurato: Argo CD richiede AWS Identity Center per l'autenticazione. Gli utenti locali non sono supportati. Se non hai configurato AWS Identity Center, consulta Guida introduttiva a AWS Identity Center per creare un'istanza di Identity Center e Aggiungi utenti e Aggiungi gruppi per creare utenti e gruppi per l'accesso ad Argo CD.

Crea la funzionalità Argo CD

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Funzionalità.
4. Nella barra di navigazione a sinistra, scegli Argo CD.
5. Scegli la funzionalità Create Argo CD.
6. Per IAM Capability Role:
   - Se disponi già di un IAM Capability Role, selezionalo dal menu a discesa

- Se devi creare un ruolo, scegli Crea ruolo Argo CD

  Questo apre la console IAM in una nuova scheda con policy di fiducia precompilate e accesso completo in lettura a Secrets Manager. Per impostazione predefinita, non vengono aggiunte altre autorizzazioni, ma è possibile aggiungerle se necessario. Se prevedi di utilizzare CodeCommit repository o altri AWS servizi, aggiungi le autorizzazioni appropriate prima di creare il ruolo.

  Dopo aver creato il ruolo, torna alla console EKS e il ruolo verrà selezionato automaticamente.

  > **Note**
  >
  > Se prevedi di utilizzare le integrazioni opzionali con AWS Secrets Manager o AWS CodeConnections, dovrai aggiungere le autorizzazioni al ruolo. Per esempi di policy IAM e linee guida alla configurazione, consulta the section called " AWS Secrets Manager" e. the section called " AWS CodeConnections"

7. Configura AWS l'integrazione con Identity Center:
   a. Seleziona Abilita AWS l'integrazione con Identity Center.
   b. Scegli la tua istanza di Identity Center dal menu a discesa.
   c. Configura le mappature dei ruoli per RBAC assegnando utenti o gruppi ai ruoli di Argo CD (ADMIN, EDITOR o VIEWER)
8. Scegli Create (Crea).

Inizia il processo di creazione delle funzionalità.

Verifica che la funzionalità sia attiva

1. Nella scheda Funzionalità, visualizza lo stato delle funzionalità di Argo CD.
2. Attendi che lo stato cambi da `CREATING` a`ACTIVE`.
3. Una volta attiva, la funzionalità è pronta per l'uso.

Per informazioni sullo stato delle funzionalità e sulla risoluzione dei problemi, vederethe section called "Lavorare con le funzionalità".

Accedere all'interfaccia utente di Argo CD

Dopo che la funzionalità è attiva, puoi accedere all'interfaccia utente di Argo CD:

1. Nella pagina delle funzionalità di Argo CD, scegliete Open Argo CD UI.
2. L'interfaccia utente di Argo CD si apre in una nuova scheda del browser.
3. Ora puoi creare applicazioni e gestire le distribuzioni tramite l'interfaccia utente.

Fasi successive

- the section called "Lavorare con Argo CD"- Configura gli archivi, registra i cluster e crea applicazioni
- the section called "Considerazioni su Argo CD"- Architettura multicluster e configurazione avanzata
- the section called "Lavorare con le funzionalità"- Gestite le risorse relative alle funzionalità di Argo CD

Crea una funzionalità Argo CD utilizzando la CLI AWS

Questo argomento descrive come creare una funzionalità Argo CD utilizzando la AWS CLI.

Prerequisiti

- AWS CLI: versione `2.12.3` o successiva. Per verificare la tua versione, `aws --version` esegui. Per ulteriori informazioni, consulta la Guida per l'utente all'installazione nell'interfaccia a riga di AWS comando.
- **`kubectl`**— Uno strumento da riga di comando per lavorare con i cluster Kubernetes. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".
- AWS Identity Center configurato: Argo CD richiede AWS Identity Center per l'autenticazione. Gli utenti locali non sono supportati. Se non hai configurato AWS Identity Center, consulta Guida introduttiva a AWS Identity Center per creare un'istanza di Identity Center e Aggiungi utenti e Aggiungi gruppi per creare utenti e gruppi per l'accesso ad Argo CD.

Fase 1: Creare un ruolo di capacità IAM

Crea un file di policy di fiducia:

```
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
```

Crea il ruolo IAM:

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

> **Note**
>
> Se prevedi di utilizzare le integrazioni opzionali con AWS Secrets Manager o AWS CodeConnections, dovrai aggiungere le autorizzazioni al ruolo. Per esempi di policy IAM e indicazioni sulla configurazione, consulta the section called " AWS Secrets Manager" e. the section called " AWS CodeConnections"

Fase 2: Creare la funzionalità Argo CD

Crea la risorsa di funzionalità Argo CD sul tuo cluster.

Innanzitutto, imposta le variabili di ambiente per la configurazione del tuo Identity Center:

```
# 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"
```

Crea la funzionalità con l'integrazione di Identity Center. Sostituisci *region-code* con la AWS regione in cui si trova il cluster, *my-cluster* con il nome del cluster e *idc-region-code* con il codice regionale in cui è stato configurato lo IAM Identity Center:

```
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]`idc-region-code`'"
      },
      "rbacRoleMappings": [{
        "role": "ADMIN",
        "identities": [{
          "id": "'$IDC_USER_ID'",
          "type": "SSO_USER"
        }]
      }]
    }
  }'
```

Il comando viene restituito immediatamente, ma la funzionalità impiega del tempo per diventare attiva poiché EKS crea l'infrastruttura e i componenti di funzionalità richiesti. EKS installerà le Kubernetes Custom Resource Definitions relative a questa funzionalità nel cluster non appena viene creata.

> **Note**
>
> Se ricevi un errore che indica che il cluster non esiste o non disponi delle autorizzazioni, verifica:
>
> - Il nome del cluster è corretto
> - La tua AWS CLI è configurata per la regione corretta
> - Disponi delle autorizzazioni IAM richieste

Fase 3: Verifica che la funzionalità sia attiva

Attendi che la funzionalità diventi attiva. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster e *my-cluster* con il nome del cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --query 'capability.status' \
  --output text
```

La funzionalità è pronta quando viene visualizzato lo statoACTIVE. Non continuare con il passaggio successivo fino a quando lo stato non sarà raggiuntoACTIVE.

Puoi anche visualizzare i dettagli completi delle funzionalità:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd
```

Fase 4: Verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verificate che le risorse personalizzate di Argo CD siano disponibili nel cluster:

```
kubectl api-resources | grep argoproj.io
```

Dovresti vedere i tipi `Application` di `ApplicationSet` risorse elencati.

Fasi successive

- the section called "Lavorare con Argo CD"- Configura i repository, registra i cluster e crea applicazioni
- the section called "Considerazioni su Argo CD"- Architettura multicluster e configurazione avanzata
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa di funzionalità Argo CD

Crea una funzionalità Argo CD usando eksctl

Questo argomento descrive come creare una funzionalità Argo CD usando eksctl.

> **Note**
>
> I seguenti passaggi richiedono la versione eksctl o successiva. `0.220.0` Per verificare la tua versione, esegui. `eksctl version`

Fase 1: Creare un ruolo di capacità IAM

Crea un file di policy di fiducia:

```
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
```

Crea il ruolo IAM:

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

> **Note**
>
> Per questa configurazione di base, non sono necessarie politiche IAM aggiuntive. Se prevedi di utilizzare Secrets Manager per le credenziali del repository oppure CodeConnections, dovrai aggiungere le autorizzazioni al ruolo. Per esempi di policy IAM e linee guida alla configurazione, consulta e. the section called " AWS Secrets Manager" the section called " AWS CodeConnections"

Fase 2: Ottieni la configurazione del tuo AWS Identity Center

Ottieni l'ARN e l'ID utente dell'istanza di Identity Center per la configurazione RBAC:

```
# Get your Identity Center instance ARN
aws sso-admin list-instances --query 'Instances[0].InstanceArn' --output text

# 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
```

Prendi nota di questi valori: ti serviranno nel passaggio successivo.

Fase 3: Creare un file di configurazione eksctl

Crea un archivio denominato `argocd-capability.yaml` con i seguenti contenuti. Sostituisci i valori segnaposto con il nome del cluster, la regione del cluster, l'ARN del ruolo IAM, l'ARN dell'istanza di Identity Center, la regione dell'Identity Center e l'ID utente:

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

metadata:
  name: my-cluster
  region: cluster-region-code
```

```
capabilities:
  - name: my-argocd
    type: ARGOCD
    roleArn: arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole
    deletePropagationPolicy: RETAIN
    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

È possibile aggiungere più utenti o gruppi alle mappature RBAC. Per i gruppi, usa `type: SSO_GROUP` e fornisci l'ID del gruppo. I ruoli disponibili sono `ADMIN`, `EDITOR`, e `VIEWER`.

Fase 4: Creare la funzionalità Argo CD

Applica il file di configurazione:

```
eksctl create capability -f argocd-capability.yaml
```

Il comando viene restituito immediatamente, ma la funzionalità impiega del tempo per diventare attiva.

Fase 5: Verificare che la funzionalità sia attiva

Verifica lo stato della capacità. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster e sostituiscilo *`my-cluster`* con il nome del cluster.

```
eksctl get capability \
  --region region-code \
  --cluster my-cluster \
  --name my-argocd
```

La funzionalità è pronta quando viene visualizzato lo statoACTIVE.

Fase 6: Verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verificate che le risorse personalizzate di Argo CD siano disponibili nel cluster:

```
kubectl api-resources | grep argoproj.io
```

Dovresti vedere i tipi Application di ApplicationSet risorse elencati.

Fasi successive

- the section called "Lavorare con Argo CD"- Scopri come creare e gestire applicazioni Argo CD
- the section called "Considerazioni su Argo CD"- Configura SSO e accesso multicluster
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa di funzionalità Argo CD

Concetti di Argo CD

Argo CD implementa GitOps trattando Git come l'unica fonte di verità per le implementazioni delle applicazioni. Questo argomento illustra un esempio pratico, quindi spiega i concetti fondamentali da comprendere quando si lavora con il CD EKS Capability for Argo.

Guida introduttiva ad Argo CD

Dopo aver creato la funzionalità Argo CD (vedithe section called "Crea una funzionalità Argo CD"), puoi iniziare a distribuire le applicazioni. Questo esempio illustra la registrazione di un cluster e la creazione di un'applicazione.

Fase 1: configurazione

Registra il tuo cluster (richiesto)

Registra il cluster in cui desideri distribuire le applicazioni. Per questo esempio, registreremo lo stesso cluster su cui è in esecuzione Argo CD (potete usare il nome in-cluster per motivi di compatibilità con la maggior parte degli esempi di 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**
>
> Per informazioni sulla configurazione dell'Argo CD CLI per funzionare con la funzionalità Argo CD in EKS, vedere. the section called "Utilizzo dell'Argo CD CLI con funzionalità gestite"

In alternativa, registra il cluster utilizzando un Kubernetes Secret (vedi per i dettagli). the section called "Registra i cluster"

Configura l'accesso al repository (opzionale)

Questo esempio utilizza un GitHub archivio pubblico, quindi non è richiesta alcuna configurazione del repository. Per gli archivi privati, configura l'accesso utilizzando AWS Secrets Manager o Kubernetes Secrets (vedi the section called "Configura i repository" per i dettagli). CodeConnections

Per AWS i servizi (grafici ECR for Helm e CodeCommit) CodeConnections, puoi farvi riferimento direttamente nelle risorse dell'applicazione senza creare un repository. Il Capability Role deve disporre delle autorizzazioni IAM richieste. Per informazioni dettagliate, vedi the section called "Configura i repository".

Fase 2: crea un'applicazione

Crea questo manifesto dell'applicazione in`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
```

Applica l'applicazione:

```
kubectl apply -f my-app.yaml
```

Dopo aver applicato questa applicazione, Argo CD: 1. Sincronizza l'applicazione da Git al tuo cluster (distribuzione iniziale) 2. Monitora il repository Git per le modifiche 3. Sincronizza automaticamente le modifiche successive al cluster 4. Rileva e corregge qualsiasi deriva dallo stato desiderato 5. Fornisce lo stato di salute e la cronologia di sincronizzazione nell'interfaccia utente

Visualizza lo stato dell'applicazione:

```
kubectl get application guestbook -n argocd
```

È inoltre possibile visualizzare l'applicazione utilizzando la CLI di Argo CD o l'interfaccia utente di Argo CD (accessibile dalla console EKS nella scheda Capacità del cluster).

> **Note**
>
> Quando si utilizza l'Argo CD CLI con la funzionalità gestita, specificare le applicazioni con il prefisso dello spazio dei nomi:. `argocd app get argocd/guestbook`

> **Note**
>
> Usa il nome del cluster in `destination.name` (il nome che hai usato durante la registrazione del cluster). La funzionalità gestita non supporta l'impostazione predefinita locale all'interno del cluster ()`kubernetes.default.svc`.

Concetti principali

GitOps principi e tipi di fonti

Argo CD implementa GitOps, in cui la fonte dell'applicazione è l'unica fonte attendibile per le implementazioni:

- Dichiarativo: lo stato desiderato viene dichiarato utilizzando i manifesti YAML, i grafici Helm o gli overlay Kustomize
- Versionato: ogni modifica viene tracciata con un audit trail completo
- Automatizzato: Argo CD monitora continuamente le sorgenti e sincronizza automaticamente le modifiche
- Riparazione automatica: rileva e corregge lo scostamento tra lo stato desiderato e quello effettivo del cluster

Tipi di sorgenti supportati:

- Archivi Git - GitHub GitLab, Bitbucket, CodeCommit (HTTPS, SSH o) CodeConnections
- Registri Helm: registri HTTP (simili) e registri OCI (simili) `https://aws.github.io/eks-charts` `public.ecr.aws`
- Immagini OCI: immagini dei contenitori contenenti manifesti o grafici Helm (come) `oci://registry-1.docker.io/user/my-app`

Questa flessibilità consente alle organizzazioni di scegliere fonti che soddisfino i propri requisiti di sicurezza e conformità. Ad esempio, le organizzazioni che limitano l'accesso a Git dai cluster possono utilizzare ECR per i grafici Helm o le immagini OCI.

Per ulteriori informazioni, consultate Application Sources nella documentazione del CD Argo.

Sincronizzazione e riconciliazione

Argo CD monitora continuamente le sorgenti e i cluster per rilevare e correggere le differenze:

1. Esamina le fonti delle modifiche (impostazione predefinita: ogni 6 minuti)
2. Confronta lo stato desiderato con lo stato del cluster
3. Contrassegna le applicazioni come o `Synced` `OutOfSync`

4. Sincronizza automaticamente le modifiche (se configurate) o attende l'approvazione manuale
5. Monitora lo stato delle risorse dopo la sincronizzazione

Le onde di sincronizzazione controllano l'ordine di creazione delle risorse utilizzando le annotazioni:

```
metadata:
  annotations:
    argocd.argoproj.io/sync-wave: "0"  # Default if not specified
```

Le risorse vengono applicate in ordine ondulatorio (partendo dai numeri più bassi, inclusi i numeri negativi come `-1`). Wave `0` è l'impostazione predefinita se non specificata. Ciò consente di creare dipendenze come i namespace (wave) prima delle distribuzioni (wave`-1`) prima dei servizi (wave`0`). 1

La riparazione automatica annulla automaticamente le modifiche manuali:

```
spec:
  syncPolicy:
    automated:
      selfHeal: true
```

> **Note**
>
> La funzionalità gestita utilizza il tracciamento delle risorse basato sulle annotazioni (non basato su etichette) per una migliore compatibilità con le convenzioni Kubernetes e altri strumenti.

Per informazioni dettagliate sulle fasi di sincronizzazione, gli hook e i pattern avanzati, consulta la documentazione di sincronizzazione con Argo CD.

Integrità dell'applicazione

Argo CD monitora lo stato di tutte le risorse dell'applicazione:

Stati di salute : * Integro: tutte le risorse funzionano come previsto * In corso di avanzamento - Risorse in fase di creazione o aggiornamento * Degradate - Alcune risorse non integre (pod bloccati, processi non riusciti) * Sospesa - Applicazione intenzionalmente sospesa * Mancante - Risorse definite in Git non presenti nel cluster

Argo CD include controlli di integrità integrati per le risorse Kubernetes più comuni (implementazioni, lavori, ecc.) e supporta controlli di integrità personalizzati per. StatefulSets CRDs

Lo stato dell'applicazione è determinato da tutte le sue risorse: se una risorsa lo è, l'applicazione sì. `Degraded` `Degraded`

Per ulteriori informazioni, consultate Resource Health nella documentazione del CD Argo.

Schemi multicluster

Argo CD supporta due modelli di implementazione principali:

H ub-and-spoke - Esegui Argo CD su un cluster di gestione dedicato che viene distribuito su più cluster di carichi di lavoro: * Controllo e visibilità centralizzati * Policy coerenti su tutti i cluster * Un'istanza Argo CD da gestire * Chiara separazione tra piano di controllo e carichi di lavoro

Per cluster: esegui Argo CD su ogni cluster, gestendo solo le applicazioni del cluster: * Separazione del cluster (un errore non influisce sugli altri) * Rete più semplice (nessuna comunicazione tra cluster) * Configurazione iniziale più semplice (nessuna registrazione del cluster)

Scegliete tra hub-and-spoke i team di piattaforma che gestiscono più cluster o per i team indipendenti per i cluster o quando i cluster devono essere completamente isolati.

Per una configurazione multi-cluster dettagliata, consulta. the section called "Considerazioni su Argo CD"

Progetti

I progetti forniscono il raggruppamento logico e il controllo degli accessi per le applicazioni:

- Restrizioni all'origine: limita i repository Git che possono essere utilizzati
- Restrizioni sulla destinazione: limita i cluster e i namespace a cui è possibile rivolgersi
- Restrizioni alle risorse: limita i tipi di risorse Kubernetes che possono essere distribuiti
- Integrazione RBAC: mappa i progetti all'utente e al gruppo di Identity Center AWS IDs

Le applicazioni appartengono a un singolo progetto. Se non specificato, usano il `default` progetto, che di default non ha restrizioni. Per uso in produzione, modificate il `default` progetto per limitare l'accesso e creare nuovi progetti con le restrizioni appropriate.

Per la configurazione del progetto e i modelli RBAC, vedere. the section called "Configurazione delle autorizzazioni "

Opzioni di sincronizzazione

Ottimizza il comportamento di sincronizzazione con opzioni comuni:

- `CreateNamespace=true`- Crea automaticamente lo spazio dei nomi di destinazione
- `ServerSideApply=true`- Utilizza server-side apply per una migliore risoluzione dei conflitti
- `SkipDryRunOnMissingResource=true`- Salta il dry run quando CRDs non esistono ancora (utile per le istanze kro)

```
spec:
  syncPolicy:
    syncOptions:
    - CreateNamespace=true
    - ServerSideApply=true
    - SkipDryRunOnMissingResource=true
```

Per un elenco completo delle opzioni di sincronizzazione, consultate la documentazione sulle opzioni di sincronizzazione di Argo CD.

Fasi successive

- the section called "Configura i repository"- Configura l'accesso al repository Git
- the section called "Registra i cluster"- Registra i cluster di destinazione per la distribuzione
- the section called "Crea applicazioni"- Crea la tua prima applicazione
- the section called "Considerazioni su Argo CD"- Modelli specifici di EKS, integrazione con Identity Center e configurazione multicluster
- Documentazione su Argo CD - Documentazione completa su Argo CD, inclusi hook di sincronizzazione, controlli di integrità e modelli avanzati

Configurare le autorizzazioni di Argo CD

La funzionalità gestita da Argo CD si integra con AWS Identity Center per l'autenticazione e utilizza ruoli RBAC integrati per l'autorizzazione. Questo argomento spiega come configurare le autorizzazioni per utenti e team.

Come funzionano le autorizzazioni con Argo CD

La funzionalità Argo CD utilizza AWS Identity Center per l'autenticazione e fornisce tre ruoli RBAC integrati per l'autorizzazione.

Quando un utente accede ad Argo CD:

1. Si autenticano tramite AWS Identity Center (che può essere collegato al vostro provider di identità aziendale)
2. AWS Identity Center fornisce informazioni su utenti e gruppi su Argo CD
3. Argo CD associa utenti e gruppi ai ruoli RBAC in base alla configurazione
4. Gli utenti vedono solo le applicazioni e le risorse a cui hanno il permesso di accedere

Ruoli RBAC integrati

La funzionalità Argo CD offre tre ruoli integrati che è possibile associare agli utenti e ai gruppi di AWS Identity Center. Si tratta di ruoli con ambito globale che controllano l'accesso alle risorse di Argo CD come progetti, cluster e repository.

> **Important**
>
> I ruoli globali controllano l'accesso ad Argo CD stesso, non a risorse relative al progetto come le Applicazioni. Gli utenti di EDITOR e VIEWER non possono visualizzare o gestire le applicazioni per impostazione predefinita: hanno bisogno di ruoli di progetto per accedere alle risorse relative al progetto. the section called “Ruoli del progetto e accesso all'ambito del progetto”Per i dettagli sulla concessione dell'accesso alle applicazioni e ad altre risorse relative al progetto, consulta la sezione.

AMMINISTRATORE

Accesso completo a tutte le risorse e le impostazioni di Argo CD:

- Crea, aggiorna ed elimina applicazioni e ApplicationSets in qualsiasi progetto
- Gestisci la configurazione di Argo CD
- Registra e gestisci i cluster di destinazione della distribuzione
- Configura l'accesso al repository

- Crea e gestisci progetti
- Visualizza lo stato e la cronologia di tutte le candidature
- Elenca e accedi a tutti i cluster e gli archivi

EDITOR

Può aggiornare i progetti e configurare i ruoli del progetto, ma non può modificare le impostazioni globali di Argo CD:

- Aggiorna progetti esistenti (non può creare o eliminare progetti)
- Configura i ruoli e le autorizzazioni del progetto
- Visualizza chiavi e certificati GPG
- Impossibile modificare la configurazione globale di Argo CD
- Non è possibile gestire direttamente cluster o repository
- Non è possibile visualizzare o gestire le applicazioni senza ruoli di progetto

VISUALIZZATORE

Accesso in sola lettura alle risorse del CD Argo:

- Visualizza le configurazioni del progetto
- Elenca tutti i progetti (inclusi i progetti a cui l'utente non è assegnato)
- Visualizza chiavi e certificati GPG
- Impossibile elencare cluster o repository
- Impossibile apportare modifiche
- Non è possibile visualizzare o gestire le applicazioni senza ruoli di progetto

> **Note**
>
> Per concedere agli utenti EDITOR o VIEWER l'accesso alle applicazioni, un ADMIN o EDITOR deve creare ruoli di progetto che associano i gruppi di Identity Center a autorizzazioni specifiche all'interno di un progetto.

Ruoli del progetto e accesso all'ambito del progetto

I ruoli globali (ADMIN, EDITOR, VIEWER) controllano l'accesso allo stesso Argo CD. I ruoli del progetto controllano l'accesso alle risorse e alle funzionalità all'interno di un progetto specifico, tra cui:

- Risorse: applicazioni ApplicationSets, credenziali del repository, credenziali del cluster
- Funzionalità: accesso ai log, accesso esecutivo ai pod delle applicazioni

Comprensione del modello di autorizzazione a due livelli:

- Ambito globale: i ruoli predefiniti determinano cosa possono fare gli utenti con progetti, cluster, repository e impostazioni del CD Argo
- Ambito del progetto: i ruoli del progetto determinano ciò che gli utenti possono fare con le risorse e le funzionalità all'interno di un progetto specifico

Ciò significa che:

- Gli utenti ADMIN possono accedere a tutte le risorse e le funzionalità del progetto senza configurazioni aggiuntive
- Agli utenti EDITOR e VIEWER devono essere concessi ruoli di progetto per accedere alle risorse e alle funzionalità del progetto
- Gli utenti di EDITOR possono creare ruoli di progetto per concedere a se stessi e ad altri l'accesso all'interno di progetti che possono aggiornare

Esempio di flusso di lavoro:

1. Un ADMIN associa un gruppo Identity Center al ruolo EDITOR a livello globale
2. Un AMMINISTRATORE crea un progetto per un team
3. L'EDITOR configura i ruoli del progetto all'interno di quel progetto per concedere ai membri del team l'accesso alle risorse relative al progetto
4. I membri del team (che possono avere il ruolo globale VIEWER) possono ora visualizzare e gestire le applicazioni in quel progetto in base alle autorizzazioni relative al ruolo di progetto

Per i dettagli sulla configurazione dei ruoli del progetto, consulta. the section called “Controllo degli accessi basato su progetti”

Configurare le mappature dei ruoli

Associa gli utenti e i gruppi di AWS Identity Center ai ruoli di Argo CD durante la creazione o l'aggiornamento della funzionalità.

Esempio di mappatura dei ruoli:

```
{
  "rbacRoleMapping": {
    "ADMIN": ["AdminGroup", "alice@example.com"],
    "EDITOR": ["DeveloperGroup", "DevOpsTeam"],
    "VIEWER": ["ReadOnlyGroup", "bob@example.com"]
  }
}
```

> **Note**
>
> I nomi dei ruoli fanno distinzione tra maiuscole e minuscole e devono essere maiuscole (ADMIN, EDITOR, VIEWER).

> **Important**
>
> L'integrazione di EKS Capabilities con AWS Identity Center supporta fino a 1.000 identità per funzionalità Argo CD. Un'identità può essere un utente o un gruppo.

Aggiorna le mappature dei ruoli:

```
aws eks 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" }
            ]
          }
        ]
      }
    }
  }'
```

Utilizzo dell'account amministratore

L'account amministratore è progettato per la configurazione iniziale e le attività amministrative come la registrazione di cluster e la configurazione dei repository.

Quando l'account amministratore è appropriato:

- Configurazione e configurazione delle funzionalità iniziali
- Sviluppo individuale o dimostrazioni rapide
- Attività amministrative (registrazione del cluster, configurazione del repository, creazione di progetti)

Procedure consigliate per l'account amministratore:

- Non affidare i token dell'account al controllo delle versioni
- Ruota immediatamente i token se esposti
- Limita l'utilizzo dei token dell'account alle attività di configurazione e amministrative
- Imposta tempi di scadenza brevi (massimo 12 ore)
- È possibile creare solo 5 token di account alla volta

Quando utilizzare invece l'accesso basato su progetti:

- Ambienti di sviluppo condivisi con più utenti
- Qualsiasi ambiente che assomigli alla produzione
- Quando hai bisogno di audit trail che indichino chi ha eseguito le azioni
- Quando è necessario imporre restrizioni sulle risorse o limiti di accesso

Per ambienti di produzione e scenari multiutente, utilizza il controllo degli accessi basato su progetti con ruoli RBAC dedicati mappati ai gruppi di Identity Center. AWS

Controllo degli accessi basato su progetti

Usa Argo CD Projects (AppProject) per fornire un controllo granulare degli accessi e l'isolamento delle risorse per i team.

> **Important**
>
> Prima di assegnare utenti o gruppi a ruoli specifici del progetto, è necessario mapparli a un ruolo globale di Argo CD (ADMIN, EDITOR o VIEWER) nella configurazione delle funzionalità. Gli utenti non possono accedere ad Argo CD senza una mappatura globale dei ruoli, anche se sono assegnati a ruoli di progetto.
> Prendi in considerazione la possibilità di associare gli utenti al ruolo VIEWER a livello globale, quindi concedi autorizzazioni aggiuntive tramite ruoli specifici del progetto. Ciò fornisce un accesso di base e consente al contempo un controllo granulare a livello di progetto.

I progetti forniscono:

- Restrizioni all'origine: limita i repository Git che possono essere utilizzati
- Restrizioni sulla destinazione: limita i cluster e i namespace a cui è possibile rivolgersi
- Restrizioni sulle risorse: limita i tipi di risorse Kubernetes che possono essere distribuiti
- Integrazione RBAC: mappa i progetti ai gruppi di AWS Identity Center o ai ruoli di Argo CD

Esempio di progetto per l'isolamento del team:

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
  namespace: argocd
spec:
  description: Team A applications

  # Required: Specify which namespaces this project watches for Applications
  sourceNamespaces:
  - argocd
```

```
  # 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
```

Namespace di origine

Quando si utilizza la funzionalità EKS Argo CD, il `spec.sourceNamespaces` campo è obbligatorio nelle definizioni. AppProject Questo campo specifica quale spazio dei nomi può contenere applicazioni o ApplicationSets che fanno riferimento a questo progetto.

> **Important**
>
> La funzionalità EKS Argo CD supporta solo un singolo spazio dei nomi per le applicazioni e, in genere, lo spazio ApplicationSets dei nomi specificato durante la creazione della funzionalità. `argocd` Ciò differisce dal CD open source Argo che supporta più namespace.

AppProject configurazione

Tutti AppProjects devono includere lo spazio dei nomi configurato della funzionalità in: `sourceNamespaces`

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
```

```
metadata:
  name: team-a-project
  namespace: argocd
spec:
  description: Applications for Team A

  # Required: Specify the capability's configured namespace
 (configuration.argoCd.namespace)
  sourceNamespaces:
    - argocd  # Must match your capability's namespace configuration

  # Source repositories this project can deploy from
  sourceRepos:
    - 'https://github.com/my-org/team-a-*'

  # Destination restrictions
  destinations:
    - namespace: 'team-a-*'
      server: arn:aws:eks:us-west-2:111122223333:cluster/my-cluster
```

> **Note**
>
> Se ometti lo spazio dei nomi della funzionalità da`sourceNamespaces`, le applicazioni o ApplicationSets in tale spazio dei nomi non possono fare riferimento a questo progetto, con conseguenti errori di distribuzione.

Assegna utenti ai progetti:

I ruoli di progetto garantiscono agli utenti EDITOR e VIEWER l'accesso alle risorse del progetto (applicazioni ApplicationSets, repository e credenziali del cluster) e alle funzionalità (log, exec). Senza ruoli di progetto, questi utenti non possono accedere a queste risorse anche se dispongono di un accesso globale ai ruoli.

Gli utenti ADMIN hanno accesso a tutte le applicazioni senza bisogno di ruoli di progetto.

Esempio: concessione dell'accesso all'applicazione ai membri del team

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
```

```
  namespace: argocd
spec:
  # ... project configuration ...

  sourceNamespaces:
  - argocd

  # Project roles grant Application-level access
  roles:
  - name: developer
    description: Team A developers - can manage Applications
    policies:
    - p, proj:team-a:developer, applications, *, team-a/*, allow
    - p, proj:team-a:developer, clusters, get, *, allow  # See cluster names in UI
    groups:
    - 686103e0-f051-7068-b225-e6392b959d9e  # Identity Center group ID

  - name: viewer
    description: Team A viewers - read-only Application access
    policies:
    - p, proj:team-a:viewer, applications, get, team-a/*, allow
    - p, proj:team-a:viewer, clusters, get, *, allow  # See cluster names in UI
    groups:
    - 786203e0-f051-7068-b225-e6392b959d9f  # Identity Center group ID
```

> **Note**
>
> Includi ruoli `clusters, get, *, allow` nel progetto per consentire agli utenti di visualizzare i nomi dei cluster nell'interfaccia utente. Senza questa autorizzazione, il cluster di destinazione viene visualizzato come «sconosciuto».

Comprensione delle politiche relative ai ruoli del progetto:

Il formato della politica è: `p, proj:<project>:<role>, <resource>, <action>, <object>, <allow/deny>`

Politiche relative alle risorse:

- `applications, , team-a/, allow`- Accesso completo a tutte le applicazioni del progetto team-a
- `applications, get, team-a/*, allow`- Accesso in sola lettura alle applicazioni

- `applications, sync, team-a/*, allow`- Può sincronizzare le applicazioni ma non creare/eliminare
- `applications, delete, team-a/*, allow`- Può eliminare le applicazioni (usare con cautela)
- `applicationsets, , team-a/, allow`- Accesso completo a ApplicationSets
- `repositories, *, *, allow`- Accesso alle credenziali del repository
- `clusters, *, *, allow`- Accesso alle credenziali del cluster

Politiche di capacità:

- `logs, , team-a/, allow`- Accesso ai registri delle applicazioni
- `exec, , team-a/, allow`- Accesso esecutivo ai pod delle applicazioni

> **Note**
>
> Gli utenti di EDITOR possono creare ruoli di progetto per concedere a se stessi e ad altri le autorizzazioni all'interno dei progetti che possono aggiornare. Ciò consente ai responsabili del team di controllare l'accesso alle risorse relative al progetto per il proprio team senza richiedere l'intervento dell'AMMINISTRATORE.

> **Note**
>
> Utilizza il gruppo Identity Center IDs (non i nomi dei gruppi) nel campo. `groups` È inoltre possibile utilizzare l'utente Identity Center IDs per l'accesso dei singoli utenti. Trovali IDs nella console di AWS Identity Center o utilizzando la AWS CLI.

Schemi di autorizzazione comuni

Modello 1: team di amministrazione con accesso completo

```
{
  "rbacRoleMapping": {
    "ADMIN": ["PlatformTeam", "SRETeam"]
  }
```

```
}
```

Gli utenti ADMIN possono visualizzare e gestire tutte le risorse relative al progetto senza configurazioni aggiuntive.

Schema 2: i responsabili del team gestiscono i progetti, gli sviluppatori accedono tramite i ruoli di progetto

```
{
  "rbacRoleMapping": {
    "ADMIN": ["PlatformTeam"],
    "EDITOR": ["TeamLeads"],
    "VIEWER": ["AllDevelopers"]
  }
}
```

1. ADMIN crea progetti per ogni team
2. I team leader (EDITOR) configurano i ruoli del progetto per garantire ai propri sviluppatori l'accesso alle risorse del progetto (applicazioni ApplicationSets, credenziali) e alle funzionalità (log, exec)
3. Gli sviluppatori (VIEWER) possono accedere solo alle risorse e alle funzionalità consentite dai rispettivi ruoli di progetto

Modello 3: accesso basato sul team con ruoli di progetto

1. ADMIN crea progetti e mappa i leader del team al ruolo di EDITOR a livello globale
2. I team leader (EDITOR) assegnano ai membri del team i ruoli di progetto all'interno dei loro progetti
3. Ai membri del team è richiesto solo il ruolo globale di VIEWER: i ruoli di progetto forniscono l'accesso alle risorse e alle funzionalità del progetto

```
{
  "rbacRoleMapping": {
    "ADMIN": ["PlatformTeam"],
    "EDITOR": ["TeamLeads"],
    "VIEWER": ["AllDevelopers"]
  }
}
```

Best practice

Utilizza i gruppi anziché i singoli utenti: associa i gruppi di AWS Identity Center ai ruoli di Argo CD anziché ai singoli utenti per una gestione più semplice.

Inizia con il privilegio minimo: inizia con l'accesso VIEWER e concedi EDITOR o ADMIN secondo necessità.

Usa i progetti per l'isolamento del team: crea ambienti separati AppProjects per team o ambienti diversi per far rispettare i limiti.

Sfrutta la federazione di Identity Center: configura AWS Identity Center per la federazione con il tuo provider di identità aziendale per una gestione centralizzata degli utenti.

Revisioni periodiche degli accessi: rivedi periodicamente le mappature dei ruoli e le assegnazioni dei progetti per garantire livelli di accesso appropriati.

Limita l'accesso al cluster: ricorda che Argo CD RBAC controlla l'accesso alle risorse e alle operazioni di Argo CD, ma non corrisponde a Kubernetes RBAC. Gli utenti con accesso ad Argo CD possono distribuire applicazioni nei cluster a cui Argo CD ha accesso. Limita i cluster a cui Argo CD può accedere e utilizza le restrizioni sulla destinazione del progetto per controllare dove possono essere distribuite le applicazioni.

AWS autorizzazioni di servizio

Per utilizzare AWS i servizi direttamente nelle risorse dell'applicazione (senza creare risorse di repository), collega le autorizzazioni IAM richieste al Capability Role.

Grafici ECR for Helm:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
```

```
      "Resource": "*"
    }
  ]
}
```

CodeCommit archivi:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "arn:aws:codecommit:region:account-id:repository-name"
    }
  ]
}
```

CodeConnections (GitHub GitLab, Bitbucket):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection"
      ],
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-id"
    }
  ]
}
```

the section called "Configura i repository"Per i dettagli sull'utilizzo di queste integrazioni, consulta la pagina.

Fasi successive

- the section called "Lavorare con Argo CD"- Scopri come creare applicazioni e gestire le distribuzioni

- the section called "Concetti di Argo CD"- Comprendi i concetti di Argo CD, inclusi i progetti
- the section called "Considerazioni sulle funzionalità EKS"- Esamina le migliori pratiche di sicurezza per quanto riguarda le funzionalità

Lavorare con Argo CD

Con Argo CD, definisci le applicazioni nei repository Git e Argo CD le sincronizza automaticamente con i tuoi cluster Kubernetes. Ciò consente l'implementazione di applicazioni dichiarative e controllate dalla versione con rilevamento automatico delle deviazioni.

Prerequisiti

Prima di lavorare con Argo CD, è necessario:

- È stato creato un cluster EKS con funzionalità Argo CD (vedi) the section called "Crea una funzionalità Argo CD"
- Un repository Git contenente i manifesti di Kubernetes
- `kubectl`configurato per comunicare con il tuo cluster

Attività comuni

I seguenti argomenti illustrano le attività più comuni di Argo CD:

the section called "Configura i repository"- Configura Argo CD per accedere ai tuoi repository Git usando AWS Secrets Manager o AWS CodeConnections Kubernetes Secrets.

the section called "Registra i cluster"- Registra i cluster di destinazione in cui Argo CD distribuirà le applicazioni.

the section called "Progetti"- Organizza le applicazioni e applica i limiti di sicurezza utilizzando Projects per ambienti multi-tenant.

the section called "Crea applicazioni"- Crea applicazioni che vengono distribuite dai repository Git con politiche di sincronizzazione automatizzate o manuali.

the section called "ApplicationSets"- ApplicationSets Utilizzatelo per distribuire applicazioni in più ambienti o cluster utilizzando modelli e generatori.

Accedi all'interfaccia utente di Argo CD

Accedi all'interfaccia utente di Argo CD tramite la console EKS:

1. Apri la console Amazon EKS
2. Seleziona il tuo cluster
3. Scegli la scheda Funzionalità
4. Scegli Argo CD
5. Scegli Open Argo CD UI

L'interfaccia utente fornisce la topologia visiva dell'applicazione, lo stato e la cronologia della sincronizzazione, lo stato e gli eventi delle risorse, i controlli di sincronizzazione manuale e la gestione delle applicazioni.

Documentazione upstream

Per informazioni dettagliate sulle funzionalità di Argo CD:

- Documentazione Argo CD - Guida utente completa
- Specifiche dell'applicazione: riferimento completo all'API dell'applicazione
- ApplicationSet Guida: ApplicationSet modelli ed esempi
- Argo CD GitHub - Codice sorgente ed esempi

Configurare l'accesso al repository

Prima di distribuire le applicazioni, configura Argo CD per accedere ai tuoi repository Git e ai registri cartografici Helm. Argo CD supporta diversi metodi di autenticazione per GitHub, GitLab Bitbucket ed ECR. AWS CodeCommit AWS

> **Note**
>
> Per le integrazioni dirette dei AWS servizi (grafici ECR Helm, CodeCommit repository e CodeConnections), è possibile farvi riferimento direttamente nelle risorse dell'applicazione senza creare configurazioni di repository. Il Capability Role deve disporre delle autorizzazioni IAM richieste. Per informazioni dettagliate, vedi the section called “Configurazione delle autorizzazioni ”.

Prerequisiti

- È stato creato un cluster EKS con funzionalità Argo CD
- Archivi Git contenenti manifesti di Kubernetes
- `kubectl`configurato per comunicare con il tuo cluster

> **Note**
>
> AWS CodeConnections può connettersi a server Git situati nel AWS Cloud o in locale. Per ulteriori informazioni, consulta AWS CodeConnections.

Metodi di autenticazione

Metodo	Caso d'uso	Autorizzazioni IAM richieste
Integrazione diretta con i servizi AWS		
CodeCommit	Integrazione diretta con i AWS CodeCommit repository Git. Non è necessaria alcuna configurazione del repository.	`codecommit:GitPull`
CodeConnections	Connect o Bitbucket con autenticazione gestita. GitHub GitLab Richiede la configurazione della connessione.	`codeconnections:UseConnection`
Artefatti ECR OCI	Integrazione diretta con AWS ECR per grafici e immagini manifest OCI Helm. Non è necessaria alcuna configurazione del repository.	`arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly`
Configurazione del repository con credenziali		
AWS Secrets Manager (nome utente/token)	Memorizza token o password di accesso personali. Abilita la rotazione delle credenziali senza accesso a Kubernetes.	`arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess`

Metodo	Caso d'uso	Autorizzazioni IAM richieste
AWS Secrets Manager (chiave SSH)	Usa l'autenticazione tramite chiave SSH. Abilita la rotazione delle credenziali senza accesso a Kubernetes.	`arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess`
AWS Secrets Manager (GitHub App)	GitHub Autenticazione dell'app con chiave privata. Abilita la rotazione delle credenziali senza accesso a Kubernetes.	`arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess`
Kubernetes Secret	Metodo Argo CD standard che utilizza segreti all'interno del cluster	Nessuna (autorizzazioni gestite da EKS Access Entry con Kubernetes RBAC)

AWS Accesso diretto ai servizi

Per quanto riguarda AWS i servizi, è possibile farvi riferimento direttamente nelle risorse dell'applicazione senza creare configurazioni del repository. Il Capability Role deve disporre delle autorizzazioni IAM richieste.

CodeCommit repository

Fai riferimento ai CodeCommit repository direttamente nelle applicazioni:

```
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
```

Autorizzazioni necessarie per il ruolo di capacità:

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codecommit:GitPull",
      "Resource": "arn:aws:codecommit:region:account-id:repository-name"
    }
  ]
}
```

CodeConnections

Reference GitHub o GitLab tramite Bitbucket repository. CodeConnections Il formato dell'URL del repository è derivato dall'ARN della CodeConnections connessione.

Il formato dell'URL del repository è:

```
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
```

Autorizzazioni necessarie per il ruolo di capacità:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codeconnections:UseConnection",
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-id"
    }
  ]
}
```

Grafici ECR Helm

ECR memorizza i grafici Helm come artefatti OCI. Argo CD supporta due modi per farvi riferimento:

Formato Helm (consigliato per i grafici 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
```

Nota: non includere il `oci://` prefisso quando si utilizza il formato Helm. Usa il `chart` campo per specificare il nome del grafico.

Formato OCI (per artefatti OCI con manifesti 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
```

Nota: includi il prefisso quando usi il formato OCI. `oci://` Usa il `path` campo invece di. `chart`

Autorizzazioni Capability Role richieste: allega la policy gestita:

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

Questa politica include le autorizzazioni ECR necessarie:`ecr:GetAuthorizationToken`,, `ecr:BatchGetImage` e. `ecr:GetDownloadUrlForLayer`

Utilizzo di AWS Secrets Manager

Archivia le credenziali del repository in Secrets Manager e fai riferimento ad esse nelle configurazioni di Argo CD Repository. L'utilizzo di Secrets Manager consente la rotazione automatica delle credenziali senza richiedere l'accesso RBAC a Kubernetes: le credenziali possono essere ruotate utilizzando le autorizzazioni IAM verso Secrets Manager e Argo CD legge automaticamente i valori aggiornati.

> **Note**
>
> Per il riutilizzo delle credenziali su più repository (ad esempio, tutti gli archivi di un'organizzazione), utilizza i modelli di credenziali di repository con. GitHub `argocd.argoproj.io/secret-type: repo-creds` Ciò offre un'esperienza utente migliore rispetto alla creazione di segreti di repository individuali. Per ulteriori informazioni, consultate Repository Credentials nella documentazione di Argo CD.

Autenticazione con nome utente e token

Per gli archivi HTTPS con token di accesso o password personali:

Crea il segreto in 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"}'
```

Campi opzionali del certificato client TLS (per server Git privati):

```
aws secretsmanager create-secret \
  --name argocd/my-private-repo \
  --secret-string '{
    "username":"your-username",
    "token":"your-token",

 "tlsClientCertData":"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCi4uLgotLS0tLUVORCBDRVJUSUZJQ0FURS0tL
```

```
 "tlsClientCertKey":"LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCi4uLgotLS0tLUVORCBQUklWQVRFIEtFWS0tLS
  }'
```

> **Note**
>
> I `tlsClientCertKey` valori `tlsClientCertData` and devono essere codificati in base 64.

Crea un Repository Secret facendo riferimento a 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
```

autenticazione con chiave SSH

Per l'accesso Git basato su SSH, memorizza la chiave privata come testo semplice (non JSON):

Crea il segreto con la chiave privata 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-----"
```

Crea un repository segreto per 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
```

GitHub Autenticazione dell'app

Per GitHub l'autenticazione dell'app con una chiave privata:

Crea il segreto con le credenziali GitHub dell'app:

```
aws secretsmanager create-secret \
  --name argocd/github-app \
  --description "GitHub App credentials for Argo CD" \
  --secret-string '{

 "githubAppPrivateKeySecret":"LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQouLi4KLS0tLS1FTkQgUlNBI
    "githubAppID":"123456",
    "githubAppInstallationID":"12345678"
  }'
```

> **Note**
>
> Il `githubAppPrivateKeySecret` valore deve essere codificato in base64.

Campo opzionale per Enterprise: GitHub

```
aws secretsmanager create-secret \
  --name argocd/github-enterprise-app \
  --secret-string '{

 "githubAppPrivateKeySecret":"LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQouLi4KLS0tLS1FTkQgUlNBI
    "githubAppID":"123456",
```

```
    "githubAppInstallationID":"12345678",
    "githubAppEnterpriseBaseUrl":"https://github.example.com/api/v3"
  }'
```

Crea un repository segreto per GitHub l'app:

```
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
```

Modelli di credenziali di repository

Per il riutilizzo delle credenziali su più repository (ad esempio, tutti gli archivi di GitHub un'organizzazione o di un utente), utilizza i modelli di credenziali di repository con. `argocd.argoproj.io/secret-type: repo-creds` Ciò offre una migliore esperienza utente rispetto alla creazione di segreti di repository individuali per ogni repository.

Crea un modello di credenziali di repository:

```
apiVersion: v1
kind: Secret
metadata:
  name: github-org-creds
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repo-creds
stringData:
  type: git
  url: https://github.com/your-org
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/github-org-
AbCdEf
```

Questo modello di credenziali si applica a tutti gli archivi che corrispondono al prefisso URL. `https://github.com/your-org` È quindi possibile fare riferimento a qualsiasi repository di questa organizzazione in Applicazioni senza creare segreti aggiuntivi.

Per ulteriori informazioni, consultate Repository Credentials nella documentazione del CD Argo.

> **Important**
>
> Assicurati che al tuo IAM Capability Role sia `arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess` associata la policy gestita o autorizzazioni equivalenti, incluse `secretsmanager:GetSecretValue` le autorizzazioni di decrittografia KMS. Vedi per la configurazione delle policy IAMthe section called "Considerazioni su Argo CD".

Usando AWS CodeConnections

Per CodeConnections l'integrazione, vedithe section called " AWS CodeConnections".

CodeConnections fornisce l'autenticazione gestita per GitHub e Bitbucket senza memorizzare le credenziali. GitLab

Utilizzo di Kubernetes Secrets

Archivia le credenziali direttamente in Kubernetes utilizzando il metodo Argo CD standard.

Per HTTPS con token di accesso personale:

```
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
```

Per 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-----
```

CodeCommit repository

Per AWS CodeCommit, concedi le CodeCommit autorizzazioni a IAM Capability Role ()`codecommit:GitPull`.

Configura il repository:

```
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
```

Per una configurazione dettagliata delle policy IAM, consultathe section called "Considerazioni su Argo CD".

Verifica la connessione al repository

Controlla lo stato della connessione tramite l'interfaccia utente di Argo CD in Impostazioni → Repository. L'interfaccia utente mostra lo stato della connessione e gli eventuali errori di autenticazione.

I Repository Secrets non includono informazioni sullo stato.

Risorse aggiuntive

- the section called "Registra i cluster"- Registra i cluster di destinazione per le distribuzioni
- the section called "Crea applicazioni"- Crea la tua prima applicazione
- the section called "Considerazioni su Argo CD"- Autorizzazioni IAM e configurazione di sicurezza
- Archivi privati: riferimento alla configurazione del repository upstream

Registra i cluster di destinazione

Registra i cluster per consentire ad Argo CD di distribuire applicazioni su di essi. È possibile registrare lo stesso cluster su cui è in esecuzione Argo CD (cluster locale) o cluster remoti in account o regioni diversi. Una volta registrato, un cluster rimarrà in uno stato di connessione sconosciuto fino a quando non si crea un'applicazione all'interno di quel cluster. Per creare un'applicazione Argo CD dopo la registrazione del cluster, consultathe section called "Crea applicazioni".

Prerequisiti

- È stato creato un cluster EKS con funzionalità Argo CD
- `kubectl`configurato per comunicare con il cluster
- Per i cluster remoti: autorizzazioni IAM e voci di accesso appropriate

Registra il cluster locale

Per distribuire le applicazioni nello stesso cluster su cui è in esecuzione Argo CD, registratelo come obiettivo di distribuzione.

> **Important**
>
> La funzionalità Argo CD non registra automaticamente il cluster locale. È necessario registrarlo in modo esplicito per distribuire le applicazioni nello stesso cluster. È possibile utilizzare il nome del cluster `in-cluster` per motivi di compatibilità con la maggior parte degli esempi di Argo CD online.

> **Note**
>
> Un EKS Access Entry viene creato automaticamente per il cluster locale con l'Argo CD Capability Role, ma per impostazione predefinita non vengono concesse autorizzazioni RBAC per Kubernetes. Ciò segue il principio del privilegio minimo: è necessario configurare in modo esplicito le autorizzazioni richieste da Argo CD in base al caso d'uso. Ad esempio, se si utilizza questo cluster solo come hub Argo CD per gestire cluster remoti, non sono necessarie autorizzazioni di distribuzione locale. Per le opzioni di configurazione, consulta la sezione relativa ai requisiti RBAC di Access Entry di seguito.

Utilizzo dell'Argo CD CLI:

```
argocd cluster add <cluster-context-name> \
  --aws-cluster-name arn:aws:eks:us-west-2:111122223333:cluster/my-cluster \
  --name local-cluster
```

Utilizzo di un segreto Kubernetes:

```
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
```

Applica la configurazione:

```
kubectl apply -f local-cluster.yaml
```

> **Note**
>
> Utilizza l'ARN del cluster EKS nel `server` campo, non l'URL del server API Kubernetes. La funzionalità gestita richiede ARNs l'identificazione dei cluster. L'impostazione predefinita non `kubernetes.default.svc` è supportata.

Registra cluster remoti

Per eseguire la distribuzione su cluster remoti:

Fase 1: Creare la voce di accesso sul cluster remoto

Sostituisci *`region-code`* con la AWS regione in cui si trova il cluster remoto, sostituisci *`remote-cluster`* con il nome del cluster remoto e sostituisci l'ARN con il tuo ruolo di funzionalità Argo CD ARN.

```
aws eks create-access-entry \
  --region region-code \
  --cluster-name remote-cluster \
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --type STANDARD
```

Fase 2: Associare una politica di accesso alle autorizzazioni RBAC di Kubernetes

L'Access Entry richiede le autorizzazioni RBAC di Kubernetes per Argo CD per distribuire le applicazioni. Per iniziare rapidamente, puoi utilizzare: `AmazonEKSClusterAdminPolicy`

```
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
```

> **Important**
>
> `AmazonEKSClusterAdminPolicy`Fornisce l'accesso completo all'amministratore del cluster (equivalente `asystem:masters`). È utile per iniziare, ma non deve essere utilizzato in produzione. Per gli ambienti di produzione, utilizza autorizzazioni più restrittive associando

l'Access Entry a gruppi Kubernetes personalizzati e creando ruoli o associazioni appropriati. ClusterRole Consulta la sezione sulla configurazione di produzione di seguito per la configurazione con privilegi minimi.

Fase 3: Registrare il cluster in Argo CD

Utilizzo dell'Argo CD CLI:

```
argocd cluster add <cluster-context-name> \
  --aws-cluster-name arn:aws:eks:us-west-2:111122223333:cluster/remote-cluster \
  --name remote-cluster
```

Utilizzo di un segreto Kubernetes:

```
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
```

Applica la configurazione:

```
kubectl apply -f remote-cluster.yaml
```

Cluster tra account

Per eseguire la distribuzione su cluster con account diversi: AWS

1. Nell'account di destinazione, creare un Access Entry sul cluster EKS di destinazione utilizzando l'ARN Argo CD IAM Capability Role dall'account di origine come principale
2. Associa una politica di accesso alle autorizzazioni Kubernetes RBAC appropriate
3. Registra il cluster in Argo CD utilizzando il relativo ARN del cluster EKS

Non è richiesta la creazione di ruoli IAM aggiuntivi o la configurazione delle policy di fiducia: EKS Access Entries gestisce l'accesso tra account.

Il formato ARN del cluster include la regione, quindi le distribuzioni interregionali utilizzano lo stesso processo delle distribuzioni nella stessa regione.

Verifica la registrazione del cluster

Visualizza i cluster registrati:

```
kubectl get secrets -n argocd -l argocd.argoproj.io/secret-type=cluster
```

Oppure controlla lo stato del cluster nell'interfaccia utente di Argo CD in Impostazioni → Cluster.

Cluster privati

La funzionalità Argo CD fornisce un accesso trasparente a cluster EKS completamente privati senza richiedere peering VPC o configurazioni di rete specializzate.

AWS gestisce automaticamente la connettività tra la funzionalità Argo CD e i cluster remoti privati.

È sufficiente registrare il cluster privato utilizzando la relativa ARN: non è richiesta alcuna configurazione di rete aggiuntiva.

Accedi ai requisiti RBAC di ingresso

Quando si crea una funzionalità Argo CD, viene creata automaticamente una voce EKS Access Entry per il ruolo Capability, ma per impostazione predefinita non vengono concesse autorizzazioni RBAC per Kubernetes. Questa progettazione intenzionale segue il principio del privilegio minimo: casi d'uso diversi richiedono autorizzazioni diverse.

Ad esempio: * Se si utilizza il cluster solo come hub Argo CD per gestire cluster remoti, non sono necessarie autorizzazioni di distribuzione locale* Se si distribuiscono applicazioni localmente, è necessario l'accesso in lettura a livello di cluster e l'accesso in scrittura a namespace specifici * Se è necessario creare, sono necessarie autorizzazioni di amministratore del cluster aggiuntive CRDs

È necessario configurare in modo esplicito le autorizzazioni richieste da Argo CD in base alle proprie esigenze.

Autorizzazioni minime per Argo CD

Argo CD necessita di due tipi di autorizzazioni per funzionare senza errori:

Autorizzazioni di lettura (a livello di cluster): Argo CD deve essere in grado di leggere tutti i tipi di risorse e le definizioni di risorse personalizzate (CRDs) in tutto il cluster per:

- Scoperta delle risorse e controlli dello stato
- Rilevamento della deriva tra lo stato desiderato e quello effettivo
- Convalida delle risorse prima della distribuzione

Autorizzazioni di scrittura (specifiche dello spazio dei nomi): Argo CD richiede le autorizzazioni di creazione, aggiornamento ed eliminazione per le risorse definite in Applicazioni:

- Implementa i carichi di lavoro delle applicazioni (implementazioni, servizi, ecc.) ConfigMaps
- Applica risorse personalizzate (CRDs specifiche per le tue applicazioni)
- Gestisci il ciclo di vita delle applicazioni

Configurazione rapida

Per iniziare rapidamente, in ambienti di test o sviluppo, utilizza: `AmazonEKSClusterAdminPolicy`

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name my-cluster \
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

> **Important**
>
> `AmazonEKSClusterAdminPolicy`Fornisce l'accesso completo all'amministratore del cluster (equivalente `asystem:masters`), inclusa la possibilità di creare CRDs, modificare risorse a livello di cluster e distribuirle in qualsiasi spazio dei nomi. Questa funzionalità è utile per lo sviluppo, POCs ma non deve essere utilizzata in produzione. Per la produzione, utilizzate la configurazione con privilegi minimi riportata di seguito.

Configurazione di produzione con privilegi minimi

Per gli ambienti di produzione, crea un RBAC Kubernetes personalizzato che garantisca:

- Accesso in lettura a livello di cluster a tutte le risorse (per il rilevamento e i controlli di integrità)
- Accesso in scrittura specifico per lo spazio dei nomi (per le distribuzioni)

Fase 1: Associare Access Entry a un gruppo Kubernetes personalizzato

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name my-cluster \
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy \
  --access-scope type=namespace,namespaces=app-namespace
```

Fase 2: ClusterRole Creazione per l'accesso in lettura

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: argocd-read-all
rules:
# Read access to all resources for discovery and health checks
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["get", "list", "watch"]
```

Fase 3: Creazione di un ruolo per l'accesso in scrittura ai namespace delle applicazioni

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: argocd-deploy
  namespace: app-namespace
rules:
# Full access to deploy application resources
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
```

Fase 4: Associare i ruoli al gruppo Kubernetes

```
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: ClusterRoleBinding
metadata:
  name: argocd-read-all
subjects:
- kind: Group
  name: eks-access-entry:arn:aws:iam::111122223333:role/ArgoCDCapabilityRole
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: argocd-read-all
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: argocd-deploy
  namespace: app-namespace
subjects:
- kind: Group
  name: eks-access-entry:arn:aws:iam::111122223333:role/ArgoCDCapabilityRole
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: argocd-deploy
  apiGroup: rbac.authorization.k8s.io
```

Note

Il formato del nome del gruppo per Access Entries è `eks-access-entry:` seguito dall'ARN principale. Ripetere l'operazione RoleBinding per ogni namespace in cui Argo CD deve distribuire le applicazioni.

Important

Argo CD deve essere in grado di leggere tutti i tipi di risorse del cluster a fini di verifica e individuazione dello stato, anche se viene distribuito solo su namespace specifici. Senza accesso in lettura a livello di cluster, Argo CD mostrerà errori durante il controllo dello stato delle applicazioni.

Limita l'accesso ai cluster con Projects

Usa Projects per controllare in quali cluster e namespace le applicazioni possono essere distribuite configurando i cluster e i namespace di destinazione consentiti in: `spec.destinations`

```
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'
```

Per informazioni dettagliate, vedi the section called “Progetti”.

Risorse aggiuntive

- the section called “Progetti”- Organizza le applicazioni e applica i limiti di sicurezza
- the section called “Crea applicazioni”- Implementa la tua prima applicazione
- the section called “ApplicationSets”- Esegui la distribuzione su più cluster con ApplicationSets
- the section called “Considerazioni su Argo CD”- Modelli multi-cluster e configurazione tra più account
- Configurazione dichiarativa del cluster - Riferimento alla configurazione del cluster upstream

Lavorare con Argo CD Projects

Argo CD Projects (AppProject) forniscono il raggruppamento logico e il controllo degli accessi per le applicazioni. I progetti definiscono i repository Git, i cluster di destinazione e i namespace che le applicazioni possono utilizzare, abilitando limiti di multi-tenancy e sicurezza nelle istanze Argo CD condivise.

Quando usare Projects

Usa Projects per:

- Separa le applicazioni per team, ambiente o unità aziendale
- Limita i repository da cui i team possono eseguire l'implementazione
- Limita i cluster e i namespace in cui i team possono effettuare l'implementazione
- Applica le quote di risorse e i tipi di risorse consentiti
- Fornisci l'implementazione self-service delle applicazioni con guardrail

Progetto predefinito

Ogni funzionalità di Argo CD include un `default` progetto che consente l'accesso a tutti i repository, i cluster e i namespace. Sebbene sia utile per i test iniziali, crea progetti dedicati con restrizioni esplicite per l'uso in produzione.

Per i dettagli sulla configurazione predefinita del progetto e su come limitarla, consultate The Default Project nella documentazione di Argo CD.

Creazione di un progetto

Crea un progetto applicando una `AppProject` risorsa al tuo cluster.

Esempio: progetto specifico per il team

```
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
```

Applica il progetto:

```
kubectl apply -f team-a-project.yaml
```

Configurazione del progetto

Archivi di origine

Controlla quali repository Git le applicazioni di questo progetto possono utilizzare:

```
spec:
  sourceRepos:
    - 'https://github.com/my-org/app-*'  # Wildcard pattern
    - 'https://github.com/my-org/infra'  # Specific repo
```

È possibile utilizzare caratteri jolly e schemi di negazione (`!`prefisso) per consentire o negare repository specifici. Per i dettagli, consultate Gestione dei progetti nella documentazione di Argo CD.

Restrizioni sulla destinazione

Limite in cui le applicazioni possono essere distribuite:

```
spec:
  destinations:
    - name: prod-cluster  # Specific cluster by name
      namespace: production
    - name: '*'  # Any cluster
```

```
      namespace: team-a-*  # Namespace pattern
```

> **Important**
>
> Utilizza nomi di cluster e modelli di namespace specifici anziché caratteri jolly per i progetti di produzione. In questo modo si evitano implementazioni accidentali in cluster o namespace non autorizzati.

È possibile utilizzare caratteri jolly e schemi di negazione per controllare le destinazioni. Per i dettagli, consultate Gestione dei progetti nella documentazione del CD Argo.

Restrizioni sulle risorse

Controlla quali tipi di risorse Kubernetes possono essere implementati:

Risorse con ambito cluster:

```
spec:
  clusterResourceWhitelist:
    - group: ''
      kind: Namespace
    - group: 'rbac.authorization.k8s.io'
      kind: Role
```

Risorse con ambito namespace:

```
spec:
  namespaceResourceWhitelist:
    - group: 'apps'
      kind: Deployment
    - group: ''
      kind: Service
    - group: ''
      kind: ConfigMap
    - group: 's3.services.k8s.aws'
      kind: Bucket
```

Usa le liste nere per negare risorse specifiche:

```
spec:
```

```
  namespaceResourceBlacklist:
    - group: ''
      kind: Secret  # Prevent direct Secret creation
```

Assegna applicazioni ai progetti

Quando crei un'applicazione, specifica il progetto nel `spec.project` campo:

```
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
```

Le applicazioni senza un progetto specificato utilizzano il `default` progetto.

Ruoli del progetto e RBAC

I progetti possono definire ruoli personalizzati per un controllo granulare degli accessi. Associa i ruoli del progetto agli utenti e ai gruppi di AWS Identity Center nella configurazione delle funzionalità per controllare chi può sincronizzare, aggiornare o eliminare le applicazioni.

Esempio: progetto con ruoli di sviluppatore e amministratore

```
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
```

Per i dettagli sui ruoli del progetto, sui token JWT per le CI/CD pipeline e sulla configurazione RBAC, consulta Project Roles nella documentazione di Argo CD.

Schemi comuni

Progetti basati sull'ambiente

Crea progetti separati per ogni 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
```

Progetti basati su team

Isola i team con progetti dedicati:

```
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: '*'
```

Progetti multicluster

Implementa su più cluster con politiche coerenti:

```
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
```

Best practice

Inizia con progetti restrittivi: inizia con autorizzazioni limitate ed espandi secondo necessità anziché iniziare con un accesso ampio.

Usa modelli di namespace: sfrutta i caratteri jolly nelle restrizioni dello spazio dei nomi (ad esempio`team-a-*`) per consentire flessibilità pur mantenendo i limiti.

Progetti di produzione separati: utilizza progetti dedicati per la produzione con controlli più rigorosi e politiche di sincronizzazione manuale.

Scopi del progetto di documentazione: utilizza il `description` campo per spiegare a cosa serve ogni progetto e chi dovrebbe utilizzarlo.

Rivedi regolarmente le autorizzazioni del progetto: controlla periodicamente i progetti per assicurarti che le restrizioni siano ancora in linea con le esigenze del team e i requisiti di sicurezza.

Risorse aggiuntive

- the section called "Configurazione delle autorizzazioni "- Configurazione dell'integrazione tra RBAC e Identity Center
- the section called "Crea applicazioni"- Crea applicazioni all'interno dei progetti
- the section called "ApplicationSets"- Utilizzabile ApplicationSets con Projects per implementazioni multi-cluster
- Documentazione sui progetti Argo CD - Riferimento completo a upstream

Crea applicazioni

Le applicazioni rappresentano distribuzioni nei cluster di destinazione. Ogni applicazione definisce una fonte (repository Git) e una destinazione (cluster e namespace). Una volta applicato, Argo CD creerà le risorse specificate dai manifest nel repository Git nello spazio dei nomi del cluster. Le applicazioni spesso specificano le distribuzioni dei carichi di lavoro, ma possono gestire qualsiasi risorsa Kubernetes disponibile nel cluster di destinazione.

Prerequisiti

- È stato creato un cluster EKS con funzionalità Argo CD
- Accesso al repository configurato (vedi) the section called "Configura i repository"
- Cluster di destinazione registrato (vedithe section called "Registra i cluster")

- `kubectl`configurato per comunicare con il cluster

Crea un'applicazione di base

Definisci un'applicazione che viene distribuita da un repository 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:
    name: in-cluster
    namespace: default
```

> **Note**
>
> Usalo `destination.name` con il nome del cluster che hai usato durante la registrazione del cluster (come `in-cluster` per il cluster locale). Il `destination.server` campo funziona anche con il cluster EKS ARNs, ma si consiglia di utilizzare i nomi dei cluster per una migliore leggibilità.

Applica l'applicazione:

```
kubectl apply -f application.yaml
```

Visualizza lo stato dell'applicazione:

```
kubectl get application guestbook -n argocd
```

Configurazione dell'origine

Archivio Git:

```
spec:
  source:
    repoURL: https://github.com/example/my-app
    targetRevision: main
    path: kubernetes/manifests
```

Tag o commit Git specifici:

```
spec:
  source:
    targetRevision: v1.2.0  # or commit SHA
```

Diagramma di 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
```

Grafico Helm con valori dal repository Git esterno (modello multi-source):

```
spec:
  sources:
  - repoURL: https://github.com/example/helm-charts
    targetRevision: main
    path: charts/my-app
    helm:
      valueFiles:
      - $values/environments/production/values.yaml
  - repoURL: https://github.com/example/config-repo
    targetRevision: main
    ref: values
```

Per ulteriori informazioni, consultate Helm Value Files from External Git Repository nella documentazione del CD Argo.

Diagramma Helm tratto da ECR:

```
spec:
  source:
    repoURL: oci://account-id.dkr.ecr.region.amazonaws.com/repository-name
    targetRevision: chart-version
    chart: chart-name
```

Se il Capability Role dispone delle autorizzazioni ECR richieste, il repository viene utilizzato direttamente e non è richiesta alcuna configurazione del repository. Per informazioni dettagliate, vedi the section called "Configura i repository".

Repository Git da CodeCommit:

```
spec:
  source:
    repoURL: https://git-codecommit.region.amazonaws.com/v1/repos/repository-name
    targetRevision: main
    path: kubernetes/manifests
```

Se il Capability Role dispone delle CodeCommit autorizzazioni richieste, il repository viene utilizzato direttamente e non è richiesta alcuna configurazione del repository. Per informazioni dettagliate, vedi the section called "Configura i repository".

Repository Git da CodeConnections:

```
spec:
  source:
    repoURL: https://codeconnections.region.amazonaws.com/git-http/account-
id/region/connection-id/owner/repository.git
    targetRevision: main
    path: kubernetes/manifests
```

Il formato dell'URL del repository è derivato dall'ARN della CodeConnections connessione. Se il Capability Role dispone delle CodeConnections autorizzazioni richieste ed è configurata una connessione, il repository viene utilizzato direttamente e non è richiesta alcuna configurazione del repository. Per informazioni dettagliate, vedi the section called "Configura i repository".

Personalizza:

```
spec:
  source:
    repoURL: https://github.com/example/kustomize-app
    targetRevision: main
    path: overlays/production
    kustomize:
      namePrefix: prod-
```

Politiche di sincronizzazione

Controlla il modo in cui Argo CD sincronizza le applicazioni.

Sincronizzazione manuale (impostazione predefinita):

Le applicazioni richiedono l'approvazione manuale per la sincronizzazione:

```
spec:
  syncPolicy: {}  # No automated sync
```

Attiva manualmente la sincronizzazione:

```
kubectl patch application guestbook -n argocd \
  --type merge \
  --patch '{"operation": {"initiatedBy": {"username": "admin"}, "sync": {}}}'
```

Sincronizzazione automatica:

Le applicazioni si sincronizzano automaticamente quando vengono rilevate modifiche a Git:

```
spec:
  syncPolicy:
    automated: {}
```

Autoriparazione:

Ripristina automaticamente le modifiche manuali al cluster:

```
spec:
  syncPolicy:
```

```
    automated:
      selfHeal: true
```

Se abilitato, Argo CD ripristina qualsiasi modifica manuale apportata direttamente al cluster, assicurando che Git rimanga la fonte della verità.

Potatura:

Elimina automaticamente le risorse rimosse da Git:

```
spec:
  syncPolicy:
    automated:
      prune: true
```

> **Warning**
>
> Il pruning eliminerà le risorse dal cluster. Usare con cautela negli ambienti di produzione.

Sincronizzazione automatica combinata:

```
spec:
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - CreateNamespace=true
```

Riprova la configurazione:

Configura il comportamento dei tentativi di ripetizione per le sincronizzazioni non riuscite:

```
spec:
  syncPolicy:
    retry:
      limit: 5  # Number of failed sync attempts; unlimited if less than 0
      backoff:
        duration: 5s  # Amount to back off (default unit: seconds, also supports "2m",
 "1h")
```

```
        factor: 2  # Factor to multiply the base duration after each failed retry
        maxDuration: 3m  # Maximum amount of time allowed for the backoff strategy
```

Ciò è particolarmente utile per le risorse che dipendono dal fatto di CRDs essere state create per prime o quando si lavora con istanze kro in cui il CRD potrebbe non essere immediatamente disponibile.

Opzioni di sincronizzazione

Configurazione di sincronizzazione aggiuntiva:

Crea lo spazio dei nomi se non esiste:

```
spec:
  syncPolicy:
    syncOptions:
    - CreateNamespace=true
```

Salta la corsa a secco per le risorse mancanti:

Utile quando si applicano risorse che dipendono da CRDs ciò che non esistono ancora (come le istanze kro):

```
spec:
  syncPolicy:
    syncOptions:
    - SkipDryRunOnMissingResource=true
```

Questo può essere applicato anche a risorse specifiche utilizzando un'etichetta sulla risorsa stessa.

Convalida le risorse prima di applicarle:

```
spec:
  syncPolicy:
    syncOptions:
    - Validate=true
```

Applica solo non sincronizzato:

```
spec:
```

```
  syncPolicy:
    syncOptions:
    - ApplyOutOfSyncOnly=true
```

Funzionalità di sincronizzazione avanzate

Argo CD supporta funzionalità di sincronizzazione avanzate per implementazioni complesse:

- Sync waves: controlla l'ordine di creazione delle risorse con le annotazioni `argocd.argoproj.io/sync-wave`
- Sync hooks: esegui i lavori prima o dopo la sincronizzazione con `argocd.argoproj.io/hook` le annotazioni (PreSync,,) PostSync SyncFail
- Valutazione dello stato delle risorse: controlli di integrità personalizzati per le risorse specifiche dell'applicazione

Per i dettagli, consultate Sync Waves e Resource Hooks nella documentazione del CD Argo.

Ignora le differenze

Impedisci ad Argo CD di sincronizzare campi specifici gestiti da altri controller (come HPA che gestisce le repliche):

```
spec:
  ignoreDifferences:
  - group: apps
    kind: Deployment
    jsonPointers:
    - /spec/replicas
```

Per i dettagli sugli schemi di ignoramento e sulle esclusioni dei campi, consultate Diffing Customization nella documentazione del CD Argo.

Implementazione multiambiente

Implementa la stessa applicazione in più ambienti:

Sviluppo:

```
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:
    name: dev-cluster
    namespace: my-app
```

Produzione:

```
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:
    name: prod-cluster
    namespace: my-app
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

Monitora e gestisci le applicazioni

Visualizza lo stato dell'applicazione:

```
kubectl get application my-app -n argocd
```

Accedi all'interfaccia utente di Argo CD:

Apri l'interfaccia utente di Argo CD tramite la console EKS per visualizzare la topologia dell'applicazione, lo stato di sincronizzazione, lo stato delle risorse e la cronologia di distribuzione. Vedi the section called "Lavorare con Argo CD" per le istruzioni di accesso all'interfaccia utente.

Applicazioni di rollback:

Ripristina una revisione precedente utilizzando l'interfaccia utente di Argo CD, l'Argo CD CLI o aggiornando le specifiche `targetRevision` dell'applicazione a un commit o tag Git precedente.

Utilizzo dell'Argo CD CLI:

```
argocd app rollback argocd/my-app <revision-id>
```

> **Note**
>
> Quando si utilizza l'Argo CD CLI con la funzionalità gestita, specificare le applicazioni con il prefisso dello spazio dei nomi:. `namespace/appname`

Per ulteriori informazioni, consulta il rollback dell'app argocd nella documentazione di Argo CD.

Risorse aggiuntive

- the section called "Progetti"- Organizza le applicazioni con Projects per ambienti multi-tenant
- the section called "ApplicationSets"- Implementa su più cluster con modelli
- Specifiche dell'applicazione: riferimento completo all'API dell'applicazione
- Opzioni di sincronizzazione: configurazione di sincronizzazione avanzata

Usa ApplicationSets

ApplicationSets genera più applicazioni da modelli, consentendoti di distribuire la stessa applicazione su più cluster, ambienti o namespace con un'unica definizione di risorsa.

Prerequisiti

- È stato creato un cluster EKS con funzionalità Argo CD
- Accesso al repository configurato (vedi) the section called "Configura i repository"
- `kubectl`configurato per comunicare con il cluster

> **Note**
>
> Non sono necessari cluster di destinazione multipli per ApplicationSets. È possibile utilizzare generatori diversi dal generatore di cluster (come i generatori list, git o matrix) per distribuire applicazioni senza cluster remoti.

Come funziona ApplicationSets

ApplicationSets utilizzate i generatori per produrre parametri, quindi applicate tali parametri a un modello di applicazione. Ogni set di parametri generati crea un'applicazione.

Generatori comuni per le implementazioni EKS:

- Generatore di elenchi: definisce in modo esplicito cluster e parametri per ogni ambiente
- Generatore di cluster: implementa automaticamente su tutti i cluster registrati
- Generatore Git: genera applicazioni dalla struttura del repository
- Generatore di matrici: combina generatori per implementazioni multidimensionali
- Generatore di fusione: unisce i parametri di più generatori

Per un riferimento completo al generatore, consultate la Documentazione. ApplicationSet

Generatore di elenchi

Esegui la distribuzione su più cluster con configurazione esplicita:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook-all-clusters
  namespace: argocd
spec:
  generators:
  - list:
      elements:
      - environment: dev
        replicas: "2"
      - environment: staging
        replicas: "3"
      - 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:
        name: '{{environment}}-cluster'
        namespace: guestbook
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

> **Note**
>
> Utilizzalo `destination.name` con i nomi dei cluster per una migliore leggibilità. ARNs Se necessario, il `destination.server` campo funziona anche con il cluster EKS.

Questo crea tre applicazioni: `guestbook-dev`guestbook-staging, eguestbook-prod.

Generatore di cluster

Implementa automaticamente su tutti i cluster registrati:

```
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
```

Questo crea automaticamente un'applicazione per ogni cluster registrato.

Cluster di filtri:

Utilizza `matchLabels` per includere cluster specifici o `matchExpressions` per escludere cluster:

```
spec:
  generators:
  - clusters:
      selector:
        matchLabels:
          environment: production
        matchExpressions:
        - key: skip-appset
          operator: DoesNotExist
```

Generatori Git

I generatori Git creano applicazioni basate sulla struttura del repository:

- Generatore di directory: distribuisci ogni directory come applicazione separata (utile per i microservizi)
- Generatore di file: genera applicazioni da file di parametri (utile per le implementazioni multi-tenant)

Esempio: implementazione di microservizi

```
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:
        name: my-cluster
        namespace: '{{path.basename}}'
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
        syncOptions:
        - CreateNamespace=true
```

Per i dettagli sui generatori Git e sulla configurazione basata su file, consultate Git Generator nella documentazione del CD Argo.

Generatore di matrici

Combina più generatori da implementare su più dimensioni (ambienti × cluster):

```
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:
        name: '{{name}}'
        namespace: 'app-{{environment}}'
```

Per i dettagli sulla combinazione dei generatori, consultate Matrix Generator nella documentazione di Argo CD.

Implementazione in più regioni

Implementa su cluster in più regioni:

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: global-app
  namespace: argocd
spec:
  generators:
  - list:
      elements:
      - clusterName: prod-us-west
        region: us-west-2
      - clusterName: prod-us-east
```

```
        region: us-east-1
      - clusterName: 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:
        name: '{{clusterName}}'
        namespace: app
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

Gestisci ApplicationSets

Visualizza ApplicationSets e genera applicazioni:

```
kubectl get applicationsets -n argocd
kubectl get applications -n argocd -l argocd.argoproj.io/application-set-
name=<applicationset-name>
```

Aggiorna un ApplicationSet:

Modifica le ApplicationSet specifiche e riapplica. Argo CD aggiorna automaticamente tutte le applicazioni generate:

```
kubectl apply -f applicationset.yaml
```

Eliminare un ApplicationSet:

```
kubectl delete applicationset <name> -n argocd
```

> **Warning**
>
> Eliminazione ed ApplicationSet eliminazione di tutte le applicazioni generate. Se tali applicazioni lo hanno fatto`prune: true`, le relative risorse verranno eliminate anche dai cluster di destinazione.
> Per preservare le risorse distribuite durante l'eliminazione di un file ApplicationSet, impostate su `true` nelle `.syncPolicy.preserveResourcesOnDeletion` specifiche. ApplicationSet Per ulteriori informazioni, consultate Application Pruning & Resource Deletion nella documentazione del CD Argo.

> **Important**
>
> La ApplicationSets funzionalità di Argo CD prevede considerazioni sulla sicurezza di cui è necessario tenere conto prima di utilizzarla. ApplicationSets Per ulteriori informazioni, consultate la sezione ApplicationSet Sicurezza nella documentazione del CD Argo.

Risorse aggiuntive

- the section called "Progetti"- Organizza ApplicationSets con progetti
- the section called "Crea applicazioni"- Comprendi la configurazione dell'applicazione
- ApplicationSet Documentazione: riferimenti e modelli completi del generatore
- Generator Reference - Specifiche dettagliate del generatore

Considerazioni su Argo CD

Questo argomento tratta importanti considerazioni sull'utilizzo del CD EKS Capability for Argo, tra cui pianificazione, autorizzazioni, autenticazione e modelli di distribuzione multicluster.

Pianificazione

Prima di distribuire Argo CD, considerate quanto segue:

Strategia di archiviazione: stabilite dove verranno archiviati i manifesti dell'applicazione (CodeCommit,, GitHub, GitLab Bitbucket). Pianifica la struttura del repository e la strategia di ramificazione per diversi ambienti.

Strategia RBAC: pianifica quali team o utenti devono avere accesso come amministratore, editor o visualizzatore. Associali ai gruppi di AWS Identity Center o ai ruoli di Argo CD.

Architettura multicluster: stabilisci se gestirai più cluster da una singola istanza Argo CD. Prendi in considerazione l'utilizzo di un cluster di gestione dedicato per Argo CD.

Organizzazione delle applicazioni: pianifica come strutturerai le applicazioni e ApplicationSets. Prendi in considerazione l'utilizzo di progetti per organizzare le applicazioni per team o ambiente.

Criteri di sincronizzazione: decidi se le applicazioni devono sincronizzarsi automaticamente o richiedere l'approvazione manuale. La sincronizzazione automatica è comune per lo sviluppo, quella manuale per la produzione.

Permissions

Per informazioni dettagliate su IAM Capability Roles, sulle policy di fiducia e sulle best practice di sicurezza, consulta the section called "Capacità, ruolo IAM" ethe section called "Considerazioni sulle funzionalità EKS".

Panoramica del ruolo di IAM Capability

Quando crei una risorsa di funzionalità Argo CD, fornisci un ruolo di capacità IAM. A differenza di ACK, Argo CD gestisce principalmente le risorse Kubernetes, non le risorse direttamente. AWS Tuttavia, lo IAM Capability Role è necessario per:

- Accesso agli archivi Git privati in CodeCommit
- Integrazione con AWS Identity Center per l'autenticazione
- Accesso ai AWS segreti in Secrets Manager (se configurato)
- Implementazioni tra cluster su altri cluster EKS

CodeCommit integrazione

Se utilizzi i CodeCommit repository, allega una policy con autorizzazioni di lettura:

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

```
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Important**
>
> Per uso in produzione, limita il `Resource` campo a un repository specifico ARNs invece di utilizzarlo. `"*"`
> Esempio:
>
> ```
> "Resource": "arn:aws:codecommit:us-west-2:111122223333:my-app-repo"
> ```
>
> Ciò limita l'accesso della funzionalità Argo CD solo ai repository che deve gestire.

Integrazione di Secrets Manager

Se stai archiviando le credenziali del repository in Secrets Manager, allega la policy gestita per l'accesso in lettura:

```
arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess
```

Questa politica include le autorizzazioni necessarie:`secretsmanager:GetSecretValue`, e le autorizzazioni di `secretsmanager:DescribeSecret` decrittografia KMS.

Configurazione di base

Per le funzionalità di base di Argo CD con repository Git pubblici, non sono richieste policy IAM aggiuntive oltre alla policy di trust.

Autenticazione

AWS Integrazione con Identity Center

La funzionalità gestita da Argo CD si integra direttamente con AWS Identity Center (precedentemente AWS SSO), consentendoti di utilizzare il tuo provider di identità esistente per l'autenticazione.

Quando configuri l'integrazione con Identity Center: AWS

1. Gli utenti accedono all'interfaccia utente di Argo CD tramite la console EKS
2. Si autenticano utilizzando AWS Identity Center (che può essere collegato al vostro provider di identità aziendale)
3. AWS Identity Center fornisce informazioni su utenti e gruppi su Argo CD
4. Argo CD associa utenti e gruppi ai ruoli RBAC in base alla configurazione
5. Gli utenti vedono solo le applicazioni e le risorse a cui hanno il permesso di accedere

Semplificazione dell'accesso con i set di autorizzazioni di Identity Center

AWS Identity Center offre due percorsi di autenticazione distinti quando si lavora con Argo CD:

Autenticazione tramite API Argo CD: Identity Center fornisce l'autenticazione SSO all'interfaccia utente e all'API di Argo CD. Questo viene configurato tramite le mappature dei ruoli RBAC della funzionalità Argo CD.

Accesso al cluster EKS: la funzionalità Argo CD utilizza il ruolo IAM fornito dal cliente per l'autenticazione con i cluster EKS tramite voci di accesso. Queste voci di accesso possono essere configurate manualmente per aggiungere o rimuovere autorizzazioni.

È possibile utilizzare i set di autorizzazioni di Identity Center per semplificare la gestione delle identità consentendo a una singola identità di accedere sia ai cluster Argo CD che EKS. Ciò riduce il sovraccarico richiedendo di gestire una sola identità su entrambi i sistemi, anziché mantenere credenziali separate per l'accesso ad Argo CD e l'accesso al cluster.

Mappature dei ruoli RBAC

Argo CD dispone di ruoli integrati che è possibile mappare agli utenti e ai gruppi di AWS Identity Center:

ADMIN: accesso completo a tutte le applicazioni e le impostazioni. Può creare, aggiornare ed eliminare applicazioni. Può gestire la configurazione di Argo CD.

EDITOR: Può creare e modificare applicazioni. Non è possibile modificare le impostazioni di Argo CD o eliminare applicazioni.

VIEWER: accesso in sola lettura alle applicazioni. Può visualizzare lo stato e la cronologia dell'applicazione. Non è possibile apportare modifiche.

> **Note**
>
> I nomi dei ruoli fanno distinzione tra maiuscole e minuscole e devono essere maiuscole (ADMIN, EDITOR, VIEWER).

> **Important**
>
> L'integrazione di EKS Capabilities con AWS Identity Center supporta fino a 1.000 identità per funzionalità Argo CD. Un'identità può essere un utente o un gruppo.

Implementazioni multicluster

La funzionalità gestita da Argo CD supporta implementazioni multi-cluster, consentendo di gestire le applicazioni in cluster di sviluppo, staging e produzione da una singola istanza Argo CD.

Come funziona il multicluster

Quando si registrano cluster aggiuntivi con Argo CD:

1. Crei segreti del cluster che fanno riferimento ai cluster EKS di destinazione tramite ARN
2. Create applicazioni o destinate a ApplicationSets cluster diversi
3. Argo CD si connette a ciascun cluster per distribuire e controllare le risorse
4. Puoi visualizzare e gestire tutti i cluster da un'unica interfaccia utente di Argo CD

Prerequisiti per il multicluster

Prima di registrare cluster aggiuntivi:

- Crea una voce di accesso sul cluster di destinazione per il ruolo di funzionalità Argo CD
- Garantite la connettività di rete tra la funzionalità Argo CD e i cluster di destinazione
- Verifica le autorizzazioni IAM per accedere ai cluster di destinazione

Registra un cluster

Registra i cluster utilizzando Kubernetes Secrets nel namespace. `argocd`

Ottieni l'ARN del cluster di destinazione. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster di destinazione e *target-cluster* sostituiscilo con il nome del cluster di destinazione.

```
aws eks describe-cluster \
  --region region-code \
  --name target-cluster \
  --query 'cluster.arn' \
  --output text
```

Crea un segreto del cluster utilizzando l'ARN del 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

Utilizza l'ARN del cluster EKS nel `server` campo, non l'URL del server API Kubernetes. La funzionalità gestita richiede l'identificazione dei cluster ARNs di destinazione.

Applica il segreto:

```
kubectl apply -f cluster-secret.yaml
```

Configura Access Entry sul cluster di destinazione

Il cluster di destinazione deve disporre di un Access Entry che conceda al ruolo di funzionalità Argo CD l'autorizzazione a distribuire le applicazioni. Sostituisci *region-code* con la AWS regione in cui

si trova il cluster di destinazione, sostituisci *target-cluster* con il nome del cluster di destinazione e sostituisci l'ARN con il tuo ruolo di funzionalità Argo CD ARN.

```
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**
>
> Per l'uso in produzione, prendi in considerazione l'utilizzo di gruppi Kubernetes più restrittivi anziché. `system:masters`

Accesso privato al cluster

La funzionalità gestita da Argo CD può essere implementata su cluster EKS completamente privati senza richiedere peering VPC o configurazioni di rete specializzate. AWS gestisce automaticamente la connettività tra la funzionalità Argo CD e i cluster remoti privati. Assicurati che i controlli di accesso al repository e le politiche RBAC di Argo CD siano configurati correttamente.

Implementazioni su più account

Per le distribuzioni tra account, aggiungi l'Argo CD IAM Capability Role dall'account di origine all'EKS Access Entry del cluster di destinazione:

1. Nell'account di destinazione, create un Access Entry sul cluster EKS di destinazione
2. Usa l'ARN Argo CD IAM Capability Role dell'account di origine come principale
3. Configura le autorizzazioni RBAC Kubernetes appropriate per l'Access Entry
4. Registra il cluster di destinazione in Argo CD utilizzando il relativo ARN del cluster EKS

Non è richiesta la creazione di ruoli IAM aggiuntivi o la configurazione delle policy di fiducia: EKS Access Entries gestisce l'accesso tra account.

Best practice

Usa le fonti dichiarative come fonte di verità: archivia tutti i manifesti delle tue applicazioni in fonti dichiarative (repository Git, registri Helm o immagini OCI), abilitando il controllo delle versioni, gli audit trail e la collaborazione.

Implementa un RBAC adeguato: utilizza l'integrazione di AWS Identity Center per controllare chi può accedere e gestire le applicazioni in Argo CD. Argo CD supporta un controllo granulare degli accessi alle risorse all'interno delle applicazioni (implementazioni, pod e segreti). ConfigMaps

Utilizzo ApplicationSets per implementazioni multiambiente: da utilizzare per distribuire applicazioni su più cluster o namespace ApplicationSets con configurazioni diverse.

Gestione del ciclo di vita

Politiche di sincronizzazione delle applicazioni

Controlla il modo in cui Argo CD sincronizza le applicazioni:

Sincronizzazione manuale: le applicazioni richiedono l'approvazione manuale per sincronizzare le modifiche. Consigliata per ambienti di produzione.

Sincronizzazione automatica: le applicazioni si sincronizzano automaticamente quando vengono rilevate modifiche a Git. Comune per gli ambienti di sviluppo e gestione temporanea.

Correzione automatica: ripristina automaticamente le modifiche manuali apportate al cluster. Assicura che lo stato del cluster corrisponda a Git.

Potatura: elimina automaticamente le risorse rimosse da Git. Usalo con cautela in quanto ciò può eliminare le risorse.

Integrità dell'applicazione

Argo CD monitora continuamente lo stato delle applicazioni:

- Salutare: tutte le risorse funzionano come previsto
- Progressione: le risorse vengono create o aggiornate
- Degradate: alcune risorse non sono salutari
- Sospesa: l'applicazione è in pausa
- Mancante: le risorse non sono presenti nel cluster

Sincronizza finestre

Configura le finestre di sincronizzazione per controllare quando le applicazioni possono essere sincronizzate:

- Consenti le sincronizzazioni solo durante le finestre di manutenzione
- Blocca le sincronizzazioni durante l'orario lavorativo
- Pianifica le sincronizzazioni automatiche per orari specifici
- Utilizza le finestre di sincronizzazione in situazioni in cui devi apportare modifiche e interrompere qualsiasi sincronizzazione (scenari break-glass)

Configurazione Webhook per una sincronizzazione più rapida

Per impostazione predefinita, Argo CD esegue il polling dei repository Git ogni 6 minuti per rilevare le modifiche. Per implementazioni più reattive, configura i webhook Git per attivare sincronizzazioni immediate quando vengono inviate le modifiche.

I webhook offrono diversi vantaggi:

- Risposta di sincronizzazione immediata quando viene inviato il codice (secondi anziché minuti)
- Riduzione del sovraccarico di polling e miglioramento delle prestazioni del sistema
- Uso più efficiente dei limiti di velocità delle API
- Esperienza utente migliore con feedback più rapidi

Endpoint Webhook

L'URL del webhook segue lo schema`${serverUrl}/api/webhook`, `serverUrl` dov'è l'URL del server Argo CD.

Ad esempio, se l'URL del server Argo CD è`https://abc123.eks-capabilities.us-west-2.amazonaws.com`, l'URL del webhook è:

```
https://abc123.eks-capabilities.us-west-2.amazonaws.com/api/webhook
```

Configura i webhook tramite il provider Git

GitHub: Nelle impostazioni del repository, aggiungi un webhook con l'URL del webhook Argo CD. Imposta il tipo di contenuto su `application/json` e seleziona «Just the push event».

GitLab: Nelle impostazioni del progetto, aggiungi un webhook con l'URL del webhook di Argo CD. Abilita «Eventi push» e, facoltativamente, «Tag eventi push».

Bitbucket: nelle impostazioni del repository, aggiungi un webhook con l'URL del webhook di Argo CD. Seleziona «Repository push» come trigger.

CodeCommit: crea una EventBridge regola Amazon che si attiva in base alle modifiche dello stato del CodeCommit repository e invia notifiche all'endpoint webhook Argo CD.

Per istruzioni dettagliate sulla configurazione dei webhook, consulta Argo CD Webhook Configuration.

> **Note**
>
> I webhook completano il polling, non lo sostituiscono. Argo CD continua a sondare gli archivi come meccanismo di riserva nel caso in cui le notifiche dei webhook vengano perse.

Fasi successive

- the section called "Lavorare con Argo CD"- Scopri come creare e gestire applicazioni Argo CD
- the section called "Risoluzione dei problemi"- Risolvi i problemi relativi ad Argo CD
- the section called "Lavorare con le funzionalità"- Gestite la vostra risorsa dedicata alle funzionalità di Argo CD

Risolvi i problemi relativi alle funzionalità di Argo CD

Questo argomento fornisce linee guida per la risoluzione dei problemi del CD EKS Capability for Argo, compresi i controlli dello stato delle funzionalità, i problemi di sincronizzazione delle applicazioni, l'autenticazione del repository e le implementazioni multi-cluster.

> **Note**
>
> Le funzionalità EKS sono completamente gestite ed eseguite all'esterno del cluster. Non avete accesso ai log del server Argo CD o al namespace. `argocd` La risoluzione dei problemi si concentra sullo stato delle funzionalità, sullo stato dell'applicazione e sulla configurazione.

La funzionalità è ATTIVA ma le applicazioni non si sincronizzano

Se la funzionalità Argo CD mostra `ACTIVE` lo stato ma le applicazioni non si sincronizzano, controlla lo stato della funzionalità e lo stato dell'applicazione.

Verifica lo stato della funzionalità:

È possibile visualizzare i problemi relativi allo stato e allo stato delle funzionalità nella console EKS o utilizzando la AWS CLI.

Console:

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Osservabilità.
4. Scegli Monitora cluster.
5. Scegli la scheda Capacità per visualizzare lo stato e lo stato di tutte le funzionalità.

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
```

Cause comuni:

- Repository non configurato: repository Git non aggiunto al CD Argo
- Autenticazione fallita: chiave SSH, token o credenziali non validi CodeCommit
- Applicazione non creata: non esistono risorse applicative nel cluster
- Politica di sincronizzazione: è richiesta la sincronizzazione manuale (la sincronizzazione automatica non è abilitata)
- Autorizzazioni IAM: autorizzazioni mancanti per il nostro Secrets CodeCommit Manager

Controlla lo stato dell'applicazione:

```
# 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}'
```

Verifica le condizioni della domanda:

```
# 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}'
```

Applicazioni bloccate nello stato «In corso»

Se un'applicazione viene visualizzata `Progressing` ma non arriva mai`Healthy`, controlla lo stato delle risorse e gli eventi dell'applicazione.

Controlla lo stato delle risorse:

```
# 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"
```

Cause comuni:

- Implementazione non pronta: i pod non si avviano o le sonde di prontezza non funzionano
- Dipendenze tra le risorse: risorse in attesa che altre risorse siano pronte
- Errori di estrazione delle immagini: le immagini dei contenitori non sono accessibili
- Risorse insufficienti: il cluster non dispone di CPU o memoria per i pod

Verifica la configurazione del cluster di destinazione (per configurazioni a più cluster):

```
# 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
```

Errori di autenticazione del repository

Se Argo CD non può accedere ai tuoi repository Git, verifica la configurazione di autenticazione.

Per CodeCommit i repository:

Verifica che IAM Capability Role disponga delle CodeCommit autorizzazioni:

```
# 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
```

Il ruolo richiede l'`codecommit:GitPull`autorizzazione per i repository.

Per i repository Git privati:

Verifica che le credenziali del repository siano configurate correttamente:

```
# Check repository secret exists
kubectl get secret -n argocd repo-secret-name -o yaml
```

Assicurati che il segreto contenga le credenziali di autenticazione corrette (chiave SSH, token o nome utente/password).

Per i repository che utilizzano 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
```

Problemi di distribuzione su più cluster

Se le applicazioni non vengono distribuite su cluster remoti, verifica la registrazione del cluster e la configurazione dell'accesso.

Controlla la registrazione del 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
```

Assicurati che il `server` campo contenga l'ARN del cluster EKS, non l'URL dell'API Kubernetes.

Verifica l'accesso al cluster di destinazione:

Sul cluster di destinazione, verifica che Argo CD Capability Role abbia un Access Entry:

```
# 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
```

Controlla le autorizzazioni IAM per più account:

Per le distribuzioni tra account, verifica che Argo CD Capability Role abbia un Access Entry sul cluster di destinazione. La funzionalità gestita utilizza EKS Access Entries per l'accesso su più account, non l'assunzione di ruoli IAM.

Per ulteriori informazioni sulla configurazione multi-cluster, consulta. the section called "Registra i cluster"

Fasi successive

- the section called "Considerazioni su Argo CD"- Considerazioni e best practice su Argo CD

- the section called "Lavorare con Argo CD"- Crea e gestisci applicazioni Argo CD
- the section called "Registra i cluster"- Configurazione di implementazioni multi-cluster
- the section called "Risoluzione dei problemi relativi alle funzionalità"- Guida generale alla risoluzione dei problemi relativi alle funzionalità

Confronto tra la funzionalità EKS per Argo CD e Argo CD autogestita

EKS Capability for Argo CD offre un'esperienza Argo CD completamente gestita che può essere eseguita in EKS. Per un confronto generale tra EKS Capabilities e soluzioni autogestite, vedi. the section called "Considerazioni" Questo argomento si concentra sulle differenze specifiche di Argo CD, tra cui l'autenticazione, la gestione di più cluster e il supporto delle funzionalità upstream.

Differenze rispetto alla versione upstream di Argo CD

EKS Capability for Argo CD si basa sull'upstream Argo CD, ma differisce nel modo in cui è accessibile, configurato e integrato con i servizi. AWS

RBAC e autenticazione: la funzionalità include tre ruoli RBAC (amministratore, editor, visualizzatore) e utilizza AWS Identity Center per l'autenticazione anziché l'autenticazione integrata di Argo CD. Configura le mappature dei ruoli tramite il `rbacRoleMapping` parametro della funzionalità per mappare i gruppi di Identity Center ai ruoli di Argo CD, non tramite Argo CD. `argocd-rbac-cm` ConfigMap L'interfaccia utente di Argo CD è ospitata con un proprio URL diretto (lo trovi nella console EKS nella scheda Capacità del cluster) e l'accesso all'API utilizza l' AWS autenticazione e l'autorizzazione tramite IAM.

Configurazione del cluster: la funzionalità non configura automaticamente il cluster o le hub-and-spoke topologie locali. È possibile configurare i cluster di destinazione della distribuzione e le voci di accesso EKS. La funzionalità supporta solo i cluster Amazon EKS come obiettivi di distribuzione utilizzando il cluster EKS ARNs (non il server API Kubernetes). URLs La funzionalità non aggiunge automaticamente il cluster locale (`kubernetes.default.svc`) come obiettivo di distribuzione: per eseguire la distribuzione nello stesso cluster in cui viene creata la funzionalità, è necessario registrare esplicitamente quel cluster utilizzando il relativo ARN.

Accesso remoto semplificato ai cluster: la funzionalità semplifica le implementazioni multi-cluster utilizzando EKS Access Entries per concedere l'accesso Argo CD ai cluster remoti, eliminando la necessità di configurare IAM Roles for Service Accounts (IRSA) o impostare ipotesi di ruoli IAM tra account. La funzionalità fornisce inoltre un accesso trasparente a cluster EKS completamente

privati senza richiedere il peering VPC o una configurazione di rete specializzata AWS : gestisce automaticamente la connettività tra la funzionalità Argo CD e i cluster remoti privati.

Integrazione diretta dei AWS servizi: la funzionalità fornisce l'integrazione diretta con i AWS servizi tramite le autorizzazioni IAM di Capability Role. È possibile fare riferimento ai CodeCommit repository, ai grafici ECR Helm e CodeConnections direttamente nelle risorse dell'applicazione senza creare configurazioni di repository. Ciò semplifica l'autenticazione ed elimina la necessità di gestire credenziali separate per i servizi. AWS Per informazioni dettagliate, vedi the section called "Configura i repository".

Supporto per lo spazio dei nomi: la funzionalità richiede di specificare un unico spazio dei nomi in cui creare l'applicazione Argo CD e le risorse personalizzate ApplicationSet. AppProject

> **Note**
>
> Questa restrizione dello spazio dei nomi si applica solo alle risorse personalizzate di Argo CD (Application,,). ApplicationSet AppProject I carichi di lavoro delle applicazioni possono essere distribuiti su qualsiasi namespace in qualsiasi cluster di destinazione. Ad esempio, se si crea la funzionalità con lo spazio dei nomi`argocd`, tutte le applicazioni CRs devono essere create nello spazio dei `argocd` nomi, ma tali applicazioni possono distribuire carichi di lavoro in,, o qualsiasi altro spazio dei nomi. `default` `production` `staging`

> **Note**
>
> La funzionalità gestita ha requisiti specifici per l'utilizzo e AppProject la configurazione della CLI:
>
> - Quando si utilizza l'Argo CD CLI, specificare le applicazioni con il prefisso dello spazio dei nomi: `argocd app sync namespace/appname`
> - AppProject le risorse devono specificare `.spec.sourceNamespaces` per definire quali namespace il progetto può controllare per le applicazioni (in genere impostate sullo spazio dei nomi specificato durante la creazione della funzionalità)
> - Le annotazioni di tracciamento delle risorse utilizzano il formato: `namespace_appname:group/kind:namespace/name`

Funzionalità non supportate: le seguenti funzionalità non sono disponibili nella funzionalità gestita:

- Plugin di gestione della configurazione (CMPs) per la generazione di manifest personalizzati
- Script Lua personalizzati per la valutazione dello stato delle risorse (sono supportati i controlli integrati per le risorse standard)
- Il controller delle notifiche
- Provider SSO personalizzati (è supportato solo AWS Identity Center, inclusa l'identità federata di terze parti tramite AWS Identity Center)
- Estensioni dell'interfaccia utente e banner personalizzati
- Accesso diretto e altre configurazioni `argocd-cm` `argocd-params` ConfigMaps
- Modifica del timeout di sincronizzazione (fissato a 120 secondi)

Compatibilità: le applicazioni ApplicationSets funzionano in modo identico alla versione upstream di Argo CD senza modifiche ai manifesti. La funzionalità utilizza lo stesso Kubernetes APIs e CRDs quindi strumenti come quelli funzionano allo stesso modo. `kubectl` La funzionalità supporta pienamente applicazioni e GitOps flussi di lavoro con sincronizzazione automatica ApplicationSets, implementazioni multi-cluster, policy di sincronizzazione (automatizzate, prune, self-heal), wave e hook di sincronizzazione, valutazione dello stato di salute per risorse Kubernetes standard, funzionalità di rollback, sorgenti di repository Git (HTTPS e SSH), Helm, Kustomize e semplici manifesti YAML, credenziali delle app, progetti per multi-tenancy ed esclusioni e inclusioni di risorse usioni. GitHub

Utilizzo dell'Argo CD CLI con funzionalità gestite

L'Argo CD CLI funziona allo stesso modo dell'Argo CD upstream per la maggior parte delle operazioni, ma l'autenticazione e la registrazione del cluster sono diverse.

Prerequisiti

Installa l'Argo CD CLI seguendo le istruzioni di installazione upstream.

Configurazione

Configura la CLI utilizzando le variabili di ambiente:

1. Ottieni l'URL del server Argo CD dalla console EKS (nella scheda Capacità del cluster) o utilizzando la AWS CLI. Il `https://` prefisso deve essere rimosso:

```
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 | sed 's|^https://||')
```

2. Genera un token di account dall'interfaccia utente di Argo CD (Impostazioni → Account → amministratore → Genera nuovo token), quindi impostalo come variabile di ambiente:

```
export ARGOCD_AUTH_TOKEN="your-token-here"
```

> **Important**
>
> Questa configurazione utilizza il token dell'account di amministrazione per i flussi di lavoro iniziali di configurazione e sviluppo. Per i casi d'uso in produzione, utilizza ruoli e token relativi al progetto per seguire il principio del privilegio minimo. Per ulteriori informazioni sulla configurazione dei ruoli di progetto e dell'RBAC, vedere. the section called "Configurazione delle autorizzazioni "

1. Imposta l'opzione gRPC richiesta:

```
export ARGOCD_OPTS="--grpc-web"
```

Con queste variabili di ambiente impostate, è possibile utilizzare l'Argo CD CLI senza `argocd login` il comando.

Differenze principali

La funzionalità gestita presenta le seguenti limitazioni CLI:

- `argocd admin` i comandi non sono supportati (richiedono l'accesso diretto al pod)
- `argocd login` non è supportato (usa invece i token dell'account o del progetto)
- `argocd cluster add` richiede il `--aws-cluster-name` flag con l'ARN del cluster EKS

Esempio: registrare un cluster

Registra un cluster EKS per la distribuzione delle applicazioni:

```
# 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
```

Per la documentazione completa della CLI di Argo CD, consultate il riferimento alla CLI di Argo CD.

Percorso di migrazione

È possibile migrare da Argo CD autogestito alla funzionalità gestita:

1. Controllate la configurazione attuale di Argo CD per verificare la presenza di funzionalità non supportate (controller di notifica, controlli di integrità personalizzati CMPs, estensioni dell'interfaccia utente)
2. Ridimensiona i controller Argo CD autogestiti fino a zero repliche per evitare conflitti
3. Crea una risorsa con funzionalità Argo CD sul tuo cluster
4. Esporta le tue applicazioni esistenti ApplicationSets e AppProjects
5. Esegui la migrazione delle credenziali del repository, dei segreti del cluster e dei modelli di credenziali di repository (record)
6. Se utilizzi chiavi GPG, certificati TLS o host noti SSH, migra anche queste configurazioni
7. Aggiorna `destination.server` i campi per utilizzare i nomi dei cluster o i cluster EKS ARNs
8. Applicali all'istanza Argo CD gestita
9. Verifica che le applicazioni si sincronizzino correttamente
10. Disattiva l'installazione di Argo CD autogestita

La funzionalità gestita utilizza lo stesso CD di Argo APIs e le stesse definizioni di risorse, quindi i manifesti esistenti funzionano con modifiche minime.

Fasi successive

- the section called "Crea una funzionalità Argo CD"- Create una risorsa con funzionalità Argo CD
- the section called "Lavorare con Argo CD"- Implementa la tua prima applicazione
- the section called "Considerazioni su Argo CD"- Configurazione AWS dell'integrazione con Identity Center

Composizione delle risorse con kro (Kube Resource Orchestrator)

kro (Kube Resource Orchestrator) è un progetto open source nativo di Kubernetes che consente di definire Kubernetes personalizzati utilizzando una configurazione semplice e diretta. APIs Con kro, puoi configurare facilmente nuove funzionalità personalizzate che creano un gruppo di oggetti Kubernetes e le operazioni logiche tra di essi. APIs

Con EKS Capabilities, kro è completamente gestito da AWS, eliminando la necessità di installare, mantenere e scalare i controller kro sui cluster.

Come funziona kro

kro introduce una Custom Resource Definition (CRD) chiamata `ResourceGraphDefinition` (RGD) che consente la creazione semplice e semplificata di Kubernetes personalizzati. APIs Quando crei un file`ResourceGraphDefinition`, kro utilizza estensioni Kubernetes native per crearne e gestirne di nuove nel tuo cluster. APIs A partire da questa specifica di singola risorsa, kro creerà e registrerà per te un nuovo CRD in base alle tue specifiche e si adatterà per gestire le tue risorse personalizzate appena definite.

RGDs può includere più risorse e kro determinerà le interdipendenze e l'ordine delle risorse, quindi non è necessario. È possibile utilizzare una sintassi semplice per inserire la configurazione da una risorsa all'altra, semplificando notevolmente le composizioni ed eliminando la necessità di operatori «collanti» nel cluster. Con kro, le tue risorse personalizzate possono includere risorse Kubernetes native e qualsiasi definizione di risorsa personalizzata () installata nel cluster. CRDs

kro supporta un singolo tipo di risorsa principale:

- ResourceGraphDefinition (RGD): definisce una risorsa personalizzata Kubernetes, incapsulando una o più risorse Kubernetes native o personalizzate sottostanti

Oltre a questa risorsa, kro creerà e gestirà il ciclo di vita delle risorse personalizzate create con essa, nonché tutte le risorse che le costituiscono.

kro si integra perfettamente con AWS Controllers for Kubernetes (ACK), consentendoti di comporre risorse per il carico di lavoro con risorse per creare astrazioni di livello superiore. AWS Ciò consente di creare i propri elementi costitutivi del cloud, semplificando la gestione delle risorse e abilitando modelli riutilizzabili con impostazioni di configurazione predefinite e immutabili basate sugli standard organizzativi.

Vantaggi di kro

kro consente ai team della piattaforma di creare Kubernetes personalizzati APIs che compongono più risorse in astrazioni di livello superiore. Ciò semplifica la gestione delle risorse consentendo agli sviluppatori di implementare applicazioni complesse utilizzando risorse personalizzate semplici, standardizzate e con versioni diverse. Definisci modelli riutilizzabili per combinazioni di risorse comuni, consentendo una creazione coerente di risorse in tutta l'organizzazione.

kro utilizza il Common Expression Language (CEL) in Kubernetes per trasferire valori tra le risorse e incorporare la logica condizionale, offrendo flessibilità nella composizione delle risorse. Puoi comporre sia le risorse Kubernetes che le AWS risorse gestite da ACK in modo unificato e personalizzato, abilitando definizioni complete di applicazioni e infrastrutture. APIs

kro supporta la configurazione dichiarativa tramite i manifesti di Kubernetes, abilitando i GitOps flussi di lavoro e l'infrastruttura come pratiche di codice che si integrano perfettamente con i processi di sviluppo esistenti. Come parte di EKS Managed Capabilities, kro è completamente gestito da AWS, eliminando la necessità di installare, configurare e mantenere i controller kro sui cluster.

Esempio: creazione di un ResourceGraphDefinition

L'esempio seguente mostra una procedura semplice `ResourceGraphDefinition` che crea un'applicazione Web con una distribuzione e un servizio:

```
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 gli utenti creano istanze della risorsa `WebApplication` personalizzata, kro crea automaticamente le risorse di distribuzione e servizio corrispondenti, gestendone il ciclo di vita insieme alla risorsa personalizzata.

Integrazione con altre funzionalità gestite da EKS

kro si integra con altre funzionalità gestite da EKS.

- AWS Controllers for Kubernetes (ACK): usa kro per comporre le risorse ACK in astrazioni di livello superiore, semplificando la gestione delle risorse. AWS
- Argo CD: utilizza Argo CD per gestire l'implementazione di risorse personalizzate kro su più cluster, abilitando flussi di lavoro per gli elementi costitutivi della piattaforma e gli stack di applicazioni. GitOps

Guida introduttiva a kro

Per iniziare a usare EKS Capability for kro:

1. Crea una risorsa con funzionalità kro sul tuo cluster EKS tramite la AWS console, la AWS CLI o la tua infrastruttura preferita come strumento di codice.
2. Crea ResourceGraphDefinitions (RGDs) che definisce le tue composizioni personalizzate APIs e di risorse.

3. Applica istanze delle tue risorse personalizzate per fornire e gestire Kubernetes e risorse sottostanti. AWS

Crea una funzionalità kro

Questo argomento spiega come creare una funzionalità kro sul tuo cluster Amazon EKS.

Prerequisiti

Prima di creare una funzionalità kro, assicurati di avere:

- Un cluster Amazon EKS esistente che esegue una versione di Kubernetes supportata (sono supportate tutte le versioni con supporto standard ed esteso)
- Autorizzazioni IAM sufficienti per creare risorse di funzionalità sui cluster EKS
- (Per CLI/EksCtl) Lo strumento CLI appropriato installato e configurato

> **Note**
>
> A differenza di ACK e Argo CD, kro non richiede autorizzazioni IAM aggiuntive oltre alla policy di fiducia. kro opera interamente all'interno del cluster e non effettua chiamate API. AWS Tuttavia, è comunque necessario fornire a un IAM Capability Role la policy di fiducia appropriata. Per informazioni sulla configurazione delle autorizzazioni RBAC di Kubernetes per kro, consulta. the section called “Configurazione delle autorizzazioni ”

Scegli il tuo strumento

Puoi creare una funzionalità kro usando Console di gestione AWS, AWS CLI o eksctl:

- the section called “Console”- Usa la console per un'esperienza guidata
- the section called “ AWS CLI”- Usa la AWS CLI per lo scripting e l'automazione
- the section called “eksctl”- Usa eksctl per un'esperienza nativa di Kubernetes

Cosa succede quando crei una funzionalità kro

Quando crei una funzionalità kro:

1. EKS crea il servizio di capacità kro e lo configura per monitorare e gestire le risorse nel cluster
2. Le definizioni di risorse personalizzate (CRDs) sono installate nel cluster
3. Viene creata automaticamente una voce di accesso per il tuo IAM Capability Role `AmazonEKSKROPolicy` che concede le autorizzazioni per la gestione ResourceGraphDefinitions e le relative istanze (vedi) the section called "Considerazioni sulle funzionalità EKS"
4. La funzionalità presuppone l'IAM Capability Role fornito dall'utente (utilizzato solo per la relazione di fiducia)
5. kro inizia a controllare `ResourceGraphDefinition` le risorse e le relative istanze
6. Lo stato della capacità cambia da a `CREATING` `ACTIVE`

Una volta attivo, è possibile creare o ResourceGraphDefinitions definire modelli personalizzati APIs e crearne APIs delle istanze.

> **Note**
>
> La voce di accesso creata automaticamente include la voce `AmazonEKSKROPolicy` che concede a kro le autorizzazioni di gestione ResourceGraphDefinitions e le relative istanze. Per consentire a kro di creare le risorse Kubernetes sottostanti definite nell'utente ResourceGraphDefinitions (come le risorse Deployments, Services o ACK), è necessario configurare politiche di accesso aggiuntive. Per ulteriori informazioni sulle voci di accesso e su come configurare autorizzazioni aggiuntive, consulta e. the section called "Configurazione delle autorizzazioni " the section called "Considerazioni sulle funzionalità EKS"

Fasi successive

Dopo aver creato la funzionalità kro:

- the section called "concetti kro"- Comprendi i concetti di kro e la composizione delle risorse
- the section called "concetti kro"- Scopri SimpleSchema le espressioni CEL e i modelli di composizione delle risorse

Crea una funzionalità kro utilizzando la console

Questo argomento descrive come creare una funzionalità kro (Kube Resource Orchestrator) utilizzando. Console di gestione AWS

Crea la funzionalità kro

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.
2. Seleziona il nome del cluster per aprire la pagina dei dettagli del cluster.
3. Scegli la scheda Funzionalità.
4. Nella barra di navigazione a sinistra, scegli kro (Kube Resource Orchestrator).
5. Scegli la funzionalità Create kro.
6. Per IAM Capability Role:
   - Se disponi già di un IAM Capability Role, selezionalo dal menu a discesa
   - Se devi creare un ruolo, scegli Create kro role

   Questo apre la console IAM in una nuova scheda con policy di fiducia precompilate. Il ruolo non richiede autorizzazioni IAM aggiuntive poiché kro opera interamente all'interno del tuo cluster.

   Dopo aver creato il ruolo, torna alla console EKS e il ruolo verrà selezionato automaticamente.

   > **Note**
   >
   > A differenza di ACK e Argo CD, kro non richiede autorizzazioni IAM aggiuntive oltre alla politica di fiducia. kro opera interamente all'interno del cluster e non effettua chiamate API. AWS
7. Scegli Create (Crea).

Inizia il processo di creazione delle funzionalità.

Verifica che la funzionalità sia attiva

1. Nella scheda Funzionalità, visualizza lo stato delle capacità di kro.
2. Attendi che lo stato cambi da `CREATING` a`ACTIVE`.
3. Una volta attiva, la funzionalità è pronta per l'uso.

Per informazioni sullo stato delle funzionalità e sulla risoluzione dei problemi, vederethe section called "Lavorare con le funzionalità".

Concedi le autorizzazioni per gestire le risorse Kubernetes

Quando crei una funzionalità kro, viene creato automaticamente un EKS Access Entry con`AmazonEKSKROPolicy`, che consente a kro di gestire e le relative istanze. ResourceGraphDefinitions Tuttavia, per impostazione predefinita non viene concessa alcuna autorizzazione per creare le risorse Kubernetes sottostanti (come Deployments, Services, ecc.) definite nel tuo. ConfigMaps ResourceGraphDefinitions

Questa progettazione intenzionale segue il principio del privilegio minimo: diversi richiedono autorizzazioni diverse. ResourceGraphDefinitions Devi configurare in modo esplicito le autorizzazioni o le esigenze in base alle risorse che gestirai. ResourceGraphDefinitions

Per iniziare rapidamente, in ambienti di test o sviluppo, usa: `AmazonEKSClusterAdminPolicy`

1. Nella console EKS, accedi alla scheda Accesso del cluster.
2. In Voci di accesso, trova la voce relativa al tuo ruolo kro capability (avrà il ruolo ARN che hai creato in precedenza).
3. Scegli la voce di accesso per aprirne i dettagli.
4. Nella sezione Politiche di accesso, scegli Associa politica di accesso.
5. Seleziona `AmazonEKSClusterAdminPolicy` dall'elenco delle politiche.
6. Per l'ambito di accesso, seleziona Cluster.
7. Selezionare Associate (Associa).

> **Important**
>
> `AmazonEKSClusterAdminPolicy`Concede ampie autorizzazioni per creare e gestire tutte le risorse Kubernetes, inclusa la possibilità di creare qualsiasi tipo di risorsa in tutti i namespace. Questa funzionalità è utile per lo sviluppo, ma non deve essere utilizzata in produzione POCs . Per la produzione, crea politiche RBAC personalizzate che concedano solo le autorizzazioni necessarie per le ResourceGraphDefinitions risorse specifiche che gestirai. Per indicazioni sulla configurazione delle autorizzazioni con privilegi minimi, consulta e. the section called “Configurazione delle autorizzazioni ” the section called “Considerazioni sulle funzionalità EKS”

Verifica che siano disponibili risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate kro siano disponibili nel tuo cluster.

Utilizzo della console

1. Accedi al tuo cluster nella console Amazon EKS
2. Scegli la scheda Risorse
3. Scegli Estensioni
4. Scegliere CustomResourceDefinitions

Dovresti vedere il tipo di `ResourceGraphDefinition` risorsa elencato.

Usare kubectl

```
kubectl api-resources | grep kro.run
```

Dovresti vedere il tipo di `ResourceGraphDefinition` risorsa elencato.

Fasi successive

- the section called "concetti kro"- Comprendi i concetti kro e la composizione delle risorse
- the section called "concetti kro"- Scopri SimpleSchema le espressioni CEL e i modelli di composizione
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa kro Capability

Crea una funzionalità kro utilizzando la CLI AWS

Questo argomento descrive come creare una funzionalità kro (Kube Resource Orchestrator) utilizzando la CLI. AWS

Prerequisiti

- AWS CLI: versione `2.12.3` o successiva. Per verificare la tua versione, `aws --version` esegui. Per ulteriori informazioni, consulta la Guida per l'utente all'installazione nell'interfaccia a riga di AWS comando.

- **`kubectl`**— Uno strumento da riga di comando per lavorare con i cluster Kubernetes. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".

Fase 1: creazione di un ruolo IAM Capability

Crea un file di policy di fiducia:

```
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
```

Crea il ruolo IAM:

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

> **Note**
>
> A differenza di ACK e Argo CD, kro non richiede autorizzazioni IAM aggiuntive. kro opera interamente all'interno del cluster e non effettua chiamate API. AWS Il ruolo è necessario solo per stabilire un rapporto di fiducia con il servizio EKS Capabilities.

Fase 2: Creare la funzionalità kro

Crea la risorsa di funzionalità kro sul tuo cluster. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster (ad esempious-west-2) e *my-cluster* con il nome del 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
```

Il comando viene restituito immediatamente, ma la funzionalità impiega del tempo per diventare attiva poiché EKS crea l'infrastruttura e i componenti di funzionalità richiesti. EKS installerà le Kubernetes Custom Resource Definitions relative a questa funzionalità nel cluster non appena viene creata.

> **Note**
>
> Se ricevi un errore che indica che il cluster non esiste o non disponi delle autorizzazioni, verifica:
>
> - Il nome del cluster è corretto
> - La tua AWS CLI è configurata per la regione corretta
> - Disponi delle autorizzazioni IAM richieste

Fase 3: Verifica che la funzionalità sia attiva

Attendi che la funzionalità diventi attiva. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster e sostituiscilo *my-cluster* con il nome del cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --query 'capability.status' \
  --output text
```

La funzionalità è pronta quando viene visualizzato lo statoACTIVE.

Puoi anche visualizzare i dettagli completi delle funzionalità:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro
```

Passaggio 4: concedere le autorizzazioni per gestire le risorse Kubernetes

Quando crei una funzionalità kro, viene creato automaticamente un EKS Access Entry conAmazonEKSKROPolicy, che consente a kro di gestire e le relative istanze. ResourceGraphDefinitions Tuttavia, per impostazione predefinita non viene concessa alcuna autorizzazione per creare le risorse Kubernetes sottostanti (come Deployments, Services, ecc.) definite nel tuo. ConfigMaps ResourceGraphDefinitions

Questa progettazione intenzionale segue il principio del privilegio minimo: diversi richiedono autorizzazioni diverse. ResourceGraphDefinitions Ad esempio: * A ResourceGraphDefinition che crea solo risorse ACK ConfigMaps e Secrets necessita di autorizzazioni diverse rispetto a una che crea distribuzioni e servizi * A ResourceGraphDefinition che crea risorse ACK necessita delle autorizzazioni per quelle risorse personalizzate specifiche * Alcuni potrebbero ResourceGraphDefinitions leggere solo le risorse esistenti senza crearne di nuove

È necessario configurare in modo esplicito le autorizzazioni o le esigenze in base alle risorse che gestirai. ResourceGraphDefinitions

Configurazione rapida

Per iniziare rapidamente, in ambienti di test o sviluppo, usa: AmazonEKSClusterAdminPolicy

Ottieni il ruolo di capacità ARN:

```
CAPABILITY_ROLE_ARN=$(aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --query 'capability.roleArn' \
  --output text)
```

Associa la politica di amministrazione del 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

`AmazonEKSClusterAdminPolicy`Concede ampie autorizzazioni per creare e gestire tutte le risorse Kubernetes, inclusa la possibilità di creare qualsiasi tipo di risorsa in tutti i namespace. Questa funzionalità è utile per lo sviluppo, ma non deve essere utilizzata in produzione POCs . Per la produzione, crea politiche RBAC personalizzate che concedano solo le autorizzazioni necessarie per le ResourceGraphDefinitions risorse specifiche che gestirai. Per indicazioni sulla configurazione delle autorizzazioni con privilegi minimi, consulta e. the section called "Configurazione delle autorizzazioni " the section called "Considerazioni sulle funzionalità EKS"

Passaggio 5: Verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate kro siano disponibili nel tuo cluster:

```
kubectl api-resources | grep kro.run
```

Dovresti vedere il tipo di `ResourceGraphDefinition` risorsa elencato.

Fasi successive

- the section called "concetti kro"- Comprendi i concetti kro e la composizione delle risorse
- the section called "concetti kro"- Scopri SimpleSchema le espressioni CEL e i modelli di composizione
- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa kro Capability

Crea una funzionalità kro usando eksctl

Questo argomento descrive come creare una funzionalità kro (Kube Resource Orchestrator) usando eksctl.

> **Note**
>
> I seguenti passaggi richiedono la versione eksctl o successiva. `0.220.0` Per verificare la tua versione, esegui. `eksctl version`

Fase 1: Creare un ruolo di capacità IAM

Crea un file di policy di fiducia:

```
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
```

Crea il ruolo IAM:

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

> **Note**
>
> A differenza di ACK e Argo CD, kro non richiede autorizzazioni IAM aggiuntive oltre alla policy di fiducia. kro opera interamente all'interno del cluster e non effettua chiamate API. AWS

Fase 2: Creare la funzionalità kro

Crea la funzionalità kro usando eksctl. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster e sostituiscilo *my-cluster* con il nome del cluster.

```
eksctl create capability \
  --region region-code \
  --cluster my-cluster \
  --name my-kro \
  --type KRO \
  --role-arn arn:aws:iam::[.replaceable]111122223333:role/KROCapabilityRole
```

Il comando viene restituito immediatamente, ma la funzionalità impiega del tempo per diventare attiva.

Fase 3: Verificare che la funzionalità sia attiva

Verifica lo stato della capacità. Sostituiscilo *region-code* con la AWS regione in cui si trova il cluster e sostituiscilo *my-cluster* con il nome del cluster.

```
eksctl get capability \
  --region region-code \
  --cluster my-cluster \
  --name my-kro
```

La funzionalità è pronta quando viene visualizzato lo statoACTIVE.

Passaggio 4: concedere le autorizzazioni per gestire le risorse Kubernetes

Per impostazione predefinita, kro può solo creare e gestire le relative istanze. ResourceGraphDefinitions Per consentire a kro di creare e gestire le risorse Kubernetes sottostanti definite nel tuo ResourceGraphDefinitions, associa la policy di accesso alla voce di `AmazonEKSClusterAdminPolicy` accesso della funzionalità.

Ottieni il ruolo di capacità ARN:

```
CAPABILITY_ROLE_ARN=$(aws eks describe-capability \
  --region region-code \
  --cluster my-cluster \
  --name my-kro \
  --query 'capability.roleArn' \
  --output text)
```

Associa la politica di amministrazione del 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**
>
> `AmazonEKSClusterAdminPolicy`Concede ampie autorizzazioni per creare e gestire tutte le risorse Kubernetes e ha lo scopo di semplificare l'avvio. Per l'uso in produzione, crea politiche RBAC più restrittive che concedano solo le autorizzazioni necessarie per le risorse specifiche che gestirai. ResourceGraphDefinitions Per indicazioni sulla configurazione delle autorizzazioni con privilegi minimi, consulta e. the section called "Configurazione delle autorizzazioni " the section called "Considerazioni sulle funzionalità EKS"

Passaggio 5: Verifica della disponibilità di risorse personalizzate

Dopo che la funzionalità è attiva, verifica che le risorse personalizzate kro siano disponibili nel tuo cluster:

```
kubectl api-resources | grep kro.run
```

Dovresti vedere il tipo di `ResourceGraphDefinition` risorsa elencato.

Fasi successive

- the section called "concetti kro"- Comprendi i concetti kro e la composizione delle risorse
- the section called "concetti kro"- Scopri SimpleSchema le espressioni CEL e i modelli di composizione

- the section called "Lavorare con le funzionalità"- Gestisci la tua risorsa kro Capability

concetti kro

kro consente ai team della piattaforma di creare Kubernetes personalizzati APIs che compongono più risorse in astrazioni di livello superiore. Questo argomento illustra un esempio pratico, quindi spiega i concetti fondamentali da comprendere quando si lavora con EKS Capability for kro.

Guida introduttiva a kro

Dopo aver creato la funzionalità kro (vedithe section called "Crea una capacità kro"), puoi iniziare a crearne un APIs utilizzo personalizzato ResourceGraphDefinitions nel tuo cluster.

Ecco un esempio completo che crea una semplice astrazione di un'applicazione 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
```

Dopo averlo applicato ResourceGraphDefinition, i team applicativi possono creare applicazioni Web utilizzando la vostra API semplificata:

```
apiVersion: kro.run/v1alpha1
kind: WebApplication
metadata:
  name: my-app
spec:
  name: my-app
  replicas: 5
```

kro crea automaticamente Deployment and Service con la configurazione appropriata. Poiché `image` non è specificato, utilizza il valore predefinito `nginx:latest` dello schema.

Concetti principali

> **Important**
>
> kro esegue la convalida ResourceGraphDefinitions al momento della creazione, non in fase di esecuzione. Quando crei un RGD, kro convalida la sintassi CEL, controlla il tipo delle

espressioni rispetto agli schemi Kubernetes effettivi, verifica l'esistenza dei campi e rileva le dipendenze circolari. Ciò significa che gli errori vengono rilevati immediatamente quando si crea l'RGD, prima che qualsiasi istanza venga distribuita.

ResourceGraphDefinition

A ResourceGraphDefinition (RGD) definisce un'API Kubernetes personalizzata specificando:

- Schema: la struttura dell'API che utilizza il SimpleSchema formato (nomi di campo, tipi, impostazioni predefinite, convalida)
- Risorse: modelli per il Kubernetes sottostante o risorse da creare AWS
- Dipendenze: come le risorse si relazionano tra loro (rilevate automaticamente dai riferimenti di campo)

Quando applichi un RGD, kro registra una nuova Custom Resource Definition (CRD) nel tuo cluster. I team applicativi possono quindi creare istanze della vostra API personalizzata e kro si occupa della creazione e della gestione di tutte le risorse sottostanti.

Per ulteriori informazioni, consulta ResourceGraphDefinition Panoramica nella documentazione di kro.

SimpleSchema formato

SimpleSchema fornisce un modo semplificato per definire gli schemi API senza richiedere conoscenze su OpenAPI:

```
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
```

SimpleSchema supporta`string`, `integer`e `boolean` `number` tipi con vincoli come`required`,,/`default`, `minimum` e. `maximum` `enum` `pattern`

Per ulteriori informazioni, consulta la documentazione SimpleSchemadi kro.

espressioni CEL

kro utilizza il Common Expression Language (CEL) per fare riferimento ai valori in modo dinamico e aggiungere logica condizionale. Le espressioni CEL sono racchiuse `${ }` e possono essere utilizzate in due modi:

Espressioni autonome - L'intero valore del campo è una singola espressione:

```
spec:
  replicas: ${schema.spec.replicaCount}  # Expression returns integer
  labels: ${schema.spec.labelMap}        # Expression returns object
```

Il risultato dell'espressione sostituisce l'intero valore del campo e deve corrispondere al tipo previsto del campo.

Modelli di stringhe: una o più espressioni incorporate in una stringa:

```
metadata:
  name: "${schema.spec.prefix}-${schema.spec.name}"  # Multiple expressions
  annotation: "Created by ${schema.spec.owner}"      # Single expression in string
```

Tutte le espressioni nei modelli di stringhe devono restituire stringhe. Usa `string()` per convertire altri tipi:`"replicas-${string(schema.spec.count)}"`.

Riferimenti sui campi: accedi ai valori delle specifiche dell'istanza utilizzando`schema.spec`:

```
template:
  metadata:
    name: ${schema.spec.name}-deployment
    namespace: ${schema.metadata.namespace}  # Can also reference metadata
  spec:
    replicas: ${schema.spec.replicas}
```

Accesso facoltativo ai campi: da utilizzare `?` per campi che potrebbero non esistere:

```
# For ConfigMaps or Secrets with unknown structure
value: ${configmap.data.?DATABASE_URL}

# For optional status fields
ready: ${deployment.status.?readyReplicas > 0}
```

Se il campo non esiste, l'espressione restituisce `null` invece di fallire.

Risorse condizionali: includi le risorse solo quando sono soddisfatte le condizioni:

```
resources:
- id: ingress
  includeWhen:
    - ${schema.spec.enableIngress == true}
  template:
    # ... ingress configuration
```

Il `includeWhen` campo accetta un elenco di espressioni booleane. Tutte le condizioni devono essere vere affinché la risorsa possa essere creata. Attualmente, `includeWhen` può fare riferimento solo `schema.spec` ai campi.

Trasformazioni - Trasforma i valori utilizzando operatori e funzioni ternari:

```
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)}
```

Riferimenti tra risorse - Valori di riferimento provenienti da altre risorse:

```
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 si fa riferimento a un'altra risorsa in un'espressione CEL, si crea automaticamente una dipendenza. kro assicura che la risorsa referenziata venga creata per prima.

Per ulteriori informazioni, consulta CEL Expressions nella documentazione di kro.

Dipendenze tra le risorse

kro deduce automaticamente le dipendenze dalle espressioni CEL: non si specifica l'ordine, si descrivono le relazioni. Quando una risorsa fa riferimento a un'altra utilizzando un'espressione CEL, kro crea una dipendenza e determina l'ordine di creazione corretto.

```
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
```

L'espressione `${bucket.spec.name}` crea una dipendenza. kro crea un grafico aciclico diretto (DAG) di tutte le risorse e le relative dipendenze, quindi calcola un ordine topologico per la creazione.

Ordine di creazione: le risorse vengono create in ordine topologico (prima le dipendenze).

Creazione parallela: le risorse senza dipendenze vengono create contemporaneamente.

Ordine di eliminazione: le risorse vengono eliminate in ordine topologico inverso (prima le persone dipendenti).

Dipendenze circolari: non consentite: kro rifiuta con dipendenze circolari durante la convalida. ResourceGraphDefinitions

È possibile visualizzare l'ordine di creazione calcolato:

```
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.topologicalOrder}'
```

Per maggiori informazioni, vedi Inferenza del grafico nella documentazione di kro.

Composizione con ACK

kro funziona perfettamente con EKS Capability for ACK per comporre AWS risorse con risorse 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}
```

Questo modello consente di creare AWS risorse, estrarne i dettagli (ARNs URLs, endpoint) e inserirle nella configurazione dell'applicazione, il tutto gestito come una singola unità.

Per ulteriori schemi di composizione ed esempi avanzati, consulta. the section called "Considerazioni"

Fasi successive

- the section called "Considerazioni"- Scopri i pattern specifici di EKS, l'RBAC e l'integrazione con ACK e Argo CD
- kro Documentation - Documentazione kro completa che include espressioni CEL avanzate, modelli di convalida e risoluzione dei problemi

Configura le autorizzazioni kro

A differenza di ACK e Argo CD, kro non richiede autorizzazioni IAM. kro opera interamente all'interno del cluster Kubernetes e non effettua chiamate API. AWS Controlla l'accesso alle risorse kro utilizzando Kubernetes RBAC standard.

Come funzionano le autorizzazioni con kro

kro utilizza due tipi di risorse Kubernetes con ambiti diversi:

ResourceGraphDefinitions: risorse con ambito cluster che definiscono la personalizzazione. APIs In genere gestite dai team della piattaforma che progettano e gestiscono gli standard organizzativi.

Istanze: risorse personalizzate con ambito namespace create da. ResourceGraphDefinitions Può essere creato da team di applicazioni con autorizzazioni RBAC appropriate.

Per impostazione predefinita, la funzionalità kro dispone delle autorizzazioni per gestire le relative istanze tramite ResourceGraphDefinitions la politica di accesso. `AmazonEKSKROPolicy` Tuttavia, kro richiede autorizzazioni aggiuntive per creare e gestire le risorse Kubernetes sottostanti definite nell'utente ResourceGraphDefinitions (come Deployments, Services o risorse ACK). È necessario concedere queste autorizzazioni tramite le politiche di accesso o Kubernetes RBAC. Per i dettagli sulla concessione di queste autorizzazioni, consulta kro arbitrary resource permissions.

Autorizzazioni del team della piattaforma

I team della piattaforma necessitano delle autorizzazioni per creare e gestire. ResourceGraphDefinitions

Esempio ClusterRole per i team della piattaforma:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kro-platform-admin
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
  verbs: ["*"]
```

Associati ai membri del team della piattaforma:

```
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
```

Autorizzazioni del team dell'applicazione

I team applicativi necessitano delle autorizzazioni per creare istanze di risorse personalizzate nei propri namespace.

Esempio di ruolo per i team delle applicazioni:

```
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"]
```

Associarsi ai membri del team applicativo:

```
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**
>
> Il gruppo di API nel ruolo (`kro.run`in questo esempio) deve corrispondere a quello `apiVersion` definito nello schema ResourceGraphDefinition dell'utente.

Accesso in sola lettura

Concedi l'accesso in sola lettura alla visualizzazione ResourceGraphDefinitions e alle istanze senza autorizzazioni di modifica.

ClusterRoleSola lettura:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kro-viewer
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
```

```
  verbs: ["get", "list", "watch"]
```

Ruolo di sola lettura per le istanze:

```
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"]
```

Accesso a più namespace

Concedi ai team delle applicazioni l'accesso a più namespace utilizzando with. ClusterRoles RoleBindings

ClusterRole per l'accesso a più namespace:

```
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"]
```

Associa a namespace specifici:

```
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
```

Best practice

Principio del privilegio minimo: concedi solo le autorizzazioni minime necessarie per le responsabilità di ogni team.

Usa i gruppi anziché i singoli utenti: associa i ruoli ai gruppi anziché ai singoli utenti per una gestione più semplice.

Aspetti distinti relativi alla piattaforma e all'applicazione: i team della piattaforma gestiscono le istanze ResourceGraphDefinitions, i team delle applicazioni gestiscono le istanze.

Isolamento dello spazio dei nomi: utilizza i namespace per isolare team o ambienti diversi, con RBAC che controlla l'accesso a ciascun namespace.

Accesso in sola lettura per il controllo: Fornisci l'accesso in sola lettura ai team di sicurezza e conformità per scopi di controllo.

Fasi successive

- the section called “concetti kro”- Comprendi i concetti kro e la composizione delle risorse
- the section called “concetti kro”- Comprendi SimpleSchema, le espressioni CEL e i modelli di composizione
- the section called “Considerazioni sulle funzionalità EKS”- Esamina le migliori pratiche di sicurezza per quanto riguarda le funzionalità

considerazioni kro per EKS

Questo argomento tratta importanti considerazioni sull'utilizzo di EKS Capability per kro, tra cui quando utilizzare la composizione delle risorse, i modelli RBAC e l'integrazione con altre funzionalità EKS.

Quando usare kro

kro è progettato per creare modelli di infrastruttura riutilizzabili e personalizzati APIs che semplificano la gestione delle risorse complesse.

Usa kro quando hai bisogno di:

- Crea piattaforme self-service semplificate per i team applicativi APIs
- Standardizza i modelli di infrastruttura tra i team (database, backup e monitoraggio)
- Gestisci le dipendenze delle risorse e trasferisci i valori tra le risorse
- Crea astrazioni personalizzate che nascondono la complessità dell'implementazione
- Componi più risorse ACK in blocchi costitutivi di livello superiore
- Abilita i GitOps flussi di lavoro per stack di infrastrutture complessi

Non usare kro quando:

- Gestione di risorse semplici e autonome (usa direttamente le risorse ACK o Kubernetes)
- È necessaria una logica di runtime dinamica (kro è dichiarativo, non imperativo)
- Le risorse non hanno dipendenze o configurazioni condivise

kro eccelle nella creazione di «percorsi asfaltati», schemi ponderati e riutilizzabili che consentono ai team di implementare correttamente infrastrutture complesse.

Schemi RBAC

kro consente la separazione delle preoccupazioni tra i team della piattaforma che creano le istanze ResourceGraphDefinitions e i team delle applicazioni che creano le istanze.

Responsabilità del team di piattaforma

I team della piattaforma creano e gestiscono ResourceGraphDefinitions (RGDs) che definiscono la personalizzazione APIs.

Autorizzazioni necessarie:

- Crea, aggiorna, elimina ResourceGraphDefinitions
- Gestisci i tipi di risorse sottostanti (implementazioni, servizi, risorse ACK)
- Accesso a tutti i namespace in cui verranno utilizzati RGDs

Esempio ClusterRole per il team della piattaforma:

```
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"]
```

Per una configurazione RBAC dettagliata, vedere. the section called "Configurazione delle autorizzazioni "

Responsabilità del team di applicazione

I team applicativi creano istanze di risorse personalizzate definite da RGDs senza la necessità di comprenderne la complessità sottostante.

Autorizzazioni necessarie:

- Creare, aggiornare ed eliminare istanze di risorse personalizzate
- Accesso in lettura al loro namespace
- Nessun accesso alle risorse sottostanti o RGDs

Vantaggi:

- I team utilizzano strumenti semplici e di alto livello APIs
- I team della piattaforma controllano i dettagli dell'implementazione
- Riduzione del rischio di errori di configurazione

- Onboarding più rapido per i nuovi membri del team

Integrazione con altre funzionalità EKS

Composizione di risorse ACK

kro è particolarmente potente se combinato con ACK per creare modelli di infrastruttura.

Schemi comuni:

- Applicazione con archiviazione: bucket S3+coda SQS+configurazione delle notifiche
- Stack di database: istanza RDS + gruppo di parametri + gruppo di sicurezza + segreto Secrets Manager
- Rete: VPC + sottoreti + tabelle di routing + gruppi di sicurezza + gateway NAT
- Elaborazione con storage: EC2 istanza + volumi EBS + profilo di istanza IAM

kro gestisce l'ordine delle dipendenze, passa i valori tra le risorse (come le stringhe di connessione) ARNs e gestisce l'intero ciclo di vita come una singola unità.

Per esempi di composizione di risorse ACK, vedi. the section called “Concetti ACK”

GitOps con Argo CD

Usa il CD EKS Capability for Argo per distribuire entrambe le RGDs istanze dai repository Git.

Organizzazione del repository:

- Repo della piattaforma: contiene contenuti ResourceGraphDefinitions gestiti dal team della piattaforma
- Repo di applicazioni: contengono istanze di risorse personalizzate gestite dai team delle app
- Repo condiviso: contiene sia le istanze che le istanze per le RGDs organizzazioni più piccole

Considerazioni:

- Implementa RGDs prima delle istanze (le onde di sincronizzazione di Argo CD possono esserti utili)
- Utilizzate progetti Argo CD separati per i team di piattaforme e applicazioni
- Il team della piattaforma controlla l'accesso al repository RGD
- I team applicativi hanno accesso in sola lettura alle definizioni RGD

Per ulteriori informazioni su Argo CD, vedere. the section called "Lavorare con Argo CD"

Organizzando ResourceGraphDefinitions

RGDs Organizza per scopo, complessità e proprietà.

Per scopo:

- Infrastruttura: stack di database, rete, archiviazione
- Applicazione: app Web APIs, lavori in batch
- Piattaforma: servizi condivisi, monitoraggio, registrazione

Per complessità:

- Semplice: 2-3 risorse con dipendenze minime
- Moderato: 5-10 risorse con alcune dipendenze
- Complesso: più di 10 risorse con dipendenze complesse

Convenzioni di denominazione:

- Usa nomi descrittivi:, `webapp-with-database` `s3-notification-queue`
- Includi la versione nel nome per annullare le modifiche: `webapp-v2`
- Usa prefissi coerenti RGDs per: `platform-` `app-`

Strategia del namespace:

- RGDs sono con ambito cluster (non con namespace)
- Le istanze hanno uno spazio dei nomi
- Usa i selettori dello spazio dei nomi per controllare dove è possibile creare le istanze RGDs

Controllo delle versioni e aggiornamenti

Piano per l'evoluzione di RGD e la migrazione delle istanze.

Aggiornamenti RGD:

- Modifiche continue: aggiorna RGD in atto (aggiungi campi opzionali, nuove risorse con IncludeWhen)
- Ultime modifiche: crea un nuovo RGD con un nome diverso (webapp-v2)
- Deprecazione: contrassegna il vecchio con annotazioni, comunica la cronologia della migrazione RGDs

Migrazione delle istanze:

- Crea nuove istanze con RGD aggiornato
- Convalida il corretto funzionamento delle nuove istanze
- Elimina le vecchie istanze
- kro gestisce automaticamente gli aggiornamenti delle risorse sottostanti

Migliori pratiche:

- Verifica innanzitutto le modifiche RGD in ambienti non di produzione
- Usa il controllo semantico delle versioni nei nomi RGD per le modifiche principali
- Modifiche di interruzione dei documenti e percorsi di migrazione
- Fornisci esempi di migrazione per i team applicativi

Convalida e test

Convalida RGDs prima dell'implementazione in produzione.

Strategie di convalida:

- Convalida dello schema: kro convalida automaticamente la struttura RGD
- Istanze dry-run: crea istanze di test nei namespace di sviluppo
- Test di integrazione: verifica che le risorse composte funzionino insieme
- Applicazione delle politiche: utilizza i controller di ammissione per far rispettare gli standard organizzativi

Problemi comuni da testare:

- Dipendenze e ordinamento delle risorse

- Scambio di valore tra risorse (espressioni CEL)
- Inclusione condizionale delle risorse (includeWhen)
- Propagazione dello stato dalle risorse sottostanti
- Autorizzazioni RBAC, ad esempio la creazione

Documentazione upstream

Per informazioni dettagliate sull'uso di kro:

- Guida introduttiva a kro - Creazione ResourceGraphDefinitions
- Espressioni CEL - Scrittura di espressioni CEL
- kro Guides - Modelli di composizione avanzati
- Risoluzione dei problemi - Risoluzione dei problemi e debug

Fasi successive

- the section called "Configurazione delle autorizzazioni "- Configura RBAC per i team di piattaforme e applicazioni
- the section called "concetti kro"- Comprendi i concetti kro e il ciclo di vita delle risorse
- the section called "Risoluzione dei problemi"- Risolvi i problemi di kro
- the section called "Concetti ACK"- Scopri le risorse ACK per la composizione
- the section called "Lavorare con Argo CD"- Distribuzione RGDs e istanze con GitOps

Risolvi i problemi relativi alle funzionalità kro

Questo argomento fornisce linee guida per la risoluzione dei problemi di EKS Capability for kro, compresi i controlli dello stato delle capacità, le autorizzazioni RBAC, gli errori di espressione CEL e i problemi di composizione delle risorse.

> **Note**
>
> Le funzionalità EKS sono completamente gestite ed eseguite all'esterno del cluster. Non hai accesso ai log dei controller o al `kro-system` namespace. La risoluzione dei problemi si concentra sullo stato delle funzionalità, sulla configurazione RBAC e sullo stato delle risorse.

Le funzionalità sono ATTIVE ma ResourceGraphDefinitions non funzionano

Se la tua funzionalità kro mostra `ACTIVE` lo stato ma ResourceGraphDefinitions non stai creando risorse sottostanti, controlla lo stato della capacità, le autorizzazioni RBAC e lo stato delle risorse.

Verifica lo stato delle capacità:

È possibile visualizzare i problemi relativi allo stato e allo stato delle funzionalità nella console EKS o utilizzando la AWS CLI.

Console:

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Osservabilità.
4. Scegli Monitora cluster.
5. Scegli la scheda Capacità per visualizzare lo stato e lo stato di tutte le funzionalità.

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
```

Cause comuni:

- Autorizzazioni RBAC mancanti: kro non dispone delle autorizzazioni per creare le risorse Kubernetes sottostanti
- Espressioni CEL non valide: errori di sintassi in ResourceGraphDefinition
- Dipendenze dalle risorse: risorse dipendenti non pronte
- Convalida dello schema: l'istanza non soddisfa i requisiti dello schema RGD

Verifica le autorizzazioni RBAC:

```
# Check if capability has cluster admin policy
kubectl get accessentry -A | grep kro
```

Se la funzionalità non dispone delle autorizzazioni richieste, associala alla voce di accesso della `AmazonEKSClusterAdminPolicy` funzionalità kro o crea politiche RBAC più restrittive per l'uso in produzione. Per informazioni dettagliate, vedi the section called "Configurazione delle autorizzazioni ".

Verifica lo stato: 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}'
```

ResourceGraphDefinitions hanno tre condizioni di stato chiave:

- `ResourceGraphAccepted`- Se l'RGD ha superato la convalida (sintassi CEL, controllo del tipo, esistenza del campo)
- `KindReady`- Se il CRD per l'API personalizzata è stato generato e registrato
- `ControllerReady`- Se kro sta monitorando attivamente le istanze della tua API personalizzata

In caso `ResourceGraphAccepted` affermativo`False`, controllate il messaggio relativo alla condizione per eventuali errori di convalida, ad esempio campi sconosciuti, mancate corrispondenze tra i tipi o dipendenze circolari.

Istanze create ma le risorse sottostanti non vengono visualizzate

Se esistono istanze di risorse personalizzate ma le risorse Kubernetes sottostanti (Deployments, Services ConfigMaps) non vengono create, verifica che kro disponga delle autorizzazioni e verifica la presenza di errori di composizione.

Controlla lo stato dell'istanza:

```
# 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}'
```

Le istanze hanno un `state` campo che mostra lo stato di alto livello:

- `ACTIVE`- L'istanza viene eseguita correttamente
- `IN_PROGRESS`- L'istanza è in fase di elaborazione o riconciliazione
- `FAILED`- Impossibile riconciliare l'istanza
- `DELETING`- L'istanza viene eliminata
- `ERROR`- Si è verificato un errore durante l'elaborazione

Le istanze hanno anche quattro condizioni di stato:

- `InstanceManaged`- I finalizzatori e le etichette sono impostati correttamente
- `GraphResolved`- Grafico di runtime creato e risorse risolte
- `ResourcesReady`- Tutte le risorse create e pronte
- `Ready`- Stato generale dell'istanza (lo diventa solo `True` quando tutte le condizioni secondarie lo sono`True`)

Concentrati sulla `Ready` condizione per determinare lo stato dell'istanza. In caso `Ready` `False` affermativo, controlla le condizioni secondarie per identificare quale fase non è riuscita.

Verifica le autorizzazioni RBAC:

La funzionalità kro richiede le autorizzazioni per creare le risorse Kubernetes sottostanti definite nel tuo. ResourceGraphDefinitions

```
# Check if the capability has the AmazonEKSClusterAdminPolicy
kubectl get accessentry -A | grep kro
```

Se mancano le autorizzazioni, associale alla voce di accesso della `AmazonEKSClusterAdminPolicy` funzionalità kro o crea politiche RBAC più restrittive per l'uso in produzione. Per informazioni dettagliate, vedi the section called “Configurazione delle autorizzazioni ”.

Errori di espressione CEL

Gli errori delle espressioni CEL vengono rilevati al momento ResourceGraphDefinition della creazione, non al momento della creazione delle istanze. kro convalida tutta la sintassi CEL, controlla i tipi delle espressioni rispetto agli schemi Kubernetes e verifica l'esistenza dei campi quando crei l'RGD.

Errori comuni di convalida CEL:

- Riferimento al campo non definito: riferimento a un campo che non esiste nello schema o nella risorsa
- Mancata corrispondenza del tipo: l'espressione restituisce un tipo errato (ad esempio, stringa in cui è previsto un numero intero)
- Sintassi non valida: parentesi, virgolette o operatori mancanti nell'espressione CEL
- Tipo di risorsa sconosciuto: riferimento a un CRD che non esiste nel cluster

Verifica lo stato di convalida 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
```

In caso `ResourceGraphAccepted` `False` affermativo, il messaggio di condizione contiene l'errore di convalida.

Esempi di espressioni CEL valide:

```
# 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}
```

Le dipendenze delle risorse non si risolvono

kro deduce automaticamente le dipendenze dalle espressioni CEL e crea le risorse nell'ordine corretto. Se le risorse non vengono create come previsto, controlla l'ordine di dipendenza e la disponibilità delle risorse.

Visualizza l'ordine di creazione calcolato:

```
# See the order kro will create resources
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.topologicalOrder}'
```

Questo mostra l'ordine calcolato in base ai riferimenti delle espressioni CEL tra le risorse.

Verifica la disponibilità delle risorse:

```
# 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}'
```

Verifica le condizioni ReadyWhen (se utilizzate):

Il campo `readyWhen` è facoltativo. Se non specificato, le risorse sono considerate pronte immediatamente dopo la creazione. Se hai definito `readyWhen` delle condizioni, verifica che controllino correttamente la disponibilità delle risorse:

```
resources:
```

```
  - id: deployment
    readyWhen:
      - ${deployment.status.availableReplicas == deployment.spec.replicas}
```

Controlla gli eventi relativi alle risorse:

```
# View events for the underlying resources
kubectl get events -n namespace --sort-by='.lastTimestamp'
```

Errori di convalida dello schema

Se le istanze non vengono create a causa di errori di convalida dello schema, verifica che l'istanza soddisfi i requisiti dello schema RGD.

Controlla gli errori di convalida:

```
# 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"
```

Problemi di convalida comuni:

- Campi obbligatori mancanti: l'istanza non fornisce tutti i campi dello schema obbligatori
- Mancata corrispondenza del tipo: fornisce una stringa dove è previsto un numero intero
- Valore enum non valido: utilizzo di un valore non presente nell'elenco consentito
- Mancata corrispondenza del modello: la stringa non corrisponde al modello regex

Rivedi lo schema RGD:

```
# View the schema definition
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.spec.schema}'
```

Assicurati che l'istanza fornisca tutti i campi obbligatori con i tipi corretti.

Fasi successive

- the section called "Considerazioni"- considerazioni e best practice su kro

- the section called "Configurazione delle autorizzazioni "- Configura RBAC per i team di piattaforme e applicazioni
- the section called "concetti kro"- Comprendi i concetti kro e il ciclo di vita delle risorse
- the section called "Risoluzione dei problemi relativi alle funzionalità"- Guida generale alla risoluzione dei problemi relativi alle funzionalità

Confronto tra EKS Capability for kro e kro autogestito

EKS Capability for kro offre le stesse funzionalità di kro autogestito, ma con vantaggi operativi significativi. Per un confronto generale tra EKS Capabilities e soluzioni autogestite, vedi. the section called "Considerazioni"

EKS Capability for kro utilizza gli stessi controller kro upstream ed è completamente compatibile con upstream kro. ResourceGraphDefinitions, Le espressioni CEL e la composizione delle risorse funzionano in modo identico. Per la documentazione completa e gli esempi di kro, consulta la documentazione di kro.

Percorso di migrazione

È possibile migrare da kro autogestito a funzionalità gestita senza tempi di inattività.

> **Important**
>
> Prima della migrazione, assicurati che il tuo controller kro autogestito utilizzi la stessa versione di EKS Capability for kro. Verifica la versione con funzionalità nella console EKS o in uso`aws eks describe-capability`, quindi aggiorna la tua installazione autogestita in modo che corrisponda. In questo modo si evitano problemi di compatibilità durante la migrazione.

1. Aggiorna il tuo controller kro autogestito da utilizzare per i contratti di locazione elettorali `kube-system` per i leader:

```
helm upgrade --install kro \
  oci://ghcr.io/awslabs/kro/kro-chart \
  --namespace kro \
  --set leaderElection.namespace=kube-system
```

Questo sposta il contratto di locazione del controllore`kube-system`, permettendo alla capacità gestita di coordinarsi con esso.

2. Crea la funzionalità kro sul tuo cluster (vedi) the section called "Crea una capacità kro"
3. La funzionalità gestita riconosce le istanze esistenti ResourceGraphDefinitions e prende il sopravvento sulla riconciliazione
4. Ridimensiona gradualmente o rimuovi le implementazioni kro autogestite:

```
helm uninstall kro --namespace kro
```

Questo approccio consente a entrambi i controller di coesistere in sicurezza durante la migrazione. La funzionalità gestita adotta ResourceGraphDefinitions automaticamente istanze precedentemente gestite da kro autogestito, garantendo una riconciliazione continua senza conflitti.

Fasi successive

- the section called "Crea una capacità kro"- Crea una risorsa con funzionalità kro
- the section called "concetti kro"- Comprendi i concetti kro e la composizione delle risorse

Risoluzione dei problemi delle funzionalità EKS

Questo argomento fornisce indicazioni generali sulla risoluzione dei problemi per EKS Capabilities, inclusi controlli dello stato delle funzionalità, problemi comuni e collegamenti alla risoluzione di problemi specifici delle funzionalità.

> **Note**
>
> Le funzionalità EKS sono completamente gestite e vengono eseguite all'esterno del cluster. Non hai accesso ai log dei controller o ai namespace dei controller. La risoluzione dei problemi si concentra sullo stato delle funzionalità, sullo stato delle risorse e sulla configurazione.

Un approccio generale alla risoluzione dei problemi

Per la risoluzione dei problemi relativi a EKS Capabilities, seguite questo approccio generale:

1. Verifica dello stato delle funzionalità: utilizza `aws eks describe-capability` per visualizzare lo stato delle funzionalità e i problemi di integrità
2. Verifica lo stato delle risorse: controlla le risorse Kubernetes (CRDs) che hai creato per verificare le condizioni e gli eventi relativi allo stato
3. Rivedi le autorizzazioni IAM: assicurati che il ruolo Capability disponga delle autorizzazioni necessarie
4. Verifica la configurazione: verifica che la configurazione specifica della funzionalità sia corretta

Verifica lo stato delle funzionalità

Tutte le funzionalità EKS forniscono informazioni sanitarie tramite la console EKS e l'`describe-capability`API.

Console:

1. Apri la console Amazon EKS a https://console.aws.amazon.com/eks/ home#/clusters.
2. Seleziona il nome del cluster.
3. Scegli la scheda Osservabilità.
4. Scegli Monitora cluster.
5. Scegli la scheda Capacità per visualizzare lo stato e lo stato di tutte le funzionalità.

La scheda Capacità mostra:

- Nome e tipo di funzionalità
- Stato attuale
- Problemi di salute, con descrizione

AWS CLI:

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability-name
```

La risposta include:

- status: stato attuale della capacità (`CREATING`,`ACTIVE`,`UPDATING`, `DELETINGCREATE_FAILED`,`UPDATE_FAILED`)
- salute: informazioni sanitarie, inclusi eventuali problemi rilevati dalla funzionalità

Stati di funzionalità comuni

CREAZIONE: La funzionalità è in fase di configurazione.

ATTIVO: La funzionalità è attiva e pronta all'uso. Se le risorse non funzionano come previsto, controlla lo stato delle risorse e le autorizzazioni IAM.

AGGIORNAMENTO: vengono applicate le modifiche alla configurazione. Attendi che lo stato ritorni a`ACTIVE`.

CREATE_FAILED o UPDATE_FAILED: durante l'installazione o l'aggiornamento si è verificato un errore. Consulta la sezione dedicata alla salute per i dettagli. Cause comuni:

- La politica di fiducia dei ruoli IAM è errata o mancante
- Il ruolo IAM non esiste o non è accessibile
- Problemi di accesso al cluster
- Parametri di configurazione non validi

Verifica lo stato delle risorse Kubernetes

EKS Capabilities crea e gestisce Kubernetes Custom Resource Definitions () nel tuo cluster. CRDs Durante la risoluzione dei problemi, controlla lo stato delle risorse che hai creato:

```
# 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
```

Le condizioni sullo stato delle risorse forniscono informazioni su:

- Se la risorsa è pronta
- Eventuali errori riscontrati
- Stato attuale di riconciliazione

Verifica le autorizzazioni IAM e l'accesso al cluster

Molti problemi di funzionalità derivano da problemi di autorizzazione IAM o dalla mancata configurazione dell'accesso al cluster. Verifica sia le autorizzazioni di Capability Role che le voci di accesso al cluster.

Controlla le autorizzazioni dei ruoli IAM

Verifica che il ruolo Capability disponga delle autorizzazioni necessarie:

```
# 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'
```

La politica di fiducia deve consentire al responsabile del `capabilities.eks.amazonaws.com` servizio:

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

```
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Controlla le voci di accesso e le politiche di accesso di EKS

Tutte le funzionalità richiedono voci di accesso e politiche di accesso EKS adeguate nel cluster in cui operano.

Verifica che esista Access Entry:

```
aws eks list-access-entries \
  --cluster-name my-cluster \
  --region region-code
```

Cerca il Capability Role ARN nell'elenco. Se manca, la funzionalità non può accedere al cluster.

Controlla le politiche di accesso allegate alla voce:

```
aws eks list-associated-access-policies \
  --cluster-name my-cluster \
  --principal-arn arn:aws:iam::111122223333:role/my-capability-role \
  --region region-code
```

Tutte le funzionalità richiedono politiche di accesso appropriate:

- ACK: necessita delle autorizzazioni per creare e gestire le risorse Kubernetes
- kro: necessita delle autorizzazioni per creare e gestire le risorse Kubernetes
- Argo CD: richiede le autorizzazioni per creare e gestire le applicazioni e richiede le voci di accesso su cluster di destinazione remoti per le implementazioni multi-cluster

Per le implementazioni multicluster di Argo CD:

Se effettui la distribuzione su cluster remoti, verifica che il Capability Role abbia un Access Entry su ogni cluster di destinazione:

```
# 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 l'Access Entry non è presente in un cluster di destinazione, Argo CD non può distribuire applicazioni su di esso. the section called "Registra i cluster"Per i dettagli di configurazione, vedere.

Risoluzione dei problemi specifici per funzionalità

Per una guida dettagliata alla risoluzione dei problemi specifica per ogni tipo di funzionalità:

- the section called "Risoluzione dei problemi"- Risolvi i problemi relativi alla creazione di risorse ACK, alle autorizzazioni IAM e all'accesso tra account
- the section called "Risoluzione dei problemi"- Risolvi i problemi di sincronizzazione delle applicazioni, autenticazione del repository e implementazioni multi-cluster
- the section called "Risoluzione dei problemi"- Risoluzione dei problemi ResourceGraphDefinitions, delle espressioni CEL e delle autorizzazioni RBAC

Problemi comuni a tutte le funzionalità

Capacità bloccata nello stato CREATING

Se una funzionalità rimane attiva `CREATING` più a lungo del previsto:

1. Verifica lo stato delle funzionalità per problemi specifici nella console (Osservabilità > Monitor cluster > scheda Capacità) o utilizzando la AWS CLI:

   ```
   aws eks describe-capability \
     --region region-code \
     --cluster-name my-cluster \
     --capability-name my-capability-name \
     --query 'capability.health'
   ```

2. Verifica che il ruolo IAM esista e abbia la politica di fiducia corretta
3. Assicurati che il tuo cluster sia accessibile e integro

4. Verifica la presenza di eventuali problemi a livello di cluster che potrebbero impedire la configurazione delle funzionalità

Risorse non create o aggiornate

Se la funzionalità è `ACTIVE` presente ma le risorse non vengono create o aggiornate:

1. Controlla lo stato della risorsa per verificare eventuali condizioni di errore
2. Verifica le autorizzazioni IAM per i AWS servizi specifici (ACK) o i repository (Argo CD)
3. Controlla le autorizzazioni RBAC per la creazione di risorse sottostanti (kro)
4. Rivedi le specifiche delle risorse per verificare eventuali errori di convalida

Lo stato delle capacità mostra problemi

Se `describe-capability` presenta problemi di salute:

1. Leggi attentamente le descrizioni dei problemi: spesso indicano il problema specifico
2. Risolvi la causa principale (autorizzazioni IAM, errori di configurazione, ecc.)
3. La funzionalità verrà ripristinata automaticamente una volta risolto il problema

Fasi successive

- the section called "Lavorare con le funzionalità"- Gestisci le risorse funzionali
- the section called "Risoluzione dei problemi"- Risoluzione dei problemi specifici per ACK
- the section called "Risoluzione dei problemi"- Risoluzione dei problemi specifici di Argo CD
- the section called "Risoluzione dei problemi"- risoluzione dei problemi specifici per kro
- the section called "Considerazioni sulle funzionalità EKS"- Le migliori pratiche di sicurezza per quanto riguarda le funzionalità

Organizzazione e monitoraggio delle risorse del cluster

Questo capitolo include i seguenti argomenti per aiutarti a gestire il cluster. Puoi inoltre visualizzare informazioni relative alle risorse Kubernetes tramite la Console di gestione AWS.

- La dashboard Kubernetes è un'interfaccia utente generica basata sul web per i cluster Kubernetes. Consente agli utenti di gestire le applicazioni in esecuzione nel cluster e risolverne i problemi, nonché di gestire il cluster stesso. Per ulteriori informazioni, consulta il repository GitHub della dashboard Kubernetes.
- the section called "Metrics Server" – Il server dei parametri Kubernetes è un aggregatore dei dati di utilizzo delle risorse nel cluster. Non viene implementato per impostazione predefinita nel cluster, ma viene utilizzato dai componenti aggiuntivi di Kubernetes, ad esempio la dashboard Kubernetes e the section called "Horizontal Pod Autoscaler". In questo argomento viene descritto come installare il server dei parametri.
- the section called "Implementazione di app con Helm" – Il programma di gestione del pacchetto Helm per Kubernetes consente di installare e gestire le applicazioni sul cluster Kubernetes. Questo argomento consente di installare ed eseguire i file binari Helm per consentire l'installazione e la gestione di grafici utilizzando la CLI Helm nel computer.
- the section called "Assegnazione di tag alle risorse ": per semplificare la gestione delle risorse Amazon EKS, è possibile assegnare metadati personalizzati a ciascuna risorsa sotto forma di tag. Questo argomento descrive i tag e mostra come crearli.
- the section called "Quote di servizio": l'account AWS dispone delle seguenti quote di default, precedentemente definite limiti, per ogni servizio AWS. Scopri le quote per Amazon EKS e come aumentarle.

Monitorare e ottimizzare i costi dei cluster Amazon EKS

Il monitoraggio dei costi è un aspetto essenziale della gestione dei cluster Kubernetes su Amazon EKS. Acquisendo visibilità sui costi dei cluster, è possibile ottimizzare l'utilizzo delle risorse, fissare budget e prendere decisioni relative alle implementazioni basate sui dati. Amazon EKS offre due soluzioni di monitoraggio dei costi, ognuna con vantaggi unici, per tracciare e allocare i costi in modo efficace:

Dati di ripartizione dei costi di fatturazione AWS per Amazon EKS: questa funzionalità nativa si integra perfettamente con la Console AWS per la fatturazione, consentendo di analizzare e allocare i

costi utilizzando la stessa interfaccia e gli stessi flussi di lavoro già noti utilizzati per altri servizi AWS. Con la ripartizione dei costi, è possibile ottenere approfondimenti sui costi di Kubernetes direttamente insieme alle altre spese AWS, semplificando l'ottimizzazione globale dei costi in tutto l'ambiente AWS. È anche possibile utilizzare le funzionalità di fatturazione AWS esistenti come Cost Categories e Cost Anomaly Detection per migliorare ulteriormente le capacità di gestione dei costi. Per ulteriori informazioni, consultare la pagina Understanding split cost allocation data della Guida per l'utente di Gestione dei costi e fatturazione AWS.

Kubecost: Amazon EKS supporta Kubecost, uno strumento di monitoraggio dei costi di Kubernetes. Kubecost offre un approccio ricco di funzionalità e nativo di Kubernetes al monitoraggio dei costi, con analisi granulari dei costi in base alle risorse Kubernetes, consigli per l'ottimizzazione dei costi e dashboard e report pronti all'uso. Kubecost recupera inoltre dati accurati sui prezzi grazie all'integrazione con il Report di costi e utilizzo AWS, garantendo una visione precisa dei costi di Amazon EKS. Informazioni su come installare Kubecost. Consultare la pagina Kubecost di AWS Marketplace per ulteriori informazioni su come ottenere un abbonamento gratuito a Kubecost.

Visualizza i costi per Pod nella AWS fatturazione con allocazione dei costi suddivisa

Monitoraggio dei costi tramite dati di allocazione dei costi AWS suddivisi per Amazon EKS

Puoi utilizzare i dati di allocazione dei costi AWS suddivisi per Amazon EKS per ottenere una visibilità granulare dei costi per i tuoi cluster Amazon EKS. In questo modo è possibile analizzare, ottimizzare e ottenere il chargeback dei costi e dell'utilizzo delle applicazioni Kubernetes. Allochi i costi delle applicazioni a singole unità aziendali e team in base alla EC2 CPU e alle risorse di memoria Amazon utilizzate dalla tua applicazione Kubernetes. I dati di ripartizione dei costi per Amazon EKS offrono visibilità sul costo per pod e consentono di aggregare i dati relativi ai costi per pod utilizzando namespace, cluster e altre primitive Kubernetes. Di seguito sono riportati alcuni esempi di primitive Kubernetes che è possibile utilizzare per analizzare i dati di ripartizione dei costi di Amazon EKS.

- Nome cluster
- Implementazione
- Namespace
- Nodo
- Nome del carico di lavoro
- Tipo di carico di lavoro

Sono supportati anche i tag di ripartizione dei costi definiti dall'utente. Per ulteriori informazioni sull'utilizzo dei dati di allocazione dei costi suddivisi, consulta Comprendere i dati di allocazione dei costi suddivisi nella Billing User Guide. AWS

Configurare i report di costi e utilizzo

È possibile attivare la suddivisione dei dati di allocazione dei costi per EKS nella console di gestione dei costi, nell'interfaccia a riga di AWS comando o in. AWS SDKs

Utilizzare la seguente procedura per i dati di ripartizione dei costi:

1. Attiva i Dati di ripartizione dei costi. Per ulteriori informazioni, vedere Abilitazione dei dati di allocazione suddivisa dei costi nella Guida per l'utente del rapporto sui AWS costi e sull'utilizzo.
2. Includere i dati in un report nuovo o esistente.
3. Visualizzare il report. Per visualizzare i file di report in Amazon Simple Storage Service, puoi utilizzare la console Gestione costi e fatturazione.

Installare Kubecost

Amazon EKS supporta Kubecost, che è possibile utilizzare per monitorare i costi suddivisi per risorse Kubernetes compresi pod, nodi, namespace ed etichette. Questo argomento illustra l'installazione di Kubecost e l'accesso alla relativa dashboard.

Amazon EKS fornisce un pacchetto AWS ottimizzato di Kubecost per la visibilità dei costi dei cluster. Puoi utilizzare i contratti di AWS supporto esistenti per ottenere assistenza. Per ulteriori informazioni sulle versioni di Kubecost disponibili, consultare the section called "Informazioni su Kubecost".

> **Note**
>
> Kubecost v3 introduce importanti miglioramenti architetturali, tra cui prestazioni notevolmente più veloci e funzionalità di automazione avanzate. Scopri di più su Kubecost v3.
> Kubecost v2 introduce molte nuove funzionalità importanti. Ulteriori informazioni su Kubecost v2.

Per ulteriori informazioni su Kubecost, consultare la documentazione di Kubecost e le domande frequenti.

Installare il pacchetto Kubecost ottimizzato per Amazon EKS

È possibile utilizzare una delle seguenti procedure per installare il pacchetto Kubecost ottimizzato per Amazon EKS:

- Prima di iniziare, si consiglia di consultare la pagina Kubecost - Architecture Overview per comprendere il funzionamento di Kubecost su Amazon EKS.
- Se non si conosce Amazon EKS, consigliamo di utilizzare il componente aggiuntivo Amazon EKS perché semplifica l'installazione del pacchetto Kubecost ottimizzato per Amazon EKS. Per ulteriori informazioni, consultare la pagina Deploying Kubecost on an Amazon EKS cluster using Amazon EKS add-on.
- Per personalizzare l'installazione, è possibile configurare il pacchetto Kubecost ottimizzato per Amazon EKS con Helm. Per ulteriori informazioni, consultare Deploying Kubecost on an Amazon EKS cluster using Helm nella documentazione di Kubecost.

> **Important**
>
> Per Kubecost v3, la posizione della mappa di Helm è stata modificata in. `public.ecr.aws/kubecost/kubecost` Se stai effettuando l'aggiornamento dalla v2, aggiorna di conseguenza i riferimenti del repository Helm.

> **Note**
>
> Per le implementazioni multi-cluster con Kubecost v3, è necessario uno storage di oggetti compatibile con S3 (S3 per clienti EKS) per lo storage delle metriche.AWS Questo sostituisce lo storage compatibile con Prometheus utilizzato nella versione 2. Per ulteriori informazioni, consulta Installazione multi-cluster nella documentazione di Kubecost.

Accedere alla dashboard Kubecost

Una volta completata la configurazione del pacchetto Kubecost ottimizzato per Amazon EKS, dovrebbe essere possibile accedere alla dashboard Kubecost. Per ulteriori informazioni, consulta the section called "Accedere alla dashboard Kubecost".

Accedere alla dashboard Kubecost

Prerequisiti

1. Assicurarsi che lo stato dei pod relativi a kubecost sia "In esecuzione".

```
kubectl get pods --namespace kubecost
```

Accedere alla dashboard Kubecost

1. Sul dispositivo, abilitare l'inoltro della porta per esporre la dashboard Kubecost.
   - Se kubecost v3 è installato usando helm:

     ```
     kubectl port-forward deployment/kubecost-frontend 9090 --namespace kubecost
     ```

   - Se kubecost v1 o v2 è installato utilizzando helm:

     ```
     kubectl port-forward deployment/kubecost-cost-analyzer 9090 --namespace kubecost
     ```

   - Se kubecost è installato utilizzando il componente aggiuntivo Amazon EKS:

     ```
     kubectl port-forward deployment/cost-analyzer 9090 --namespace kubecost
     ```

     In alternativa, è possibile utilizzare AWS Load Balancer Controller per esporre Kubecost e utilizzare Amazon Cognito per l'autenticazione, l'autorizzazione e la gestione degli utenti. Per ulteriori informazioni, consultare l'articolo How to use Application Load Balancer and Amazon Cognito to authenticate users for your Kubernetes web apps.
2. Sullo stesso dispositivo su cui hai completato il passaggio precedente, apri un browser Web e inserisci il seguente indirizzo.

   ```
   http://localhost:9090
   ```

   Viene visualizzata la pagina Panoramica di Kubecost nel browser. La raccolta dei parametri da parte di Kubecost potrebbe richiedere dai 5 ai 10 minuti (o più), a seconda delle dimensioni del cluster. È possibile visualizzare la spesa Amazon EKS, inclusi i costi cumulativi dei cluster, costi delle risorse Kubernetes associate e spesa mensile aggregata.

3. Per tenere traccia dei costi a livello di cluster, tagga le tue risorse Amazon EKS per la fatturazione. Per ulteriori informazioni, consulta the section called "Tagging delle risorse per la fatturazione".
   - Ripartizione dei costi: visualizza i costi mensili di Amazon EKS e i costi cumulativi per ciascuno dei tuoi namespace e altre dimensioni negli ultimi sette giorni. Questo è utile per capire quali parti della tua applicazione contribuiscono alla spesa di Amazon EKS.
   - Asset: visualizza i costi degli asset AWS infrastrutturali associati alle tue risorse Amazon EKS.

Informazioni su Kubecost

Amazon EKS fornisce un pacchetto AWS ottimizzato di Kubecost per la visibilità dei costi dei cluster. Amazon EKS supporta Kubecost, che è possibile utilizzare per monitorare i costi suddivisi per risorse Kubernetes compresi pod, nodi, namespace ed etichette.

Questo argomento illustra le versioni di Kubecost disponibili e le differenze tra i vari livelli. EKS supporta Kubecost versione 1, versione 2 e versione 3. Ogni versione è disponibile in diversi livelli. È possibile utilizzare il pacchetto Kubecost ottimizzato per Amazon EKS per i cluster Amazon EKS senza costi aggiuntivi. È possibile che ti venga addebitato un costo per l'utilizzo di AWS servizi associati, come Amazon Managed Service for Prometheus. Inoltre, puoi utilizzare i contratti di AWS assistenza esistenti per ottenere assistenza.

In qualità di amministratore della piattaforma Kubernetes e leader finanziario, è possibile usare Kubecost per visualizzare una ripartizione degli addebiti di Amazon EKS, allocare i costi e ottenere il chargeback di unità organizzative come i team responsabili delle applicazioni. È possibile fornire ai team interni e alle unità aziendali dati sui costi trasparenti e accurati basati sulla AWS fattura effettiva. Inoltre, puoi anche ottenere consigli personalizzati per l'ottimizzazione dei costi in base all'ambiente dell'infrastruttura e ai modelli di utilizzo all'interno dei cluster. Per ulteriori informazioni su Kubecost, consultare la documentazione di Kubecost.

Qual è la differenza tra il pacchetto personalizzato di Kubecost e la versione gratuita di Kubecost (nota anche come)? OpenCost

AWS e Kubecost hanno collaborato per offrire una versione personalizzata di Kubecost. Questa versione include un sottoinsieme di funzionalità commerciali senza costi aggiuntivi. Consultare le tabelle seguenti per le funzionalità incluse nel pacchetto personalizzato di Kubecost.

Kubecost v3

Qual è la differenza tra Kubecost v2 e v3?

Kubecost 3.0 è un importante aggiornamento dell'architettura che offre prestazioni notevolmente più veloci, scalabilità migliorata e funzionalità di ottimizzazione proattiva. La modifica più significativa è la migrazione a un ClickHouse database, che sostituisce DuckDB dalla versione 2.8, che fornisce query sostanzialmente più veloci e prestazioni più affidabili su larga scala. Kubecost 3.0 introduce anche un agente unificato che combina le funzionalità Kubecost e Cloudability, eliminando la dipendenza da Prometheus e riducendo l'ingombro di memoria mantenendo la compatibilità. OpenCost

> **Important**
>
> Consulta la documentazione di Kubecost prima di eseguire l'aggiornamento alla v3. La migrazione dalla v2 richiede un'attenta pianificazione e può influire sulla disponibilità dei report durante la transizione. La posizione della mappa di Helm è stata modificata in. `public.ecr.aws/kubecost/kubecost`

Principali miglioramenti architettonici nella v3:

- ClickHouse Database: sostituisce DuckDB per query notevolmente più veloci e una migliore scalabilità
- Unified Agent: combina le funzionalità di Kubecost e Cloudability, eliminando la dipendenza da Prometheus
- Storage compatibile con S3 per più cluster: per le implementazioni multi-cluster, la v3 utilizza lo storage di oggetti compatibile con S3 (AWS S3 per i clienti EKS) anziché lo storage compatibile con Prometheus come Amazon Managed Service for Prometheus. L' FinOps agente estrae i parametri dall'API Kubernetes e li trasferisce a uno storage compatibile con S3, quindi l'Aggregator estrae i dati, esegue le fasi di derivazione e visualizza i risultati nel frontend. Per ulteriori informazioni, consulta la Guida all'installazione di più cluster e ai cluster secondari nella documentazione di Kubecost.
- Ingombro di memoria ridotto: requisiti di risorse notevolmente inferiori pur mantenendo la funzionalità
- Architettura semplificata: topologia pod a contenitore singolo per scalabilità indipendente e maggiore resilienza
- Automazione avanzata: dimensionamento automatizzato delle richieste di container con consapevolezza di più cluster e profili personalizzati

Vantaggi del pacchetto ottimizzato per Amazon EKS nella versione 3:

Il pacchetto Kubecost ottimizzato per Amazon EKS continua a essere disponibile senza costi aggiuntivi ed è esente dal nuovo limite di spesa di 100.000 USD introdotto nel piano gratuito Kubecost v3. Gli utenti EKS mantengono l'accesso completo a tutte le funzionalità di spesa di Kubernetes indipendentemente dai livelli di spesa.

Confronto delle funzionalità principali:

Funzionalità	Kubecost gratuito, livello 3.0	Pacchetto Kubecost 3.0 ottimizzato per Amazon EKS	Kubecost Enterprise 3.0
Visibilità costo del cluster	Cluster illimitati, con un costo limitato a 100.000 USD, da spendere in 30 giorni	Multicluster unificato senza limiti di spesa	Cluster unificati e illimitati in un numero illimitato di ambienti (ad esempio multi-clo ud)
Backend del database	ClickHouse (locale)	ClickHouse con storage compatibile con S3 per metriche multi-cluster	ClickHouse con opzioni di database personalizzate
Performance	Query notevolmente più veloci rispetto alla v2	Query notevolmente più veloci rispetto alla v2	Query notevolmente più veloci rispetto alla v2
Impronta di memoria	Ridotto rispetto alla v2 (nessuna dipendenza da Prometheus)	Ridotto rispetto alla v2 (nessuna dipendenza da Prometheus)	Ridotto rispetto alla v2 (nessuna dipendenza da Prometheus)
Dimensionamento automatico delle richieste di container	Disponibile (limitato a 250 core)	Disponibile senza limiti di base	Disponibile senza limiti di base
Limiti di spesa	$100.000 USD in 30 giorni	Nessun limite di spesa	Nessun limite di spesa

Funzionalità	Kubecost gratuito, livello 3.0	Pacchetto Kubecost 3.0 ottimizzato per Amazon EKS	Kubecost Enterprise 3.0
Automazione multicluster	Limitato	Conoscenza completa di più cluster con messaggistica sicura	Conoscenza completa di più cluster con messaggistica sicura

Kubecost v2

Qual è la differenza tra Kubecost v1 e v2?

Kubecost 2.0 è un grande aggiornamento rispetto alle versioni precedenti e include nuove importanti funzionalità tra cui un nuovo backend dell'API. Nota che Allocation e Assets APIs sono completamente compatibili con le versioni precedenti. Consultare la documentazione di Kubecost per garantire una transizione fluida. Per l'elenco completo dei miglioramenti, consultare l'annuncio di Kubecost v2.0 e le note di rilascio complete.

> **Important**
>
> Consultare la documentazione di Kubecost prima dell'aggiornamento. L'aggiornamento può compromettere la disponibilità dei report.

Confronto delle funzionalità principali:

Funzionalità	Kubecost 2.0 livello gratuito	Pacchetto Kubecost 2.0 ottimizzato per Amazon EKS	Kubecost 2.0 Enterprise
Visibilità costo del cluster	Cluster illimitati fino a 250 core	Multi-cluster unificato senza limiti core se integrato con il servizio gestito da Amazon per Prometheus	Cluster unificati e illimitati in un numero illimitato di ambienti (ad esempio multi-clo ud)

Funzionalità	Kubecost 2.0 livello gratuito	Pacchetto Kubecost 2.0 ottimizzato per Amazon EKS	Kubecost 2.0 Enterprise
Implementazione	Utente ospitato	Utente ospitato	Ospitata dall'utente, ospitata da Kubecost (tenant dedicato), SaaS
Database supportati	Prometheus locale	Servizio gestito da Amazon per Prometheus o Prometheus locale	Qualsiasi variante e database personali zzato Prometheus
Supporto per la conservazione del database (parametri non elaborati)	15 giorni	Dati storici illimitati	Dati storici illimitati
API Kubecost e conservazione UI (ETL)	15 giorni	15 giorni	Illimitato
Visibilità del cloud ibrido	-	Cluster Amazon EKS e Amazon EKS Anywhere	Multi-cloud e cloud ibrido
Avvisi e report ricorrenti	Supporto solo sul cluster primario, limitato a 250 core	Avvisi di efficienza, avvisi sul budget, avvisi di modifica della spesa e altri avvisi per tutti i cluster	Avvisi di efficienza, avvisi sul budget, avvisi di modifica della spesa e altri avvisi per tutti i cluster
Report salvati.	-	Report che utilizzano 15 giorni di parametri	Report che utilizzano dati storici e parametri illimitati

Funzionalità	Kubecost 2.0 livello gratuito	Pacchetto Kubecost 2.0 ottimizzato per Amazon EKS	Kubecost 2.0 Enterprise
Integrazione della fatturazione cloud	Supporto solo sul cluster primario, limitato a 250 core	Supporto tariffario personalizzato per AWS (inclusi più cluster e più account)	Supporto tariffario personalizzato per qualsiasi cloud
Suggerimenti per i risparmi	Supporto solo sul cluster primario, limitato a 250 core	Approfondimenti sul cluster primario, ma senza limite di 250 core	Informazioni su più cluster
Governance: audit	-	-	Verifica gli eventi relativi ai costi storici
Supporto Single Sign-On (SSO)	-	Amazon Cognito supportato	Okta, Auth0, PingID e qualsiasi altra cosa personalizzata KeyCloak
Controllo degli accessi basato sul ruolo (RBAC) con SAML 2.0	-	-	Okta, Auth0, PingID e qualsiasi altra cosa personalizzata KeyCloak
Formazione e onboarding aziendali	-	-	Servizio completo di formazione e onboarding FinOps
Team	-	-	Sì

Nuove caratteristiche:

Le seguenti funzionalità hanno limiti legati ai parametri:

- Aggregatore Kubecost

- Monitoraggio della rete
- Operazioni Kubecost
- Raccolte
- Rilevamento anomalie
- Richiesta di dimensionamento ottimale dei container
- Previsioni Kubecost
- Completamento automatico per filtraggio e aggregazione

Limiti dei parametri:

Metrica	Kubecost 2.0 livello gratuito	Pacchetto Kubecost 2.0 ottimizzato per Amazon EKS	Kubecost 2.0 Enterprise
Dimensione del cluster	Cluster illimitati fino a 250 core	Illimitato	Illimitato
Conservazione dei parametri	15 giorni	15 giorni	Illimitato
Supporto multi-cluster	Non disponibile	Disponibilità	Disponibilità
Limiti di core	250 core per cluster	Nessun limite di core	Nessun limite di core

Kubecost v1

Funzionalità	Kubecost livello gratuito	Pacchetto Kubecost ottimizzato per Amazon EKS	Kubecost Enterprise
Distribuzione	Utente ospitato	Utente ospitato	Ospitata dall'utente o ospitata da Kubecost (SaaS)

Funzionalità	Kubecost livello gratuito	Pacchetto Kubecost ottimizzato per Amazon EKS	Kubecost Enterprise
Numero di cluster supportati	Illimitato	Illimitato	Illimitato
Database supportati	Prometheus locale	Prometheus locale o servizio gestito da Amazon per Prometheus	Prometheus, servizio gestito da Amazon per Prometheus, Cortex o Thanos
Supporto per la conservazione dei database	15 giorni	Dati storici illimitati	Dati storici illimitati
Conservazione API Kubecost (ETL)	15 giorni	15 giorni	Dati storici illimitati
Visibilità costo del cluster	Cluster singoli	Multicluster unificato	Multicluster unificato
Visibilità del cloud ibrido	-	Cluster Amazon EKS e Amazon EKS Anywhere	Supporto multi-cloud e cloud ibrido
Avvisi e report ricorrenti	-	Avvisi di efficienza, avvisi sul budget, avvisi di modifica della spesa e altro ancora	Avvisi di efficienza, avvisi sul budget, avvisi di modifica della spesa e altro ancora
Report salvati.	-	Report che utilizzano dati di 15 giorni	Report che utilizzano dati storici illimitati
Integrazione della fatturazione cloud	Obbligatorio per ogni singolo cluster	Supporto tariffario personalizzato per AWS (inclusi più cluster e più account)	Supporto tariffario personalizzato per AWS (inclusi più cluster e più account)

Funzionalità	Kubecost livello gratuito	Pacchetto Kubecost ottimizzato per Amazon EKS	Kubecost Enterprise
Suggerimenti per i risparmi	Informazioni su un singolo cluster	Informazioni su un singolo cluster	Informazioni su più cluster
Governance: audit	-	-	Verifica gli eventi relativi ai costi storici
Supporto Single Sign-On (SSO)	-	Amazon Cognito supportato	Okta, Auth0, PingID, KeyCloak
Controllo degli accessi basato sul ruolo (RBAC) con SAML **2.0**	-	-	Okta, Auth0, PingID, Keycloak
Formazione e onboarding aziendali	-	-	Servizio completo di formazione e onboarding FinOps

Domande frequenti

Consultare le seguenti domande frequenti e relative risposte sull'utilizzo di Kubecost con Amazon EKS.

Cos'è la funzionalità di conservazione dell'API Kubecost (ETL)?

La funzionalità ETL Kubecost aggrega e organizza i parametri per evidenziare la visibilità dei costi a vari livelli di granularità (come `namespace-level`, `pod-level` e `deployment-level`). Per il pacchetto Kubecost ottimizzato per Amazon EKS, i clienti ottengono dati e approfondimenti dai parametri degli ultimi 15 giorni.

Cos'è la funzionalità di avvisi e segnalazioni ricorrenti? Quali avvisi e report include?

Gli avvisi Kubecost consentono ai team di ricevere aggiornamenti in tempo reale sulla spesa Kubernetes, nonché sulla spesa per il cloud. I report ricorrenti consentono ai team di ricevere

visualizzazioni personalizzate della cronologia della spesa Kubernetes e per il cloud. Entrambi sono configurabili utilizzando l'UI Kubecost o i valori Helm. Supportano e-mail, Slack e Microsoft Teams.

Cosa includono i report salvati?

I report salvati Kubecost sono visualizzazioni predefinite dei parametri di costi ed efficienza. Includono il costo per cluster, namespace, etichetta e altro ancora.

Cos'è l'integrazione della fatturazione nel cloud?

L'integrazione con la AWS fatturazione APIs consente a Kubecost di visualizzare out-of-cluster i costi (come Amazon S3). Inoltre, consente a Kubecost di riconciliare le previsioni interne al cluster di Kubecost con i dati di fatturazione effettivi, per tenere conto dell'utilizzo spot, dei Savings Plans e degli sconti aziendali.

Cosa includono i consigli di risparmio?

Kubecost offre approfondimenti e automazione per aiutare gli utenti a ottimizzare l'infrastruttura Kubernetes e le relative spese.

È previsto un costo per questa funzionalità?

No. È possibile utilizzare il pacchetto Kubecost ottimizzato per Amazon EKS senza costi aggiuntivi. Se desideri funzionalità Kubecost aggiuntive non incluse, puoi acquistare una licenza Enterprise di Kubecost tramite AWS Marketplace o direttamente da Kubecost.

È disponibile supporto per il pacchetto Kubecost ottimizzato per Amazon EKS?

Sì, solo se si utilizza il pacchetto Kubecost ottimizzato per Amazon EKS.

Come si può richiedere supporto per il pacchetto Kubecost ottimizzato per Amazon EKS?

Puoi aprire una richiesta di supporto con il team di AWS supporto all'indirizzo Contatti AWS.

È necessaria una licenza per l'utilizzo delle funzionalità di Kubecost fornite dall'integrazione Amazon EKS?

No.

Posso integrare AWS Kubecost con Cost and Usage Report per un reporting più accurato?

Sì. Puoi configurare Kubecost in modo che acquisisca i dati dal rapporto sui AWS costi e sull'utilizzo per ottenere una visibilità accurata dei costi, inclusi sconti, prezzi Spot, prezzi delle istanze

riservate e altro. Per ulteriori informazioni, consultare la pagina AWS Cloud Billing Integration della documentazione di Kubecost.

Questa versione supporta la gestione dei costi dei cluster Kubernetes autogestiti su Amazon? EC2

No. Il pacchetto Kubecost ottimizzato per Amazon EKS è compatibile solo con i cluster Amazon EKS.

Kubecost può tenere traccia dei costi di Amazon EKS su AWS Fargate?

Kubecost si impegna al massimo per mostrare la visibilità dei costi del cluster per Amazon EKS su Fargate, ma con una precisione inferiore rispetto ad Amazon EKS su Amazon. EC2 Ciò è dovuto principalmente alla differenza nella modalità di fatturazione dell'utilizzo. Con Amazon EKS su Fargate, vengono fatturate le risorse utilizzate. Con Amazon EKS sui EC2 nodi Amazon, ti vengono fatturate le risorse assegnate. Kubecost calcola il costo di un EC2 nodo Amazon in base alle specifiche del nodo, che includono CPU, RAM e storage temporaneo. Con Fargate, i costi vengono calcolati in base alle risorse richieste per i pod Fargate.

Come si possono ottenere aggiornamenti e nuove versioni di Kubecost?

È possibile aggiornare la versione di Kubecost utilizzando le procedure di aggiornamento standard di Helm. Per Kubecost v3, le versioni più recenti sono disponibili nella nuova posizione cartografica di Helm. `public.ecr.aws/kubecost/kubecost` Le versioni precedenti (v1 e v2) rimangono disponibili nella Amazon ECR Public Gallery.

> **Important**
>
> Quando esegui l'aggiornamento a Kubecost v3, tieni presente che la posizione della mappa di Helm è cambiata da a. `public.ecr.aws/kubecost/cost-analyzer` `public.ecr.aws/kubecost/kubecost` Aggiorna di conseguenza i riferimenti del tuo repository Helm.

La CLI di **`kubectl-cost`** è supportata? Come posso installarlo?

Sì. `Kubectl-cost` è uno strumento open source di Kubecost (licenza Apache 2.0) che offre l'accesso tramite CLI ai parametri di ripartizione dei costi di Kubernetes. Per installarlo`kubectl-cost`, vedi Installazione su GitHub.

L'interfaccia utente di Kubecost è supportata? Come posso accedervi?

Kubecost offre un pannello di controllo web a cui è possibile accedere tramite l'inoltro alla porta `kubectl`, un ingresso o un bilanciatore del carico. Puoi anche usare AWS Load Balancer Controller per esporre Kubecost e usare Amazon Cognito per l'autenticazione, l'autorizzazione e la gestione degli utenti. Per ulteriori informazioni, consulta Come usare Application Load Balancer e Amazon Cognito per autenticare gli utenti per le tue app web Kubernetes sul blog. AWS

Il nuovo limite di spesa di 100.000 dollari in Kubecost v3 influisce sugli utenti di Amazon EKS?

No. Il limite di spesa di 100.000 USD per 30 giorni introdotto nel piano gratuito Kubecost v3 non si applica agli utenti del pacchetto Kubecost ottimizzato per Amazon EKS. Gli utenti EKS mantengono l'accesso completo a tutte le funzionalità di spesa di Kubernetes indipendentemente dai livelli di spesa.

Quali sono i principali miglioramenti delle prestazioni di Kubecost v3?

Kubecost v3 introduce sostanziali miglioramenti delle prestazioni attraverso il backend del ClickHouse database, che fornisce query notevolmente più veloci rispetto al DuckDB utilizzato nella versione 2.8. Inoltre, l'architettura unificata degli agenti elimina la dipendenza da Prometheus, riducendo l'ingombro della memoria e mantenendo al contempo la piena funzionalità e compatibilità. OpenCost

Quale backend di storage utilizza Kubecost v3 per le implementazioni multi-cluster?

Kubecost v3 utilizza lo storage di oggetti compatibile con S3 (AWS S3 per clienti EKS) per lo storage di metriche multi-cluster, sostituendo lo storage compatibile con Prometheus utilizzato nella v2. L' FinOps agente raccoglie le metriche dall'API Kubernetes e le invia a uno storage compatibile con S3. L'Aggregator recupera quindi questi dati, esegue calcoli dei costi e visualizza i risultati nel frontend. Per istruzioni dettagliate sulla configurazione multicluster, consulta la Guida all'installazione di più cluster e ai cluster secondari nella documentazione di Kubecost.

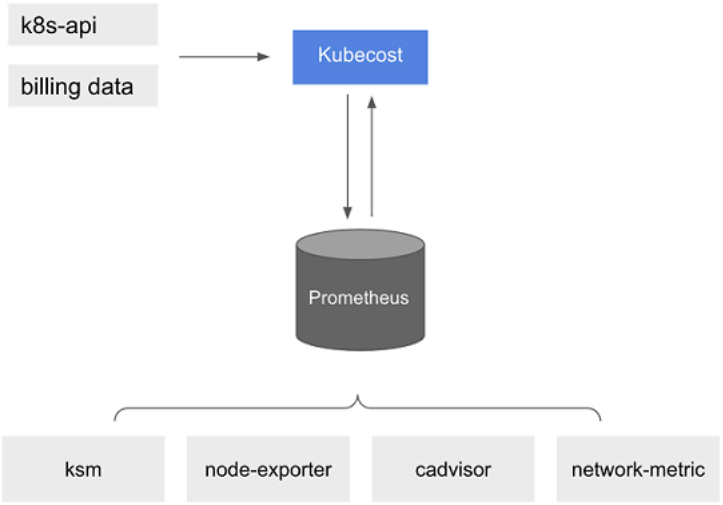
Posso effettuare l'aggiornamento direttamente da Kubecost v1 a v3?

No. L'aggiornamento diretto dalla v1 alla v3 non è supportato. È necessario prima eseguire l'aggiornamento alla v2, quindi migrare alla v3. Consulta la documentazione di Kubecost per indicazioni dettagliate sulla migrazione, poiché il processo richiede un'attenta pianificazione e può influire sulla disponibilità dei report durante la transizione.

Funzionalità Kubecost aggiuntive

- Le seguenti funzionalità sono disponibili in Kubecost v1, v2 e v3.
  - Metriche dei costi di esportazione: il monitoraggio ottimizzato dei costi di Amazon EKS viene implementato con Kubecost. Nelle versioni 1 e 2, Kubecost si integra con Prometheus per

l'archiviazione e l'elaborazione delle metriche. Nella v3, Kubecost utilizza un database per migliorare notevolmente le prestazioni mantenendo al contempo la compatibilità. ClickHouse OpenCost Per le implementazioni multi-cluster nella v3, le metriche vengono archiviate nello storage di oggetti compatibile con S3 (AWS S3 per i clienti EKS) anziché nello storage compatibile con Prometheus. Kubecost legge le metriche, esegue calcoli di allocazione dei costi e fornisce dati tramite la propria interfaccia utente. APIs L'architettura varia in base alla versione ma mantiene funzionalità coerenti.



Puoi scrivere query per inserire i dati di Kubecost nel tuo attuale sistema di business intelligence per ulteriori analisi. È anche possibile usarlo come origine dati per la dashboard Grafana attuale al fine di visualizzare i costi dei cluster Amazon EKS che i team interni conoscono. Per saperne di più su come scrivere query, consulta la documentazione di OpenCost configurazione o usa come riferimenti i modelli JSON Grafana di esempio nel repository Github di Kubecost.

- AWS Integrazione dei report su costi e utilizzo: per eseguire calcoli di allocazione dei costi per il cluster Amazon EKS, Kubecost recupera le informazioni pubbliche sui prezzi di AWS servizi e AWS risorse dall'API Price List. AWS Puoi anche integrare AWS Kubecost con Cost and Usage Report per migliorare l'accuratezza delle informazioni sui prezzi specifiche del tuo account. AWS Queste informazioni includono programmi di discount aziendali, utilizzo di istanze riservate, Savings Plans e utilizzo spot. Per saperne di più su come funziona l'integrazione del AWS Cost and Usage Report, consulta AWS Cloud Billing Integration nella documentazione di Kubecost.

Visualizza l'utilizzo delle risorse con Kubernetes Metrics Server

Kubernetes Metrics Server è un aggregatore dei dati di utilizzo delle risorse nel cluster e non viene implementato nei cluster Amazon EKS per impostazione predefinita. Per ulteriori informazioni, consulta Kubernetes Metrics Server on. GitHub Il Metrics Server è comunemente usato da altri componenti aggiuntivi di Kubernetes, come le implementazioni per scalare i pod con Horizontal Pod Autoscaler o il pannello di controllo di Kubernetes. Per ulteriori informazioni, consulta la pipeline di parametri delle risorse nella documentazione di Kubernetes. In questo argomento viene descritto come implementare Kubernetes Metrics Server sul cluster Amazon EKS.

> **Important**
>
> Le metriche sono destinate all' point-in-timeanalisi e non sono una fonte accurata per l'analisi storica. Non possono essere utilizzati come soluzione di monitoraggio o per altri scopi di dimensionamento non automatico. Per ulteriori informazioni sul monitoraggio, consulta la pagina Monitoraggio dei cluster.

Considerazioni

- Se distribuisci manualmente Kubernetes Metrics Server sui nodi Fargate utilizzando il manifesto, configura l'implementazione `metrics-server` per utilizzare una porta diversa da quella predefinita di `10250`. Questa porta è riservata a Fargate. La versione del componente aggiuntivo di Amazon EKS di Metrics Server è preconfigurata per utilizzare la porta `10251`.
- Assicurati che i gruppi di sicurezza e la rete ACLs consentano la porta `10250` tra i `metrics-server` Pod e tutti gli altri nodi e Pod. Il Kubernetes Metrics Server utilizza ancora la porta `10250` per raccogliere metriche da altri endpoint del cluster. Se esegui l'implementazione sui nodi Fargate, consenti sia la porta Metrics Server alternativa configurata che la porta `10250`.

Implementa come componente aggiuntivo della community con i componenti aggiuntivi di Amazon EKS

Novità: ora puoi distribuire Metrics Server come componente aggiuntivo della community utilizzando la console AWS o Amazon EKS. APIs

Implementa con console AWS

1. Apri il tuo cluster EKS nella console AWS
2. Nella scheda "Add-ons", seleziona Get More Add-ons.
3. Dalla sezione "Community add-ons", seleziona Metrics Server, quindi Avanti
4. EKS determina la versione appropriata del componente aggiuntivo per il cluster. È possibile modificare la versione utilizzando il menu a discesa Versione.
5. Seleziona Avanti e poi Crea per installare il componente aggiuntivo.

Risorse aggiuntive

Ulteriori informazioni su the section called "Componenti aggiuntivi della community".

Puoi installare o aggiornare i componenti aggiuntivi della community come fai con gli altri componenti aggiuntivi di Amazon EKS.

- the section called "Creare un componente aggiuntivo"
- the section called "Aggiornamento di un componente aggiuntivo"
- the section called "Rimozione di un componente aggiuntivo"

Implementa con manifesto

Novità: ora puoi distribuire Metrics Server come componente aggiuntivo della community utilizzando la console AWS o Amazon EKS. APIs Queste istruzioni di installazione del manifesto verranno archiviate.

1. implementare Metrics Server con il comando seguente:

   ```
   kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
   ```

   Se utilizzi Fargate, dovrai modificare questo file. Nella configurazione predefinita, Metric Server utilizza la porta 10250. Questa porta è riservata a Fargate. Sostituisci i riferimenti alla porta 10250 in components.yaml con un'altra porta, ad esempio 10251.
2. Verifica che l'implementazione `metrics-server` esegua il numero di pod desiderato con il comando seguente:

```
kubectl get deployment metrics-server -n kube-system
```

Di seguito viene riportato un output di esempio.

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           6m
```

3. Verifica che il server delle metriche funzioni visualizzando l'utilizzo delle risorse (CPU/memoria) dei nodi.

```
kubectl top nodes
```

4. Se ricevi il messaggio di errore `Error from server (Forbidden)`, devi aggiornare la configurazione RBAC di Kubernetes. La tua identità RBAC di Kubernetes richiede autorizzazioni sufficienti per leggere le metriche del cluster. Verifica le autorizzazioni minime richieste per l'API Kubernetes per la lettura delle metriche. GitHub Scopri come concedere alle identità AWS IAM, come i ruoli, l'accesso a Kubernetes. APIs

Implementazione di applicazioni con Helm su Amazon EKS

Il programma di gestione del pacchetto Helm per Kubernetes consente di installare e gestire le applicazioni sul cluster Kubernetes. Per ulteriori informazioni, consultare la documentazione di Helm. Questo argomento consente di installare ed eseguire i file binari Helm per consentire l'installazione e la gestione di grafici utilizzando la CLI Helm nel sistema locale.

> **Important**
>
> Prima di installare grafici Helm nel cluster Amazon EKS, è necessario configurare `kubectl` per l'utilizzo di Amazon EKS. Se non hai già eseguito questa operazione, consultare the section called "Accedere al cluster con kubectl" prima di continuare. Se il comando seguente va a buon fine per il cluster, la configurazione è corretta.
>
> ```
> kubectl get svc
> ```

1. Eseguire il comando appropriato per il sistema operativo client.

- Se utilizzi macOS con Homebrew, installa i file binari con il comando seguente.

```
brew install helm
```

- Per ulteriori opzioni di installazione, consulta Installing Helm nella documentazione di Helm.

> **Note**
>
> Se ricevi un messaggio che indica la necessità di installare prima `openssl`, è possibile installarlo con il seguente comando.

```
sudo yum install openssl
```

1. Per selezionare il nuovo file binario in `PATH`, chiudere la finestra del terminale corrente e aprirne una nuova.
2. Guarda la versione di Helm che hai installato.

```
helm version --template='{{ .Version }}{{ "\n" }}'
```

Di seguito viene riportato un output di esempio:

```
v3.17.2
```

3. Assicurati che la versione installata sia compatibile con la tua versione del cluster. Per ulteriori informazioni, consulta la versione supportata di Skew. Ad esempio, se esegui `3.17.x`, la versione di Kubernetes supportata non dovrebbe superare l'intervallo di `1.29.x ~ 1.32.x`.
4. A questo punto, è possibile eseguire qualsiasi comando Helm (ad esempio `helm install` *`chart-name`*) per installare, modificare, eliminare o interrogare i grafici Helm nel cluster. Se sei un nuovo utente di Helm e non hai un grafico specifico da installare, puoi effettuare le seguenti operazioni:
   - Sperimentare installando un grafico di esempio. Consulta Install an Example Chart nella Quickstart Guide di Helm.
   - Crea un grafico di esempio e invialo ad Amazon ECR. Per ulteriori informazioni, consultare Invio di un grafico Helm nella Guida per l'utente del registro del container di Amazon Elastic.
   - Installare un grafico Amazon EKS dal repository GitHub eks-charts o da ArtifactHub.

Organizzazione delle risorse Amazon EKS con tag

È possibile utilizzare tag per aiutarti a gestire le risorse Amazon EKS. In questo argomento viene fornita una panoramica della funzione tag e viene illustrato come creare tag.

Argomenti

- Nozioni di base sui tag
- Assegnazione di tag alle risorse
- Limitazioni applicate ai tag
- Tagging delle risorse per la fatturazione
- Utilizzo di tag tramite la console
- Utilizzo di tag tramite la CLI, l'API o eksctl

> **Note**
>
> I tag sono un tipo di metadati separati dalle etichette e dalle annotazioni di Kubernetes. Per ulteriori informazioni sugli altri tipi di metadati, consulta le sezioni seguenti nella documentazione di Kubernetes:
>
> - Etichette e selettori
> - Annotazioni

Nozioni di base sui tag

Un tag è un'etichetta che assegni a una risorsa AWS. Ciascun tag è formato da una chiave e da un valore opzionale.

Puoi utilizzare i tag per categorizzare le risorse AWS. Ad esempio, puoi categorizzarle in base a scopo, proprietario o ambiente. In presenza di un numero elevato di risorse dello stesso tipo, si possono usare i tag assegnati a una risorsa specifica per identificarla rapidamente. Ad esempio, puoi definire un set di tag per i cluster Amazon EKS per monitorare il proprietario di ogni cluster e il livello di stack. Consigliamo di definire un set coerente di chiavi di tag per ciascun tipo di risorsa. Puoi cercare e filtrare le risorse in base ai tag aggiunti.

Dopo aver aggiunto un tag, puoi modificarne le chiavi e i valori oppure rimuovere i tag da una risorsa in qualsiasi momento. Se elimini una risorsa, verranno eliminati anche tutti i tag a essa associati.

I tag non hanno alcun significato semantico per Amazon EKS e vengono interpretati rigorosamente come una stringa di caratteri. È possibile impostare il valore di un tag su una stringa vuota. Tuttavia, non è possibile impostare il valore di un tag su null. Se aggiungi un tag con la stessa chiave di un tag esistente a una risorsa specifica, il nuovo valore sovrascrive quello precedente.

Se utilizzi AWS Identity and Access Management (IAM), puoi controllare quali utenti nel tuo account AWS dispongono dell'autorizzazione per gestire i tag.

Assegnazione di tag alle risorse

I tag sono disponibili per le risorse Amazon EKS seguenti:

- cluster
- gruppi di nodi gestiti
- profili Fargate

Puoi aggiungere un tag a queste risorse utilizzando le opzioni seguenti:

- Se utilizzi la console Amazon EKS, puoi applicare tag alle risorse nuove o esistenti in qualsiasi momento. A tale scopo, è possibile utilizzare la sheda Tag nella pagina della risorsa interessata. Per ulteriori informazioni, consulta the section called "Utilizzo di tag tramite la console".
- Se utilizzi `eksctl`, puoi applicare i tag alle risorse quando vengono create utilizzando l'opzione `--tags`.
- Se utilizzi l'AWS CLI, l'API Amazon EKS o un SDK AWS, è possibile applicare i tag alle nuove risorse mediante il parametro `tags` nell'operazione API rilevante. Inoltre, è possibile applicare tag a risorse utilizzando l'operazione API `TagResource`. Per ulteriori informazioni, consultare TagResource.

Quando utilizzi alcune azioni per la creazione di risorse, puoi anche specificare i tag per la risorsa nello stesso momento in cui la crei. Se i tag non possono essere applicati durante la creazione di una risorsa, la risorsa non viene creata. Mediante questo meccanismo ti assicuri che le risorse a cui desideri applicare tag al momento della creazione vengono create con tag specifici o non vengono create affatto. Se aggiungi tag alle risorse al momento della creazione, non è necessario eseguire script di assegnazione di tag personalizzati dopo la creazione di una risorsa.

I tag non si propagano ad altre risorse associate alla risorsa creata. Ad esempio, i tag del profilo di Fargate non vengono propagati ad altre risorse associate a tale profilo, come i pod pianificati con esso.

Limitazioni applicate ai tag

Ai tag si applicano le limitazioni seguenti:

- È possibile associare un massimo di 50 tag a una risorsa.
- Le chiavi dei tag non possono essere ripetute per una risorsa. Ogni chiave di tag deve essere univoca e può avere un solo valore.
- Le chiavi di tag possono contenere fino a 128 caratteri in UTF-8.
- Ogni valore può contenere fino a 256 caratteri UTF-8.
- Se più servizi AWS e risorse utilizzano il tuo schema di tagging, limita i tipi di caratteri che usi. Alcuni servizi potrebbero avere restrizioni sui caratteri consentiti. I caratteri generalmente consentiti sono lettere, numeri, spazi e i simboli seguenti: `+ - = . _ : / @`.
- I valori e le chiavi dei tag rispettano la distinzione tra maiuscole e minuscole.
- Non utilizzare `aws:`, `AWS:` o qualsiasi combinazione di maiuscole o minuscole come prefisso per chiavi o valori. Questi sono prenotati solo per l'uso AWS. Non puoi modificare né eliminare le chiavi o i valori di tag con tale prefisso. I tag con questo prefisso non vengono conteggiati per il limite del numero di tag per risorsa.

Tagging delle risorse per la fatturazione

Quando applichi i tag ai cluster Amazon EKS, puoi utilizzarli per l'allocazione dei costi nel tuo Rapporti su costi e utilizzo. I dati di misurazione nel Rapporti su costi e utilizzo mostrano l'utilizzo in tutti i processi dei cluster di Amazon EKS. Per ulteriori informazioni, consulta AWS cost and usage report nella Guida per l'utente della fatturazione AWS.

Il tag di allocazione dei costi generato da AWS, specificamente `aws:eks:cluster-name`, consente di suddividere i costi delle istanze Amazon EC2 per ogni singolo cluster Amazon EKS in Cost Explorer. Tuttavia, questo tag non rileva le spese del piano di controllo (control-plane). Il tag viene aggiunto automaticamente alle istanze Amazon EC2 che fanno parte di un cluster Amazon EKS. Questo comportamento si verifica indipendentemente dal fatto che venga eseguito il provisioning delle istanze tramite gruppi di nodi gestiti da Amazon EKS, Karpenter o direttamente con Amazon EC2. Questo tag specifico non viene conteggiato ai fini del limite di 50 tag. Per utilizzare il tag, il

proprietario dell'account deve attivarlo nella Console AWS per la fatturazione o utilizzando l'API. Quando il proprietario di un account di gestione di AWS Organizations attiva il tag, questo viene attivato anche per tutti gli account membri dell'organizzazione.

Puoi organizzare le informazioni di fatturazione in base alle risorse con gli stessi valori di chiave di tag. Puoi ad esempio applicare tag a numerose risorse con un nome di applicazione specifico, quindi organizzare le informazioni di fatturazione. In questo modo, puoi visualizzare il costo totale dell'applicazione in più servizi. Per ulteriori informazioni sulla configurazione di un report di allocazione dei costi mediante i tag, consulta The Monthly Cost Allocation Report nella Guida per l'utente della fatturazione AWS.

> **Note**
>
> Se hai appena abilitato la reportistica, i dati relativi al mese corrente saranno disponibili per la visualizzazione dopo 24 ore.

Cost Explorer è uno strumento di reporting disponibile come parte di Piano gratuito AWS. È possibile utilizzare Cost Explorer per visualizzare i grafici delle tue risorse Amazon EKS degli ultimi 13 mesi. Puoi anche prevedere le spese per i prossimi tre mesi. Puoi visualizzare i modelli relativi a quanto spendi in risorse AWS nel tempo. Ad esempio, puoi utilizzarlo per identificare aree che richiedono ulteriore studio e visualizzare le tendenze che puoi utilizzare per comprendere i costi. Puoi anche specificare intervalli di tempo per i dati e visualizzare i dati temporali per mese o per giorno.

Utilizzo di tag tramite la console

Con la console Amazon EKS è possibile gestire i tag associati ai gruppi di nodi gestiti o ai cluster nuovi o esistenti.

Quando selezioni una pagina relativa alle risorse nella console Amazon EKS, viene visualizzato l'elenco delle risorse corrispondenti. Ad esempio, se nel riquadro di navigazione sinistro si seleziona Cluster, nella console viene visualizzato l'elenco dei cluster Amazon EKS. Quando si seleziona una risorsa in uno di questi elenchi, (ad esempio un cluster specifico), se la risorsa supporta i tag, sarà possibile visualizzare e gestire i tag nella scheda Tag.

L'Editor di tag della Console di gestione AWS consente inoltre di gestire i tag modo unificato. Per ulteriori informazioni, consulta Tagging your AWS resources with Tag Editor nella Guida per l'utente dell'editor di tag AWS.

Aggiunta di tag a una risorsa in fase di creazione

É possibile aggiungere tag ai gruppi di nodi gestiti, ai cluster Amazon EKS ed ai profili Fargate durante la creazione. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ".

Aggiunta ed eliminazione di tag in una risorsa

Puoi aggiungere o eliminare i tag associati ai cluster direttamente dalla pagina della risorsa.

1. Aprire la Console Amazon EKS.
2. Seleziona la Regione AWS da utilizzare nella barra di navigazione.
3. Nel pannello di navigazione a sinistra, seleziona Cluster.
4. Scegliere un cluster specifico.
5. Scegliere la scheda Tags (Tag) quindi scegliere Manage tags (Gestisci tag).
6. Nella pagina Gestisci i tag, aggiungi o elimina i tag in base alle esigenze.
   - Per aggiungere un tag, scegli Add tag (Aggiungi tag). Specificare la chiave e il valore per ogni tag.
   - Per eliminare un tag, scegli Rimuovi tag.
7. Ripeti la procedura per ogni tag da aggiungere o eliminare.
8. Scegliere Update (Aggiorna) per terminare.

Utilizzo di tag tramite la CLI, l'API o `eksctl`

Utilizza i seguenti comandi dell'AWS CLI oppure le operazioni API Amazon EKS per aggiungere, aggiornare, elencare ed eliminare i tag per le risorse. È possibile utilizzare solo `eksctl` per aggiungere tag e contemporaneamente creare le nuove risorse con un unico comando.

Attività	CLI AWS	AWS Tools for Windows PowerShell	Azione API
Aggiungere sovrascrivere uno o più tag.	tag-resource	Add-EKSResourceTag	TagResource
Eliminare uno o più tag.	untag-resource	Remove-EKSResourceTag	UntagResource

I seguenti esempi mostrano come aggiungere o rimuovere tag alle o dalle risorse utilizzando l'AWS CLI.

Esempio 1: applicazione di un tag a un cluster esistente

Il comando seguente applica un tag a un cluster esistente.

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

Esempio 2: rimozione di un tag a un cluster esistente

Il comando seguente elimina un tag da un cluster esistente.

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Esempio 3: elencazione dei tag di una risorsa

Il comando seguente elenca i tag associati a una risorsa esistente.

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

Quando utilizzi alcune azioni per la creazione di risorse, puoi anche specificare i tag per la risorsa nello stesso momento in cui la crei. Le seguenti operazioni supportano l'assegnazione di tag durante la creazione di una risorsa.

Attività	CLI AWS	AWS Tools for Windows PowerShell	Azione API	eksctl
Creazione di un cluster	create-cluster	New-EKSCluster	CreateCluster	`create cluster`
Creazione di un gruppo di nodi gestito*	create-nodegroup	New-EKSNodegroup	CreateNodegroup	`create nodegroup`
Creazione di un profilo Fargate	create-fargate-profile	New-EKSFargateProfile	CreateFargateProfile.html	`create fargatepr ofile`

- Se desideri taggare anche le istanze Amazon EC2 durante la creazione di un gruppo di nodi gestito, crea il gruppo di nodi gestito utilizzando un modello di avvio. Per ulteriori informazioni, consulta the section called "Assegnazione di tag a istanze Amazon EC2". Se le istanze esistono già, è possibile taggare manualmente le istanze. Per ulteriori informazioni, consulta Tagging your resources nella Guida per l'utente di Amazon EC2.

Visualizzazione e gestione di quote di servizio di Amazon EKS e Fargate

Amazon EKS si è integrato con Service Quotas, un AWS servizio che puoi utilizzare per visualizzare e gestire le quote da una posizione centrale. Per ulteriori informazioni, consulta Cos'è Service Quotas? nella Guida per l'utente di Service Quotas. Con l'integrazione di Service Quotas, puoi cercare rapidamente il valore delle tue quote di servizio Amazon EKS e AWS Fargate utilizzando la CLI e. Console di gestione AWS AWS

Visualizza le quote dei servizi EKS nel Console di gestione AWS

1. Apri la console Service Quotas.
2. Nel riquadro di navigazione a sinistra, scegli AWS servizi.
3. Dall'elenco dei AWS servizi, cerca e seleziona Amazon Elastic Kubernetes Service (Amazon EKS) o Fargate. AWS

   Nell'elenco delle quote di servizio, puoi visualizzare il nome della quota di servizio, il valore applicato (se disponibile), la quota AWS predefinita e se il valore della quota è regolabile.
4. Per visualizzare ulteriori informazioni su una quota di servizio, ad esempio la descrizione, scegli il nome della quota.
5. (Facoltativo) Per richiedere un aumento della quota, seleziona la quota che desideri aumentare, seleziona Richiedi un aumento della quota, inserisci o seleziona le informazioni richieste e seleziona Richiedi.

Per utilizzare meglio le quote di servizio utilizzando la Console di gestione AWS, consulta la Service Quotas User Guide. Per richiedere un aumento delle quote, consultare Richiesta di aumento delle quote nella Guida per l'utente di Service Quotas.

Visualizza le quote dei servizi EKS con la CLI AWS

Eseguire questo comando per visualizzare le quote 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
```

Eseguire questo comando per visualizzare le quote di 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**
>
> La quota restituita è il numero di attività Amazon ECS o Amazon EKS Pods che possono essere eseguite contemporaneamente su Fargate in questo account nella regione corrente. AWS

Per lavorare di più con le quote di servizio utilizzando la AWS CLI, consulta service-quotas nel AWS CLI Command Reference. Per richiedere un aumento della quota, consulta il request-service-quota-increasecomando nella AWS CLI Command Reference.

Service Quotas di Amazon EKS

AWS consiglia di utilizzare il Console di gestione AWS per visualizzare le quote correnti. Per ulteriori informazioni, consulta the section called "Visualizza le quote dei servizi EKS nel Console di gestione AWS".

Per visualizzare le quote di servizio EKS predefinite, consulta gli endpoint e le quote di Amazon Elastic Kubernetes Service nella Guida generale. AWS

Queste quote del servizio sono elencate in Amazon Elastic Kubernetes Service (Amazon EKS) nella console Service Quotas. Per chiedere un incremento della quota per i valori indicati come regolabili, consulta Richiesta di incremento di una quota nella Guida per l'utente di Service Quotas.

> **Note**
>
> Le modifiche ai seguenti componenti non sono supportate in Service Quotas: * Associazioni di Pod Identity per cluster. the section called "Pod Identity"Per i limiti, consulta. * CIDRs per reti a nodi remoti o reti pod remoti per nodi ibridi. Per i limiti, vederethe section called "Nodi ibridi".

AWS Quote di servizio Fargate

Il servizio AWS Fargate nella console Service Quotas elenca diverse quote di servizio. È possibile configurare gli allarmi che avvisano quando l'uso si avvicina a una quota di servizio. Per ulteriori informazioni, consulta the section called "Creazione di un CloudWatch allarme per monitorare le metriche di utilizzo delle risorse di Fargate".

AWS I nuovi account potrebbero avere quote iniziali inferiori che possono aumentare nel tempo. Fargate monitora costantemente l'utilizzo dell'account all'interno di ciascuna AWS regione, quindi aumenta automaticamente le quote in base all'utilizzo. Puoi chiedere anche un incremento della quota per i valori indicati come regolabili. Per ulteriori informazioni, consulta Richiesta di un aumento delle quote nella Guida per l'utente di Service Quotas.

AWS consiglia di utilizzare il per visualizzare le quote correnti Console di gestione AWS . Per ulteriori informazioni, consulta the section called "Visualizza le quote dei servizi EKS nel Console di gestione AWS".

Per visualizzare le quote di servizio AWS Fargate su EKS predefinite, vedere le quote di servizio Fargate nella Guida generale. AWS

> **Note**
>
> Inoltre, Fargate applica attività Amazon ECS e quote di velocità di avvio dei pod Amazon EKS. Per ulteriori informazioni, consulta Quote di limitazione (della larghezza di banda della rete AWS Fargate nella Guida per Amazon ECS.

Sicurezza in Amazon EKS

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. Per Amazon EKS, AWS è responsabile del piano di controllo Kubernetes, che include i nodi del piano di controllo e il database. `etcd` I revisori di terze parti testano e verificano regolarmente l'efficacia della sicurezza come parte dei programmi di conformitàAWS. Per ulteriori informazioni sui programmi di conformità che si applicano ad Amazon EKS, consultare Servizi AWS coperti dal programma di conformità.
- Sicurezza nel cloud - Le seguenti aree sono sotto la responsabilità dell'utente.
  - La configurazione di sicurezza del piano dei dati, compresa la configurazione dei gruppi di sicurezza che autorizzano il passaggio del traffico dal piano di controllo Amazon EKS al VPC del cliente
  - La configurazione dei nodi di lavoro e degli stessi container
  - Il sistema operativo che ospita il nodo di lavoro (inclusi gli aggiornamenti e le patch di sicurezza)
  - Altri software applicativi associati:
    - La configurazione e la gestione dei controlli di rete, come le regole del firewall
    - La gestione di identità a livello di piattaforma e la gestione degli accessi, con o in aggiunta alle IAM
  - La riservatezza dei dati, i requisiti dell'azienda e le leggi e le normative applicabili.

Amazon EKS è certificato da svariati programmi di conformità per applicazioni regolamentate e sensibili. Amazon EKS è conforme a SOC, PCI, ISO, FedRAMP-Moderate, IRAP, C5, K-ISMS, ENS High, OSPAR, HITRUST CSF, ed è un servizio idoneo allo standard HIPAA. Per ulteriori informazioni, consulta Gestione dell'accesso.

Questa documentazione aiuta a comprendere come applicare il modello di responsabilità condivisa quando si utilizza Amazon EKS. Gli argomenti seguenti descrivono come configurare Amazon EKS

per soddisfare gli obiettivi di sicurezza e conformità. Scopri anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse Amazon EKS.

> **Note**
>
> I container Linux sono costituiti da gruppi di controllo (cgroup) e namespace che aiutano a limitare l'accesso a cui un container può accedere, ma tutti i contenitori condividono lo stesso kernel Linux dell'istanza Amazon host. EC2 L'esecuzione di un container come utente root (UID 0), o la concessione di un accesso al container a risorse host o namespace, come la rete host o il namespace PID host sono fortemente sconsigliati, in quanto ciò riduce l'efficacia dell'isolamento fornito dai container.

Argomenti

- Protezione dei cluster Amazon EKS con le best practice
- Analizzare le vulnerabilità in Amazon EKS
- Convalida della conformità per cluster di Amazon EKS
- Considerazioni di sicurezza per Amazon Elastic Container Service per Kubernetes
- Considerazioni sulla sicurezza per Kubernetes
- Considerazioni sulla sicurezza per Amazon EKS Auto Mode
- Considerazioni sulla sicurezza per EKS Capabilities
- Identity and Access Management per Amazon EKS

Protezione dei cluster Amazon EKS con le best practice

Le best practice di sicurezza di Amazon EKS si trovano nelle Best practice per la sicurezza della Guida delle best practice di Amazon EKS.

Analizzare le vulnerabilità in Amazon EKS

La sicurezza è una considerazione fondamentale per la configurazione e la manutenzione di cluster e applicazioni Kubernetes. Di seguito sono elencate le risorse per analizzare la configurazione di sicurezza dei cluster EKS, le risorse per verificare la presenza di vulnerabilità e le integrazioni con i servizi AWS che possono eseguire l'analisi.

Il Center for Internet Security (CIS) Benchmark per Amazon EKS

Il Center for Internet Security (CIS) Benchmark fornisce indicazioni per le configurazioni di sicurezza di Amazon EKS. Il benchmark:

- È applicabile ai nodi Amazon EC2 (gestiti e autogestiti) in cui sei responsabile delle configurazioni di sicurezza dei componenti Kubernetes.
- Fornisce un modo standard approvato dalla community per garantire di aver configurato in modo sicuro il cluster Kubernetes e i nodi quando si utilizza Amazon EKS.
- È costituito da quattro sezioni: configurazione della registrazione del piano di controllo, configurazioni di sicurezza dei nodi, criteri e servizi gestiti.
- Supporta tutte le versioni Kubernetes attualmente disponibili in Amazon EKS e può essere eseguito utilizzando kube-bench, uno strumento open source standard per il controllo della configurazione utilizzando il benchmark CIS sui cluster Kubernetes.

Per ulteriori informazioni, consulta Presentazione del benchmark CIS Amazon EKS.

Per una pipeline `aws-sample` automatizzata al fine di aggiornare il gruppo di nodi con un AMI con benchmark CIS, consulta EKS-Optimized AMI Hardening Pipeline.

Versioni della piattaforma Amazon EKS

Le versioni della piattaforma Amazon EKS rappresentano le funzionalità del piano di controllo del cluster, tra cui i flag del server API di Kubernetes abilitati e l'attuale versione della patch di Kubernetes. I nuovi cluster vengono implementati con la versione più recente della piattaforma. Per i dettagli, consulta EKS platform-versions.

È possibile aggiornare un cluster Amazon EKS alla versione più recente di Kubernetes. Man mano che nuove versioni di Kubernetes diventano disponibili in Amazon EKS, ti consigliamo di aggiornare tempestivamente i cluster in modo da usare la versione più recente disponibile. Per ulteriori informazioni sulle versioni di Kubernetes in EKS, consulta Amazon EKS supported versions.

Elenco delle vulnerabilità del sistema operativo

Elenco vulnerabilità di AL2023

Tieni traccia degli eventi di sicurezza o di privacy per Amazon Linux 2023 nel Centro di sicurezza di Amazon Linux o iscriviti al feed RSS associato. Gli eventi relativi alla sicurezza e alla privacy

includono una panoramica del problema, i pacchetti e le istruzioni per l'aggiornamento delle istanze per correggere il problema.

Elenco delle vulnerabilità di Amazon Linux 2

Tieni traccia degli eventi di sicurezza o di privacy per Amazon Linux 2 nel Centro di sicurezza di Amazon Linux o iscriviti al feed RSS associato. Gli eventi relativi alla sicurezza e alla privacy includono una panoramica del problema, i pacchetti e le istruzioni per l'aggiornamento delle istanze per correggere il problema.

Rilevamento dei nodi con Amazon Inspector

È possibile utilizzare Amazon Inspector per verificare la presenza di accessibilità alla rete non intenzionale dei nodi e le vulnerabilità sulle istanze Amazon EC2.

Rilevamento di cluster e nodi con Amazon GuardDuty

Amazon GuardDuty è un servizio di rilevamento delle minacce che monitora continuamente gli account, i container, i carichi di lavoro e i dati all'interno dell'ambiente AWS. Tra le altre funzionalità, GuardDuty offre le due funzionalità seguenti che rilevano potenziali minacce ai cluster EKS: Protezione EKS e Monitoraggio del runtime.

Per ulteriori informazioni, consulta the section called "Amazon GuardDuty".

Convalida della conformità per cluster di Amazon EKS

Per sapere se un servizio AWS è coperto da programmi di conformità specifici, consulta AWS services in Scope by Compliance Program e scegli il programma di conformità desiderato. Per informazioni generali, consulta Programmi di conformità di AWS.

Puoi scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Downloading Reports in AWS Artifact.

La tua responsabilità per la compliance quando si utilizzano i servizi AWS è determinata dalla riservatezza dei dati, dagli obiettivi di conformità dell'azienda e dalle normative vigenti. Per semplificare il rispetto della conformità, AWS offre i seguenti servizi:

- Governance e conformità per la sicurezza: queste guide all'implementazione di soluzioni illustrano considerazioni relative all'architettura e i passaggi per implementare le funzionalità di sicurezza e conformità.

- Riferimenti sui servizi conformi ai requisiti HIPAA: elenca i servizi HIPAA idonei. Non tutti i servizi AWS sono conformi ai requisiti HIPAA.
- Risorse per la conformità di AWS: questa raccolta di workbook e guide potrebbe essere utile al settore e alla posizione.
- AWSGuide alla conformità dei clienti: comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione dei servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).
- Valutazione delle risorse con le regole nella Guida per gli sviluppatori di AWS: il servizio AWS Config valuta il livello di conformità delle configurazioni delle risorse con pratiche interne, linee guida e regolamenti industriali.
- AWS Security Hub: questo servizio AWS fornisce una vista completa dello stato di sicurezza in AWS. Security Hub utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.
- Amazon GuardDuty: questo servizio AWS rileva potenziali minacce ad account AWS, carichi di lavoro, container e dati monitorando l'ambiente alla ricerca di attività sospette e dannose. GuardDuty soddisfa i requisiti di rilevamento delle intrusioni imposti da determinati framework di conformità e garantisce il rispetto di vari standard come il PCI DSS.
- AWS Audit Manager: questo servizio AWS aiuta a controllare continuamente l'utilizzo di AWS per semplificare la gestione dei rischi e la conformità alle normative e agli standard di settore.

Considerazioni di sicurezza per Amazon Elastic Container Service per Kubernetes

Di seguito sono riportate alcune considerazioni relative alla sicurezza del cloud, in quanto influiscono su Amazon EKS.

Argomenti

- Sicurezza dell'infrastruttura in Amazon EKS
- Comprendere la resilienza nei cluster Amazon EKS
- Prevenzione del problema "confused deputy" tra servizi in Amazon EKS

Sicurezza dell'infrastruttura in Amazon EKS

In quanto servizio gestito, Amazon Elastic Kubernetes Service è protetto dalla sicurezza di rete globale di AWS. Per informazioni sui servizi di sicurezza AWSe su come AWSprotegge l'infrastruttura, consulta la pagina Sicurezza del cloud AWS. Per progettare l'ambiente AWSutilizzando le best practice per la sicurezza dell'infrastruttura, consulta la pagina Protezione dell'infrastruttura nel Pilastro della sicurezza di AWSWell-Architected Framework.

Utilizzare le chiamate API pubblicate di AWS per accedere a Amazon EKS tramite la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa, puoi utilizzare AWS Security Token Service (AWS STS) per generare le credenziali di sicurezza temporanee per firmare le richieste.

Quando si crea un cluster Amazon EKS, specificare anche le sottoreti VPC; che devono essere utilizzate dal cluster. Amazon EKS richiede sottoreti in almeno due zone di disponibilità. È consigliabile un VPC con sottoreti pubbliche e private in modo che Kubernetes possa creare bilanciatori del carico pubblici nelle sottoreti pubbliche, che bilancino il carico del traffico ai pod in esecuzione su nodi che si trovano in sottoreti private.

Per ulteriori informazioni sulle considerazioni per il VPC, consultare the section called "Requisiti del VPC e delle sottoreti".

Se si crea il proprio VPC e i gruppi di nodi con i modelli AWS CloudFormation forniti nella procedura dettagliata Get started with Amazon EKS, il piano di controllo e i gruppi di sicurezza dei nodi di lavoro vengono configurati con le impostazioni consigliate.

Per ulteriori informazioni sulle considerazioni per i gruppi di sicurezza, consultare the section called "Requisiti relativi al gruppo di sicurezza".

Quando si crea un nuovo cluster, Amazon EKS crea un endpoint per il server API Kubernetes gestito utilizzato per comunicare con il cluster (usando strumenti di gestione Kubernetes, ad esempio `kubectl`). Per impostazione predefinita, questo endpoint del server API è pubblico in Internet

e l'accesso al server API è protetto utilizzando una combinazione di AWS Identity and Access Management (AWS IAM) e dal controllo degli accessi basato sul ruolo (RBAC) Kubernetes nativo.

Puoi abilitare l'accesso privato al server API Kubernetes in modo che tutte le comunicazioni tra i nodi di lavoro e il server API rimangano all'interno del VPC. È possibile limitare gli indirizzi IP che possono accedere al server API da Internet o disabilitare completamente l'accesso a Internet al server API.

Per ulteriori informazioni sulla modifica dell'accesso all'endpoint del cluster, consultare the section called "Modifica dell'accesso all'endpoint del cluster".

È possibile implementare policy di rete Kubernetes con CNI di Amazon VPC o strumenti di terze parti come Project Calico. Per ulteriori informazioni sull'utilizzo della CNI di Amazon VPC per le policy di rete, consulta la pagina the section called "Policy di Kubernetes". Project Calico è un progetto open source di terze parti. Per ulteriori informazioni, consulta la documentazione di Project Calico.

Accesso ad Amazon EKS tramite AWS PrivateLink

È possibile utilizzare AWS PrivateLink per creare una connessione privata tra il tuo VPC e Amazon Elastic Kubernetes Service. È possibile accedere ad Amazon EKS come se fosse nel tuo VPC, senza utilizzare un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione a collegamento diretto AWS. Le istanze nel tuo VPC non richiedono indirizzi IP pubblici per l'accesso ad Amazon EKS.

Stabilisci questa connessione privata creando un endpoint di interfaccia alimentato da AWS PrivateLink. In ciascuna sottorete viene creata un'interfaccia di rete endpoint da abilitare per l'endpoint di interfaccia. Queste sono interfacce di rete gestite dal richiedente che fungono da punto di ingresso per il traffico destinato ad Amazon EKS.

Per ulteriori informazioni, consulta Accedere ai servizi AWS tramite AWS PrivateLink nella Guida AWS PrivateLink.

Prima di iniziare

Prima di iniziare, assicurati di aver eseguito le seguenti attività:

- Esamina Accesso a un servizio AWS utilizzando un endpoint VPC nella Guida AWS PrivateLink

Considerazioni

- Supporto e limitazioni: gli endpoint dell'interfaccia Amazon EKS consentono l'accesso sicuro a tutte le azioni dell'API Amazon EKS dal tuo VPC, ma presentano limitazioni specifiche, ovvero non

supportano l'accesso alle API Kubernetes, poiché dispongono di un endpoint privato separato né è possibile configurare Amazon EKS in modo che sia accessibile solo tramite l'endpoint di interfaccia.

- Prezzi: l'utilizzo degli endpoint di interfaccia per Amazon EKS comporta tariffe AWS PrivateLink standard: tariffe orarie per ciascun endpoint fornito in ciascuna zona di disponibilità, costi di elaborazione dei dati per il traffico attraverso l'endpoint. Per ulteriori informazioni, consulta Prezzi AWS PrivateLink.
- Sicurezza e controllo degli accessi: consigliamo di migliorare la sicurezza e il controllo dell'accesso con queste configurazioni aggiuntive: utilizza le policy degli endpoint VPC per controllare l'accesso ad Amazon EKS tramite l'endpoint dell'interfaccia, associa i gruppi di sicurezza alle interfacce di rete degli endpoint per gestire il traffico, utilizza i log di flusso VPC per acquisire e monitorare il traffico IP da e verso gli endpoint dell'interfaccia, con log pubblicabili su Amazon CloudWatch o Amazon S3. Per ulteriori informazioni, consulta Controllare l'accesso agli endpoint VPC utilizzando le policy degli endpoint e Registrare il traffico IP utilizzando log di flusso VPC.
- Opzioni di connettività: gli endpoint di interfaccia offrono opzioni di connettività flessibili utilizzando l'accesso on-premises (connetti il data center on-premises a un VPC con l'endpoint dell'interfaccia utilizzando AWS Direct Connect o AWS VPN da sito a sito) o tramite connettività inter-VPC (utilizza AWS Transit Gateway o il peering VPC per connettere altri VPC al VPC con l'endpoint di interfaccia, mantenendo il traffico all'interno della rete AWS).
- Supporto versione IP: gli endpoint creati prima di agosto 2024 supportano solo IPv4 utilizzando eks.region.amazonaws.com. I nuovi endpoint creati dopo agosto 2024 supportano IPv4 e IPv6 dual-stack (ad esempio, eks.region.amazonaws.com, eks.region.api.aws).
- Disponibilità regionale: AWS PrivateLink per l'API EKS non è disponibile nelle regioni Asia Pacifico (Malesia) (ap-southeast-5), Asia Pacifico (Thailandia) (ap-southeast-7), Messico (Centrale) (mx-central-1) e Asia Pacifica (Taipei) (ap-east-2). AWS Il supporto PrivateLink per eks-auth (EKS Pod Identity) è disponibile nella regione Asia Pacifico (Malesia) (ap-southeast-5).

Crea un endpoint di interfaccia per Amazon EKS

È possibile creare un endpoint di interfaccia per Amazon EKS utilizzando la console Amazon VPC o l'interfaccia a riga di comando AWS (AWS CLI). Per ulteriori informazioni, consulta Creare un endpoint VPC nella Guida di AWS PrivateLink.

Crea un endpoint di interfaccia per Amazon EKS utilizzando i seguenti nomi servizio:

API EKS

- com.amazonaws.region-code.eks

- com.amazonaws.region-code.eks-fips (per endpoint conformi a FIPS)

API di autenticazione EKS (Pod Identity EKS)

- com.amazonaws.region-code.eks-auth

Funzionalità DNS privata per gli endpoint dell'interfaccia Amazon EKS

La funzionalità DNS privata, abilitata per impostazione predefinita per gli endpoint di interfaccia di Amazon EKS e altri servizi AWS, facilita le richieste API sicure e private utilizzando nomi DNS regionali predefiniti. Questa funzionalità garantisce che le chiamate API siano instradate attraverso l'endpoint di interfaccia sulla rete AWS privata, migliorando la sicurezza e le prestazioni.

La funzionalità DNS privata si attiva automaticamente quando si crea un endpoint di interfaccia per Amazon EKS o altri servizi AWS. Per abilitarla, devi configurare correttamente il tuo VPC impostando attributi specifici:

- enableDnsHostnames: consente alle istanze all'interno del VPC di avere nomi host DNS.
- enableDnsSupport: abilita la risoluzione DNS in tutto il VPC.

Per istruzioni dettagliate per verificare o modificare queste impostazioni, consulta Visualizzare e aggiornare gli attributi DNS per il VPC.

Nomi DNS e tipi di indirizzi IP

Con la funzionalità DNS privata abilitata, è possibile utilizzare nomi DNS specifici per connetterti ad Amazon EKS e queste opzioni si evolvono nel tempo:

- eks.region.amazonaws.com: il nome DNS tradizionale, che si risolve solo negli indirizzi IPv4 prima di agosto 2024. Per gli endpoint esistenti aggiornati al dual-stack, questo nome è risolto sia in indirizzi IPv4 che IPv6.
- eks.region.api.aws: disponibile per i nuovi endpoint creati dopo agosto 2024, questo nome DNS dual-stack si risolve in indirizzi IPv4 e IPv6.

Dopo agosto 2024, i nuovi endpoint di interfaccia sono forniti con due nomi DNS ed è possibile optare per il tipo di indirizzo IP dual-stack. Per gli endpoint esistenti, l'aggiornamento a dual-stack modifica eks.region.amazonaws.com in modo che supporti sia IPv4 che IPv6.

Utilizzo della funzionalità DNS privata

Una volta configurata, la funzionalità DNS privata può essere integrata nei flussi di lavoro, offrendo le seguenti funzionalità:

- Richieste API: utilizza i nomi DNS regionali predefiniti, `eks.region.amazonaws.com` o `eks.region.api.aws`, in base alla configurazione dell'endpoint per effettuare richieste API ad Amazon EKS.
- Compatibilità delle applicazioni: le applicazioni esistenti che chiamano le API EKS non richiedono modifiche per sfruttare questa funzionalità.
- AWS CLI con Dual-Stack: per utilizzare gli endpoint dual-stack con AWS CLI, consulta la configurazione degli endpoint Dual-stack e FIPS nella Guida di riferimento agli SDK e agli strumenti AWS.
- Routing automatico: qualsiasi chiamata all'endpoint del servizio predefinito di Amazon EKS è instradata automaticamente tramite l'endpoint di interfaccia, garantendo una connettività privata e sicura.

Comprendere la resilienza nei cluster Amazon EKS

L'infrastruttura globale di AWS è basata su Regioni e zone di disponibilità AWS. Le Regioni AWS forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite reti altamente ridondanti, a bassa latenza e con throughput elevato. Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, fault tolerant e scalabili rispetto alle infrastrutture a data center singolo o multiplo.

Amazon EKS esegue ed effettua il dimensionamento di istanze del piano di controllo Kubernetes in molteplici zone di disponibilità AWS per garantire un'elevata disponibilità. Amazon EKS dimensiona automaticamente le istanze del piano di controllo in base al carico, rileva e sostituisce le istanze del piano di controllo non integre e fornisce loro le patch automaticamente. Dopo aver avviato un aggiornamento della versione, Amazon EKS aggiorna il piano di controllo per conto dell'utente, mantenendone un'elevata disponibilità durante l'aggiornamento.

Questo piano di controllo (control-plane) è costituito da almeno due istanze del server API e tre istanze `etcd` eseguite su tre zone di disponibilità all'interno di una Regione AWS. Amazon EKS:

- Monitora attivamente il carico sulle istanze del piano di controllo e le ridimensiona automaticamente per garantire prestazioni elevate.

- Rileva e sostituisce automaticamente le istanze del piano di controllo (control-plane) non integre, riavviandole nelle zone di disponibilità all'interno della Regione AWS in base alle esigenze.
- Sfrutta l'architettura delle Regioni AWS al fine di mantenere elevata disponibilità. Per questo motivo, Amazon EKS è in grado di offrire un SLA per la disponibilità dell'endpoint del server API.

Per ulteriori informazioni sulle regioni e le zone di disponibilità AWS, consultare Infrastruttura globale di AWS.

Prevenzione del problema "confused deputy" tra servizi in Amazon EKS

Con "confused deputy" si intende un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire una certa operazione può costringere un'entità con più privilegi a eseguire tale operazione. Nel AWS, l'impersonificazione tra servizi può portare al confuso problema del vice. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare che ciò accada, AWS mette a disposizione strumenti che consentono di proteggere i dati relativi a tutti i servizi, con responsabili del servizio a cui è stato concesso l'accesso alle risorse del vostro account.

Consigliamo di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy delle risorse per limitare le autorizzazioni con cui Amazon Elastic Kubernetes Service (Amazon EKS) fornisce un altro servizio alla risorsa.

`aws:SourceArn`

Utilizza `aws:SourceArn` per associare una sola risorsa all'accesso tra servizi.

`aws:SourceAccount`

Utilizza `aws:SourceAccount` se desideri consentire l'associazione di qualsiasi risorsa in tale account all'uso tra servizi.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. Se l'ARN completo della risorsa non è noto o si scelgono più risorse, utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con caratteri jolly (*) per le parti sconosciute dell'ARN. Ad esempio, `arn:aws:<servicename>:*:<123456789012>:*`.

Se il valore `aws:SourceArn` non contiene l'ID account, ad esempio un ARN di un bucket Amazon S3, è necessario utilizzare sia `aws:SourceAccount` che `aws:SourceArn` per limitare le autorizzazioni.

Prevenzione del problema "confused deputy" tra servizi nei cluster Amazon EKS

richiesto un ruolo IAM del cluster Amazon EKS per ogni cluster. I cluster Kubernetes gestiti da Amazon EKS utilizzano questo ruolo per gestire i nodi e il provider cloud legacy utilizza questo ruolo per creare bilanciatori del carico con Elastic Load Balancing per i servizi. Queste operazioni del cluster possono influire solo sullo stesso account, quindi consigliamo di limitare ogni ruolo del cluster a quel cluster e a quell'account. Questa è un'applicazione specifica della AWS raccomandazione di seguire il principio del privilegio minimo nel proprio account.

Formato dell'ARN dell'origine

Il valore di `aws:SourceArn` deve essere l'ARN di un cluster EKS nel formato `arn:aws:eks:`*`region`*`:`*`account`*`:cluster/`*`cluster-name`*. Ad esempio, `arn:aws: eks:us-west-2:123456789012:cluster/my-cluster`.

Formato della policy di attendibilità per i ruoli del cluster EKS

L'esempio seguente mostra il modo in cui è possibile utilizzare le chiavi di contesto `aws:SourceArn` e `aws:SourceAccount` delle condizioni globali in Amazon EKS per prevenire il problema "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"
        }
```

```
      }
    }
  ]
}
```

Considerazioni sulla sicurezza per Kubernetes

Di seguito sono riportate alcune considerazioni relative alla sicurezza del cloud, in quanto influiscono su Kubernetes nei cluster Amazon EKS. Per un'analisi approfondita dei controlli e delle pratiche di sicurezza in Kubernetes, consulta Sicurezza nativa del cloud e Kubernetes nella documentazione di Kubernetes.

Argomenti

- Proteggere i carichi di lavoro con i certificati Kubernetes
- Comprendere ruoli e utenti creati da Amazon EKS tramite RBAC
- Crittografia dei segreti Kubernetes con KMS su cluster esistenti
- Utilizzare i segreti di AWS Secrets Manager con i pod di Amazon EKS
- Crittografia a busta predefinita per tutti i dati dell'API Kubernetes

Proteggere i carichi di lavoro con i certificati Kubernetes

L'API Kubernetes Certificates automatizza il provisioning delle credenziali X.509. L'API presenta un'interfaccia a riga di comando per i client API Kubernetes che consente di richiedere e ottenere i certificati X.509 da parte di un'autorità di certificazione (CA). Puoi utilizzare la risorsa (CSR) `CertificateSigningRequest` per chiedere la firma del certificato da parte di un firmatario designato. Le richieste vengono approvate o negate prima della firma. Kubernetes supporta sia i firmatari integrati che personalizzati con comportamenti ben definiti. In questo modo, i client possono prevedere cosa succede alle loro CSR. Per ulteriori informazioni sulla firma dei certificati, consulta signing requests.

Uno dei firmatari integrati è `kubernetes.io/legacy-unknown`. L'API `v1beta1` della risorsa CSR ha onorato questo firmatario "legacy-unknown". Tuttavia, l'API `v1` stabile di CSR non consente l'impostazione di `signerName` su `kubernetes.io/legacy-unknown`.

Se desideri utilizzare l'autorità di certificazione (CA) di Amazon EKS per la generazione di certificati sul tuo cluster, devi utilizzare un firmatario personalizzato. Per utilizzare la versione `v1` dell'API

CSR e generare un nuovo certificato, devi eseguire la migrazione di tutti i manifesti e i client dell'API esistenti. I certificati esistenti creati con l'API `v1beta1` esistente sono validi e funzionanti fino alla scadenza dei certificati. Questo include gli output seguenti:

- Distribuzione di attendibilità: nessuna. Non esiste alcuna distribuzione di attendibilità o standard per questo firmatario in un cluster Kubernetes.
- Argomenti consentiti: tutti
- Estensioni x509 consentite: rispetta le estensioni di utilizzo delle chiavi e subjectAltName, scartando le altre.
- Utilizzo delle chiavi consentite: non devono includere altri utilizzi oltre ["key encipherment", "digital signature", "server auth"]

> **Note**
>
> La firma dei certificati client non è supportata.

- Scadenza/durata del certificato: 1 anno (predefinita e massima)
- Bit CA consentito/non consentito: non consentito

Generazione CSR di esempio con signerName

Questa procedura mostra come generare un certificato di servizio per il nome DNS `myserver.default.svc` utilizzando `signerName: beta.eks.amazonaws.com/app-serving`. Utilizzala come guida per il tuo ambiente.

1. Esegui il comando `openssl genrsa -out myserver.key 2048` per generare una chiave privata RSA.

```
openssl genrsa -out myserver.key 2048
```

2. Utilizza il comando seguente per generare una richiesta di certificato.

```
openssl req -new -key myserver.key -out myserver.csr -subj "/CN=myserver.default.svc"
```

3. Genera un valore `base64` per la richiesta CSR e memorizzalo in una variabile da utilizzare in un passaggio successivo.

```
base_64=$(cat myserver.csr | base64 -w 0 | tr -d "
```

```
")
```

4. Per creare un file denominato `mycsr.yaml`, esegui il comando di seguito. Nell'esempio seguente, `beta.eks.amazonaws.com/app-serving` è `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. Invia la CSR.

```
kubectl apply -f mycsr.yaml
```

6. Approva il certificato di servizio.

```
kubectl certificate approve myserver
```

7. Verifica l'emissione del certificato.

```
kubectl get csr myserver
```

   Di seguito viene riportato un output di esempio.

```
NAME       AGE     SIGNERNAME                              REQUESTOR          CONDITION
myserver   3m20s   beta.eks.amazonaws.com/app-serving   kubernetes-admin
 Approved,Issued
```

8. Esporta il certificato emesso.

```
kubectl get csr myserver -o jsonpath='{.status.certificate}'| base64 -d >
 myserver.crt
```

Comprendere ruoli e utenti creati da Amazon EKS tramite RBAC

Quando si crea un cluster Kubernetes, su tale cluster vengono create diverse identità Kubernetes predefinite per il corretto funzionamento di Kubernetes. Amazon EKS crea identità Kubernetes per ciascuno dei suoi componenti predefiniti. Le identità offrono un controllo delle autorizzazioni basato sul ruolo (RBAC) Kubernetes per i componenti del cluster. Per ulteriori informazioni, consulta Using RBAC Authorization nella documentazione di Kubernetes.

Quando si installano componenti aggiuntivi opzionali nel cluster, è possibile che vengano aggiunte identità Kubernetes supplementari al cluster. Per ulteriori informazioni sulle identità non trattate in questo argomento, consulta la documentazione del componente aggiuntivo.

Puoi visualizzare l'elenco delle identità Kubernetes create da Amazon EKS sul tuo cluster utilizzando lo strumento Console di gestione AWS o `kubectl` la riga di comando. Tutte le identità degli utenti vengono visualizzate nei registri di `kube` controllo disponibili tramite Amazon. CloudWatch

Console di gestione AWS

Prerequisito

Il principale IAM da utilizzare deve disporre delle autorizzazioni descritte alla pagina Required permissions.

Per visualizzare le identità create da Amazon EKS utilizzando Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Nell'elenco Cluster, seleziona il cluster contenente le identità da visualizzare.
3. Scegli la scheda Risorse.
4. In Resource types (Tipi di risorse), scegli Authorization (Autorizzazione).
5. Scegli, ClusterRoles, ClusterRoleBindings, Ruoli o RoleBindings. Tutte le risorse con il prefisso eks sono create da Amazon EKS. Le risorse di identità supplementari create da Amazon EKS sono:
   - La band ClusterRolesi ClusterRoleBindingchiama aws-node. Le risorse aws-node supportano il plugin CNI di Amazon VPC per Kubernetes, che Amazon EKS installa su tutti i cluster.
   - Un ClusterRolenome vpc-resource-controller-rolee un ClusterRoleBindingnome. vpc-resource-controller-rolebinding Queste risorse supportano il Amazon VPC resource controller (Controller risorse Amazon VPC), che Amazon EKS installa su tutti i cluster.

Oltre alle risorse visualizzate nella console, nel cluster esistono le seguenti identità utente speciali, anche se non sono visibili nella configurazione del cluster:

- **`eks:cluster-bootstrap`** : utilizzato per operazioni `kubectl` durante il bootstrap del cluster.
- **`eks:support-engineer`** : utilizzato per le operazioni di gestione del cluster.

6. Scegli una risorsa specifica per visualizzarne i dettagli. Per impostazione predefinita, le informazioni sono mostrate in Vista strutturata. Nell'angolo superiore destro della pagina dei dettagli puoi scegliere la Raw view (Visualizzazione non elaborata) per vedere tutte le informazioni per la risorsa.

Kubectl

Prerequisito

L'entità che utilizzi (AWS Identity and Access Management (IAM) o OpenID Connect (OIDC)) per elencare le risorse Kubernetes nel cluster deve essere autenticata da IAM o dal tuo provider di identità OIDC. All'entità devono essere concesse autorizzazioni per utilizzare i verbi Kubernetes `get` e `list` per le risorse `Role`, `ClusterRole`, `RoleBinding` e `ClusterRoleBinding` nel cluster che devono essere utilizzate dall'entità. Per ulteriori informazioni sulla concessione alle entità IAM dell'accesso al tuo cluster, consulta the section called "Accesso API Kubernetes". Per ulteriori informazioni sulla concessione dell'accesso al cluster alle entità autenticate dal gestore OIDC, consultare the section called "Collegare il provider OIDC".

Come visualizzare le identità create da Amazon EKS tramite **`kubectl`**

Esegui il comando per il tipo di risorsa da visualizzare. Tutte le risorse restituite con il prefisso eks sono create da Amazon EKS. Oltre alle risorse restituite nell'output dai comandi, nel cluster esistono le seguenti identità utente speciali, anche se non sono visibili nella configurazione del cluster:

- **`eks:cluster-bootstrap`** : utilizzato per operazioni `kubectl` durante il bootstrap del cluster.
- **`eks:support-engineer`** : utilizzato per le operazioni di gestione del cluster.

ClusterRoles— `ClusterRoles` sono limitati al cluster, quindi qualsiasi autorizzazione concessa a un ruolo si applica alle risorse in qualsiasi spazio dei nomi Kubernetes del cluster.

Il comando seguente restituisce tutti i `ClusterRoles` Kubernetes creati da Amazon EKS nel cluster.

```
kubectl get clusterroles | grep eks
```

Oltre ai `ClusterRoles` restituiti nell'output con prefisso, esistono i seguenti `ClusterRoles`.

- **`aws-node`** : questo `ClusterRole` supporta il plugin CNI di Amazon VPC per Kubernetes, che Amazon EKS installa su tutti i cluster.
- **`vpc-resource-controller-role`** : questo `ClusterRole` supporta il controller di risorse Amazon VPC, che Amazon EKS installa su tutti i cluster.

Per visualizzare la specifica di a`ClusterRole`, sostituiscila *`eks:k8s-metrics`* nel comando seguente con una `ClusterRole` restituita nell'output del comando precedente. L'esempio seguente restituisce la specifica per *`eks:k8s-metrics`*`ClusterRole`.

```
kubectl describe clusterrole eks:k8s-metrics
```

Di seguito viene riportato un output di esempio.

```
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` rientrano nell'ambito del cluster.

Il comando seguente restituisce tutti i `ClusterRoleBindings` Kubernetes creati da Amazon EKS nel cluster.

```
kubectl get clusterrolebindings | grep eks
```

Oltre ai `ClusterRoleBindings` restituiti nell'output, esistono i seguenti `ClusterRoleBindings`.

- **`aws-node`** : questo `ClusterRoleBinding` supporta il plugin CNI di Amazon VPC per Kubernetes, che Amazon EKS installa su tutti i cluster.

- **`vpc-resource-controller-rolebinding`** : questo `ClusterRoleBinding` supporta il controller di risorse Amazon VPC, che Amazon EKS installa su tutti i cluster.

Per visualizzare la specifica di a`ClusterRoleBinding`, sostituiscila *`eks:k8s-metrics`* nel comando seguente con una `ClusterRoleBinding` restituita nell'output del comando precedente. L'esempio seguente restituisce la specifica per *`eks:k8s-metrics`*`ClusterRoleBinding`.

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

Di seguito viene riportato un output di esempio.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Roles: i `Roles` sono racchiusi in un namespace Kubernetes. Tutti i `Roles` creati da Amazon EKS rientrano nell'ambito dello spazio nomi `kube-system`.

Il comando seguente restituisce tutti i `Roles` Kubernetes creati da Amazon EKS nel cluster.

```
kubectl get roles -n kube-system | grep eks
```

Per visualizzare la specifica di a`Role`, sostituiscila *`eks:k8s-metrics`* nel comando seguente con il nome di a `Role` restituito nell'output del comando precedente. L'esempio seguente restituisce la specifica per *`eks:k8s-metrics`*`Role`.

```
kubectl describe role eks:k8s-metrics -n kube-system
```

Di seguito viene riportato un output di esempio.

```
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` sono riconducibili a uno spazio dei nomi Kubernetes. Tutti i `RoleBindings` creati da Amazon EKS rientrano nell'ambito dello spazio nomi `kube-system`.

Il comando seguente restituisce tutti i `RoleBindings` Kubernetes creati da Amazon EKS nel cluster.

```
kubectl get rolebindings -n kube-system | grep eks
```

Per visualizzare le specifiche di a`RoleBinding`, sostituiscilo *`eks:k8s-metrics`* nel comando seguente con un valore `RoleBinding` restituito nell'output del comando precedente. L'esempio seguente restituisce la specifica per *`eks:k8s-metrics`*`RoleBinding`.

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

Di seguito viene riportato un output di esempio.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Crittografia dei segreti Kubernetes con KMS su cluster esistenti

> **Important**
>
> Questa procedura si applica solo a cluster EKS che eseguono la versione 1.27 o precedente di Kubernetes. Se si esegue la versione 1.28 o successiva di Kubernetes, i segreti Kubernetes sono protetti con crittografia a busta per impostazione predefinita. Per ulteriori

informazioni, consulta the section called "Crittografia a busta predefinita per tutti i dati dell'API Kubernetes".

Se abiliti la crittografia dei segreti, i segreti Kubernetes vengono crittografati utilizzando la chiave KMS selezionata. AWS La chiave KMS deve soddisfare le condizioni seguenti:

- Simmetria
- Possibilità di crittografia e decrittografia dei dati
- Creato nella stessa regione del cluster AWS
- Se la chiave KMS è stata creata in un account diverso, il principale IAM dovrà avere accesso alla stessa.

Per ulteriori informazioni, consulta Consentire ai responsabili IAM di altri account di utilizzare una chiave KMS nella Key Management Service Developer Guide.AWS

Warning

Non è possibile disabilitare la crittografia dei segreti dopo averla abilitata. Questa operazione è irreversibile.

eksctl

Questa procedura si applica solo a cluster EKS che eseguono la versione 1.27 o precedente di Kubernetes. Per ulteriori informazioni, consulta the section called "Crittografia a busta predefinita per tutti i dati dell'API Kubernetes".

È possibile abilitare la crittografia in due modi:

- Aggiungere la crittografia al cluster con un singolo comando.

  Per ricrittografare automaticamente i segreti, esegui il comando seguente.

```
eksctl utils enable-secrets-encryption \
    --cluster my-cluster \
    --key-arn arn:aws: kms:region-code:account:key/key
```

Per disattivare la ricrittografia automatica dei segreti, esegui il comando seguente.

```
eksctl utils enable-secrets-encryption
    --cluster my-cluster \
    --key-arn arn:aws: kms:region-code:account:key/key \
    --encrypt-existing-secrets=false
```

- Aggiungi la crittografia al cluster con un file `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
```

Per ricrittografare automaticamente i segreti, esegui il comando seguente.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

Per disattivare la ricrittografia automatica dei segreti, esegui il comando seguente.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-
secrets=false
```

Console di gestione AWS

a. Questa procedura si applica solo a cluster EKS che eseguono la versione 1.27 o precedente di Kubernetes. Per ulteriori informazioni, consulta the section called "Crittografia a busta predefinita per tutti i dati dell'API Kubernetes".

b. Aprire la Console Amazon EKS.

c. Scegli il cluster a cui aggiungere la crittografia KMS.

d. Scegli la scheda Overview (Panoramica) (selezionata per impostazione predefinita).

e. Scorri verso il basso fino a Secrets encryption (Crittografia dei segreti) e scegli Enable (Abilita).

f. Seleziona una chiave nell'elenco a discesa e fare clic sul pulsante Enable (Abilita). Se non sono elencate chiavi, è necessario crearne una prima. Per ulteriori informazioni, consultare Creazione chiavi

g. Fai clic sul pulsante Confirm (Conferma) per utilizzare la chiave scelta.

AWS CLI

a. Questa procedura si applica solo a cluster EKS che eseguono la versione 1.27 o precedente di Kubernetes. Per ulteriori informazioni, consulta the section called "Crittografia a busta predefinita per tutti i dati dell'API Kubernetes".

b. Associa la configurazione di crittografia segreta al tuo cluster utilizzando il seguente AWS comando CLI. Sostituire i valori di esempio con i propri valori.

```
aws eks associate-encryption-config \
    --cluster-name my-cluster \
    --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws: kms:region-code:account:key/key"}}]'
```

Di seguito viene riportato un output di esempio.

```
{
  "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. È possibile monitorare lo stato dell'aggiornamento di crittografia con il comando seguente. Usa il `cluster name` specifico e il valore `update ID` restituito nell'output precedente. Quando viene visualizzato lo stato `Successful`, l'aggiornamento è completo.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id 3141b835-8103-423a-8e68-12c2521ffa4d
```

Di seguito viene riportato un output di esempio:

```
{
  "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. Per verificare che la crittografia sia attivata nel cluster, eseguire il comando `describe-cluster`. La risposta contiene una stringa `EncryptionConfig`.

```
aws eks describe-cluster --region region-code --name my-cluster
```

Dopo aver abilitato la crittografia nel cluster, sarà necessario crittografare tutti i segreti esistenti con la nuova chiave:

> **Note**
>
> È necessario eseguire il comando seguente soltanto se si utilizza `eksctl` e la crittografia automatica dei segreti è disattivata.

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-
encryption-timestamp="time value"
```

Warning

Se si abilita la crittografia dei segreti per un cluster esistente e la chiave KMS utilizzata non viene mai eliminata, non sarà possibile ripristinare il cluster. L'eliminazione della chiave KMS metterà definitivamente il cluster in uno stato degradato. Per ulteriori informazioni, consulta Eliminazione delle chiavi AWS KMS.

Note

Per impostazione predefinita, il `create-key` comando crea una chiave KMS di crittografia simmetrica con una politica chiave che fornisce all'amministratore principale dell'account l'accesso alle azioni e alle AWS risorse KMS. Se si desidera ridurre l'ambito delle autorizzazioni, assicurarsi che le azioni `kms:DescribeKey` e `kms:CreateGrant` siano consentite nella policy per il principale che effettua la chiamata all'API `create-cluster`. Per cluster che utilizzano la crittografia a busta KMS, sono necessarie le autorizzazioni `kms:CreateGrant`. La condizione non `kms:GrantIsForAWSResource` è supportata per l' CreateCluster azione e non deve essere utilizzata nelle politiche KMS per controllare `kms:CreateGrant` le autorizzazioni degli utenti che eseguono. CreateCluster

Utilizzare i segreti di AWS Secrets Manager con i pod di Amazon EKS

Per mostrare i segreti di Secrets Manager e i parametri dall'archivio parametri come file montati in pod di Amazon EKS, è possibile utilizzare AWS Secrets e Configuration Provider (ASCP) per il driver CSI Kubernetes Secrets Store.

Con ASCP, è possibile archiviare e gestire i segreti in Secrets Manager e recuperarli tramite i carichi di lavoro in esecuzione su Amazon EKS. È possibile utilizzare i ruoli e i criteri IAM per limitare l'accesso ai propri segreti a specifici pod Kubernetes in un cluster. ASCP recupera la Pod identity e scambia l'identità per un ruolo IAM. ASCP assume il ruolo IAM del pod e quindi può recuperare i segreti da Secrets Manager autorizzati per tale ruolo.

Se si utilizza la rotazione automatica di Secrets Manager per i propri segreti, è anche possibile utilizzare la funzione di rotation reconciler del Driver CSI di Secrets Store, per assicurarsi di recuperare il segreto più recente da Secrets Manager.

> **Note**
>
> I gruppi di nodi AWS Fargate (Fargate) non sono supportati.

Per ulteriori informazioni, consulta Rotazione dei segreti di Secrets Manager in Amazon EKS nella AWS Guida per l'utente di Secrets Manager.

Crittografia a busta predefinita per tutti i dati dell'API Kubernetes

Amazon Elastic Kubernetes Service (Amazon EKS) fornisce la crittografia a busta predefinita di tutti i dati dell'API Kubernetes per i cluster EKS che eseguono Kubernetes versione 1.28 o successiva.

La crittografia a busta protegge i dati archiviati con il server API Kubernetes. Ad esempio, la crittografia a busta si applica alla configurazione del cluster Kubernetes, ad esempio. `ConfigMaps` La crittografia a busta non si applica ai dati sui nodi o sui volumi EBS. EKS in precedenza supportava la crittografia dei segreti Kubernetes, mentre ora la crittografia a busta si estende a tutti i dati dell'API Kubernetes.

Ciò fornisce un'esperienza gestita e predefinita che implementa defense-in-depth le tue applicazioni Kubernetes e non richiede alcuna azione da parte tua.

Amazon EKS utilizza AWS Key Management Service (KMS) con il provider Kubernetes KMS v2 per questo ulteriore livello di sicurezza con una chiave di proprietà di Amazon Web Services e la possibilità di portare la propria chiave gestita dal cliente (CMK) da KMS. AWS

Comprensione della crittografia a busta

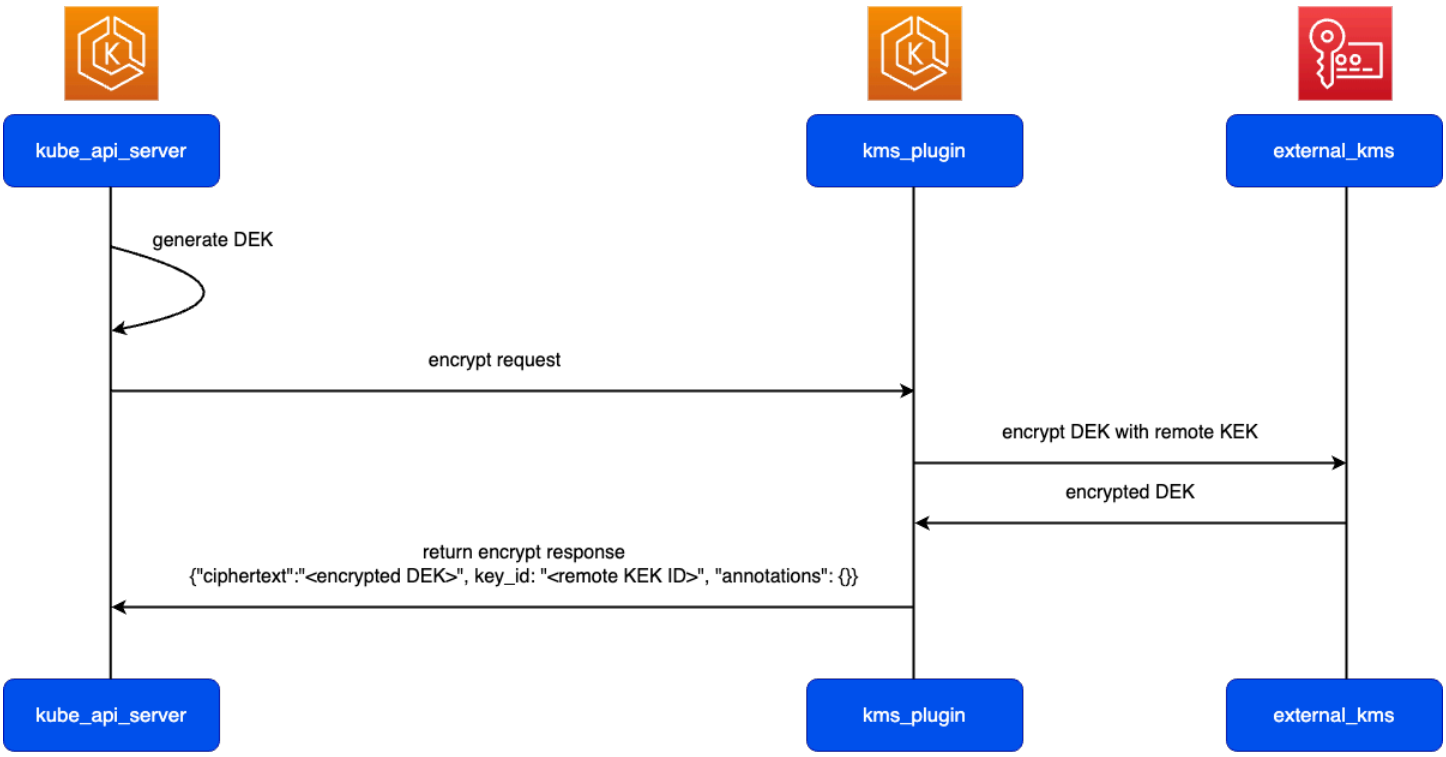
La crittografia delle buste è il processo di crittografia dei dati in testo semplice con una chiave di crittografia dei dati (DEK) prima che vengano inviati al datastore (etcd) e quindi di crittografia del DEK con una chiave KMS principale archiviata in un sistema KMS (AWS KMS) remoto e gestito centralmente. Questa è una defense-in-depth strategia perché protegge i dati con una chiave di crittografia (DEK) e quindi aggiunge un altro livello di sicurezza proteggendo tale DEK con una chiave di crittografia separata e archiviata in modo sicuro chiamata chiave di crittografia (KEK).

In che modo Amazon EKS abilita la crittografia predefinita delle buste con KMS v2 e KMS AWS

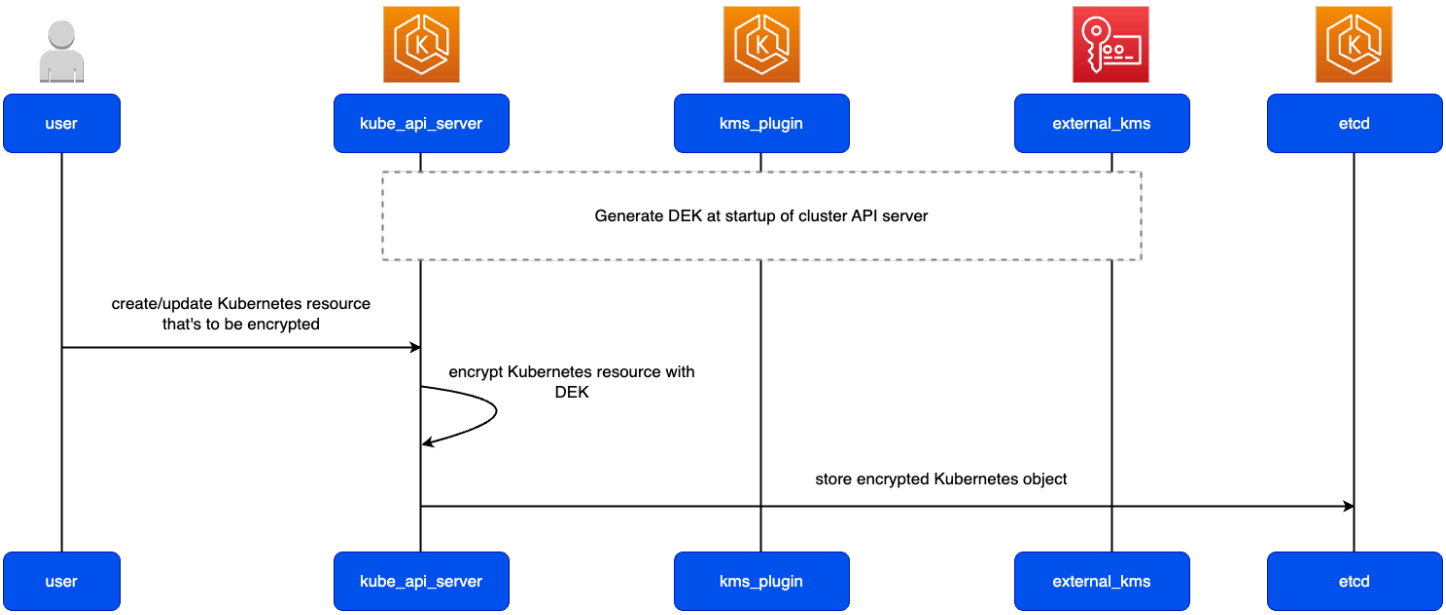
Amazon EKS utilizza KMS v2 per implementare la crittografia a busta predefinita per tutti i dati dell'API nel piano di controllo (control-plane) Kubernetes gestito prima che vengano conservati nel database etcd. All'avvio, il server API del cluster genera una chiave di crittografia dei dati (DEK) da un seed segreto combinato con dati generati casualmente. Inoltre, all'avvio, il server API effettua una chiamata al plug-in KMS per crittografare il seme DEK utilizzando una chiave di crittografia a chiave remota (KEK) di KMS. AWS Si tratta di una chiamata una tantum eseguita all'avvio del server API e durante la rotazione KEK. Il server API memorizza quindi nella cache il seed della DEK crittografato. Successivamente, il server API utilizza il seed DEK memorizzato nella cache per generarne un altro monouso DEKs basato su una Key Derivation Function (KDF). Ciascuno di questi dati generati DEKs viene quindi utilizzato una sola volta per crittografare una singola risorsa Kubernetes prima che venga archiviata in etcd. Con l'uso di un seed della DEK crittografato e memorizzato nella cache in KMS v2, il processo di crittografia delle risorse Kubernetes nel server API è sia più performante che conveniente.

Per impostazione predefinita, questa KEK è di proprietà di AWS, ma puoi opzionalmente importare la tua da KMS. AWS

Il diagramma riportato di seguito mostra la generazione e la crittografia di una DEK all'avvio del server API.



Il diagramma di alto livello riportato di seguito mostra la crittografia di una risorsa Kubernetes prima che venga archiviata in etcd.



Domande frequenti

In che modo la crittografia a busta predefinita migliora il livello di sicurezza del mio cluster EKS?

Questa funzionalità riduce la superficie e il periodo di tempo in cui i metadati e i contenuti dei clienti non vengono crittografati. Con la crittografia a busta predefinita, i metadati e i contenuti dei clienti sono solo temporaneamente non crittografati nella memoria di kube-apiserver prima di essere archiviati in etcd. La memoria del kube-apiserver è protetta tramite il sistema Nitro. Amazon EKS utilizza solo istanze EC2 basate su Nitro per il piano di controllo (control-plane) Kubernetes gestito. Queste istanze hanno un design di controllo di sicurezza che impediscono a qualsiasi sistema o persona di accedere alla memoria.

Quale versione di Kubernetes devo eseguire per avere questa funzionalità?

Per abilitare la crittografia a busta predefinita, il cluster Amazon EKS deve eseguire Kubernetes versione 1.28 o successiva.

I miei dati sono comunque sicuri se utilizzo una versione del cluster Kubernetes che non supporta questa funzionalità?

Sì. In, la sicurezza è la nostra massima priorità. AWS Basiamo tutta la nostra trasformazione e innovazione digitale sulle pratiche operative di sicurezza più elevate e restiamo impegnati a innalzare tale livello.

Tutti i dati archiviati in etcd sono crittografati a livello di disco per ogni cluster EKS, indipendentemente dalla versione di Kubernetes in esecuzione. EKS utilizza chiavi root che generano chiavi di crittografia del volume gestite dal servizio EKS. Inoltre, ogni cluster Amazon EKS viene eseguito in un VPC isolato utilizzando macchine virtuali specifiche del cluster. Grazie a questa architettura e alle nostre pratiche in materia di sicurezza operativa, Amazon EKS ha ottenuto diversi livelli e standard di conformità, tra cui l'idoneità SOC 1,2,3, PCI-DSS, ISO e HIPAA. Questi livelli e standard di conformità vengono mantenuti per tutti i cluster EKS con o senza crittografia a busta predefinita.

Come funziona la crittografia a busta in Amazon EKS?

All'avvio, il server API del cluster genera una chiave di crittografia dei dati (DEK) da un seed segreto combinato con dati generati casualmente. Inoltre, all'avvio, il server API effettua una chiamata al plug-in KMS per crittografare il DEK utilizzando una chiave di crittografia a chiave remota (KEK) di KMS. AWS Si tratta di una chiamata una tantum eseguita all'avvio del server API e durante la rotazione KEK. Il server API memorizza quindi nella cache il seed della DEK crittografato.

Successivamente, il server API utilizza il seed DEK memorizzato nella cache per generare un altro codice singolo DEKs basato su una Key Derivation Function (KDF). Ciascuno di questi dati generati DEKs viene quindi utilizzato una sola volta per crittografare una singola risorsa Kubernetes prima che venga archiviata in etcd.

È importante notare che sono state effettuate chiamate aggiuntive dal server API per verificare lo stato e la normale funzionalità dell'integrazione KMS. AWS Questi controlli sanitari aggiuntivi sono visibili nel tuo AWS CloudTrail.

Devo fare qualcosa o modificare le autorizzazioni affinché questa funzionalità sia abilitata nel mio cluster EKS?

Non è necessario eseguire alcuna operazione. La crittografia a busta in Amazon EKS è ora una configurazione predefinita abilitata in tutti i cluster che eseguono Kubernetes versione 1.28 o successiva. L'integrazione AWS KMS è stabilita dal server API Kubernetes gestito da. AWS Ciò significa che non è necessario configurare alcuna autorizzazione per iniziare a utilizzare la crittografia KMS per il cluster.

Come posso sapere se la crittografia a busta predefinita è abilitata sul mio cluster?

Se esegui la migrazione per utilizzare la tua CMK, puoi visualizzare l'ARN della chiave KMS associata al tuo cluster. Inoltre, puoi visualizzare i registri degli AWS CloudTrail eventi associati all'uso della CMK del tuo cluster.

Se il cluster utilizza una chiave AWS di proprietà, questa verrà descritta in dettaglio nella console EKS (escluso l'ARN della chiave).

È possibile AWS accedere alla chiave AWS di proprietà utilizzata per la crittografia predefinita delle buste in Amazon EKS?

No. AWS dispone di rigorosi controlli di sicurezza in Amazon EKS che impediscono a chiunque di accedere a qualsiasi chiave di crittografia in testo semplice utilizzata per proteggere i dati nel database etcd. Queste misure di sicurezza vengono applicate anche alla chiave KMS proprietaria. AWS

La crittografia a busta predefinita è abilitata nel mio cluster EKS esistente?

Se esegui il cluster Amazon EKS con Kubernetes versione 1.28 o successiva, la crittografia a busta predefinita di tutti i dati dell'API Kubernetes è abilitata. Per i cluster esistenti, Amazon EKS utilizza l'`eks:kms-storage-migrator`RBAC ClusterRole per migrare dati che in precedenza non erano crittografati in busta in etcd verso questo nuovo stato di crittografia.

Cosa significa se ho già abilitato la crittografia a busta per i segreti nel mio cluster EKS?

Se disponi di una chiave gestita dal cliente (CMK) in KMS che è stata utilizzata per crittografare a busta i tuoi segreti Kubernetes, quella stessa chiave verrà utilizzata come KEK per la crittografia a busta di tutti i tipi di dati dell'API Kubernetes nel cluster.

La gestione di un cluster EKS con crittografia a busta predefinita comporta costi aggiuntivi?

Non ci sono costi aggiuntivi associati al piano di controllo (control-plane) Kubernetes gestito se utilizzi una chiave di proprietà di Amazon Web Services per la crittografia a busta predefinita. Per impostazione predefinita, ogni cluster EKS che esegue Kubernetes versione 1.28 o successiva utilizza una chiave di proprietà di Amazon Web Service. Tuttavia, se utilizzi la tua chiave KMS, verranno applicati i AWS normali prezzi KMS.

Quanto costa utilizzare la mia chiave AWS KMS per crittografare i dati dell'API Kubernetes nel mio cluster?

Paghi 1 USD al mese per archiviare qualsiasi chiave personalizzata creata o importata in KMS. Addebiti KMS per le richieste di crittografia e decrittografia. È disponibile un piano gratuito di 20.000 richieste al mese per account e paghi 0,03 USD per 10.000 richieste superiori al piano gratuito al mese. Questo vale per tutti gli utilizzi del KMS per un account, quindi il costo dell'utilizzo della tua chiave AWS KMS sul cluster sarà influenzato dall'utilizzo di questa chiave su altri cluster o risorse all'interno del tuo account. AWS

Gli addebiti di KMS saranno più alti ora che la mia chiave gestita dal cliente (CMK) viene utilizzata per crittografare a busta tutti i dati dell'API Kubernetes oltre ai segreti?

No. La nostra implementazione con KMS v2 riduce significativamente il numero di chiamate effettuate a AWS KMS. Ciò a sua volta ridurrà i costi associati alla CMK indipendentemente dal fatto che i dati Kubernetes aggiuntivi vengano crittografati o decrittografati nel cluster EKS.

Come spiegato in precedenza, il seed della DEK generato utilizzato per la crittografia delle risorse Kubernetes viene archiviato localmente nella cache del server API Kubernetes dopo essere stato crittografato con la KEK remota. Se il seed DEK crittografato non si trova nella cache del server API, il server API chiamerà AWS KMS per crittografare il seme DEK. Il server API memorizza quindi nella cache il seed della DEK crittografato per utilizzi futuri nel cluster senza chiamare KMS. Allo stesso modo, per le richieste di decrittografia, il server API chiamerà AWS KMS per la prima richiesta di decrittografia, dopodiché il seme DEK decrittografato verrà memorizzato nella cache e utilizzato per future operazioni di decrittografia.

Per ulteriori informazioni, consulta KEP-3299: KMS v2 Improvements in the Kubernetes Enhancements on. GitHub

Posso utilizzare la stessa chiave CMK per più cluster Amazon EKS?

Sì. Per utilizzare nuovamente una chiave, puoi collegarla a un cluster nella stessa regione associando l'ARN al cluster durante la creazione. Tuttavia, se utilizzi la stessa CMK per più cluster EKS, devi adottare le misure necessarie per impedire la disabilitazione arbitraria della CMK. In caso contrario, una CMK disabilitata associata a più cluster EKS avrà un ambito di impatto più ampio sui cluster a seconda di tale chiave.

Cosa succede al mio cluster EKS se la mia CMK non è più disponibile dopo aver abilitato la crittografia a busta predefinita?

Se disabiliti una chiave KMS, non può essere utilizzata in nessuna operazione di crittografiariabiliti. Senza l'accesso a una CMK esistente, il server API non sarà in grado di crittografare e rendere persistenti gli oggetti Kubernetes appena creati, nonché di decrittografare qualsiasi oggetto Kubernetes precedentemente crittografato archiviato in etcd. Se la CMK è disabilitata, il cluster verrà immediatamente messo in unhealthy/degraded uno stato, a quel punto non saremo in grado di adempiere al nostro impegno di servizio finché non riabiliterai la CMK associata.

Quando una CMK è disabilitata, riceverai notifiche sullo stato di degrado del tuo cluster EKS e sulla necessità di abilitare nuovamente la CMK entro 30 giorni dalla sua disabilitazione per garantire il corretto ripristino delle risorse del piano di controllo (control-plane) Kubernetes.

Come posso proteggere il mio cluster EKS dall'impatto di una CMK? disabled/deleted

Per proteggere i cluster EKS da tali eventi, gli amministratori della chiave devono gestire l'accesso alle operazioni della chiave KMS utilizzando policy IAM con un principio di privilegio minimo per ridurre il rischio di qualsiasi disabilitazione o eliminazione arbitraria delle chiavi associate ai cluster EKS. Inoltre, puoi impostare un CloudWatch allarme per ricevere una notifica sullo stato della tua CMK.

Il mio cluster EKS verrà ripristinato se abilito nuovamente la CMK?

Per garantire il corretto ripristino del cluster EKS, consigliamo vivamente di abilitare di nuovo la CMK entro i primi 30 giorni dalla sua disabilitazione. Tuttavia, il corretto ripristino del cluster EKS dipenderà anche dal fatto che subisca o meno modifiche sostanziali all'API a causa di un aggiornamento automatico di Kubernetes che può avvenire mentre il cluster è in uno stato. unhealthy/degraded

Perché il mio cluster EKS è in unhealthy/degraded uno stato dopo aver disabilitato la CMK?

Il server API del piano di controllo EKS utilizza una chiave DEK crittografata e memorizzata nella memoria del server API per crittografare tutti gli oggetti durante le create/update operazioni prima che vengano archiviati in etcd. Quando un oggetto esistente viene recuperato da etcd, il server API utilizza la stessa chiave DEK memorizzata nella cache e decrittografa l'oggetto della risorsa Kubernetes. Se disabiliti la CMK, il server API non avrà alcun impatto immediato a causa della chiave DEK archiviata nella cache nella memoria del server API. Tuttavia, quando l'istanza del server API viene riavviata, non avrà un DEK memorizzato nella cache e dovrà chiamare KMS per le operazioni di crittografia e decrittografia. AWS Senza una CMK, questo processo fallirà con un codice di errore KMS_KEY_DISABLED, che impedisce il corretto avvio del server API.

Cosa succede al cluster EKS se elimino la CMK?

L'eliminazione della chiave CMK associata al cluster EKS ne comprometterà lo stato di integrità in modo non ripristinabile. Senza la CMK del cluster, il server API non sarà più in grado di crittografare e rendere persistenti i nuovi oggetti Kubernetes, nonché di decrittografare gli oggetti precedentemente crittografati archiviati nel database etcd. Dovresti procedere con l'eliminazione di una chiave CMK per il cluster EKS solo quando hai la certezza di non dover più utilizzare il cluster EKS.

Tieni presente che se la CMK non viene trovata (KMS_KEY_NOT_FOUND) o le concessioni per la CMK associata al cluster vengono revocate (KMS_GRANT_REVOKED), il cluster non sarà ripristinabile. Per ulteriori informazioni sullo stato del cluster e sui codici di errore, consulta Integrità del cluster e codici di errore con percorsi di risoluzione. FAQs

Mi verrà comunque addebitato il costo di un cluster degraded/unhealthy EKS perché ho disabilitato o eliminato la mia CMK?

Sì. Sebbene il piano di controllo EKS non sia utilizzabile in caso di CMK disabilitato, AWS continuerà a utilizzare risorse infrastrutturali dedicate allocate al cluster EKS fino a quando non verrà eliminato dal cliente. Inoltre, il nostro impegno di servizio non si applicherà in tali circostanze perché si tratterà di un'operazione volontaria o passiva da parte del cliente che impedisce il normale stato di integrità e il funzionamento del cluster EKS.

Il mio cluster EKS può essere aggiornato automaticamente quando si trova in unhealthy/degraded uno stato a causa di una CMK disabilitata?

Sì. Tuttavia, se il tuo cluster ha una CMK disabilitata, avrai a disposizione un periodo di 30 giorni per abilitarla nuovamente. In questo periodo di 30 giorni, il cluster Kubernetes non verrà aggiornato

automaticamente. Tuttavia, se questo periodo scade e non hai abilitato nuovamente la CMK, il cluster verrà aggiornato automaticamente alla versione successiva (n+1) con supporto standard, seguendo il ciclo di vita della versione Kubernetes in EKS.

Ti consigliamo vivamente di riattivare rapidamente una CMK disabilitata quando vieni a conoscenza di un cluster interessato. Tieni presente che, sebbene EKS aggiorni automaticamente questi cluster interessati, non è garantito che vengano ripristinati correttamente, soprattutto se il cluster viene sottoposto a più aggiornamenti automatici, poiché ciò potrebbe includere modifiche all'API Kubernetes e comportamenti imprevisti nel processo di bootstrap del server API.

Posso utilizzare un alias della chiave KMS?

Sì. Amazon EKS supporta l'utilizzo di alias delle chiavi KMS. Un alias è un nome descrittivo per una chiave Amazon Web Service KMS. Ad esempio, un alias ti consente di fare riferimento a una chiave KMS come my-key invece di **`1234abcd-12ab-34cd-56ef-1234567890ab`**.

Posso comunque eseguire il backup e il ripristino delle risorse del cluster utilizzando la mia soluzione di backup Kubernetes?

Sì. Puoi utilizzare una soluzione di backup Kubernetes (come Velero) per il disaster recovery, la migrazione e la protezione dei dati del cluster Kubernetes. Se esegui una soluzione di backup Kubernetes che accede alle risorse del cluster tramite il server API, tutti i dati recuperati dall'applicazione verranno decrittografati prima di raggiungere il client. Ciò consentirà di ripristinare le risorse del cluster in un altro cluster Kubernetes.

Considerazioni sulla sicurezza per Amazon EKS Auto Mode

Questo argomento descrive l'architettura di sicurezza, i controlli e le best practice per Amazon EKS Auto Mode. Man mano che le organizzazioni implementano applicazioni containerizzate su scala, mantenere una solida posizione di sicurezza diventa sempre più complesso. EKS Auto Mode implementa controlli di sicurezza automatizzati e si integra con i servizi di sicurezza AWS per aiutarti a proteggere l'infrastruttura, i carichi di lavoro e i dati del cluster. Attraverso funzionalità di sicurezza integrate come la gestione forzata del ciclo di vita dei nodi e l'implementazione automatica delle patch, EKS Auto Mode ti aiuta a rispettare le best practice di sicurezza riducendo al contempo il sovraccarico operativo.

Prima di procedere con questo argomento, assicurati di conoscere i concetti base di EKS Auto Mode e di aver esaminato i prerequisiti per abilitare EKS Auto Mode sui cluster. Per informazioni generali sulla sicurezza di Amazon EKS, consulta Sicurezza.

Amazon EKS Auto Mode si fonda sulle basi di sicurezza esistenti di Amazon EKS, introducendo al contempo controlli di sicurezza automatizzati aggiuntivi per le istanze gestite EC2.

Sicurezza e autenticazione delle API

Amazon EKS Auto Mode utilizza meccanismi di sicurezza della piattaforma AWS per proteggere e autenticare le chiamate all'API Amazon EKS.

- L'accesso all'API Kubernetes è protetto tramite voci di accesso EKS, che si integrano con le identità AWS IAM.
  - Per ulteriori informazioni, consulta the section called "Voci di accesso".
- I clienti possono implementare un controllo granulare degli accessi all'endpoint dell'API Kubernetes tramite la configurazione delle voci di accesso EKS.

Sicurezza di rete

Amazon EKS Auto Mode supporta più livelli di sicurezza di rete:

- Integrazione con VPC
  - Funziona all'interno del cloud privato virtuale (VPC) di Amazon
  - Supporta configurazioni VPC personalizzate e layout di sottorete
  - Abilita la rete privata tra i componenti del cluster
  - Per ulteriori informazioni, consulta Managing security responsibilities for Amazon Virtual Private Cloud
- Policy di rete
  - Supporto nativo per le policy di rete di Kubernetes
  - Capacità di definire regole granulari del traffico di rete
  - Per ulteriori informazioni, consulta the section called "Policy di Kubernetes"

Sicurezza delle istanze gestite di EC2

Amazon EKS Auto Mode opera su istanze gestite EC2 con i seguenti controlli di sicurezza:

Sicurezza EC2

- Le istanze gestite di EC2 mantengono le funzionalità di sicurezza di Amazon EC2.

- Per ulteriori informazioni sulle istanze gestite da EC2, consulta Security in Amazon EC2.

Gestione del ciclo di vita delle istanze

Le istanze gestite EC2 operate da EKS Auto Mode hanno una durata massima di 21 giorni. Amazon EKS Auto Mode termina automaticamente le istanze che superano questa durata. Questo limite del ciclo di vita aiuta a prevenire variazioni di configurazione e mantiene il livello di sicurezza.

Protezione dei dati

- Amazon EC2 Instance Storage è crittografata; questa archiviazione è allegata direttamente all'istanza. Per ulteriori informazioni, consulta Data protection in Amazon EC2.
- Amazon EKS Auto Mode gestisce i volumi collegati alle istanze EC2 al momento della creazione, compresi i volumi root e di dati. EKS Auto Mode non gestisce completamente i volumi EBS creati utilizzando le funzionalità di storage persistente di Kubernetes.

Gestione delle patch

- Amazon EKS Auto Mode applica automaticamente le patch alle istanze gestite.
- Le patch includono:
  - Aggiornamenti del sistema operativo
  - Patch di sicurezza
  - Componenti di Amazon EKS Auto Mode

> **Note**
>
> I clienti hanno la responsabilità di proteggere e aggiornare i carichi di lavoro in esecuzione su queste istanze.

Controlli sugli accessi

- L'accesso diretto alle istanze è limitato:
  - L'accesso SSH non è disponibile.
  - L'accesso ad AWS Systems Manager Session Manager (SSM) non è disponibile.

- Le operazioni di gestione vengono eseguite tramite l'API Amazon EKS e l'API Kubernetes.

Gestione automatizzata delle risorse

Amazon EKS Auto Mode non gestisce completamente i volumi di Amazon Elastic Block Store (Amazon EBS) creati utilizzando le funzionalità di storage persistente di Kubernetes. Inoltre, EKS Auto Mode non gestisce il bilanciatore del carico elastico (ELB). Amazon EKS Auto Mode automatizza le attività di routine per queste risorse.

Sicurezza dell'archiviazione

- AWS consiglia di abilitare la crittografia per i volumi EBS forniti dalle funzionalità di archiviazione persistente di Kubernetes. Per ulteriori informazioni, consulta the section called "Crea StorageClass".
- Crittografia inattiva tramite AWS KMS
- È possibile configurare il proprio account AWS affinché applichi la crittografia alle nuove copie di volumi e snapshot EBS create. Per ulteriori informazioni, consulta Enable Amazon EBS encryption by default nella Guida per l'utente di Amazon EBS.
- Per ulteriori informazioni, consulta Security in Amazon EBS.

Sicurezza del bilanciatore del carico

- Configurazione automatizzata dei bilanciatori del carico elastico
- Gestione dei certificati SSL/TLS tramite l'integrazione di Gestione certificati AWS
- Automazione dei gruppi di sicurezza per il controllo degli accessi al bilanciatore del carico
- Per ulteriori informazioni, consulta Security in Elastic Load Balancing.

Best practice di sicurezza

La sezione seguente descrive le best practice di sicurezza per Amazon EKS Auto Mode.

- Esamina regolarmente le policy IAM AWS e le voci di accesso EKS.
- Implementa schemi di accesso con privilegi minimi per i carichi di lavoro.

- Monitora l'attività del cluster tramite AWS CloudTrail e Amazon CloudWatch. Per ulteriori informazioni, consulta the section called " AWS CloudTrail" e the section called "Amazon CloudWatch".
- Usa AWS Security Hub per la valutazione del livello di sicurezza.
- Implementa gli standard di sicurezza dei pod appropriati per i carichi di lavoro.

Considerazioni sulla sicurezza per EKS Capabilities

Questo argomento tratta importanti considerazioni sulla sicurezza per EKS Capabilities, tra cui la configurazione dei ruoli IAM, le autorizzazioni Kubernetes e i modelli architettonici per le implementazioni multi-cluster e la gestione delle risorse tra account. AWS

EKS Capabilities utilizza una combinazione di ruoli IAM, voci di accesso EKS e Kubernetes RBAC per fornire un accesso sicuro ai AWS servizi, alle risorse Kubernetes in cluster e alle integrazioni con Secrets Manager e altri servizi. AWS CodeConnections AWS AWS

Capacità, ruolo IAM

Quando crei una capacità, fornisci un ruolo di funzionalità IAM che EKS utilizza per eseguire azioni per tuo conto. Questo ruolo deve:

- Avere lo stesso AWS account del cluster e della risorsa di capacità
- Adottate una politica di fiducia che consenta al responsabile del `capabilities.eks.amazonaws.com` servizio di assumere il ruolo
- Disponi delle autorizzazioni IAM appropriate al tipo di funzionalità e al caso d'uso, a seconda delle tue esigenze. Per informazioni dettagliate sulle autorizzazioni IAM richieste, consultathe section called " AWS CodeConnections", e the section called " AWS Secrets Manager" the section called "Configurazione delle autorizzazioni "

È consigliabile considerare l'ambito dei privilegi richiesti per il caso d'uso specifico e concedere solo le autorizzazioni necessarie per soddisfare i requisiti. Ad esempio, quando si utilizza EKS Capability for Kube Resource Orchestrator, potrebbe non essere richiesta alcuna autorizzazione IAM, mentre quando si utilizza EKS Capability for AWS Controllers for Kubernetes è possibile concedere l'accesso completo a uno o più servizi. AWS

> **Important**
>
> Sebbene alcuni casi d'uso possano giustificare l'uso di ampi privilegi amministrativi, segui il principio del privilegio minimo concedendo solo le autorizzazioni IAM minime richieste per il tuo caso d'uso specifico, limitando l'accesso a risorse specifiche utilizzando chiavi ARNs e condizioni anziché utilizzare autorizzazioni wildcard.

Per informazioni dettagliate sulla creazione e configurazione dei ruoli IAM con funzionalità, consulta. the section called "Capacità, ruolo IAM"

Voci di accesso EKS

Quando crei una funzionalità con un ruolo IAM, Amazon EKS crea automaticamente una voce di accesso per quel ruolo nel tuo cluster. Questa voce di accesso concede alla funzionalità di base le autorizzazioni Kubernetes per funzionare.

> **Note**
>
> Le voci di accesso vengono create per il cluster in cui viene creata la funzionalità. Per le distribuzioni di Argo CD su cluster remoti, è necessario creare voci di accesso su tali cluster con le autorizzazioni appropriate per consentire alla funzionalità di Argo CD di distribuire e gestire le applicazioni.

La voce di accesso include:

- Il ruolo di IAM (ARN) come principale
- Politiche di accesso specifiche per funzionalità che concedono le autorizzazioni Kubernetes di base
- Ambito appropriato (a livello di cluster o ambito dello spazio dei nomi) in base al tipo di funzionalità

> **Note**
>
> Per Argo CD, le autorizzazioni relative allo spazio dei nomi vengono concesse allo spazio dei nomi specificato nella configurazione delle funzionalità (il valore predefinito è). `argocd`

Politiche di accesso predefinite per funzionalità

Ogni tipo di funzionalità concede al ruolo di capacità le autorizzazioni richieste, impostando diverse politiche di accesso predefinite come segue:

kro

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSKROPolicy`(con ambito cluster)

  Concede le autorizzazioni per guardare, gestire ResourceGraphDefinitions e creare istanze di risorse personalizzate definite da. RGDs

ACK

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSACKPolicy`(con ambito cluster)

  Concede le autorizzazioni per creare, leggere, aggiornare ed eliminare risorse personalizzate ACK in tutti i namespace.

CD Argo

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSArgoCDClusterPolicy`(con ambito cluster)

  Concede autorizzazioni a livello di cluster per Argo CD per individuare risorse e gestire oggetti con ambito cluster.

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSArgoCDPolicy`(con ambito namespace)

  Concede autorizzazioni a livello di namespace per Argo CD per distribuire e gestire applicazioni. Ha come ambito lo spazio dei nomi specificato nella configurazione delle funzionalità (il valore predefinito è). `argocd`

Per informazioni più dettagliate, vederethe section called "Rivedere le policy di accesso".

Autorizzazioni Kubernetes aggiuntive

Alcune funzionalità potrebbero richiedere autorizzazioni Kubernetes aggiuntive oltre alle politiche di accesso predefinite. Puoi concedere queste autorizzazioni utilizzando uno dei seguenti metodi:

- Criteri di accesso: Associa criteri gestiti aggiuntivi alla voce di accesso

- Kubernetes RBAC: creazione `Role` di `ClusterRole` associazioni per l'utente Kubernetes della funzionalità

Autorizzazioni di lettura segrete ACK

Alcuni controller ACK devono leggere i segreti di Kubernetes per recuperare dati sensibili come le password dei database. I seguenti controller ACK richiedono un accesso segreto in lettura:

- `acm`, `acmpca`, `documentdb`, `memorydb`, `mq`, `rds`, `secretsmanager`

Per concedere autorizzazioni di lettura segrete:

1. Associa la politica di `arn:aws:eks::aws:cluster-access-policy/AmazonEKSSecretReaderPolicy` accesso alla voce di accesso della funzionalità
2. Estendi la policy a namespace specifici in cui le risorse ACK faranno riferimento ai segreti o concederanno l'accesso a livello di cluster

> **Important**
>
> Le autorizzazioni di lettura segrete sono limitate ai namespace specificati durante l'associazione della politica di immissione degli accessi. Ciò consente di limitare i segreti a cui la funzionalità può accedere.

o autorizzazioni arbitrarie per le risorse

kro richiede le autorizzazioni per creare e gestire le risorse definite nel tuo. ResourceGraphDefinitions Per impostazione predefinita, kro può solo guardare e gestire se stesso. RGDs

Per concedere a kro i permessi per creare risorse:

Opzione 1: Accedere alle politiche di accesso

Associa politiche di accesso predefinite come `AmazonEKSAdminPolicy` o `AmazonEKSEditPolicy` all'immissione di accesso della funzionalità.

Opzione 2: Kubernetes RBAC

Crea un account `ClusterRoleBinding` che conceda all'utente Kubernetes della funzionalità le autorizzazioni necessarie:

```
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**
>
> Il nome utente Kubernetes per kro segue lo schema: `arn:aws:sts::ACCOUNT_ID:assumed-role/ROLE_NAME/KRO` Il nome della sessione `/KRO` (in maiuscolo) viene impostato automaticamente dalla funzionalità EKS kro.

Autorizzazioni IAM richieste per funzionalità

kro (Kube Resource Orchestrator)

Non sono richieste autorizzazioni IAM. Puoi creare un ruolo di capacità senza policy allegate. kro richiede solo le autorizzazioni RBAC di Kubernetes.

ACK AWS (Controller per Kubernetes)

Richiede le autorizzazioni per gestire le AWS risorse che ACK creerà e gestirà. È necessario definire l'ambito delle autorizzazioni per servizi, azioni e risorse specifici in base alle proprie esigenze. Per informazioni dettagliate sulla configurazione delle autorizzazioni ACK, incluse le migliori pratiche di produzione con IAM Role Selectors, consulta. the section called "Configurazione delle autorizzazioni "

CD Argo

Per impostazione predefinita, non sono richieste autorizzazioni IAM. Potrebbero essere necessarie autorizzazioni opzionali per:

- AWS Secrets Manager: se si archiviano le credenziali del repository Git in Secrets Manager
- AWS CodeConnections: Se si utilizza CodeConnections per l'autenticazione del repository Git
- Amazon ECR: se si utilizzano grafici Helm archiviati in formato OCI in Amazon ECR

Best practice di sicurezza

Privilegio minimo IAM

Concedi alle tue risorse funzionali solo le autorizzazioni necessarie per il tuo caso d'uso. Ciò non significa che non sia possibile concedere ampie autorizzazioni amministrative alle proprie funzionalità, se necessario. In questi casi, è necessario gestire l'accesso a tali risorse in modo appropriato.

Ruoli relativi alle capacità:

- ACK: Quando possibile, limita le autorizzazioni IAM a AWS servizi e risorse specifici di cui i tuoi team hanno bisogno, in base ai casi d'uso e ai requisiti
- Argo CD: limita l'accesso a specifici repository Git e namespace Kubernetes
- kro: Richiede un ruolo di capacità per la policy di fiducia, ma non sono necessarie autorizzazioni IAM (utilizza solo il cluster RBAC)

Esempio: invece`"Resource": "*"`, specifica modelli per risorse o gruppi di risorse specifici.

```
"Resource": [
  "arn:aws:s3:::my-app-*",
  "arn:aws:rds:us-west-2:111122223333:db:prod-*"
]
```

Utilizza le chiavi delle condizioni IAM per limitare ulteriormente l'accesso:

```
"Condition": {
  "StringEquals": {
```

```
    "aws:ResourceTag/Environment": "production"
  }
}
```

Per ulteriori informazioni sulla configurazione IAM, consulta la sezione Considerazioni per ciascuna funzionalità.

Isolamento dello spazio dei nomi per i segreti di Argo CD

La funzionalità gestita di Argo CD ha accesso a tutti i segreti di Kubernetes all'interno dello spazio dei nomi configurato (impostazione predefinita:). `argocd` Per mantenere una posizione di sicurezza ottimale, segui queste pratiche di isolamento dello spazio dei nomi:

- Conserva solo i segreti relativi ad Argo CD all'interno dello spazio dei nomi Argo CD
- Evita di archiviare segreti applicativi non correlati nello stesso spazio dei nomi di Argo CD
- Utilizzate namespace separati per i segreti delle applicazioni che non sono necessari per le operazioni di Argo CD

Questo isolamento garantisce che l'accesso segreto di Argo CD sia limitato alle sole credenziali necessarie per l'autenticazione del repository Git e altre operazioni specifiche di Argo CD.

RBAC Kubernetes

Controlla quali utenti e account di servizio possono creare e gestire le risorse funzionali. È consigliabile distribuire le risorse di funzionalità in namespace dedicati con politiche RBAC appropriate.

Esempio: ruolo RBAC per l'utilizzo con ACK, che consente la gestione delle risorse S3 Bucket nel 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"]
```

Registrazione di controllo

CloudTrail: Tutte le operazioni dell'API EKS Capability (creazione, aggiornamento, eliminazione) vengono registrate. AWS CloudTrail

Abilita la CloudTrail registrazione per tenere traccia di:

- Chi ha creato o modificato le funzionalità
- Quando sono cambiate le configurazioni delle funzionalità
- Quali ruoli di capacità sono in uso

Accesso alla rete ed endpoint VPC

Accesso privato all'API Argo CD

È possibile limitare l'accesso al server API Argo CD associando uno o più endpoint VPC all'endpoint Argo CD ospitato. Ciò consente la connettività privata dall'interno del tuo VPC senza attraversare la rete Internet pubblica. L'endpoint VPC fornisce l'accesso sia all'interfaccia utente web di Argo CD che all'API Argo CD (incluso l'accesso CLI).

> **Note**
>
> Endpoint VPC connessi agli endpoint API Argo CD ospitati (utilizzando le funzionalità eks). *region*.amazonaws.com) non supportano le politiche degli endpoint VPC.

Distribuzione su cluster privati

La funzionalità Argo CD può implementare applicazioni su cluster EKS completamente privati, offrendo un vantaggio operativo significativo eliminando la necessità di peering VPC o configurazioni di rete complesse. Tuttavia, nel progettare questa architettura, considerate che Argo CD estrae la configurazione dai repository Git (che possono essere pubblici) e la applica ai vostri cluster privati.

Assicuratevi di:

- Usa repository Git privati per carichi di lavoro sensibili
- Implementa controlli di accesso e autenticazione adeguati al repository Git
- Rivedi e approva le modifiche tramite richieste pull prima dell'unione

- Prendi in considerazione l'utilizzo delle finestre di sincronizzazione di Argo CD per controllare quando possono avvenire le distribuzioni
- Monitora i registri di controllo di Argo CD per rilevare eventuali modifiche non autorizzate alla configurazione

Conformità

Le funzionalità EKS sono completamente gestite e dispongono delle certificazioni di conformità di Amazon EKS.

Per informazioni aggiornate sulla conformità, consulta AWS Services in Scope by Compliance Program.

Fasi successive

- the section called "Configurazione delle autorizzazioni "- Configura le autorizzazioni IAM per ACK
- the section called "Configurazione delle autorizzazioni "- Configura Kubernetes RBAC per kro
- the section called "Configurazione delle autorizzazioni "- Configura l'integrazione di Identity Center per Argo CD
- the section called "Risoluzione dei problemi relativi alle funzionalità"- Risolvi i problemi di sicurezza e autorizzazione

Identity and Access Management per Amazon EKS

AWS Identity and Access Management (IAM) è AWS un servizio che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi è autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) a utilizzare risorse Amazon EKS. IAM è un AWS servizio che puoi utilizzare senza costi aggiuntivi.

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in Amazon EKS.

Utente del servizio:: se si utilizza il servizio Amazon EKS per svolgere il proprio lavoro, l'amministratore fornisce le credenziali e le autorizzazioni necessarie. All'aumentare del numero di caratteristiche Amazon EKS utilizzate per il lavoro, potrebbero essere necessarie ulteriori

autorizzazioni. La comprensione della gestione dell'accesso consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una caratteristica in Amazon EKS, consultare the section called "Risoluzione dei problemi".

Amministratore del servizio: se si è responsabile delle risorse Amazon EKS presso la propria azienda, probabilmente si dispone dell'accesso completo a Amazon EKS. Il compito dell'utente è determinare le caratteristiche e le risorse di Amazon EKS a cui gli utenti del servizio devono accedere. Devi quindi inviare le richieste all'amministratore IAM per modificare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la propria azienda può utilizzare IAM con Amazon EKS, consulta the section called "Amazon EKS e IAM".

Amministratore IAM: se si è amministratore IAM, potrebbe essere interessante ottenere informazioni su come scrivere policy per gestire l'accesso ad Amazon EKS. Per visualizzare policy basate su identità Amazon EKS di esempio che possono essere utilizzate in IAM, consulta the section called "Policy basate sull'identità".

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. È necessario autenticarsi (effettuare l'accesso AWS) come utente root dell' AWS account, come utente IAM o assumendo un ruolo IAM.

Puoi accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS Gli utenti di IAM Identity Center (IAM Identity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Quando accedi AWS utilizzando la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al Console di gestione AWS o al portale di AWS accesso. Per ulteriori informazioni sull'accesso AWS, vedi Come accedere al tuo AWS account nella Guida per l'utente di AWS accesso.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le tue richieste utilizzando le tue credenziali. Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta Signing AWS API request nella IAM User Guide.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) AWS nella Guida per l'utente IAM.

AWS account (utente root)

Quando si crea un AWS account, si inizia con un'identità di accesso che ha accesso completo a tutti i AWS servizi e le risorse dell'account. Questa identità è denominata utente root dell' AWS account ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzale per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente IAM.

Utenti e gruppi IAM

Un utente IAM è un'identità all'interno del tuo AWS account che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, se si hanno casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni di set di utenti di grandi dimensioni. Ad esempio, potresti avere un gruppo denominato IAMAdminse concedere a quel gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente IAM.

Ruoli IAM

Un ruolo IAM è un'identità all'interno del tuo AWS account che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Puoi assumere temporaneamente un ruolo IAM in Console di gestione AWS cambiando ruolo. Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API o utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato - Per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center.
- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni AWS servizi, puoi allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.
- Accesso tra servizi: alcuni AWS servizi utilizzano le funzionalità di altri AWS servizi. Ad esempio, quando effettui una chiamata in un servizio, è normale che quel servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, un ruolo di servizio oppure un ruolo collegato al servizio.
  - Sessioni di accesso inoltrato (FAS): quando utilizzi un utente o un ruolo IAM per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che chiama un AWS servizio, in combinazione con il servizio richiedente per

effettuare richieste ai AWS servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri AWS servizi o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta Forward access sessions.

- Ruolo di servizio - Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta Creazione di un ruolo per delegare le autorizzazioni a un AWS servizio nella Guida per l'utente IAM.
- Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un servizio. AWS Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati al servizio vengono visualizzati nell' AWS account e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati al servizio, ma non modificarle.

- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un ruolo IAM per gestire le credenziali temporanee per le applicazioni in esecuzione su un' EC2 istanza e che effettuano richieste AWS CLI AWS o API. Questa soluzione è preferibile alla memorizzazione delle chiavi di accesso all'interno dell'istanza. EC2 Per assegnare un AWS ruolo a un' EC2 istanza e renderlo disponibile per tutte le sue applicazioni, create un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull' EC2 istanza di ottenere credenziali temporanee. Per ulteriori informazioni, consulta Usare un ruolo IAM per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella IAM User Guide.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente IAM.

Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e collegandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dalla Console di gestione AWS, dalla AWS CLI o dall' AWS API.

Policy basate sull'identità

Le policy basate sull'identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli nel tuo AWS account. Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente IAM.

Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy di bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio specificare un'entità principale. I principali possono includere account, utenti, ruoli, utenti federati o AWS servizi.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non puoi utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Elenchi di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3, AWS WAF e Amazon VPC sono esempi di servizi che supportano. ACLs Per ulteriori informazioni ACLs, consulta la panoramica della lista di controllo degli accessi (ACL) nella Amazon Simple Storage Service Developer Guide.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai più tipi di policy comuni.

- Limiti delle autorizzazioni - Un limite delle autorizzazioni è una funzionalità avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate sull'identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per maggiori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Policy di controllo del servizio (SCPs): SCPs sono politiche JSON che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in Organizations AWS . AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più AWS account di proprietà della tua azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. L'SCP limita le autorizzazioni per le entità negli account dei membri, incluso ogni AWS utente root dell'account. Per ulteriori informazioni su Organizations and SCPs, consulta le politiche di controllo dei servizi nella AWS Organizations User Guide.
- Policy di sessione - Le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate sull'identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Funzionamento di Amazon EKS con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon EKS, è necessario comprendere quali funzioni IAM sono disponibili per l'uso con Amazon EKS. Per avere una visione di alto livello di come Amazon EKS e altri AWS servizi funzionano con IAM, consulta AWS i servizi che funzionano con IAM nella IAM User Guide.

Argomenti

- Policy basate su identità Amazon EKS
- Policy basate sulle risorse Amazon EKS
- Autorizzazione basata su tag Amazon EKS
- Ruoli IAM di Amazon EKS

Policy basate su identità Amazon EKS

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Amazon EKS supporta specifiche operazioni, risorse e chiavi di condizione. Per informazioni su tutti gli elementi utilizzati in una policy JSON, consultare Documentazione di riferimento degli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

Azioni

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Le azioni politiche in genere hanno lo stesso nome dell'operazione AWS API associata. Ci sono alcune eccezioni, ad esempio le operazioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includere le operazioni in una policy per concedere le autorizzazioni per eseguire l'operazione associata.

Le operazioni delle policy in Amazon EKS utilizzano il seguente prefisso prima dell'operazione: `eks:`. Ad esempio, per concedere a qualcuno l'autorizzazione per ottenere le informazioni descrittive su un cluster Amazon EKS, includere l'operazione `DescribeCluster` nella policy. Le istruzioni della policy devono includere un elemento `Action` o `NotAction`.

Per specificare più operazioni in una sola istruzione, separa ciascuna di esse con una virgola come mostrato di seguito:

```
"Action": ["eks:action1", "eks:action2"]
```

È possibile specificare più azioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le azioni che iniziano con la parola `Describe`, includi la seguente azione:

```
"Action": "eks:Describe*"
```

Per visualizzare un elenco di operazioni di Amazon EKS, consulta Operazioni definite da Amazon Elastic Kubernetes Service in Service Authorization Reference.

Resources

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Le istruzioni devono includere un elemento `Resource` o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). È possibile eseguire questa operazione per operazioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

La risorsa del cluster Amazon EKS dispone del seguente ARN.

```
 arn:aws: eks:region-code:account-id:cluster/cluster-name
```

Per ulteriori informazioni sul formato di ARNs, consulta Amazon resource names (ARNs) e AWS service namespaces.

Ad esempio, per specificare il cluster con il nome *my-cluster* nell'istruzione, utilizzare il seguente ARN:

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/my-cluster"
```

Per specificare tutti i cluster che appartengono a un account e a una AWS regione specifici, usa il carattere jolly (*):

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/*"
```

Alcune operazioni Amazon EKS, ad esempio quelle utili per la creazione di risorse, non possono essere eseguite su una risorsa specifica. In questi casi, è necessario utilizzare il carattere jolly (*).

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di risorse Amazon EKS e relativi ARNs, consulta Resources defined by Amazon Elastic Kubernetes Service nel Service Authorization Reference. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consultare Operazioni definite da Amazon Elastic Kubernetes Service.

Chiavi di condizione

Amazon EKS definisce il proprio set di chiavi di condizione e supporta anche l'uso di alcune chiavi di condizione globali. Per visualizzare tutte le chiavi di condizione AWS globali, consulta AWS Global Condition Context Keys nella IAM User Guide.

È possibile impostare le chiavi di condizione quando si associa un provider OpenID Connect al cluster. Per ulteriori informazioni, consulta the section called “Policy IAM di esempio”.

Tutte le operazioni Amazon EC2 supportano le chiavi di condizione `aws:RequestedRegion` e `ec2:Region`. Per ulteriori informazioni, consulta Esempio: limitazione dell'accesso a una AWS regione specifica.

Per un elenco di chiavi di condizione di Amazon EKS, consultare Condizioni per Amazon Elastic Kubernetes Service in Service Authorization Reference. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consultare Operazioni definite da Amazon Elastic Kubernetes Service.

Esempi

Per visualizzare esempi di policy basate su identità Amazon EKS, consultare the section called "Policy basate sull'identità".

Quando si crea un cluster Amazon EKS, il principale IAM che crea il cluster riceve automaticamente le autorizzazioni `system:masters` nella configurazione del controllo degli accessi basato sul ruolo (RBAC) nel piano di controllo di Amazon EKS. Questo principale non è visualizzato in una configurazione visibile qualsiasi, quindi assicurati di tenere traccia di quale principale ha originariamente creato il cluster. Per concedere a ulteriori principali IAM la capacità di interagire con il cluster, modifica `aws-auth ConfigMap` all'interno di Kubernetes e crea `rolebinding` o `clusterrolebinding` Kubernetes con il nome di `group` specificato in `aws-auth ConfigMap`.

Per ulteriori informazioni sull'utilizzo di ConfigMap, vederethe section called "Accesso API Kubernetes".

Policy basate sulle risorse Amazon EKS

Amazon EKS non supporta policy basate su risorse.

Autorizzazione basata su tag Amazon EKS

È possibile allegare i tag alle risorse Amazon EKS o inoltrarli in una richiesta ad Amazon EKS. Per controllare l'accesso basato su tag, fornire informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/`*`key-name`* , `aws:RequestTag/`*`key-name`* o `aws:TagKeys`. Per ulteriori informazioni sull'assegnazione di tag delle risorse di Amazon EKS, consultare the section called "Assegnazione di tag alle risorse ". Per ulteriori informazioni sulle operazioni in cui è possibile utilizzare i tag nelle chiavi di condizione, consulta Operazioni definite da Amazon EKSnel manuale Service Authorization Reference.

Ruoli IAM di Amazon EKS

Un ruolo IAM è un'entità all'interno del tuo AWS account che dispone di autorizzazioni specifiche.

Utilizzo di credenziali temporanee con Amazon EKS

È possibile utilizzare credenziali temporanee per effettuare l'accesso con la federazione, assumere un ruolo IAM o un ruolo multi-account. È possibile ottenere credenziali di sicurezza temporanee chiamando operazioni API AWS STS come AssumeRoleo. GetFederationToken

Amazon EKS supporta l'uso di credenziali temporanee.

Ruoli collegati ai servizi

I ruoli collegati ai AWS servizi consentono ai servizi di accedere alle risorse di altri servizi per completare un'azione per conto dell'utente. I ruoli collegati ai servizi sono visualizzati nell'account IAM e sono di proprietà del servizio. Un amministratore può visualizzare, ma non può modificare le autorizzazioni dei ruoli collegati ai servizi.

Amazon EKS supporta i ruoli collegati ai servizi. Per maggiori dettagli su come creare e gestire i ruoli collegati ai servizi Amazon EKS, consultare the section called "Ruoli collegati ai servizi".

Ruoli dei servizi

Questa funzionalità consente a un servizio di assumere un ruolo di servizio per conto dell'utente. Questo ruolo consente al servizio di accedere alle risorse in altri servizi per completare un'azione per conto dell'utente. I ruoli dei servizi sono visualizzati nell'account IAM e sono di proprietà dell'account. Ciò significa che un amministratore IAM può modificare le autorizzazioni per questo ruolo. Tuttavia, questo potrebbe pregiudicare la funzionalità del servizio.

Amazon EKS supporta i ruoli del servizio. Per ulteriori informazioni, consultare the section called "Ruolo IAM del cluster" e the section called "Ruolo IAM del nodo".

Scelta di un ruolo IAM in Amazon EKS

Quando crei una risorsa cluster in Amazon EKS, devi scegliere un ruolo per consentire ad Amazon EKS di accedere a diverse altre AWS risorse per tuo conto. Se hai già creato un ruolo di servizio in precedenza, Amazon EKS ti fornisce un elenco di ruoli da scegliere. È importante scegliere un ruolo con policy gestite da Amazon EKS allegate ad esso. Per ulteriori informazioni, consultare the section called "Verifica della presenza di un ruolo del cluster esistente" e the section called "Verifica della presenza di un ruolo di nodo esistente".

Esempi di policy basate su identità Amazon EKS

Per impostazione predefinita, gli utenti e i ruoli IAM non dispongono dell'autorizzazione per creare o modificare risorse Amazon EKS. Inoltre, non possono eseguire attività utilizzando Console di gestione AWS la AWS CLI o AWS l'API. Un amministratore IAM deve creare policy IAM che concedono a utenti e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse specificate di cui hanno bisogno. L'amministratore deve quindi allegare queste policy a utenti o IAM che richiedono tali autorizzazioni.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consultare Creazione di policy nella scheda JSON nella Guida per l'utente di IAM.

Quando si crea un cluster Amazon EKS, il principale IAM che crea il cluster riceve automaticamente le autorizzazioni `system:masters` nella configurazione del controllo degli accessi basato sul ruolo (RBAC) nel piano di controllo di Amazon EKS. Questo principale non è visualizzato in una configurazione visibile qualsiasi, quindi assicurati di tenere traccia di quale principale ha originariamente creato il cluster. Per concedere a ulteriori principali IAM la capacità di interagire con il cluster, modifica `aws-auth ConfigMap` all'interno di Kubernetes e crea `rolebinding` o `clusterrolebinding` Kubernetes con il nome di `group` specificato in `aws-auth ConfigMap`.

Per ulteriori informazioni sull'utilizzo di ConfigMap, vederethe section called "Accesso API Kubernetes".

Argomenti

- Best practice per le policy
- Utilizzo della console Amazon EKS
- Consentire agli utenti IAM di visualizzare le loro autorizzazioni
- Crea un cluster Kubernetes sul cloud AWS
- Creazione di un cluster Kubernetes locale su un Outpost
- Aggiornare un cluster Kubernetes
- Elencare o descrivere tutti i cluster

Best practice per le policy

Le policy basate su identità determinano se qualcuno può creare, accedere o eliminare risorse Amazon EKS nell'account. Queste azioni possono comportare costi per il tuo AWS account. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche AWS gestite che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo account. AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.

- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite un AWS servizio specifico, ad esempio AWS CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle policy per IAM Access Analyzer nella Guida per l'utente IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel AWS tuo account, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungi le condizioni MFA alle policy. Per ulteriori informazioni, consulta Configurazione dell'accesso alle API protetto con MFA nella Guida per l'utente IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console Amazon EKS

Per accedere alla console Amazon EKS, un principale IAM deve disporre di un set di autorizzazioni minimo. Queste autorizzazioni consentono al responsabile di elencare e visualizzare i dettagli sulle risorse Amazon EKS nel tuo AWS account. Se crei una policy di basata su identità più restrittiva delle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per i principali associati a tale policy.

Per garantire che i principali IAM possano comunque utilizzare la console Amazon EKS, crea una policy con un nome univoco, ad esempio `AmazonEKSAdminPolicy`. Allega la politica ai principali. Per ulteriori informazioni, consulta Aggiunta e rimozione di autorizzazioni per identità IAM nella Guida per l'utente di IAM.

> **Important**
>
> La policy di esempio riportata di seguito consente a un principale di visualizzare le informazioni sulla scheda Configurazione della console. Per visualizzare le informazioni nelle schede Panoramica e Risorse di Console di gestione AWS, il principale necessita anche delle autorizzazioni Kubernetes. Per ulteriori informazioni, consulta the section called "Autorizzazioni richieste".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

Non è necessario consentire le autorizzazioni minime della console per i principali che effettuano chiamate solo alla AWS CLI o all'API. AWS Al contrario, è possibile accedere solo alle operazioni che soddisfano l'operazione API che stai cercando di eseguire.

Consentire agli utenti IAM di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o a livello di codice utilizzando la CLI o l'API AWS . 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": "*"
        }
    ]
}
```

Crea un cluster Kubernetes sul cloud AWS

Questa policy di esempio include le autorizzazioni minime richieste per creare un cluster Amazon EKS denominato *my-cluster* nella *us-west-2* AWS regione. Puoi sostituire la AWS regione con la AWS regione in cui desideri creare un cluster. Se viene visualizzato un avviso che indica che le azioni contenute nella politica non supportano le autorizzazioni a livello di risorsa e richiedono la scelta **All resources** Console di gestione AWS, è possibile ignorarlo senza problemi. Se il tuo account dispone già del ruolo *AWSServiceRoleForAmazonEKS*, puoi rimuovere l'azione `iam:CreateServiceLinkedRole` dalla policy. Se hai mai creato un cluster Amazon EKS nel tuo account, questo ruolo è già esistente, a meno che sia stato eliminato.

```
{
    "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"
        }
    ]
}
```

Creazione di un cluster Kubernetes locale su un Outpost

Questa policy di esempio include le autorizzazioni minime richieste per creare un cluster locale Amazon EKS denominato *my-cluster* su un Outpost nella *us-west-2* AWS regione. Puoi sostituire la AWS regione con la AWS regione in cui desideri creare un cluster. Se viene visualizzato un avviso che indica che le azioni contenute nella politica non supportano le autorizzazioni a livello di risorsa e richiedono la scelta **`All resources`** Console di gestione AWS, è possibile ignorarlo senza problemi. Se il tuo account dispone già del ruolo `AWSServiceRoleForAmazonEKSLocalOutpost`, puoi rimuovere l'azione `iam:CreateServiceLinkedRole` dalla policy. Se hai già creato un cluster locale Amazon EKS su Outpost nel tuo account, questo ruolo esiste già a meno che tu non l'abbia eliminato.

```
{
    "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"
        }
    ]
}
```

Aggiornare un cluster Kubernetes

Questa politica di esempio include l'autorizzazione minima richiesta per aggiornare un cluster denominato *`my-cluster`* nella regione 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"
        }
    ]
}
```

Elencare o descrivere tutti i cluster

Questa policy di esempio include le autorizzazioni minime richieste per elencare e descrivere tutti i cluster dell'account. Un principale IAM deve essere in grado di elencare e descrivere i cluster per utilizzare il comando `update-kubeconfig` AWS CLI.

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

```
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

Utilizzo di ruoli collegati ai servizi per Amazon EKS

Amazon Elastic Kubernetes Service utilizza service-linked roles di AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. I ruoli collegati ai servizi sono definiti automaticamente da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per eseguire chiamate agli altri servizi AWS per conto dell'utente.

Argomenti

- Utilizzo dei ruoli per i cluster Amazon EKS
- Utilizzo dei ruoli per i gruppi di nodi Amazon EKS
- Utilizzo dei ruoli per i profili Fargate in Amazon EKS
- Utilizzo di ruoli per connettere un cluster Kubernetes ad Amazon EKS
- Utilizzo dei ruoli per i cluster locali Amazon EKS su Outpost
- Utilizzo dei ruoli per la dashboard Amazon EKS

Utilizzo dei ruoli per i cluster Amazon EKS

Amazon Elastic Kubernetes Service utilizza ruoli collegati a servizi AWS Identity and Access Management (IAM) Ruoli collegati ai servizi. Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. I ruoli collegati ai servizi sono definiti automaticamente da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per eseguire chiamate agli altri servizi AWS per conto dell'utente.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le

autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consultare Servizi AWS che funzionano con IAM e cercare i servizi che riportano Yes (Sì) nella colonna Service-linked roles (Ruoli collegati ai servizi). Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKS`. Il ruolo consente ad Amazon EKS di gestire i cluster nell'account. Le policy allegate consentono al ruolo di gestire le seguenti risorse: interfacce di rete, gruppi di sicurezza, registri e VPC.

> **Note**
>
> Il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKS` è distinto dal ruolo richiesto per la creazione del cluster. Per ulteriori informazioni, consulta the section called "Ruolo IAM del cluster".

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKS` considera attendibili i seguenti servizi:

- `eks.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad Amazon EKS di eseguire le seguenti operazioni sulle risorse specificate:

- AmazonEKSServiceRolePolicy

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando si crea un cluster in Console di gestione AWS, AWS CLI, o AWS API, Amazon EKS crea automaticamente il ruolo collegato al servizio.

Se elimini questo ruolo collegato ai servizi, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando viene creato un cluster, Amazon EKS crea di nuovo il ruolo collegato ai servizi automaticamente.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKS`. Dopo aver creato un ruolo collegato al servizio, non potrai modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra selezionare Clusters (Cluster).

3. Se il cluster dispone di gruppi di nodi o profili Fargate, è necessario eliminarli prima di eliminare il cluster. Per ulteriori informazioni, consultare the section called "Elimina" e the section called "Eliminare profili".
4. Sulla pagina Cluster, scegliere il cluster da eliminare e scegliere Elimina.
5. Digitare il nome del cluster nella finestra di conferma eliminazione, quindi scegliere Elimina.
6. Ripetere questa procedura per tutti gli altri cluster nell'account. Attendere che tutte le operazioni di eliminazione siano completate.

Eliminazione manuale del ruolo collegato ai servizi

Utilizza la console IAM, la CLI di AWS o l'API AWS per eliminare il ruolo orientato ai servizi `AWSServiceRoleForAmazonEKS`. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EKS

Amazon EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le Regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la sezione Amazon EKS endpoints and quotas (Endpoint e quote di servizio di Amazon EKS).

Utilizzo dei ruoli per i gruppi di nodi Amazon EKS

Amazon EKS utilizza i ruoli collegati ai servizi di AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. I ruoli collegati ai servizi sono definiti automaticamente da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per eseguire chiamate agli altri servizi AWS per conto dell'utente.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consultare Servizi AWS che funzionano con IAM e cercare i servizi che riportano Yes (Sì) nella colonna Service-linked roles

(Ruoli collegati ai servizi). Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKSNodegroup`. Il ruolo consente ad Amazon EKS di gestire i gruppi di nodi nell'account. La policy `AWSServiceRoleForAmazonEKSNodegroup` allegata consente al ruolo di gestire le seguenti risorse: gruppi Auto Scaling, gruppi di sicurezza, modelli di avvio e profili dell'istanza IAM. Per ulteriori informazioni, consulta the section called "AWS politica gestita: AWSService RoleForAmazon EKSNodegroup".

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSNodegroup` considera attendibili i seguenti servizi:

- `eks-nodegroup.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad Amazon EKS di eseguire le seguenti operazioni sulle risorse specificate:

- AWSServiceRoleForAmazonEKSNodegroup

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando si crea un gruppo di nodi gestiti in Console di gestione AWS, AWS CLI o nell'API AWS, Amazon EKS crea automaticamente il ruolo collegato al servizio.

> **Important**
>
> Questo ruolo collegato al servizio può apparire nell'account, se è stata completata un'operazione in un altro servizio che utilizza le caratteristiche supportate da questo ruolo. Se si utilizzava il servizio Amazon EKS prima dell'1 Gennaio 2017, data da cui è disponibile il supporto dei ruoli collegati ai servizi, Amazon EKS ha creato il ruolo

AWSServiceRoleForAmazonEKSNodegroup nell'account. Per ulteriori informazioni, consultare Un nuovo ruolo è apparso nel mio account IAM.

Creazione di un ruolo collegato ai servizi in Amazon EKS (AWS API)

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando si crea un gruppo di nodi gestiti in Console di gestione AWS, AWS CLI o nell'API AWS, Amazon EKS crea automaticamente il ruolo collegato al servizio.

Se elimini questo ruolo collegato ai servizi, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando si crea un altro gruppo di nodi gestiti, Amazon EKS crea nuovamente il ruolo collegato al servizio.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSNodegroup`. Dopo aver creato un ruolo collegato al servizio, non potrai modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Seleziona la scheda Compute (Calcolo).
4. Nella sezione Gruppi di nodi, scegliere il gruppo di nodi da eliminare.
5. Digitare il nome del gruppo di nodi nella finestra di conferma eliminazione e quindi scegliere Elimina.
6. Ripetere questa procedura per tutti gli altri gruppi di nodi nel cluster. Attendere che tutte le operazioni di eliminazione siano completate.

Eliminazione manuale del ruolo collegato ai servizi

Utilizza la console IAM, la CLI di AWS o l'API AWS per eliminare il ruolo orientato ai servizi `AWSServiceRoleForAmazonEKSNodegroup`. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EKS

Amazon EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le Regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la sezione Amazon EKS endpoints and quotas (Endpoint e quote di servizio di Amazon EKS).

Utilizzo dei ruoli per i profili Fargate in Amazon EKS

Amazon EKS utilizza i ruoli collegati ai servizi di AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. I ruoli collegati ai servizi sono definiti automaticamente da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per eseguire chiamate agli altri servizi AWS per conto dell'utente.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consultare Servizi AWS che funzionano con IAM e cercare i servizi che riportano Yes (Sì) nella colonna Service-linked roles (Ruoli collegati ai servizi). Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKSForFargate`. Il ruolo consente ad Amazon EKS Fargate di configurare la rete VPC necessaria per i pod Fargate. Le policy collegate consentono al ruolo di creare ed eliminare le interfacce di rete elastiche e di descrivere le risorse e le interfacce di rete elastiche.

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSForFargate` considera attendibili i seguenti servizi:

- `eks-fargate.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad Amazon EKS di eseguire le seguenti operazioni sulle risorse specificate:

- AmazonEKSForFargateServiceRolePolicy

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un profilo Fargate in Console di gestione AWS, AWS CLI o nell'API AWS, Amazon EKS crea il ruolo collegato al servizio per l'utente.

> **Important**
>
> Questo ruolo collegato al servizio può apparire nell'account, se è stata completata un'operazione in un altro servizio che utilizza le caratteristiche supportate da questo ruolo. Se si utilizzava il servizio Amazon EKS prima del 13 dicembre 2019, data da cui è disponibile il supporto dei ruoli collegati ai servizi, allora Amazon EKS ha creato il ruolo

AWSServiceRoleForAmazonEKSForFargate nell'account. Per ulteriori informazioni, consulta la sezione A New role appeared in my IAM account (Nel mio account IAM è apparso un nuovo ruolo).

Creazione di un ruolo collegato ai servizi in Amazon EKS (AWS API)

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un profilo Fargate in Console di gestione AWS, AWS CLI o nell'API AWS, Amazon EKS crea il ruolo collegato al servizio per l'utente.

Se elimini questo ruolo collegato ai servizi, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando si crea un altro gruppo di nodi gestiti, Amazon EKS crea nuovamente il ruolo collegato al servizio.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSForFargate`. Dopo aver creato un ruolo collegato al servizio, non potrai modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Nella pagina Cluster selezionare il cluster.
4. Seleziona la scheda Compute (Calcolo).
5. Se sono presenti profili Fargate nella sezione Fargate profiles (Profili Fargate), selezionare ciascun profilo singolarmente, quindi scegliere Delete (Elimina).
6. Digitare il nome del cluster nella finestra di conferma eliminazione e quindi scegliere Elimina.
7. Ripetere questa procedura per tutti gli altri profili Fargate nel cluster e per ciascun cluster nell'account.

Eliminazione manuale del ruolo collegato ai servizi

Utilizza la console IAM, la AWS CLI o l'API AWS per eliminare il ruolo collegato al servizio AWSServiceRoleForAmazonEKSForFargate. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EKS

Amazon EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le Regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la sezione Amazon EKS endpoints and quotas (Endpoint e quote di servizio di Amazon EKS).

Utilizzo di ruoli per connettere un cluster Kubernetes ad Amazon EKS

Amazon EKS utilizza ruoli AWS collegati al servizio Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. I ruoli collegati ai servizi sono predefiniti da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato al servizio solo dopo avere eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consulta Servizi AWS che funzionano con IAM e cerca i servizi che riportano Yes (Sì) nella colonna Service-linked roles (Ruoli collegati ai servizi). Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKSConnector`. Il ruolo consente ad Amazon EKS di collegarsi ai cluster Kubernetes. Le policy allegate consentono al ruolo di gestire le risorse necessarie per connettersi al cluster Kubernetes registrato.

Ai fini dell'assunzione del ruolo, il ruolo collegato al servizio `AWSServiceRoleForAmazonEKSConnector` considera attendibili i seguenti servizi:

- `eks-connector.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad Amazon EKS di eseguire le seguenti operazioni sulle risorse specificate:

- AmazonEKSConnectorServiceRolePolicy

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato al servizio è necessario configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente IAM.

Questo ruolo utilizza le autorizzazioni SSM (Systems Manager) per stabilire connessioni sicure e gestire i cluster Kubernetes connessi.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non devi creare manualmente un ruolo collegato ai servizi per collegare un cluster. Quando connetti un cluster nella AWS CLI o nell' AWS API`eksctl`, Amazon EKS crea il ruolo collegato al servizio per te. Console di gestione AWS

Se elimini questo ruolo collegato al servizio, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando viene collegato un cluster, Amazon EKS crea di nuovo il ruolo collegato ai servizi automaticamente.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSConnector`. Dopo avere creato un ruolo collegato al servizio, non sarà possibile modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato al servizio nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, seleziona Cluster.
3. Nella pagina Cluster selezionare il cluster.
4. Selezionare la scheda Annulla registrazione e quindi selezionare la scheda Ok.

Eliminazione manuale del ruolo collegato ai servizi

Utilizza la console IAM, la AWS CLI o l' AWS API per eliminare il ruolo collegato al AWSService RoleForAmazon EKSConnector servizio. Per ulteriori informazioni, consultare Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Utilizzo dei ruoli per i cluster locali Amazon EKS su Outpost

Amazon Elastic Kubernetes Service utilizza ruoli collegati a servizi AWS Identity and Access Management (IAM) Ruoli collegati ai servizi. Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. I ruoli collegati ai servizi sono definiti automaticamente da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per eseguire chiamate agli altri servizi AWS per conto dell'utente.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consultare Servizi AWS che funzionano con IAM e cercare i servizi che riportano Yes (Sì) nella colonna Service-linked roles (Ruoli collegati ai servizi). Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKSLocalOutpost`. Il ruolo consente ad Amazon EKS di gestire i cluster locali nell'account. Le policy collegate consentono al ruolo di gestire le seguenti risorse: interfacce di rete, gruppi di sicurezza, registri e istanze Amazon EC2.

> **Note**
>
> Il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSLocalOutpost` è distinto dal ruolo richiesto per la creazione del cluster. Per ulteriori informazioni, consulta the section called "Ruolo IAM del cluster".

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSLocalOutpost` considera attendibili i seguenti servizi:

- `outposts.eks-local.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad Amazon EKS di eseguire le seguenti operazioni sulle risorse specificate:

- AmazonEKSServiceRolePolicy

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando si crea un cluster in Console di gestione AWS, AWS CLI, o AWS API, Amazon EKS crea automaticamente il ruolo collegato al servizio.

Se elimini questo ruolo collegato ai servizi, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando viene creato un cluster, Amazon EKS crea di nuovo il ruolo collegato ai servizi automaticamente.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSLocalOutpost`. Dopo aver creato un ruolo collegato al servizio, non è possibile modificarne il nome, perché potrebbero farvi riferimento diverse entità. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione a sinistra, scegli Clusters (Cluster) Amazon EKS.
3. Se il cluster dispone di gruppi di nodi o profili Fargate, è necessario eliminarli prima di eliminare il cluster. Per ulteriori informazioni, consultare the section called "Elimina" e the section called "Eliminare profili".
4. Sulla pagina Cluster, scegliere il cluster da eliminare e scegliere Elimina.
5. Digitare il nome del cluster nella finestra di conferma eliminazione, quindi scegliere Elimina.
6. Ripetere questa procedura per tutti gli altri cluster nell'account. Attendere che tutte le operazioni di eliminazione siano completate.

Eliminazione manuale del ruolo collegato ai servizi

Utilizza la console IAM, la CLI di AWS o l'API AWS per eliminare il ruolo orientato ai servizi `AWSServiceRoleForAmazonEKSLocalOutpost`. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EKS

Amazon EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le Regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la sezione Amazon EKS endpoints and quotas (Endpoint e quote di servizio di Amazon EKS).

Utilizzo dei ruoli per la dashboard Amazon EKS

La dashboard Amazon EKS utilizza questo ruolo collegato ai servizi per aggregare informazioni sulle risorse del cluster da più regioni e account. La dashboard si integra con AWS Organizations per leggere in modo sicuro le informazioni su più account dell'organizzazione.

Per ulteriori informazioni sulle dashboard Amazon EKS, consulta the section called "Pannello di controllo Amazon EKS".

Contesto

Amazon EKS utilizza i ruoli collegati ai servizi di AWS Identity and Access Management (IAM). Un ruolo collegato ai servizi è un tipo di ruolo IAM univoco collegato direttamente ad Amazon EKS. I ruoli collegati ai servizi sono definiti automaticamente da Amazon EKS e includono tutte le autorizzazioni richieste dal servizio per eseguire chiamate agli altri servizi AWS per conto dell'utente.

Un ruolo collegato ai servizi semplifica la configurazione di Amazon EKS perché consente di evitare l'aggiunta manuale delle autorizzazioni necessarie. Amazon EKS definisce le autorizzazioni del ruolo associato ai servizi e, salvo diversamente definito, solo Amazon EKS può assumerne il ruolo. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere allegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di Amazon EKS perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni sugli altri servizi che supportano i ruoli collegati ai servizi, consultare Servizi AWS che funzionano con IAM e cercare i servizi che riportano Yes (Sì) nella colonna Service-linked roles (Ruoli collegati ai servizi). Scegli Sì in corrispondenza di un link per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per Amazon EKS

Amazon EKS usa il ruolo collegato ai servizi denominato `AWSServiceRoleForAmazonEKSDashboard`. Il ruolo consente ad Amazon EKS di generare e visualizzare la dashboard EKS, incluse informazioni aggregate sulle risorse del cluster. La policy `AmazonEKSDashboardServiceRolePolicy` allegata consente al ruolo di gestire le seguenti risorse: gruppi Auto Scaling, gruppi di sicurezza, modelli di avvio e profili dell'istanza IAM. Per ulteriori informazioni, consulta the section called "AWS politica gestita: Amazon EKSDashboard ServiceRolePolicy".

Ai fini dell'assunzione del ruolo, il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSDashboard` considera attendibili i seguenti servizi:

- `dashboard.eks.amazonaws.com`

Per visualizzare la versione più recente del documento di policy JSON, consulta AmazonEKSDashboardServiceRolePolicy nella Guida di riferimento sulle policy gestite da AWS.

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per Amazon EKS

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando è abilitata la dashboard o nella console AWS, Amazon EKS crea il ruolo collegato ai servizi.

Per ulteriori informazioni sull'abilitazione della dashboard EKS, consulta the section called "Configurare le organizzazioni".

> **Important**
>
> Questo ruolo collegato al servizio può apparire nell'account, se è stata completata un'operazione in un altro servizio che utilizza le caratteristiche supportate da questo ruolo.

Modifica di un ruolo collegato ai servizi per Amazon EKS

Amazon EKS non consente di modificare il ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSDashboard`. Dopo aver creato un ruolo collegato al servizio, non potrai modificarne il nome perché varie entità potrebbero farvi riferimento. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per Amazon EKS

Se non è più necessario utilizzare una funzionalità o un servizio che richiede un ruolo collegato al servizio, ti consigliamo di eliminare il ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

Pulizia di un ruolo collegato ai servizi

Prima di utilizzare IAM; per eliminare un ruolo collegato al servizio, è necessario prima rimuovere qualsiasi risorsa utilizzata dal ruolo.

> **Note**
>
> Se il servizio Amazon EKS utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non riesca. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

Per ulteriori informazioni sulla disabilitazione della dashboard EKS, consulta the section called "Configurare le organizzazioni".

Eliminazione manuale del ruolo collegato ai servizi

Utilizza la console IAM, la CLI di AWS o l'API AWS per eliminare il ruolo orientato ai servizi `AWSServiceRoleForAmazonEKSDashboard`. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di Amazon EKS

Amazon EKS supporta l'utilizzo di ruoli collegati ai servizi in tutte le Regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta la sezione Amazon EKS endpoints and quotas (Endpoint e quote di servizio di Amazon EKS).

Ruolo IAM di esecuzione del pod di Amazon EKS

Il ruolo di esecuzione di Amazon EKS Pod è necessario per eseguire Pods sull'infrastruttura AWS Fargate.

Quando il cluster crea Pods sull'infrastruttura AWS Fargate, i componenti in esecuzione sull'infrastruttura Fargate devono effettuare chiamate AWS APIs per conto dell'utente. In questo modo possono eseguire azioni come estrarre le immagini dei container da Amazon ECR o indirizzare i log ad altri AWS servizi. Il ruolo di esecuzione del pod Amazon EKS fornisce le autorizzazioni IAM per eseguire questa operazione.

Quando si crea un profilo Fargate, è necessario specificare un ruolo di esecuzione del pod per i componenti Amazon EKS che vengono eseguiti su infrastruttura Fargate utilizzando il profilo. Questo ruolo è aggiunto al controllo di accesso basato sul ruolo (RBAC) di Kubernetes del cluster per l'autorizzazione. Ciò consente al `kubelet` in esecuzione sull'infrastruttura Fargate di registrarsi con il cluster Amazon EKS in modo che possa essere visualizzato nel cluster come nodo.

> **Note**
>
> Il profilo Fargate deve avere un ruolo IAM diverso rispetto ai gruppi di EC2 nodi Amazon.

> **Important**
>
> I container in esecuzione nel pod Fargate non possono assumere le autorizzazioni IAM associate a un ruolo di esecuzione del pod. Per concedere ai contenitori del tuo Fargate Pod le autorizzazioni per accedere ad altri AWS servizi, devi utilizzare i ruoli IAM per gli account di servizio.

Prima di creare un profilo Fargate, devi creare un ruolo IAM con Amazon. EKSFargate PodExecutionRolePolicy

Verificare la presenza di un ruolo di esecuzione del pod esistente configurato correttamente

Per verificare se l'account dispone già di un ruolo di esecuzione del pod Amazon EKS configurato correttamente, utilizzare la procedura indicata di seguito. Per prevenire il problema di sicurezza di un delegato confuso, è importante che il ruolo limiti l'accesso in base a `SourceArn`. È possibile modificare il ruolo di esecuzione secondo necessità per includere il supporto per i profili Fargate su altri cluster.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, cerca l'elenco dei ruoli per Amazon EKSFargate PodExecutionRole. Se il ruolo non esiste, consulta the section called "Creazione del ruolo di esecuzione del pod Amazon EKS" per crearlo. Se il ruolo è presente, selezionalo.
4. Nella EKSFargate PodExecutionRole pagina Amazon, procedi come segue:
   a. Seleziona Autorizzazioni.
   b. Assicurati che la policy gestita da EKSFargatePodExecutionRolePolicyAmazon Amazon sia associata al ruolo.
   c. Scegli Trust relationships (Relazioni di trust).
   d. Seleziona Edit trust policy (Modifica policy di attendibilità).

5. Nella pagina Edit trust policy (Modifica policy di attendibilità), verifica che la relazione di attendibilità contenga la policy seguente e una riga per i profili Fargate nel cluster. In tal caso, scegli Cancel (Annulla).

```
{
  "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 la policy corrisponde ma non contiene una riga che specifica i profili Fargate sul cluster, aggiungi la riga riportata di seguito nella parte superiore dell'oggetto `ArnLike`. Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster, *`111122223333`* con l'ID dell'account e *`my-cluster`* con il nome del cluster.

```
"aws:SourceArn": "arn:aws: eks:region-code:111122223333:fargateprofile/my-cluster/*",
```

Se la policy non corrisponde, copia la policy precedente nel modulo e scegli Update policy (Aggiorna policy). Sostituiscilo *`region-code`* con la AWS regione in cui si trova il cluster. Se desideri utilizzare lo stesso ruolo in tutte le AWS regioni del tuo account, sostituiscilo *`region-code`* con*. Sostituisci *`111122223333`* con il tuo ID account e *`my-cluster`* con il nome del cluster. Se vuoi usare lo stesso ruolo per tutti i cluster dell'account, sostituisci *`my-cluster`* con *.

Creazione del ruolo di esecuzione del pod Amazon EKS

Se non disponi già del ruolo di esecuzione di Amazon EKS Pod per il tuo cluster, puoi utilizzare Console di gestione AWS o la AWS CLI per crearlo.

Console di gestione AWS

a. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
b. Nel pannello di navigazione a sinistra, seleziona Ruoli.
c. Nella pagina Ruoli, seleziona Crea ruolo.
d. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
    i. Nella sezione Tipo di entità affidabile, scegli AWS servizio.
    ii. Dall'elenco a discesa Casi d'uso per altri AWS servizi, scegli EKS.
    iii. Scegli EKS - Pod Fargate.
    iv. Scegli Next (Successivo).
e. Nella pagina Add permissions (Aggiungi autorizzazioni), scegli Next (Successivo).
f. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:
    i. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSFargatePodExecutionRole`.
    ii. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.
    iii. Scegli Crea ruolo.
g. Nella pagina Ruoli, cerca l'elenco dei ruoli per Amazon EKSFargate PodExecutionRole. Seleziona il ruolo.
h. Nella EKSFargate PodExecutionRole pagina Amazon, procedi come segue:
    i. Scegli Trust relationships (Relazioni di trust).
    ii. Seleziona Edit trust policy (Modifica policy di attendibilità).
i. Nella pagina Modifica policy di attendibilità, effettua le operazioni seguenti:
    i. Copia e incolla il contenuto seguente nel modello Modifica policy di attendibilità. Sostituisci *`region-code`* con la AWS regione in cui si trova il cluster. Se desideri utilizzare lo stesso ruolo in tutte le AWS regioni del tuo account, sostituiscilo *`region-code`* con`*`. Sostituisci

111122223333 con il tuo ID account e *my-cluster* con il nome del cluster. Se vuoi usare lo stesso ruolo per tutti i cluster dell'account, sostituisci *my-cluster* con *.

```
{
  "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. Scegli Aggiorna policy.

AWS CLI

a. Copia e incolla il contenuto seguente in un file denominato `pod-execution-role-trust-policy.json`. Sostituisci *region-code* con la AWS regione in cui si trova il cluster. Se desideri utilizzare lo stesso ruolo in tutte le AWS regioni del tuo account, sostituiscilo *region-code* con*. Sostituisci *111122223333* con il tuo ID account e *my-cluster* con il nome del cluster. Se vuoi usare lo stesso ruolo per tutti i cluster dell'account, sostituisci *my-cluster* con *.

```
{
  "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. Creare un ruolo IAM di esecuzione del pod.

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

c. Allega la policy IAM gestita da Amazon EKS richiesta al ruolo.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Ruolo IAM di Amazon EKS Connector

Puoi connettere i cluster Kubernetes per visualizzarli nel tuo. Console di gestione AWS Per collegarsi a un cluster Kubernetes, creare un ruolo IAM.

Verificare la presenza di un ruolo del connettore EKS esistente

Per controllare se l'account dispone già di un ruolo di Amazon EKS Connector, utilizzare la procedura indicata di seguito.

1. Apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Cerca l'elenco dei ruoli per `AmazonEKSConnectorAgentRole`. Se un ruolo che include `AmazonEKSConnectorAgentRole` non esiste, consultare the section called "Creazione del ruolo agente di Amazon EKS Connector" per crearlo. Se un ruolo che include `AmazonEKSConnectorAgentRole` esiste, seleziona il ruolo per visualizzare le policy allegate.
4. Selezionare Autorizzazioni.

5. Assicurati che la policy EKSConnector AgentPolicy gestita da Amazon sia associata al ruolo. Se la policy è collegata, il ruolo del connettore Amazon EKS è configurato correttamente.
6. Scegliere Relazioni di attendibilità, quindi scegliere Modifica policy di attendibilità.
7. Verifica che la relazione di trust includa la policy seguente. Se la relazione di attendibilità corrisponde alla policy seguente, scegli Annulla. Se la relazione di attendibilità non corrisponde, copia la policy nella finestra Modifica policy di attendibilità e scegli Aggiorna policy.

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

Creazione del ruolo agente di Amazon EKS Connector

Puoi usare Console di gestione AWS o AWS CloudFormation per creare il ruolo di agente del connettore.

AWS CLI

a. Creare un file denominato `eks-connector-agent-trust-policy.json` contenente il seguente JSON da utilizzare per il ruolo IAM.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
```

```
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Creare un file denominato `eks-connector-agent-policy.json` contenente il seguente JSON da utilizzare per il ruolo 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. Crea il ruolo agente di Amazon EKS Connector utilizzando la policy di attendibilità e la policy creata negli elementi precedenti dell'elenco.

```
aws iam create-role \
     --role-name AmazonEKSConnectorAgentRole \
     --assume-role-policy-document file://eks-connector-agent-trust-policy.json
```

d. Allegare la policy al ruolo agente di 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. Salvate il seguente AWS CloudFormation modello in un file di testo sul vostro sistema locale.

> **Note**
>
> Questo modello crea anche il ruolo collegato al servizio che altrimenti verrebbe stato creato al momento della chiamata API `registerCluster`. Per informazioni dettagliate, vedi the section called "Ruolo del connettore del 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. Apri la AWS CloudFormation console.

c. Scegliere Crea stack Con nuove risorse (standard).

d. In Specify template (Specifica modello), selezionare Upload a template file (Carica un file di modello) e Choose file (Scegli file).

e. Scegliere il file creato in precedenza, quindi selezionare Next (Successivo).

f. Per Stack name (Nome pila), immettere un nome per il ruolo, ad esempio `eksConnectorAgentRole`, quindi scegliere Next (Successivo).

g. Nella pagina Configure stack options (Configura opzioni pila), scegliere Next (Successivo).

h. Nella pagina Revisione, esaminare le informazioni, accettare che la pila può creare risorse IAM, quindi scegliere Crea pila.

AWS politiche gestite per Amazon Elastic Kubernetes Service

Una politica AWS gestita è una politica autonoma creata e amministrata da AWS. AWS le politiche gestite sono progettate per fornire autorizzazioni per molti casi d'uso comuni, in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Tieni presente che le policy AWS gestite potrebbero non concedere le autorizzazioni con il privilegio minimo per i tuoi casi d'uso specifici, poiché sono disponibili per tutti i clienti. AWS Si consiglia pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i propri casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle politiche gestite. AWS Se AWS aggiorna le autorizzazioni definite in una politica AWS gestita, l'aggiornamento ha effetto su tutte le identità principali (utenti, gruppi e ruoli) a cui è associata la politica. AWS è più probabile che aggiorni una policy AWS gestita quando viene lanciato un nuovo AWS servizio o quando nuove operazioni API diventano disponibili per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWS nella Guida per l'utente di IAM.

AWS politica gestita: Amazoneks_CNI_Policy

É possibile allegare la `AmazonEKS_CNI_Policy` alle entità IAM. Prima di creare un gruppo di nodi Amazon EC2, questa policy deve essere allegata al ruolo IAM del nodo, o a un ruolo IAM che è utilizzato in modo specifico dal plugin CNI di Amazon VPC per Kubernetes. In questo modo può eseguire operazioni per conto dell'utente. Si consiglia di allegare le policy a un ruolo utilizzato solo dal plugin. Per ulteriori informazioni, consultare the section called “CNI di Amazon VPC” e the section called “Configurare per IRSA”.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`ec2:*NetworkInterface`** e **`ec2:*PrivateIpAddresses`** : consente al plugin CNI di Amazon VPC di eseguire azioni come il provisioning di interfacce di rete elastiche e indirizzi IP per i pod, così da fornire una rete per le applicazioni eseguite in Amazon EKS.
- Azioni di lettura **`ec2`**: consente al plugin CNI di Amazon VPC di eseguire azioni come descrivere istanze e sottoreti, così da visualizzare la quantità di indirizzi IP gratuiti nelle sottoreti Amazon VPC. Il CNI di VPC può utilizzare gli indirizzi IP gratuiti in ciascuna sottorete per scegliere le sottoreti con gli indirizzi IP più liberi da utilizzare durante la creazione di un'interfaccia di rete elastica.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazoneks_CNI_Policy nella Managed Policy Reference Guide. AWS

AWS politica gestita: Amazon EKSCluster Policy

È possibile allegare `AmazonEKSClusterPolicy` alle entità IAM. Prima di creare un cluster, è necessario disporre di un ruolo IAM del cluster con questa policy allegata. I cluster Kubernetes gestiti da Amazon EKS effettuano chiamate verso altri AWS servizi per tuo conto. Ciò serve a gestire le risorse utilizzate con il servizio.

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`autoscaling`** : leggi e aggiorna la configurazione di un gruppo Auto Scaling. Queste autorizzazioni non sono utilizzate da Amazon EKS, ma rimangono nella policy per la compatibilità con le versioni precedenti.
- **`ec2`** : lavora con volumi e risorse di rete associati ai nodi Amazon EC2. Ciò è necessario affinché il piano di controllo (control-plane) Kubernetes possa unire le istanze a un cluster ed eseguire dinamicamente il provisioning e la gestione dei volumi Amazon EBS richiesti dai volumi persistenti di Kubernetes.
- **`ec2`** : elimina le interfacce di rete elastiche create da CNI di VPC. Ciò è necessario affinché EKS possa ripulire le interfacce di rete elastiche che rimangono se CNI di VPC si chiude inaspettatamente.
- **`elasticloadbalancing`** : lavora con Elastic Load Balancers e aggiunge nodi come destinazioni. Ciò è necessario affinché il piano di controllo Kubernetes possa eseguire dinamicamente il provisioning degli Elastic Load Balancers (load balancer Elastico) richiesti dai servizi Kubernetes.
- **`iam`** : crea un ruolo collegato ai servizi. Ciò è necessario affinché il piano di controllo (control-plane) Kubernetes possa eseguire dinamicamente il provisioning degli Elastic Load Balancer richiesti dai servizi Kubernetes.
- **`kms`**— Leggi una chiave da KMS. AWS Questa attività è necessaria affinché il piano di controllo (control-plane) Kubernetes supporti la crittografia dei segreti di Kubernetes archiviati in `etcd`.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSCluster Policy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSDashboard ConsoleReadOnly

È possibile collegare `AmazonEKSDashboardConsoleReadOnly` alle entità IAM.

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`eks`** : accesso in sola lettura ai dati, alle risorse e alle informazioni sulle versioni del cluster della dashboard EKS. Ciò consente di visualizzare le metriche relative a EKS e i dettagli di configurazione del cluster.
- **`organizations`**- Accesso in sola lettura alle informazioni di AWS Organizations, tra cui:

- visualizzazione dei dettagli dell'organizzazione e dell'accesso al servizio
- elenco delle root organizzative, degli account e delle unità organizzative
- Visualizzazione della struttura dell'organizzazione

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSDashboard ConsoleReadOnly nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSFargate PodExecutionRolePolicy

È possibile allegare `AmazonEKSFargatePodExecutionRolePolicy` alle entità IAM. Prima di poter creare un profilo Fargate, è necessario creare un ruolo di esecuzione del pod Fargate e allegare questa policy ad esso. Per ulteriori informazioni, consultare the section called "Fase 2: creare un ruolo di esecuzione del pod Fargate" e the section called "Definisci i profili".

Questa politica concede al ruolo le autorizzazioni che forniscono l'accesso ad altre risorse di AWS servizio necessarie per eseguire Amazon EKS Pods su Fargate.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`ecr`** : consente ai pod in esecuzione su Fargate di estrarre le immagini del container memorizzate in Amazon ECR.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSFargate PodExecutionRolePolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSConnector ServiceRolePolicy

Non è possibile allegare `AmazonEKSConnectorServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon EKS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta the section called "Ruolo del connettore del cluster".

Il ruolo consente ad Amazon EKS di collegarsi ai cluster Kubernetes. Le policy allegate consentono al ruolo di gestire le risorse necessarie per connettersi al cluster Kubernetes registrato.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le seguenti attività.

- **`SSM Management`**— Crea, descrivi ed elimina le attivazioni SSM e annulla la registrazione delle istanze gestite. Ciò consente le operazioni di base di Systems Manager.
- **`Session Management`**— Avvia sessioni SSM specifiche per i cluster EKS ed esegui comandi non interattivi utilizzando il documento AmazonEks.
- **`IAM Role Passing`**— Passa i ruoli IAM specificamente al servizio SSM, controllato da una condizione che limita i ruoli passati a. `ssm.amazonaws.com`
- **`EventBridge Rules`**— Crea EventBridge regole e obiettivi, ma solo se gestiti da. `eks-connector.amazonaws.com` Le regole sono specificamente limitate a AWS SSM come fonte dell'evento.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSConnector ServiceRolePolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSFor FargateServiceRolePolicy

Non è possibile allegare `AmazonEKSForFargateServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon EKS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta `AWSServiceRoleforAmazonEKSForFargate`.

Questa policy concede ad Amazon EKS le autorizzazioni necessarie per eseguire le attività di Fargate. La policy viene utilizzato solo se si dispone di nodi Fargate.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le seguenti attività.

- **`ec2`** : crea ed elimina interfacce di rete elastiche e descrive le interfacce di rete elastiche e le risorse. Questa operazione è necessaria affinché il servizio Fargate di Amazon EKS possa configurare la rete VPC necessaria per i pod Fargate.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSFor FargateServiceRolePolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSCompute Policy

È possibile collegare `AmazonEKSComputePolicy` alle entità IAM. È possibile allegare questa policy al ruolo IAM del cluster per espandere le risorse che EKS può gestire nel tuo account.

Questa policy concede le autorizzazioni necessarie ad Amazon EKS per creare e gestire istanze EC2 per il cluster EKS e le autorizzazioni IAM necessarie per configurare EC2. Inoltre, questa policy concede ad Amazon EKS le autorizzazioni per creare il ruolo collegato al servizio Spot EC2 per conto dell'utente.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- Autorizzazioni**`ec2`** :
  - `ec2:CreateFleet` e `ec2:RunInstances`: consente la creazione di istanze EC2 e l'utilizzo di risorse EC2 specifiche (immagini, gruppi di sicurezza, sottoreti) per i nodi del cluster EKS.
  - `ec2:CreateLaunchTemplate`: consente di creare modelli di lancio EC2 per i nodi del cluster EKS.
  - La policy include anche condizioni per limitare l'utilizzo di queste autorizzazioni EC2 alle risorse contrassegnate con il nome del cluster EKS e altri tag pertinenti.
  - `ec2:CreateTags`: consente di aggiungere tag alle risorse EC2 create dalle azioni `CreateFleet`, `RunInstances` e `CreateLaunchTemplate`.
- Autorizzazioni **`iam`**:
  - `iam:AddRoleToInstanceProfile`: consente di aggiungere un ruolo IAM al profilo dell'istanza di calcolo EKS.
  - `iam:PassRole`: consente di passare i ruoli IAM necessari al servizio EC2.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSCompute Policy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSNetworking Policy

È possibile collegare `AmazonEKSNetworkingPolicy` alle entità IAM. È possibile allegare questa policy al ruolo IAM del cluster per espandere le risorse che EKS può gestire nel tuo account.

Questa policy è progettata per concedere le autorizzazioni necessarie ad Amazon EKS per creare e gestire interfacce di rete per il cluster EKS, consentendo al piano di controllo e ai nodi worker di comunicare e funzionare correttamente.

Dettagli delle autorizzazioni

Questa policy concede le seguenti autorizzazioni per consentire ad Amazon EKS di gestire le interfacce di rete per il cluster:

- Autorizzazioni dell'interfaccia di rete **`ec2`**
  - `ec2:CreateNetworkInterface`: consente la creazione di interfacce di rete EC2.
  - La policy include condizioni per limitare l'utilizzo di questa autorizzazione alle interfacce di rete contrassegnate con il nome del cluster EKS e il nome del nodo CNI di Kubernetes.
  - `ec2:CreateTags`: consente di aggiungere tag alle interfacce di rete create dall'azione `CreateNetworkInterface`.
- Autorizzazioni di gestione delle interfacce di rete **`ec2`**:
  - `ec2:AttachNetworkInterface`, `ec2:DetachNetworkInterface`: consente di collegare e scollegare interfacce di rete alle istanze EC2.
  - `ec2:UnassignPrivateIpAddresses`, `ec2:UnassignIpv6Addresses`, `ec2:AssignPrivateIpAddresses`, `ec2:AssignIpv6Addresses`: consente di gestire le assegnazioni degli indirizzi IP delle interfacce di rete.
  - Tali autorizzazioni sono limitate alle interfacce di rete contrassegnate con il nome del cluster EKS.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSNetworking Policy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSBlock StoragePolicy

È possibile collegare `AmazonEKSBlockStoragePolicy` alle entità IAM. È possibile allegare questa policy al ruolo IAM del cluster per espandere le risorse che EKS può gestire nel tuo account.

Questa policy concede le autorizzazioni necessarie ad Amazon EKS per creare, gestire e mantenere volumi e snapshot EC2 per il cluster EKS, consentendo al piano di controllo e ai nodi worker di fornire e utilizzare l'archiviazione persistente come richiesto dai carichi di lavoro Kubernetes.

Dettagli delle autorizzazioni

Questa policy IAM concede le seguenti autorizzazioni per consentire ad Amazon EKS di gestire volumi e snapshot EC2:

- Autorizzazioni per la gestione dei volumi **`ec2`**:
  - `ec2:AttachVolume`, `ec2:DetachVolume`, `ec2:ModifyVolume`, `ec2:EnableFastSnapshotRestores`: consente di allegare, scollegare, modificare e abilitare ripristini rapidi di snapshot per i volumi EC2.
  - Queste autorizzazioni sono limitate ai volumi contrassegnati con il nome del cluster EKS.
  - `ec2:CreateTags`: consente di aggiungere tag ai volumi e agli snapshot di EC2 creati dalle azioni `CreateVolume` e `CreateSnapshot`.
- Autorizzazioni per la creazione di volumi **`ec2`**:
  - `ec2:CreateVolume`: consente la creazione di nuovi volumi EC2.
  - La policy include condizioni per limitare l'utilizzo di questa autorizzazione a volumi contrassegnati con il nome del cluster EKS e altri tag pertinenti.
  - `ec2:CreateSnapshot`: consente la creazione di nuovi snapshot di volumi EC2.
  - La policy include condizioni per limitare l'utilizzo di questa autorizzazione agli snapshot contrassegnati con il nome del cluster EKS e altri tag pertinenti.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSBlock StoragePolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSLoad BalancingPolicy

È possibile collegare `AmazonEKSLoadBalancingPolicy` alle entità IAM. È possibile allegare questa policy al ruolo IAM del cluster per espandere le risorse che EKS può gestire nel tuo account.

Questa policy IAM concede le autorizzazioni necessarie ad Amazon EKS per lavorare con vari AWS servizi per gestire Elastic Load Balancers (ELBs) e risorse correlate.

Dettagli delle autorizzazioni

Le autorizzazioni principali concesse da questa policy sono:

- **`elasticloadbalancing`** : consente di creare, modificare e gestire Elastic Load Balancer e gruppi di destinazione. Ciò include le autorizzazioni per creare, aggiornare ed eliminare bilanciatori del carico, gruppi di destinazione, listener e regole.

- **`ec2`** : consente di creare e gestire gruppi di sicurezza, necessari al piano di controllo Kubernetes per unire le istanze a un cluster e gestire i volumi Amazon EBS. Consente inoltre di descrivere ed elencare risorse EC2 come istanze, sottoreti VPCs, gruppi di sicurezza e altre risorse di rete.
- **`iam`**: consente di creare un ruolo collegato al servizio per Elastic Load Balancing, necessario per il provisioning dinamico del piano di controllo Kubernetes. ELBs
- **`kms`**: Consente la lettura di una chiave da AWS KMS, necessaria affinché il piano di controllo di Kubernetes supporti la crittografia dei segreti Kubernetes archiviati in etcd.
- **`wafv2`**e **`shield`**: Consente di associare e dissociare Web e di ACLs creare/eliminare le protezioni AWS Shield per Elastic Load Balancers.
- **`cognito-idp`** , **`acm`** e **`elasticloadbalancing`** : concede le autorizzazioni per descrivere i client del pool di utenti, elencare e descrivere i certificati, e descrivere i gruppi di destinazione, necessari al piano di controllo di Kubernetes per gestire gli Elastic Load Balancer.

La policy include anche diversi controlli delle condizioni per garantire che le autorizzazioni rientrino nell'ambito dello specifico cluster EKS gestito, utilizzando il tag `eks:eks-cluster-name`.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSLoad BalancingPolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSMCPRead OnlyAccess

È possibile collegare `AmazonEKSMCPReadOnlyAccess` alle entità IAM. Questa policy fornisce l'accesso in sola lettura alle risorse di Amazon EKS e ai AWS servizi correlati, consentendo al server Amazon EKS Model Context Protocol (MCP) di eseguire operazioni di osservabilità e risoluzione dei problemi senza apportare modifiche all'infrastruttura.

Dettagli delle autorizzazioni

Questa politica include le seguenti autorizzazioni che consentono ai responsabili di completare le seguenti attività:

- **`eks`**Consente ai responsabili di descrivere ed elencare cluster EKS, gruppi di nodi, componenti aggiuntivi, voci di accesso, approfondimenti e accedere all'API Kubernetes per operazioni di sola lettura.
- **`iam`**Consente ai responsabili di recuperare informazioni sui ruoli, le policy e i relativi allegati IAM per comprendere le autorizzazioni associate alle risorse EKS.

- **`ec2`**Consente ai principali di descrivere VPCs, alle sottoreti e alle tabelle di routing per comprendere la configurazione di rete dei cluster EKS.
- **`sts`**Consente ai responsabili di recuperare le informazioni sull'identità del chiamante per scopi di autenticazione e autorizzazione.
- **`logs`**Consente ai responsabili di avviare query e recuperare i risultati delle query dai CloudWatch registri per la risoluzione dei problemi e il monitoraggio.
- **`cloudwatch`**Consente ai responsabili di recuperare i dati metrici per il monitoraggio delle prestazioni del cluster e del carico di lavoro.
- **`eks-mcp`**Consente ai responsabili di richiamare operazioni MCP e richiamare strumenti di sola lettura all'interno del server MCP Amazon EKS.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSMCPRead OnlyAccess nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSService Policy

È possibile collegare `AmazonEKSServicePolicy` alle entità IAM. I cluster creati prima del 16 aprile 2020 richiedono all'utente di creare un ruolo IAM e allegarvi questa policy. I cluster creati a partire dal 16 aprile 2020 non richiedono la creazione di un ruolo e non richiedono l'assegnazione di questa policy. Quando crei un cluster utilizzando un principale IAM che dispone dell'`iam:CreateServiceLinkedRole`autorizzazione, il ruolo AWSServiceRoleforAmazonEKS collegato ai servizi viene creato automaticamente per te. Al ruolo collegato ai servizi è EKSService RolePolicy associata la policy gestita: Amazon.

Questa policy consente ad Amazon EKS di creare e gestire le risorse necessarie per gestire i cluster Amazon EKS.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le seguenti attività.

- **`eks`** : aggiorna la versione Kubernetes del cluster dopo aver avviato un aggiornamento. Questa autorizzazione non è utilizzata da Amazon EKS, ma rimane nella policy per la compatibilità con le versioni precedenti.
- **`ec2`** : lavora con le interfacce di rete elastiche e altre risorse e tag di rete. Ciò è richiesto da Amazon EKS per configurare la rete che facilita la comunicazione tra i nodi e il piano di controllo Kubernetes. Leggi le informazioni sui gruppi di sicurezza. Aggiorna i tag sui gruppi di sicurezza.

- **`route53`** : associa un VPC a una zona ospitata. Ciò è richiesto da Amazon EKS per abilitare la rete di endpoint privati per il server API del cluster Kubernetes.
- **`logs`** : registra eventi. Ciò è necessario per consentire ad Amazon EKS di spedire i log del piano di controllo Kubernetes a. CloudWatch
- **`iam`** : crea un ruolo collegato ai servizi. Questa operazione è necessaria affinché Amazon EKS possa creare il ruolo collegato al servizio the section called "Autorizzazioni del ruolo collegato ai servizi per Amazon EKS" per conto dell'utente.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSService Policy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSService RolePolicy

Non è possibile allegare `AmazonEKSServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon EKS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta the section called "Autorizzazioni del ruolo collegato ai servizi per Amazon EKS". Quando crei un cluster utilizzando un principale IAM che dispone dell'`iam:CreateServiceLinkedRole`autorizzazione, il ruolo AWSServiceRoleforAmazonEKS collegato ai servizi viene creato automaticamente per te e questa policy gli viene allegata.

Questa policy consente al ruolo collegato al servizio di chiamare i AWS servizi per tuo conto.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le seguenti attività.

- **`ec2`** : crea e descrive le interfacce di rete elastiche e le istanze di Amazon EC2, il gruppo di sicurezza del cluster e i VPC necessari per la creazione del cluster. Per ulteriori informazioni, consulta the section called "Requisiti relativi al gruppo di sicurezza". Leggi le informazioni sui gruppi di sicurezza. Aggiorna i tag sui gruppi di sicurezza. Leggi informazioni sulle prenotazioni della capacità on demand. Leggi la configurazione del VPC, comprese le tabelle di routing e la rete, ACLs per rilevare i problemi di configurazione nell'ambito di Cluster Insights.
- Modalità automatica **`ec2`**: termina le istanze EC2 create dalla modalità automatica EKS. Per ulteriori informazioni, consulta Modalità automatica EKS.
- **`iam`** : elenca tutti i criteri gestiti allegati a un ruolo IAM. Ciò è necessario affinché Amazon EKS possa pubblicare e convalidare tutte le politiche gestite e le autorizzazioni necessarie per la creazione di un cluster.

- Associa un VPC a una zona ospitata. Ciò è richiesto da Amazon EKS per abilitare la rete di endpoint privati per il server API del cluster Kubernetes.
- Evento di registro: è necessario per consentire ad Amazon EKS di inviare i log del piano di controllo Kubernetes a. CloudWatch
- Metrica metrica: è necessaria per consentire ad Amazon EKS di inviare i log del piano di controllo Kubernetes a. CloudWatch
- **`eks`** : gestisci le voci e le politiche di accesso ai cluster, permettendo un controllo granulare su chi può accedere alle risorse EKS e quali azioni può eseguire. Ciò include l'associazione di policy di accesso standard per operazioni di elaborazione, rete, bilanciamento del carico e archiviazione.
- **`elasticloadbalancing`** : crea, gestisci ed elimina i bilanciatori del carico e i relativi componenti (listener, gruppi di destinazione, certificati) associati ai cluster EKS. Visualizza gli attributi e lo stato di integrità del bilanciatore del carico.
- **`events`**- Crea e gestisci EventBridge regole per il monitoraggio degli eventi EC2 e AWS Health relativi ai cluster EKS, abilitando risposte automatiche ai cambiamenti dell'infrastruttura e agli avvisi sullo stato.
- **`iam`** : gestisci i profili delle istanze EC2 con il prefisso "eks", inclusa la creazione, l'eliminazione e l'associazione dei ruoli, necessarie per la gestione dei nodi EKS. Consente di descrivere qualsiasi profilo di istanza per consentire agli utenti di definire profili di istanza personalizzati da utilizzare per i nodi di lavoro.
- **`pricingshield`**- Accedi alle informazioni AWS sui prezzi e allo stato di protezione Shield, abilitando la gestione dei costi e funzionalità di sicurezza avanzate per le risorse EKS.
- Pulizia delle risorse: elimina in modo sicuro le risorse con tag EKS, inclusi volumi, snapshot, modelli di avvio e interfacce di rete durante le operazioni di pulizia dei cluster.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSService RolePolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSVPCResource Controller

È possibile allegare la policy `AmazonEKSVPCResourceController` alle identità IAM. Se utilizzi gruppi di sicurezza per pod, dovrai allegare questa policy al ruolo IAM del cluster Amazon EKS perché possa eseguire operazioni per tuo conto.

Questa policy concede le autorizzazioni del ruolo cluster per gestire le interfacce di rete elastiche e gli indirizzi IP per i nodi.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`ec2`** : gestisci le interfacce di rete elastiche e gli indirizzi IP per supportare i gruppi di sicurezza dei pod e i nodi Windows.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSVPCResource Controller nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSWorker NodePolicy

É possibile allegare la `AmazonEKSWorkerNodePolicy` alle entità IAM. È necessario allegare questa policy a un Ruolo IAM del nodo che va specificato quando si creano nodi Amazon EC2 che consentono ad Amazon EKS di eseguire operazioni per conto dell'utente. Se si crea un gruppo di nodi utilizzando `eksctl`, si crea il ruolo IAM del nodo e si allega automaticamente questa policy al ruolo.

Questa policy concede ai nodi Amazon EKS Amazon EC2 le autorizzazioni per connettersi ai cluster Amazon EKS.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`ec2`** : leggi le informazioni sul volume dell'istanza e sulla rete. Ciò è necessario affinché i nodi Kubernetes possano descrivere le informazioni sulle risorse Amazon EC2 necessarie per l'unione del nodo al cluster Amazon EKS.
- **`eks`** : descrivi in modo facoltativo il cluster come parte del bootstrap del nodo.
- **`eks-auth:AssumeRoleForPodIdentity`** : consenti il recupero delle credenziali per i carichi di lavoro EKS sul nodo. Ciò è necessario per il funzionamento di EKS Pod Identity.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSWorker NodePolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSWorker NodeMinimalPolicy

Puoi collegare WorkerNodeMinimalPolicy AmazonEks alle tue entità IAM. È necessario allegare questa policy a un ruolo IAM del nodo che deve essere specificato quando si creano nodi Amazon EC2 che consentono ad Amazon EKS di eseguire operazioni per conto dell'utente.

Questa policy concede ai nodi Amazon EKS Amazon EC2 le autorizzazioni per connettersi ai cluster Amazon EKS. Questa politica ha meno autorizzazioni rispetto ad Amazon EKSWorkerNodePolicy.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- `eks-auth:AssumeRoleForPodIdentity`: consenti il recupero delle credenziali per i carichi di lavoro EKS sul nodo. Ciò è necessario per il funzionamento di EKS Pod Identity.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSWorker NodePolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: AWSService RoleForAmazon EKSNodegroup

Non è possibile allegare `AWSServiceRoleForAmazonEKSNodegroup` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon EKS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta the section called "Autorizzazioni del ruolo collegato ai servizi per Amazon EKS".

Questa policy concede l'autorizzazione ai ruoli `AWSServiceRoleForAmazonEKSNodegroup` che consentono di creare e gestire gruppi di nodi Amazon EC2 nell'account.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`ec2`** : lavora con gruppi di sicurezza, tag, prenotazioni della capacità e modelli di avvio. Ciò è necessario per i gruppi di nodi gestiti da Amazon EKS per abilitare la configurazione dell'accesso remoto e descrivere le prenotazioni della capacità che possono essere utilizzate nei gruppi di nodi gestiti. Inoltre, i gruppi di nodi gestiti da Amazon EKS creano un modello di avvio per conto dell'utente. Questo consente di configurare il gruppo Amazon EC2 Auto Scaling che supporta ogni gruppo di nodi gestito.

- **`iam`** : crea un ruolo collegato ai servizi e passa un ruolo. Ciò è richiesto dai gruppi di nodi gestiti da Amazon EKS per gestire i profili di istanza per il ruolo passato durante la creazione di un gruppo di nodi gestito. Questo profilo dell'istanza viene utilizzato dalle istanze Amazon EC2 avviate come parte di un gruppo di nodi gestito. Amazon EKS deve creare ruoli collegati al servizio per altri servizi come, ad esempio i gruppi di Amazon EC2 Auto Scaling. Queste autorizzazioni vengono utilizzate nella creazione di un gruppo di nodi gestito.
- **`autoscaling`** : lavora con gruppi Auto Scaling di sicurezza. Questo è richiesto dai gruppi di nodi gestiti da Amazon EKS per gestire il gruppo di Amazon EC2 Auto Scaling che supporta ciascun gruppo di nodi gestito. Viene anche utilizzato per supportare funzionalità come l'eliminazione dei Pod quando i nodi vengono terminati o il riciclo durante gli aggiornamenti dei gruppi di nodi e la gestione dei pool caldi configurati su gruppi di nodi gestiti.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta la Managed Policy Reference Guide AWSServiceRoleForAmazonEKSNodegroup. AWS

AWS politica gestita: Amazon EKSDashboard ServiceRolePolicy

Non è possibile allegare `AmazonEKSDashboardServiceRolePolicy` alle entità IAM. Questa policy è allegata a un ruolo collegato ai servizi che consente ad Amazon EKS di eseguire operazioni per conto dell'utente. Per ulteriori informazioni, consulta the section called “Autorizzazioni del ruolo collegato ai servizi per Amazon EKS”.

Questa policy concede l'autorizzazione ai ruoli `AWSServiceRoleForAmazonEKSDashboard` che consentono di creare e gestire gruppi di nodi Amazon EC2 nell'account.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni che consentono ad Amazon EKS di completare le attività seguenti:

- **`organizations`**— Visualizza informazioni sulla struttura e sugli account della tua AWS Organizations. Ciò include le autorizzazioni per elencare gli account dell'organizzazione, visualizzare le unità organizzative e le root, elencare gli amministratori delegati, visualizzare i servizi che hanno accesso all'organizzazione e recuperare informazioni dettagliate sull'organizzazione e sugli account.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSDashboard ServiceRolePolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EBSCSIDriver Policy

La `AmazonEBSCSIDriverPolicy` policy consente al driver Amazon EBS Container Storage Interface (CSI) di creare, modificare, copiare, allegare, scollegare ed eliminare volumi per tuo conto. Ciò include la modifica dei tag sui volumi esistenti e l'abilitazione del ripristino rapido degli snapshot (FSR) sui volumi EBS. Inoltre, concede al driver CSI EBS le autorizzazioni per creare, bloccare, ripristinare ed eliminare istantanee e per elencare istanze, volumi e istantanee.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EBSCSIDriver ServiceRolePolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EFSCSIDriver Policy

La`AmazonEFSCSIDriverPolicy`la policy consente all'Amazon EFS Container Storage Interface (CSI) di creare ed eliminare punti di accesso per tuo conto. Concede inoltre al driver CSI di Amazon EFS le autorizzazioni per elencare i punti di accesso, i file system, gli obiettivi di montaggio e le zone di disponibilità di Amazon EC2.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EFSCSIDriver ServiceRolePolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSLocal OutpostClusterPolicy

È possibile collegare questa policy alle entità IAM. Prima di creare un cluster, è necessario allegare questa policy a un ruolo cluster. I cluster Kubernetes gestiti da Amazon EKS effettuano chiamate verso altri AWS servizi per tuo conto. Ciò serve a gestire le risorse utilizzate con il servizio.

La `AmazonEKSLocalOutpostClusterPolicy` include le autorizzazioni seguenti:

- Azioni di lettura **`ec2`**: consente alle istanze del piano di controllo di descrivere le proprietà della zona di disponibilità, della tabella di routing, dell'istanza e dell'interfaccia di rete. Autorizzazioni necessarie affinché le istanze Amazon EC2 si uniscano correttamente al cluster come istanze del piano di controllo.
- **`ssm`** : consente la connessione di Systems Manager di Amazon EC2 all'istanza del piano di controllo, utilizzata da Amazon EKS per comunicare e gestire il cluster locale nel tuo account.
- **`logs`**— Consente alle istanze di inviare i log ad Amazon. CloudWatch
- **`secretsmanager`**— Consente alle istanze di ottenere ed eliminare i dati di bootstrap per le istanze del piano di controllo in modo sicuro da Secrets Manager. AWS

- **`ecr`** : consente a pod e container in esecuzione sulle istanze del piano di controllo di estrarre le immagini di container memorizzate in Amazon Elastic Container Registry.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSLocal OutpostClusterPolicy nella AWS Managed Policy Reference Guide.

AWS politica gestita: Amazon EKSLocal OutpostServiceRolePolicy

Non è possibile allegare questa policy alle entità IAM. Quando crei un cluster utilizzando un principale IAM che dispone dell'`iam:CreateServiceLinkedRole`autorizzazione, Amazon EKS crea automaticamente il ruolo collegato al servizio AWSServiceRoleforAmazonEKSLocalOutpost per te e gli allega questa policy. Questa policy consente al ruolo collegato al servizio di chiamare i AWS servizi per tuo conto per i cluster locali.

La `AmazonEKSLocalOutpostServiceRolePolicy` include le autorizzazioni seguenti:

- **`ec2`** : consente ad Amazon EKS di utilizzare le risorse di sicurezza, rete e altro tipo per avviare e gestire correttamente le istanze del piano di controllo nel tuo account.
- **`ssm`**, **`ssmmessages`** : consente la connessione di Amazon EC2 Systems Manager alle istanze del piano di controllo, utilizzata da Amazon EKS per comunicare e gestire il cluster locale nel tuo account.
- **`iam`** : consente ad Amazon EKS di gestire il profilo dell'istanza associato alle istanze del piano di controllo.
- **`secretsmanager`**- Consente ad Amazon EKS di inserire i dati di bootstrap per le istanze del piano di controllo in AWS Secrets Manager in modo che possano essere referenziati in modo sicuro durante il bootstrap dell'istanza.
- **`outposts`** : consente ad Amazon EKS di ottenere le informazioni sull'Outpost dal tuo account per avviare correttamente un cluster locale in un Outpost.

Per visualizzare la versione più recente del documento sulla policy JSON, consulta Amazon EKSLocal OutpostServiceRolePolicy nella AWS Managed Policy Reference Guide.

Amazon EKS si aggiorna alle politiche AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per Amazon EKS da quando questo servizio ha iniziato a tracciare queste modifiche.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, puoi iscriverti al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/security/iam-reference/security-iam-awsmanpol.adoc.atom
```

Modifica	Descrizione	Data
Aggiunte autorizzazioni a the section called "AWS politica gestita: AWSService RoleForAmazon EKSNodegroup".	Aggiunti `autoscaling:PutWarmPool` `autoscaling:DeleteWarmPool` e `autoscaling:DescribeWarmPool` autorizzazioni a. `AWSServiceRoleForAmazonEKSNodegroup` Ciò consente ai gruppi di nodi gestiti di Amazon EKS di gestire le risorse del pool caldo ASG sottostante per tutto il ciclo di vita del gruppo di nodi.	17 febbraio 2026
Autorizzazione aggiunta a the section called "AWS politica gestita: Amazon EKSService RolePolicy".	È stato rimosso il requisito del prefisso «eks» nel nome del profilo dell'istanza di destinazione per l'autorizzazione in. `iam:GetInstanceProfile` `AmazonEKSServiceRolePolicy` Ciò consente ad Amazon EKS Auto Mode di convalidare e utilizzare profili di istanza personalizzati NodeClasses senza richiedere il prefisso di denominazione «eks».	2 febbraio 2026

Modifica	Descrizione	Data
Sono state aggiunte le autorizzazioni ad Amazon EBSCSIDriver Policy.	È stata aggiunta `ec2:LockSnapshot` l'autorizzazione per consentire al driver CSI EBS di bloccare direttamente le istantanee EBS.	15 gennaio 2026
Introdotto the section called "AWS politica gestita: Amazon EKSMCPRead OnlyAccess".	Amazon EKS ha introdotto una nuova policy gestita `AmazonEKSMCPReadOnlyAccess` per abilitare strumenti di sola lettura nel server MCP Amazon EKS per l'osservabilità e la risoluzione dei problemi.	21 novembre 2025
Sono state aggiunte le autorizzazioni ad Amazon EBSCSIDriver Policy.	È stata aggiunta `ec2:CopyVolumes` l'autorizzazione per consentire al driver CSI EBS di copiare direttamente i volumi EBS.	17 novembre 2025
Autorizzazione aggiunta a the section called "AWS politica gestita: Amazon EKSService RolePolicy".	Aggiunte `ec2:DescribeRouteTables` e `ec2:DescribeNetworkAcls` autorizzazioni a. `AmazonEKSServiceRolePolicy` Ciò consente ad Amazon EKS di rilevare problemi di configurazione con le tabelle di routing VPC e la rete ACLs per nodi ibridi come parte delle informazioni sui cluster.	22 ottobre 2025

Modifica	Descrizione	Data
Aggiunta l'autorizzazione a AWSServiceRoleForAmazonEKSConnector	È stata aggiunta `ssmmessages:OpenDataChannel` l'autorizzazione a `AmazonEKSConnectorServiceRolePolicy`	15 ottobre 2025
Aggiunta l'autorizzazione a the section called "AWS politica gestita: Amazon EKSServiceRolePolicy"	Questo ruolo può allegare una nuova policy di accesso `AmazonEKSEventPolicy` . Autorizzazioni limitate per `ec2:DeleteLaunchTemplate` e `ec2:TerminateInstances` .	26 agosto 2025
Aggiunta l'autorizzazione a the section called "AWS politica gestita: Amazon EKSLocalOutpostServiceRolePolicy"	Aggiunta l'autorizzazione `ssmmessages:OpenDataChannel` a `AmazonEKSLocalOutpostServiceRolePolicy` .	26 giugno 2025

Modifica	Descrizione	Data
Aggiunta l'autorizzazione a the section called "AWS politica gestita: Amazon EKSCompute Policy".	Autorizzazioni aggiornate per le risorse `ec2:RunInstances` e `ec2:CreateFleet` azioni che includono le prenotazioni della capacità `arn:aws: ec2:*:*:capacity-reservation/*` . Ciò consente alla modalità automatica di Amazon EKS di avviare istanze utilizzando le prenotazioni della capacità on-demand di EC2 nel tuo account. `iam:CreateServiceLinkedRole` aggiunto per consentire alla modalità automatica di Amazon EKS di creare il ruolo `AWSServiceRoleForEC2Spot` collegato al servizio Spot EC2 per tuo conto.	20 giugno 2025
È stata aggiunta l'autorizzazione ad Amazon EKSService RolePolicy.	È stata aggiunta l'autorizzazione `ec2:DescribeCapacityReservations` per consentire alla modalità automatica di Amazon EKS di avviare istanze utilizzando le prenotazioni della capacità on-demand EC2 nell'account.	20 giugno 2025

Modifica	Descrizione	Data
Introdotto the section called "AWS politica gestita: Amazon EKSDashboard ConsoleReadOnly".	È stata introdotta una nuova policy `AmazonEKSDashboardConsoleReadOnly` .	19 giugno 2025
Introdotto the section called "AWS politica gestita: Amazon EKSDashboard ServiceRolePolicy".	È stata introdotta una nuova policy `AmazonEKSDashboardServiceRolePolicy` .	21 maggio 2025
Sono state aggiunte le autorizzazioni ad Amazon EKSCluster Policy.	È stata aggiunta l'autorizzazione `ec2:DeleteNetworkInterfaces` per consentire ad Amazon EKS di eliminare le interfacce di rete elastiche che sono lasciate indietro se il CNI di VPC si chiude inaspettatamente.	16 aprile 2025
È stata aggiunta l'autorizzazione ad Amazon EKSService RolePolicy.	Sono state aggiunte `ec2:RevokeSecurityGroupEgress` e `ec2:AuthorizeSecurityGroupEgress` autorizzazioni per consentire ai AI/ML clienti EKS di aggiungere regole Security Group Egress all'EKS Cluster SG predefinito compatibili con EFA, come parte della versione EKS 1.33.	14 aprile 2025

Modifica	Descrizione	Data
Autorizzazioni aggiunte ad Amazon EKSService RolePolicy.	Aggiunta l'autorizzazione per terminare le istanze EC2 create dalla modalità automatica di EKS.	28 febbraio 2025
Sono state aggiunte le autorizzazioni ad Amazon EBSCSIDriver Policy.	È stata aggiunta una nuova dichiarazione che autorizza il driver CSI EBS a ripristin are tutti gli snapshot. Ciò era precedentemente consentit o dalla policy esistente, ma è necessaria una nuova dichiarazione esplicita per via di una modifica nella gestione di IAM per `CreateVolume` . È stata aggiunta la possibili tà per il driver CSI EBS di modificare i tag sui volumi esistenti. Il driver CSI EBS può modificare i tag dei volumi esistenti tramite un parametro in Kubernetes. VolumeAtt ributesClasses È stata aggiunta la possibili tà per il driver CSI EBS di abilitare il ripristino rapido degli snapshot (FSR) sui volumi EBS. Il driver CSI EBS può abilitare FSR su nuovi volumi tramite parametri nelle classi di archiviazione Kubernetes.	13 gennaio 2025

Modifica	Descrizione	Data
Aggiunte autorizzazioni a the section called "AWS politica gestita: Amazon EKSLoad BalancingPolicy".	Aggiornato `AmazonEKS LoadBalancingPolic y` per consentire l'elenco e la descrizione delle risorse di rete e degli indirizzi IP.	26 dicembre 2024
Aggiunte autorizzazioni a the section called "AWS politica gestita: AWSService RoleForAmazon EKSNodegr oup".	Aggiornato `AWSServic eRoleForAmazonEKSN odegroup` per compatibilità con le regioni della Cina.	22 novembre 2024
Aggiunte autorizzazioni a the section called "AWS politica gestita: Amazon EKSLocal OutpostClusterPolicy"	È stata aggiunta l'`ec2:Descr ibeAvailabilityZon es` autorizzazione per `AmazonEKSLocalOutp ostClusterPolicy` consentire al AWS Cloud Controller Manager sul piano di controllo del cluster di identificare la zona di disponibi lità in cui si trova ogni nodo.	21 novembre 2024
Aggiunte autorizzazioni a the section called "AWS politica gestita: AWSService RoleForAmazon EKSNodegr oup".	Policy `AWSServic eRoleForAmazonEKSN odegroup` aggiornata per consentire a `ec2:Reboo tInstances` le istanze create da gruppi di nodi gestiti di Amazon EKS. Limitazione delle autorizza zioni `ec2:CreateTags` per le risorse Amazon EC2.	20 novembre 2024

Modifica	Descrizione	Data
Aggiunte autorizzazioni a the section called "AWS politica gestita: Amazon EKSService RolePolicy".	Politica AWS gestita aggiornata da EKS `AmazonEKSServiceRolePolicy`. Aggiunte le autorizzazioni per le politiche di accesso EKS, la gestione del bilanciatore del carico e la pulizia automatica delle risorse del cluster.	16 novembre 2024
Introdotto the section called "AWS politica gestita: Amazon EKSCompute Policy".	Politica AWS gestita aggiornata da EKS `AmazonEKSComputePolicy`. Autorizzazioni delle risorse aggiornate per l'azione `iam:AddRoleToInstanceProfile`.	7 novembre 2024
Introdotto the section called "AWS politica gestita: Amazon EKSCompute Policy".	AWS ha introdotto il `AmazonEKSComputePolicy`.	1 novembre 2024
Aggiunte autorizzazioni a `AmazonEKSClusterPolicy`	Aggiunta l'autorizzazione `ec2:DescribeInstanceTopology` per consentire ad Amazon EKS di allegare informazioni di topologia al nodo come etichette.	1 novembre 2024
Introdotto the section called "AWS politica gestita: Amazon EKSBlock StoragePolicy".	AWS ha introdotto il `AmazonEKSBlockStoragePolicy`.	30 ottobre 2024
Introdotto the section called "AWS politica gestita: Amazon EKSLoad BalancingPolicy".	AWS ha introdotto il `AmazonEKSLoadBalancingPolicy`.	30 ottobre 2024

Modifica	Descrizione	Data
Autorizzazioni aggiunte ad Amazon EKSService RolePolicy.	Sono state aggiunte `cloudwatch:PutMetricData` autorizzazioni per consentire ad Amazon EKS di pubblicare metriche su Amazon. CloudWatch	29 ottobre 2024
Introdotto the section called "AWS politica gestita: Amazon EKSNetworking Policy".	AWS ha introdotto il. `AmazonEKSNetworkingPolicy`	28 ottobre 2024
Aggiunte autorizzazioni a `AmazonEKSServicePolicy` e `AmazonEKSServiceRolePolicy`	Aggiunte autorizzazione `ec2:GetSecurityGroupsForVpc` e ai tag associati per consentire a EKS di leggere le informazioni sui gruppi di sicurezza e aggiornare i tag correlati.	10 ottobre 2024
Presentato Amazon EKSWorker NodeMinimalPolicy.	AWS ha introdotto il`AmazonEKSWorkerNodeMinimalPolicy` .	3 ottobre 2024
Aggiunte autorizzazioni a AWSServiceRoleForAmazonEKSNodegroup.	Aggiunte le autorizzazioni `autoscaling:ResumeProcesses` e `autoscaling:SuspendProcesses` per consentire ad Amazon EKS di sospendere e riprendere `AZRebalance` nei gruppi Auto Scaling gestiti da Amazon EKS.	21 agosto 2024

Modifica	Descrizione	Data
Aggiunte autorizzazioni a AWSServiceRoleForAmazonEKSNodegroup.	Aggiunta l'autorizzazione `ec2:DescribeCapacityReservations` per consentire ad Amazon EKS di descrivere la prenotazione della capacità nell'account dell'utente. È stata aggiunta l'autorizzazione `autoscaling:PutScheduledUpdateGroupAction` per abilitare l'impostazione del ridimensionamento pianificato sui gruppi di nodi `CAPACITY_BLOCK` .	27 giugno 2024
AmazonEKS_CNI_Policy: aggiornamento a una policy esistente	Amazon EKS ha aggiunto nuove autorizzazioni `ec2:DescribeSubnets` per consentire al plugin CNI di Amazon VPC per Kubernetes di visualizzare la quantità di indirizzi IP gratuiti nelle sottoreti Amazon VPC. Il CNI di VPC può utilizzare gli indirizzi IP gratuiti in ciascuna sottorete per scegliere le sottoreti con gli indirizzi IP più liberi da utilizzare durante la creazione di un'interfaccia di rete elastica.	4 marzo 2024

Modifica	Descrizione	Data
Amazon EKSWorker NodePolicy: aggiornamento a una politica esistente	Amazon EKS ha aggiunto nuove autorizzazioni per consentire le associazioni EKS Pod Identity. Amazon EKS Pod Identity Agent utilizza il ruolo del nodo.	26 novembre 2023
Introduzione EFSCSIDriverdella politica Amazon.	AWS ha introdotto il`AmazonEFSCSIDriver Policy` .	26 luglio 2023
Sono state aggiunte le autorizzazioni ad Amazon EKSCluster Policy.	È stata aggiunta l'autorizzazione `ec2:DescribeAvailabilityZones` per consentire ad Amazon EKS di ottenere i dettagli AZ durante l'individuazione automatica delle sottoreti nella creazione di sistemi di bilanciamento del carico.	7 febbraio 2023
Condizioni politiche aggiornate in Amazon EBSCSIDriver Policy.	Sono state rimosse condizioni della policy non valide con caratteri jolly nel campo chiave `StringLike` . È stata aggiunta, inoltre, una nuova condizione `ec2:ResourceTag/kubernetes.io/created-for/pvc/name: "*"` a `ec2:DeleteVolume` che consente al driver CSI EBS di eliminare volumi creati dal plugin nella struttura.	17 novembre 2022

Modifica	Descrizione	Data
Autorizzazioni aggiunte ad Amazon EKSLocal OutpostServiceRolePolicy.	Aggiunti `ec2:DescribeVPCAttribute` , `ec2:GetConsoleOutput` e `ec2:DescribeSecret` per consentire una migliore convalida dei prerequisiti e il controllo gestito del ciclo di vita. Sono stati inoltre aggiunti `ec2:DescribePlacementGroups` , `"arn:aws:ec2:*:*:placement-group/*"` e `ec2:RunInstances` per supportare il controllo del posizionamento delle istanze Amazon EC2 del piano di controllo su Outposts.	24 ottobre 2022
Aggiorna le autorizzazioni di Amazon Elastic Container Registry in Amazon EKSLocal OutpostClusterPolicy.	L'azione `ecr:GetDownloadUrlForLayer` è stata spostata da tutte le sezioni delle risorse a una sezione con ambito. Aggiunta la risorsa `arn:aws:ecr:*:*:repository/eks/` . Risorsa `arn:aws:ecr:` rimossa. Questa risorsa è coperta dalla risorsa `arn:aws: ecr:*:*:repository/eks/*` aggiunta.	20 ottobre 2022

Modifica	Descrizione	Data
Autorizzazioni aggiunte ad Amazon EKSLocal OutpostClusterPolicy.	È stato aggiunto il repository `arn:aws:ecr:*:*:repository/kubelet-config-updater` Amazon Elastic Container Registry in modo che le istanze del piano di controllo del cluster possano aggiornare alcuni argomenti `kubelet`.	31 agosto 2022
Presentato Amazon EKSLocal OutpostClusterPolicy.	AWS ha introdotto il `AmazonEKSLocalOutpostClusterPolicy` .	24 agosto 2022
Presentato Amazon EKSLocal OutpostServiceRolePolicy.	AWS ha introdotto il `AmazonEKSLocalOutpostServiceRolePolicy` .	23 agosto 2022
Introduzione EBSCSIDriverdella politica Amazon.	AWS ha introdotto il `AmazonEBSCSIDriverPolicy` .	4 aprile 2022
Autorizzazioni aggiunte ad Amazon EKSWorkerNodePolicy.	`ec2:DescribeInstanceTypes` Aggiunto per abilitare Amazon EKS ottimizzato in grado di AMIs rilevare automaticamente le proprietà a livello di istanza.	21 marzo 2022

Modifica	Descrizione	Data
Aggiunte autorizzazioni a AWSServiceRoleForAmazonEKSNodegroup.	Aggiunta l'autorizzazione `autoscaling:EnableMetricsCollection` per consentire ad Amazon EKS l'abilitazione della raccolta di parametri.	13 dicembre 2021
Sono state aggiunte le autorizzazioni ad Amazon EKSCluster Policy.	Aggiunta delle autorizzazioni `ec2:DescribeAccountAttributes`, `ec2:DescribeAddresses`, e `ec2:DescribeInternetGateways` per consentire ad Amazon EKS di creare un ruolo collegato al servizio per un Network Load Balancer (load balancer di Rete).	17 giugno 2021
Amazon EKS ha iniziato a monitorare le modifiche	Amazon EKS ha iniziato a tracciare le modifiche per le sue politiche AWS gestite.	17 giugno 2021

Risoluzione dei problemi di IAM

Questo argomento illustra alcuni errori comuni che si potrebbero verificare durante l'utilizzo di Amazon EKS con IAM, e il modo in cui gestirli.

AccessDeniedException

Se ricevi un messaggio `AccessDeniedException` quando chiami un'operazione AWS API, le credenziali principali IAM che stai utilizzando non dispongono delle autorizzazioni necessarie per effettuare quella chiamata.

```
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
```

Nel messaggio di esempio precedente, l'utente non dispone delle autorizzazioni per chiamare l'operazione API `DescribeCluster` di Amazon EKS. Per fornire le autorizzazioni da amministratore Amazon EKS a un principale IAM, consultare the section called "Policy basate sull'identità".

Per informazioni generali su IAM, consultare Controllo dell'accesso tramite policy nella Guida per l'utente di IMA.

Non riesci a vedere i nodi nella scheda Elaborazione o altro nella scheda Risorse e ricevi un errore nella Console di gestione AWS

È possibile che venga visualizzato un messaggio di errore della console che recita `Your current user or role does not have access to Kubernetes objects on this EKS cluster`. Assicurati che l'utente principale IAM Console di gestione AWS con cui stai utilizzando disponga delle autorizzazioni necessarie. Per ulteriori informazioni, consulta the section called "Autorizzazioni richieste".

La **`ConfigMap`** per aws-auth non concede l'accesso al cluster

L'Autenticatore AWS IAM non consente un percorso nell'ARN del ruolo utilizzato in `ConfigMap`. Pertanto, rimuovi il percorso prima di specificare `rolearn`. Ad esempio, modifica `arn:aws: iam::111122223333:role/team/developers/eks-admin` in `arn:aws: iam::111122223333:role/eks-admin` .

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un errore che indica che non disponi delle autorizzazioni a eseguire l'operazione `iam:PassRole`, è necessario aggiornare le policy per poter passare un ruolo ad Amazon EKS.

Alcuni AWS servizi consentono di trasferire un ruolo esistente a quel servizio anziché creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in Amazon EKS. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: {arn-aws}iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne al mio AWS account di accedere alle mie risorse Amazon EKS

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per consentire alle persone di accedere alle tue risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon EKS supporta queste caratteristiche, consultare the section called "Amazon EKS e IAM".
- Per scoprire come fornire l'accesso alle tue risorse su più AWS account di tua proprietà, consulta Fornire l'accesso a un utente IAM in un altro AWS account di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse ad AWS account di terze parti, consulta Fornire l'accesso agli AWS account di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente di IAM.

I container dei pod riceveranno il seguente errore: **`An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`**

I tuoi contenitori ricevono questo errore se l'applicazione effettua esplicitamente richieste all'endpoint globale AWS STS (`https://sts.amazonaws`) e il tuo account di servizio Kubernetes è configurato per utilizzare un endpoint regionale. Puoi risolvere il problema con una delle opzioni seguenti:

- Aggiorna il codice dell'applicazione per rimuovere le chiamate esplicite all'endpoint globale STS. AWS
- Aggiorna il codice dell'applicazione per effettuare chiamate esplicite agli endpoint regionali, ad esempio `https://sts.us-west-2.amazonaws.com`. L'applicazione dovrebbe avere la ridondanza integrata per scegliere una AWS regione diversa in caso di guasto del servizio nella regione. AWS Per ulteriori informazioni, consulta Managing AWS STS in an AWS Region nella IAM User Guide.
- Configura gli account del servizio per l'utilizzo dell'endpoint globale. I cluster utilizzano l'endpoint regionale per impostazione predefinita. Per ulteriori informazioni, consulta the section called "Endpoint STS".

Ruolo IAM del cluster Amazon EKS

richiesto un ruolo IAM del cluster Amazon EKS per ogni cluster. I cluster Kubernetes gestiti da Amazon EKS utilizzano questo ruolo per gestire i nodi e il provider di servizi cloud utilizza questo ruolo per creare bilanciatori del carico con Elastic Load Balancing per i servizi.

Prima di poter creare cluster Amazon EKS, devi creare un ruolo IAM con una delle seguenti policy IAM:

- EKSClusterPolitica di Amazon
- Una policy IAM personalizzata. Le autorizzazioni minime che seguono consentono al cluster di Kubernetes di gestire i nodi, ma non consentono al provider di servizi cloud di creare bilanciatori del carico con bilanciamento del carico elastico. La policy IAM personalizzata deve disporre almeno delle seguenti autorizzazioni:

```
{
  "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**
>
> Prima del 3 ottobre 2023, era richiesta la EKSClusterpolitica di Amazon sul ruolo IAM per ogni cluster.
> Prima del 16 aprile 2020, erano obbligatorie Amazon EKSService EKSCluster Policy e Amazon Policy e il nome suggerito per il ruolo era`eksServiceRole`. Con il ruolo `AWSServiceRoleForAmazonEKS` collegato ai servizi, la politica di Amazon EKSService Policy non è più richiesta per i cluster creati a partire dal 16 aprile 2020.

Verifica della presenza di un ruolo del cluster esistente

Per verificare se l'account dispone già di un ruolo del cluster Amazon EKS, puoi utilizzare la procedura indicata di seguito.

1. Apri la console IAM all'indirizzo. https://console.aws.amazon.com/iam/
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Cerca l'elenco dei ruoli per `eksClusterRole`. Se un ruolo che include `eksClusterRole` non esiste, consultare the section called "Creazione del ruolo del cluster Amazon EKS" per crearlo. Se un ruolo che include `eksClusterRole` esiste, seleziona il ruolo per visualizzare le policy allegate.
4. Selezionare Autorizzazioni.

5. Assicurati che la EKSClusterpolitica gestita da Amazon Policy sia associata al ruolo. Se la policy è allegata, il ruolo del cluster Amazon EKS è configurato correttamente.
6. Scegliere Relazioni di attendibilità, quindi scegliere Modifica policy di attendibilità.
7. Verifica che la relazione di trust includa la policy seguente. Se la relazione di attendibilità corrisponde alla policy seguente, scegli Annulla. Se la relazione di attendibilità non corrisponde, copia la policy nella finestra Modifica policy di attendibilità e scegli Aggiorna policy.

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

Creazione del ruolo del cluster Amazon EKS

È possibile utilizzare Console di gestione AWS o la AWS CLI per creare il ruolo del cluster.

Console di gestione AWS

a. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
b. Scegli Ruoli, quindi Crea ruolo.
c. In Tipo di entità affidabile, seleziona AWS servizio.
d. Dall'elenco a discesa Casi d'uso per altri AWS servizi, scegli EKS.
e. Scegli EKS - Cluster per il caso d'uso, quindi scegli Successivo.
f. Nella scheda Aggiungi autorizzazioni, scegli Successivo.
g. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `eksClusterRole`.
h. Per Descrizione, inserisci un testo descrittivo come `Amazon EKS - Cluster role`.
i. Scegli Crea ruolo.

AWS CLI

a. Copiare i seguenti contenuti in un file denominato *`cluster-trust-policy.json`*.

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

b. Crea il ruolo. Puoi sostituire *eksClusterRole* con un nome a tua scelta.

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

c. Allega la policy IAM richiesta al ruolo.

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

Ruolo IAM del nodo Amazon EKS

Il `kubelet` daemon del nodo Amazon EKS effettua chiamate per tuo AWS APIs conto. I nodi ricevono le autorizzazioni per queste chiamate API attraverso un profilo dell'istanza IAM e le policy associate. Prima di avviare i nodi e registrarli in un cluster, devi creare un ruolo IAM che i nodi possano utilizzare all'avvio. Questo requisito si applica ai nodi lanciati con l'AMI ottimizzata Amazon EKS fornita da Amazon o con qualsiasi altro nodo AMIs che intendi utilizzare. Inoltre, questo requisito si applica sia ai gruppi di nodi gestiti sia ai nodi autogestiti.

> **Note**
>
> Non è possibile utilizzare lo stesso ruolo utilizzato per creare i cluster.

Prima di creare i nodi, è necessario creare un ruolo IAM con le seguenti autorizzazioni:

- Autorizzazioni per `kubelet` descrivere EC2 le risorse Amazon nel VPC, come quelle fornite dalla policy di Amazon EKSWorker NodePolicy. Questa policy fornisce anche le autorizzazioni per il Pod Identity Agent di Amazon EKS.
- Autorizzazioni `kubelet` per l'utilizzo di immagini di container da Amazon Elastic Container Registry (Amazon ECR), come previsto dalla politica di Amazon. EC2 ContainerRegistryPullOnly Le autorizzazioni per utilizzare le immagini dei container da Amazon Elastic Container Registry (Amazon ECR) sono necessarie perché i componenti aggiuntivi integrati per le reti eseguono pod che utilizzano immagini di container provenienti da Amazon ECR.
- (Facoltativo) Autorizzazioni per consentire al Pod Identity Agent di Amazon EKS di utilizzare l'azione `eks-auth:AssumeRoleForPodIdentity` per recuperare le credenziali per i pod. Se non utilizzi Amazon EKSWorker NodePolicy, devi fornire questa autorizzazione oltre alle EC2 autorizzazioni per utilizzare EKS Pod Identity.
- (Facoltativo) Se non si usa Pod Identity di EKS o IRSA per fornire le autorizzazioni ai pod VPC CNI, è necessario fornire le autorizzazioni per il VPC CNI sul ruolo dell'istanza. Puoi utilizzare la politica `AmazonEKS_CNI_Policy`gestita (se hai creato il cluster con la `IPv4` famiglia) o una IPv6 politica creata da te (se hai creato il cluster con la `IPv6` famiglia). Invece di allegare la policy a questo ruolo, tuttavia, consigliamo di allegarla a un ruolo separato utilizzato specificamente per il componente aggiuntivo CNI di Amazon VPC. Per ulteriori informazioni sulla creazione di un ruolo separato per il componente aggiuntivo CNI di Amazon VPC, consulta the section called "Configurare per IRSA".

> **Note**
>
> I gruppi di EC2 nodi Amazon devono avere un ruolo IAM diverso rispetto al profilo Fargate. Per ulteriori informazioni, consulta the section called "Ruolo IAM per l'esecuzione del pod".

Verifica della presenza di un ruolo di nodo esistente

Per controllare se l'account dispone già di un ruolo di nodo Amazon EKS, utilizza la procedura indicata di seguito.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.

3. Ricerca `eksNodeRole`, `AmazonEKSNodeRole` o `NodeInstanceRole` nell'elenco di ruoli. Se un ruolo con uno di questi nomi non esiste, consulta the section called "Creazione del ruolo IAM del nodo Amazon EKS" per creare il ruolo. Se esiste un ruolo che contiene `eksNodeRole`, `AmazonEKSNodeRole` o `NodeInstanceRole`, seleziona il ruolo per visualizzare le policy allegate.
4. Seleziona Autorizzazioni.
5. Assicurati che le policy EC2 ContainerRegistryPullOnly gestite da Amazon EKSWorker NodePolicy e Amazon siano associate al ruolo o che sia allegata una policy personalizzata con le autorizzazioni minime.

> **Note**
>
> Se la policy AmazonEKS_CNI_Policy è allegata al ruolo, è consigliabile invece rimuoverla e allegarla a un ruolo IAM mappato all'account di servizio `aws-node` Kubernetes. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".

6. Scegli Trust relationships (Relazioni di attendibilità), quindi scegli Edit trust policy (Modifica policy di attendibilità).
7. Verifica che la relazione di trust includa la policy seguente. Se la relazione di attendibilità corrisponde alla policy seguente, scegli Annulla. Se la relazione di attendibilità non corrisponde, copia la policy nella finestra Modifica policy di attendibilità e scegli Aggiorna policy.

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

Creazione del ruolo IAM del nodo Amazon EKS

Puoi creare il ruolo IAM del nodo con Console di gestione AWS o la AWS CLI.

Console di gestione AWS

a. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
b. Nel pannello di navigazione a sinistra, seleziona Ruoli.
c. Nella pagina Ruoli, seleziona Crea ruolo.
d. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   i. Nella sezione Tipo di entità affidabile, scegli AWS servizio.
   ii. In Use case (Caso d'uso), scegli EC2.
   iii. Scegli Next (Successivo).
e. Nella pagina Aggiungi autorizzazioni, collega una policy personalizzata o esegui le operazioni seguenti:
   i. Nella casella Filtra policy, inserisci `AmazonEKSWorkerNodePolicy`.
   ii. Seleziona la casella di controllo a sinistra di Amazon EKSWorker NodePolicy nei risultati della ricerca.
   iii. Scegli Cancella filtri.
   iv. Nella casella Filtra policy, inserisci `AmazonEC2ContainerRegistryPullOnly`.
   v. Seleziona la casella di controllo a sinistra di Amazon EC2 ContainerRegistryPullOnly nei risultati della ricerca.

      La policy gestita Amazoneks_CNI_Policy o una policy creata da te devono essere collegate anche a questo ruolo o a IPv6 un ruolo diverso mappato all'account del servizio Kubernetes. `aws-node` Si consiglia di assegnare la policy al ruolo associato all'account del servizio Kubernetes anziché assegnarlo a questo ruolo. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".
   vi. Scegli Next (Successivo).
f. Nella pagina Name, review, and create (Assegna un nome, rivedi e crea), esegui le operazioni seguenti:
   i. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSNodeRole`.
   ii. In Descrizione, sostituisci il testo attuale con un testo descrittivo come `Amazon EKS - Node role`.

iii. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.

iv. Scegli Crea ruolo.

AWS CLI

a. Per creare il file `node-role-trust-relationship.json`, emetti il seguente comando:

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

b. Crea il ruolo IAM.

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

c. Allegare al ruolo IAM le due policy gestite IAM richieste.

```
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. Allega una delle seguenti policy IAM al ruolo IAM a seconda della famiglia di IP con cui hai creato il cluster. La policy deve essere allegata a questo ruolo o a un ruolo associato

all'account di servizio Kubernetes `aws-node` che viene utilizzato per il componente aggiuntivo Amazon VPC CNI per Kubernetes. Si consiglia di assegnare la policy al ruolo associato all'account del servizio Kubernetes. Per assegnare la policy al ruolo associato all'account del servizio Kubernetes, consultare the section called "Configurare per IRSA".

- IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSNodeRole
```

- IPv6

  A. Copia il testo seguente e salvalo in un file denominato `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. Creare la policy IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

C. Allega la policy IAM al ruolo IAM. Sostituisci *111122223333* con l'ID del tuo account.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

Ruolo IAM del cluster della modalità automatica di Amazon EKS

richiesto un ruolo IAM del cluster di Amazon EKS per ogni cluster. I cluster di Kubernetes gestiti da Amazon EKS utilizzano questo ruolo per automatizzare le attività di routine per lo storage, il networking e il dimensionamento automatico di elaborazione.

Prima di poter creare cluster Amazon EKS, devi creare un ruolo IAM con le policy richieste per la modalità automatica di EKS. Puoi allegare le politiche gestite AWS IAM suggerite o creare policy personalizzate con autorizzazioni equivalenti.

- EKSComputePolitica di Amazon
- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- EKSNetworkingPolitica di Amazon
- EKSClusterPolitica di Amazon

Verifica della presenza di un ruolo del cluster esistente

Per verificare se l'account dispone già di un ruolo del cluster Amazon EKS, puoi utilizzare la procedura indicata di seguito.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Cerca l'elenco dei ruoli per `AmazonEKSAutoClusterRole`. Se un ruolo che include `AmazonEKSAutoClusterRole` non esiste, consultare le istruzioni nella sezione successiva per crearlo. Se un ruolo che include `AmazonEKSAutoClusterRole` esiste, seleziona il ruolo per visualizzare le policy allegate.

4. Selezionare Autorizzazioni.
5. Assicurati che la EKSClusterpolitica gestita da Amazon Policy sia associata al ruolo. Se la policy è allegata, il ruolo del cluster Amazon EKS è configurato correttamente.
6. Scegliere Relazioni di attendibilità, quindi scegliere Modifica policy di attendibilità.
7. Verifica che la relazione di trust includa la policy seguente. Se la relazione di attendibilità corrisponde alla policy seguente, scegli Annulla. Se la relazione di attendibilità non corrisponde, copia la policy nella finestra Modifica policy di attendibilità e scegli Aggiorna policy.

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

> **Note**
>
> AWS non richiede il nome `AmazonEKSAutoClusterRole` per questo ruolo.

Creazione del ruolo del cluster Amazon EKS

È possibile utilizzare Console di gestione AWS o la AWS CLI per creare il ruolo del cluster.

Console di gestione AWS

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Scegli Ruoli, quindi Crea ruolo.
3. In Tipo di entità affidabile, seleziona AWS servizio.

4. Dall'elenco a discesa Casi d'uso per altri AWS servizi, scegli EKS.
5. Scegli EKS - Cluster per il caso d'uso, quindi scegli Successivo.
6. Nella scheda Aggiungi autorizzazioni, selezionare le policy e quindi scegliere Avanti.
   - EKSComputePolitica di Amazon
   - AmazonEKSBlockStoragePolicy
   - AmazonEKSLoadBalancingPolicy
   - EKSNetworkingPolitica di Amazon
   - EKSClusterPolitica di Amazon
7. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSAutoClusterRole`.
8. Per Descrizione, inserisci un testo descrittivo come `Amazon EKS - Cluster role`.
9. Scegli Crea ruolo.

AWS CLI

1. Copiare i seguenti contenuti in un file denominato *`cluster-trust-policy.json`*.

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

2. Crea il ruolo. Puoi sostituire *`AmazonEKSAutoClusterRole`* con un nome a tua scelta.

```
aws iam create-role \
  --role-name AmazonEKSAutoClusterRole \
```

```
    --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. Allegare le policy IAM richieste al ruolo:

EKSClusterPolitica di Amazon:

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

EKSComputePolitica di Amazon:

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

Amazon EKSBlock StoragePolicy:

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

Amazon EKSLoad BalancingPolicy:

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

EKSNetworkingPolitica di Amazon:

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

Ruolo IAM del nodo della modalità automatica di Amazon EKS

> **Note**
>
> Non puoi utilizzare lo stesso ruolo usato per creare i cluster.

Prima di creare i nodi, devi creare un ruolo IAM con le seguenti policy, o autorizzazioni equivalenti:

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

Verifica della presenza di un ruolo di nodo esistente

Per controllare se l'account dispone già di un ruolo di nodo Amazon EKS, utilizza la procedura indicata di seguito.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Cerca l'elenco dei ruoli per `AmazonEKSAutoNodeRole`. Se un ruolo con uno di questi nomi non esiste, consulta le istruzioni nella sezione successiva per creare il ruolo. Se esiste un ruolo che contiene `AmazonEKSAutoNodeRole`, selezionare il ruolo per visualizzare i criteri allegati.
4. Seleziona Autorizzazioni.
5. Assicurati che siano allegate le policy richieste sopra riportate o le policy personalizzate equivalenti.
6. Scegli Relazioni di attendibilità, quindi scegli Modifica policy di attendibilità.
7. Verifica che la relazione di trust includa la policy seguente. Se la relazione di attendibilità corrisponde alla policy seguente, scegli Annulla. Se la relazione di attendibilità non corrisponde, copia la policy nella finestra Modifica policy di attendibilità e scegli Aggiorna policy.

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

Creazione del ruolo IAM del nodo Amazon EKS

Puoi creare il ruolo IAM del nodo con Console di gestione AWS o la AWS CLI.

Console di gestione AWS

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Nella pagina Ruoli, seleziona Crea ruolo.
4. Nella pagina Seleziona un'entità attendibile, esegui le operazioni seguenti:
   a. Nella sezione Tipo di entità affidabile, scegli AWS servizio.
   b. In Use case (Caso d'uso), scegli EC2.
   c. Scegli Next (Successivo).
5. Nella pagina Aggiungi autorizzazioni, allega le policy seguenti:
   - AmazonEKSWorkerNodeMinimalPolicy
   - AmazonEC2ContainerRegistryPullOnly
6. Nella pagina Assegna un nome, rivedi e crea, esegui le operazioni seguenti:
   a. Per Nome ruolo, inserisci un nome univoco per il ruolo, ad esempio `AmazonEKSAutoNodeRole`.
   b. In Descrizione, sostituisci il testo attuale con un testo descrittivo come `Amazon EKS - Node role`.
   c. In Aggiungi tag (facoltativo), aggiungi metadati al ruolo collegando i tag come coppie chiave-valore. Per ulteriori informazioni sull'utilizzo di tag in IAM, consulta la sezione Tagging IAM resources nella Guida per l'utente di IAM.
   d. Scegli Crea ruolo.

AWS CLI

Crea il ruolo IAM del nodo

Utilizza il node-trust-policyfile.json del passaggio precedente per definire quali entità possono assumere il ruolo. Per creare il ruolo IAM del nodo, esegui il seguente comando:

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
```

```
    --assume-role-policy-document file://node-trust-policy.json
```

Annota l'ARN del ruolo

Una volta creato il ruolo, recupera e salva l'ARN del ruolo IAM del nodo. Sarà necessario usare questo ARN nei passaggi successivi. Usa il seguente comando per ottenere l'ARN:

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

Allega le policy richieste

Allega le seguenti policy AWS gestite al ruolo Node IAM per fornire le autorizzazioni necessarie:

Per collegare Amazon EKSWorkerNodeMinimalPolicy:

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

Per collegare Amazon EC2ContainerRegistryPullOnly:

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Funzionalità Amazon EKS, ruolo IAM

Le funzionalità EKS richiedono la configurazione di un ruolo IAM di capacità (o ruolo di capacità). Le funzionalità utilizzano questo ruolo per eseguire azioni sui AWS servizi e accedere alle risorse Kubernetes nel cluster tramite voci di accesso create automaticamente.

Prima di poter specificare un ruolo di capacità durante la creazione della capacità, è necessario creare il ruolo IAM con la policy di fiducia e le autorizzazioni appropriate per il tipo di funzionalità. Una volta creato, questo ruolo IAM può essere riutilizzato per un numero qualsiasi di risorse funzionali.

Requisiti del ruolo di capacità

Il ruolo di capacità deve soddisfare i seguenti requisiti:

- Il ruolo deve appartenere allo stesso AWS account del cluster e della risorsa di capacità
- Il ruolo deve avere una politica di fiducia che consenta al servizio EKS Capabilities di assumerlo

- Il ruolo deve disporre delle autorizzazioni appropriate per il tipo di capacità e i requisiti del caso d'uso (vedithe section called "Autorizzazioni per tipo di funzionalità")

Politica di fiducia per i ruoli di capacità

Tutti i ruoli di capacità devono includere la seguente politica di fiducia:

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

Questa politica di fiducia consente a EKS di:

- Assumi il ruolo di eseguire operazioni AWS API
- Tagga le sessioni per scopi di controllo e tracciamento

Autorizzazioni per tipo di funzionalità

Le autorizzazioni IAM richieste dipendono dalla funzionalità utilizzata e dal modello di implementazione.

> **Note**
>
> Per le implementazioni di produzione che utilizzano IAM Role Selectors con ACK o quando si utilizza kro o Argo CD senza integrazione di AWS servizi, Capability Role potrebbe non richiedere alcuna autorizzazione IAM oltre alla policy di fiducia.

kro (Kube Resource Orchestrator)

Non sono richieste autorizzazioni IAM. Puoi creare un ruolo di capacità senza policy allegate. kro richiede solo le autorizzazioni RBAC di Kubernetes per creare e gestire le risorse Kubernetes.

AWS Controller per Kubernetes (ACK)

ACK supporta due modelli di autorizzazione:

- Configurazione semplice (sviluppo/test): aggiungi le autorizzazioni di AWS servizio direttamente al Capability Role. È la soluzione ideale per iniziare, per le implementazioni con account singolo o quando tutti gli utenti necessitano delle stesse autorizzazioni.
- Best practice di produzione: utilizza IAM Role Selectors per implementare l'accesso con privilegi minimi. Con questo approccio, il Capability Role necessita solo dell'`sts:AssumeRole`autorizzazione per assumere ruoli specifici del servizio. Non si aggiungono autorizzazioni di AWS servizio (come S3 o RDS) al ruolo di capacità stesso: tali autorizzazioni vengono concesse a singoli ruoli IAM mappati su namespace specifici.

  I selettori di ruolo IAM consentono di:

  - Isolamento delle autorizzazioni a livello di namespace
  - Gestione delle risorse tra account
  - Ruoli IAM specifici del team
  - Modello di sicurezza con privilegi minimi

  Esempio di policy Capability Role per l'approccio IAM Role Selector:

```
{
  "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"
      ]
    }
  ]
}
```

Per una configurazione dettagliata delle autorizzazioni ACK, inclusi IAM Role Selectors, consulta. the section called "Configurazione delle autorizzazioni "

CD Argo

Per impostazione predefinita, non sono richieste autorizzazioni IAM. Potrebbero essere necessarie autorizzazioni opzionali per:

- AWS Secrets Manager: se si utilizza Secrets Manager per archiviare le credenziali del repository Git
- AWS CodeConnections: Se si utilizza CodeConnections per l'autenticazione del repository Git

Politica di esempio per Secrets Manager e 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/*"
    }
  ]
}
```

Per i requisiti di autorizzazione dettagliati di Argo CD, vederethe section called "Considerazioni su Argo CD".

Verificate la presenza di un ruolo di capacità esistente

Puoi utilizzare la seguente procedura per verificare se il tuo account dispone già di un ruolo IAM con funzionalità adatto al tuo caso d'uso.

1. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nel pannello di navigazione a sinistra, seleziona Ruoli.
3. Cerca nell'elenco dei ruoli il nome del tuo ruolo di capacità (ad esempio, `ACKCapabilityRole` o`ArgoCDCapabilityRole`).
4. Se esiste un ruolo, selezionalo per visualizzare le politiche allegate e la relazione di fiducia.
5. Scegliere Relazioni di attendibilità, quindi scegliere Modifica policy di attendibilità.
6. Verifica che la relazione di fiducia corrisponda alla politica di fiducia in materia di capacità. Se non corrisponde, aggiorna la politica di fiducia.
7. Scegli Autorizzazioni e verifica che il ruolo disponga delle autorizzazioni appropriate per il tipo di capacità e il caso d'uso.

Creazione di una funzionalità (ruolo IAM)

È possibile utilizzare Console di gestione AWS o la AWS CLI per creare un ruolo di capacità.

Console di gestione AWS

a. Apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
b. Scegli Ruoli, quindi Crea ruolo.
c. In Tipo di entità affidabile, seleziona Politica di fiducia personalizzata.
d. Copia e incolla la politica di attendibilità delle capacità nell'editor dei criteri di fiducia.
e. Scegli Next (Successivo).
f. Nella scheda Aggiungi autorizzazioni, seleziona o crea politiche appropriate per il tuo tipo di capacità (vedithe section called "Autorizzazioni per tipo di funzionalità"). Per kro, puoi saltare questo passaggio.
g. Scegli Next (Successivo).
h. Per Nome ruolo, inserisci un nome univoco per il tuo ruolo, ad esempio `ACKCapabilityRoleArgoCDCapabilityRole`, o. `kroCapabilityRole`
i. Per Descrizione, inserisci un testo descrittivo come `Amazon EKS - ACK capability role`.

j. Scegliere Crea ruolo.

AWS CLI

a. Copia la policy di affidabilità delle capacità in un file denominato`capability-trust-policy.json`.

b. Crea il ruolo. Sostituiscila `ACKCapabilityRole` con il nome del ruolo desiderato.

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

c. Allega le politiche IAM richieste al ruolo. Per ACK, allega le policy per AWS i servizi che desideri gestire. Per Argo CD, allegate le politiche per Secrets Manager o CodeConnections se necessario. Per kro, puoi saltare questo passaggio.

   Esempio per ACK con autorizzazioni S3:

```
aws iam put-role-policy \
  --role-name ACKCapabilityRole \
  --policy-name S3Management \
  --policy-document file://s3-policy.json
```

Risoluzione dei problemi relativi alle funzionalità e ai ruoli

La creazione delle funzionalità fallisce con «Ruolo IAM non valido»

Verificare che:

- Il ruolo esiste nello stesso account del cluster
- La politica di fiducia corrisponde alla politica di affidabilità delle capacità
- Hai l'`iam:PassRole`autorizzazione per il ruolo

La funzionalità mostra errori di autorizzazione

Verificare che:

- Il ruolo dispone delle autorizzazioni IAM necessarie per il tipo di funzionalità
- La voce di accesso per il ruolo esiste nel cluster
- Se necessario, vengono configurate autorizzazioni Kubernetes aggiuntive (vedi) the section called “Autorizzazioni Kubernetes aggiuntive”

Le risorse ACK falliscono con errori di «autorizzazione negata»

Verificare che:

- Il ruolo dispone delle autorizzazioni IAM necessarie per il tuo caso d'uso
- Per i controller ACK che fanno riferimento a segreti, assicurati di aver associato la policy di `AmazonEKSSecretReaderPolicy` accesso prevista ai namespace appropriati.

Per ulteriori indicazioni sulla risoluzione dei problemi, consulta. the section called "Considerazioni sulle funzionalità EKS"

Monitoraggio delle prestazioni del cluster e visualizzazione dei log

Amazon EKS mette a disposizione una serie di strumenti di monitoraggio o registrazione per l'osservazione dei dati. I dati di log di Amazon EKS possono essere trasmessi in streaming a AWS servizi o a strumenti partner per l'analisi dei dati. Sono disponibili molti servizi Console di gestione AWS che forniscono dati per la risoluzione dei problemi di Amazon EKS. Puoi anche utilizzare una soluzione open source AWS supportata per monitorare l'infrastruttura Amazon EKS.

Dopo aver selezionato Cluster nel riquadro di navigazione sinistro della console Amazon EKS, puoi visualizzare l'integrità e i dettagli del cluster selezionandone il nome e scegliendo la scheda Osservabilità. Per visualizzare i dettagli delle risorse Kubernetes esistenti implementate nel cluster, consulta the section called "Accesso alle risorse di cluster".

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di Amazon EKS e delle tue AWS soluzioni. Ti consigliamo di raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione. per consentire un debug più facile di eventuali errori in più punti. Prima di iniziare il monitoraggio di Amazon EKS, è opportuno creare un piano di monitoraggio che tenga conto dei seguenti aspetti.

- Quali sono gli obiettivi? Sono necessarie notifiche in tempo reale se i cluster si ridimensionano notevolmente?
- Di quali risorse si intende eseguire il monitoraggio?
- Con quale frequenza sarà eseguito il monitoraggio di queste risorse? L'azienda vuole rispondere rapidamente ai rischi?
- Quali strumenti verranno utilizzati? Se utilizzi già AWS Fargate come parte del lancio, puoi utilizzare il log router integrato.
- Chi eseguirà le attività di monitoraggio?
- Chi deve ricevere una notifica quando si verifica un problema?

Registrazione e monitoraggio su Amazon EKS

Amazon EKS offre strumenti integrati per la registrazione e il monitoraggio. Per le versioni supportate, la dashboard di osservabilità offre visibilità sulle prestazioni del cluster. Ti aiuta a rilevare, risolvere

e correggere rapidamente i problemi. Oltre alle funzionalità di monitoraggio, include elenchi basati sui log di audit del piano di controllo (control-plane). Il piano di controllo (control-plane) di Kubernetes espone una serie di metriche che possono essere eliminate anche all'esterno della console.

I log del piano di controllo (control-plane) registrano tutte le chiamate dell'API ai tuoi cluster, le informazioni di controllo che acquisiscono quali utenti hanno eseguito determinate azioni sui tuoi cluster, e informazioni basate sui ruoli. Per ulteriori informazioni, consulta Registrazione e monitoraggio su Amazon EKS nella Guida AWS prescrittiva.

La registrazione del piano di controllo di Amazon EKS fornisce log di audit e diagnostica direttamente dal piano di controllo di Amazon CloudWatch EKS ai log del tuo account. Questi log consentono di semplificare la protezione e l'esecuzione dei cluster. Puoi selezionare i tipi di log esatti di cui hai bisogno e i log vengono inviati come flussi di log a un gruppo per ogni cluster Amazon EKS in cui opera. CloudWatch Per ulteriori informazioni, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".

> **Note**
>
> Quando controlli i log dell'autenticatore di Amazon EKS su Amazon CloudWatch, vengono visualizzate le voci che contengono testo simile al seguente testo di esempio.
>
> ```
> level=info msg="mapping IAM role" groups="[]" role="arn:aws:
>  iam::111122223333:role/XXXXXXXXXXXXXXXXX-NodeManagerRole-XXXXXXXX"
>  username="eks:node-manager"
> ```
>
> Sono previste voci che contengono questo testo. Lo `username` è un ruolo di servizio Amazon EKS interno che esegue operazioni specifiche per i gruppi di nodi gestiti e nodi Fargate. Per la registrazione personalizzabile di basso livello, consulta Registrazione Kubernetes.

Amazon EKS è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, un ruolo o un AWS servizio in Amazon EKS. CloudTrail acquisisce tutte le chiamate API per Amazon EKS come eventi. Le chiamate acquisite includono le chiamate dalla console Amazon EKS e le chiamate di codice alle operazioni API Amazon EKS. Per ulteriori informazioni, consulta the section called " AWS CloudTrail".

Il server API Kubernetes espone una serie di parametri che sono utili per il monitoraggio e l'analisi dei dati. Per ulteriori informazioni, consulta the section called "Parametri Prometheus".

Per configurare Fluent Bit per CloudWatch i log Amazon personalizzati, consulta Configurazione di Fluent Bit nella Amazon CloudWatch User Guide.

Strumenti di registrazione e monitoraggio di Amazon EKS

Amazon Web Services offre vari strumenti che permettono di monitorare Amazon EKS. Alcuni di questi strumenti possono essere configurati in modo che eseguano automaticamente il monitoraggio, mentre altri richiedono l'intervento manuale. Ti consigliamo di automatizzare il più possibile i processi di monitoraggio nella misura in cui l'ambiente e il set di strumenti esistenti lo consentono.

La tabella seguente descrive varie opzioni dello strumento di monitoraggio.

Aree	Strumento	Description	Configurazione
Piano di controllo	Dashboard di osservabilità	Per le versioni supportate, la dashboard di osservabilità offre visibilità sulle prestazioni del cluster. Ti aiuta a rilevare, risolvere e corregger e rapidamente i problemi.	Procedura di configurazione
Applicazioni/piano di controllo (control-plane)	Prometheus	Prometheus può essere utilizzato per monitorare metriche e avvisi per le applicazi oni e il piano di controllo (control-plane).	Procedura di configurazione
Applicazioni	CloudWatch Container Insights	CloudWatch Container Insights raccoglie, aggrega e riepiloga metriche e log delle applicazi	Procedura di configurazione

Aree	Strumento	Description	Configurazione
		oni e dei microservizi containerizzati.	
Applicazioni	AWS OpenTelemetry Distro per (ADOT)	ADOT può raccoglie re e inviare metriche, dati di traccia e metadati correlati a servizi o partner di monitoraggio. AWS Può essere configura to tramite Container Insights. CloudWatch	Procedura di configurazione
Applicazioni	Amazon DevOps Guru	Amazon DevOps Guru rileva le prestazioni operative e la disponibilità a livello di nodo.	Procedura di configurazione
Applicazioni	AWS X-Ray	AWS X-Ray riceve dati di traccia sull'appl icazione. Questi dati di traccia includono le richieste in entrata e in uscita e i metadati relativi a tali richieste . Per Amazon EKS, l'implementazione richiede il OpenTelem etry componente aggiuntivo.	Procedura di configurazione

Aree	Strumento	Description	Configurazione
Applicazioni	Amazon CloudWatch	CloudWatch fornisce gratuitamente alcuni parametri di base di Amazon EKS nelle versioni supportat e. Puoi espandere questa funzional ità con CloudWatc h Observability Operator per gestire la raccolta di metriche, log e dati di tracciamento.	Procedura di configurazione

La tabella seguente descrive varie opzioni dello strumento di registrazione.

Aree	Strumento	Description	Configurazione
Piano di controllo	Dashboard di osservabilità	Per le versioni supportate, la dashboard di osservabilità mostra elenchi basati sui log di audit del piano di controllo (control-plane). Include anche collegamenti per controllare i log dei piani in Amazon CloudWatch.	Procedura di configurazione
Applicazioni	Amazon CloudWatch Container Insights	Amazon CloudWatc h Container Insights raccoglie, aggrega e	Procedura di configurazione

Aree	Strumento	Description	Configurazione
		riepiloga i parametri e i log delle tue applicazioni e microservizi containerizzati.	
Piano di controllo (control-plane)	CloudWatch Registri Amazon	Puoi inviare i log di controllo e diagnostica direttamente dal piano di controllo di Amazon CloudWatch EKS ai registri del tuo account.	Procedura di configurazione
Piano di controllo (control-plane)	AWS CloudTrail	Registra le chiamate API da parte di un utente, un ruolo o un servizio.	Procedura di configurazione
Più aree per le istanze AWS Fargate	AWS Router di registro Fargate	Per le istanze AWS Fargate, il router di log trasmette i log ai servizi o agli AWS strumenti dei partner. Utilizza AWS per Fluent Bit. I log possono essere trasmessi in streaming ad altri servizi o strumenti partner. AWS	Procedura di configurazione

Monitora il tuo cluster con la dashboard di osservabilità

La console Amazon EKS include una dashboard di osservabilità che offre visibilità nelle prestazioni del cluster. Le informazioni che fornisce aiutano a rilevare, risolvere e chiarire rapidamente i problemi. È possibile aprire la sezione applicabile del dashboard di osservabilità scegliendo un elemento nel Riepilogo di integrità e prestazioni. Questo riepilogo è incluso in diversi punti, inclusa la scheda Osservabilità.

La dashboard di osservabilità è suddivisa in diverse schede.

Riepilogo

Il riepilogo dello stato e delle prestazioni elenca la quantità di articoli in varie categorie. Ciascun numero funge da collegamento ipertestuale a una posizione nella dashboard di osservabilità con un elenco per quella categoria.

Integrità del cluster

L'integrità del cluster fornisce notifiche importanti di cui essere a conoscenza, su alcune delle quali potrebbe essere necessario intervenire il prima possibile. Con questo elenco, è possibile vedere le descrizioni e le risorse interessate. Lo stato del cluster include due tabelle: Problemi di integrità e Approfondimenti sulla configurazione. Per aggiornare lo stato dei Problemi di integrità, scegli il pulsante di aggiornamento (↻). Gli Approfondimenti sulla configurazione si aggiornano automaticamente una volta ogni 24 ore e non possono essere aggiornate manualmente.

Per ulteriori informazioni sui Problemi di integrità, consulta the section called "Domande frequenti sull'integrità del cluster e codici di errore con percorsi di risoluzione". Per ulteriori informazioni su Approfondimenti sulla configurazione, consulta the section called "Approfondimenti sui cluster".

Monitoraggio del piano di controllo

La scheda Monitoraggio del piano di controllo è divisa in tre sezioni, ognuna delle quali aiuta a monitorare e risolvere i problemi del piano di controllo del cluster.

Metriche

Per i cluster con versione Kubernetes `1.28` e successive, la sezione Metriche mostra grafici di diverse metriche raccolte per vari componenti del piano di controllo.

È possibile impostare il periodo di tempo utilizzato dall'asse X di ogni grafico effettuando le selezioni nella parte superiore della sezione. È possibile aggiornare i dati con il pulsante di aggiornamento (↻). Per ogni grafico separato, il pulsante con le ellissi verticali (⋮) apre un menu con opzioni da. CloudWatch

Queste e altre metriche sono automaticamente disponibili come metriche di monitoraggio di base nel CloudWatch namespace. `AWS/EKS` Per ulteriori informazioni, consulta la sezione Monitoraggio di base e monitoraggio dettagliato nella Amazon CloudWatch User Guide. Per ottenere metriche, visualizzazioni e approfondimenti più dettagliati, consulta Container Insights nella Amazon CloudWatch User Guide. Oppure, se si preferisce un monitoraggio basato su Prometheus, consulta the section called "Parametri Prometheus".

La tabella seguente descrive le metriche disponibili.

Metrica	Description
APIServer Richieste	Le richieste al minuto effettuate al server API.
APIServer Richieste totali 4XX	Il numero di richieste al minuto del server API con codici di risposta HTTP 4XX (errori lato client).
APIServer Richieste totali 5XX	Il numero di richieste al minuto del server API con codici di risposta HTTP 5XX (errori lato server).
APIServer Richieste totali 429	Il numero di richieste al minuto del server API con codici di risposta HTTP 429 (troppe richieste).
Dimensioni dell'archiviazione	La dimensione del database di archiviazione (`etcd`).
Tentativi del sistema di pianificazione	Il numero di tentativi di pianificare i pod in base ai risultati "Non pianificabile", "Errore" e "Pianificato".
Pod in sospeso	Il numero di pod in sospeso per tipo di coda di "attivo", "backoff", "non pianificabile" e "limitato".

Metrica	Description
Latenza delle richieste del server API	La latenza per le richieste del server API.
Richieste correnti del server API	Le richieste correnti per il server API.
Richieste webhook	Le richieste webhook al minuto.
Rifiuti delle richieste webhook	Il numero di richieste webhook che sono state rifiutate.
Latenza della richiesta webhook P99	La latenza del 99° percentile delle richieste webhook esterne di terze parti.

CloudWatch Log Insights

La sezione CloudWatch Log Insights mostra vari elenchi basati sui log di controllo del piano di controllo. I log del piano di controllo di Amazon EKS devono essere attivati per utilizzare questa funzionalità, che puoi eseguire dalla sezione Visualizza i log del piano di controllo. CloudWatch

Quando è trascorso abbastanza tempo per raccogliere i dati, è possibile Eseguire tutte le query o scegliere Esegui query per un singolo elenco alla volta. CloudWatch Ogni volta che esegui una richiesta, verrà addebitato un costo aggiuntivo. Nella parte superiore della sezione, scegli il periodo di tempo dei risultati che desideri visualizzare. Se desideri un controllo più avanzato per qualsiasi query, puoi scegliere Visualizza in. CloudWatch Ciò ti consentirà di aggiornare una query in CloudWatch base alle tue esigenze.

Per ulteriori informazioni, consulta Analyzing log data with CloudWatch Logs Insights nella Amazon CloudWatch Logs User Guide.

Visualizza i log del piano di controllo in CloudWatch

Per aggiornare i tipi di log disponibili, scegli Gestisci log. Dopo aver abilitato la registrazione, occorrono alcuni minuti prima che i log vengano visualizzati in CloudWatch Logs. Quando è trascorso tempo sufficiente, scegli uno dei link Visualizza in questa sezione per accedere al log applicabile.

Per ulteriori informazioni, consulta the section called "Registrazione dei log del piano di controllo (control-plane)".

Approfondimenti sui cluster

La tabella Migliora approfondimenti fa emergere i problemi e consiglia azioni correttive, accelerando il processo di convalida per l'aggiornamento a nuove versioni di Kubernetes. Amazon EKS esegue automaticamente la scansione dei cluster e li mette a confronto con un elenco di potenziali problemi di aggiornamento della versione Kubernetes. La tabella Migliora approfondimenti elenca i controlli approfonditi eseguiti da Amazon EKS su questo cluster, insieme ai relativi stati associati.

Amazon EKS gestisce e aggiorna periodicamente l'elenco dei controlli degli approfondimenti da eseguire sulla base delle valutazioni delle modifiche nel progetto Kubernetes e delle modifiche al servizio Amazon EKS legate alle nuove versioni. La console Amazon EKS aggiorna automaticamente lo stato di ciascun approfondimento, che può essere visualizzato nella colonna relativa all'ora dell'ultimo aggiornamento.

Per ulteriori informazioni, consulta the section called "Approfondimenti sui cluster".

Problemi di integrità dei nodi

L'agente di monitoraggio dei nodi Amazon EKS legge automaticamente i log dei nodi per rilevare problemi di integrità. Indipendentemente dall'impostazione di riparazione automatica, vengono segnalati tutti i problemi di integrità dei nodi in modo da poterli esaminare se necessario. Se un tipo di problema è elencato senza una descrizione, è possibile leggere la descrizione nel relativo elemento popover.

Quando si aggiorna la pagina, tutti i problemi risolti scompariranno dall'elenco. Se la riparazione automatica è abilitata, potresti visualizzare temporaneamente alcuni problemi di integrità che verranno risolti senza intervento da parte dell'utente. I problemi non supportati dalla riparazione automatica potrebbero richiedere un'azione manuale da parte dell'utente a seconda del tipo.

Per segnalare problemi di integrità dei nodi, il cluster deve utilizzare la modalità automatica Amazon EKS o disporre del componente aggiuntivo dell'agente di monitoraggio dei nodi. Per ulteriori informazioni, consulta the section called "Integrità del nodo".

Funzionalità EKS

La sezione Capacità mostra lo stato e lo stato delle risorse EKS Capability nel cluster. Le notifiche sullo stato e sullo stato di entrambe le funzionalità e le relative risorse Kubernetes gestite nel cluster possono essere monitorate qui. Quando si aggiorna la pagina, tutti i problemi risolti scompariranno dall'elenco.

Per ulteriori informazioni, consulta the section called "Lavorare con le funzionalità".

Monitora il traffico dei carichi di lavoro Kubernetes con Container Network Observability

Amazon EKS offre funzionalità avanzate di osservabilità della rete che forniscono informazioni più approfondite sull'ambiente di rete dei container. Queste funzionalità ti aiutano a comprendere, monitorare e risolvere meglio il tuo panorama di rete Kubernetes in. AWS Con una migliore osservabilità della rete di container, puoi sfruttare metriche granulari relative alla rete per un migliore rilevamento proattivo delle anomalie nel traffico del cluster, nei flussi cross-AZ e nei servizi. AWS Utilizzando queste metriche, puoi misurare le prestazioni del sistema e visualizzare le metriche sottostanti utilizzando il tuo stack di osservabilità preferito.

Inoltre, Amazon EKS ora fornisce visualizzazioni di monitoraggio della rete nella AWS console che accelerano e migliorano la risoluzione precisa dei problemi per un'analisi più rapida delle cause principali. Puoi anche sfruttare queste funzionalità visive per individuare i principali interlocutori e i flussi di rete che causano ritrasmissioni e timeout di ritrasmissione, eliminando i punti ciechi durante gli incidenti.

Queste funzionalità sono abilitate da Amazon CloudWatch Network Flow Monitor.

Casi d'uso

Misura le prestazioni della rete per rilevare anomalie

Diversi team si basano su uno stack di osservabilità che consente loro di misurare le prestazioni del sistema, visualizzare le metriche del sistema e allarmarsi nel caso in cui venga superata una soglia specifica. L'osservabilità della rete di container in EKS si allinea a questo aspetto, esponendo le metriche chiave del sistema che è possibile analizzare per ampliare l'osservabilità delle prestazioni di rete del sistema a livello di pod e nodo di lavoro.

Sfrutta le visualizzazioni della console per una risoluzione dei problemi più precisa

In caso di allarme proveniente dal sistema di monitoraggio, potresti voler concentrarti sul cluster e sul carico di lavoro da cui ha avuto origine il problema. A tale scopo, puoi sfruttare le visualizzazioni della console EKS che restringono l'ambito di indagine a livello di cluster e accelerano la divulgazione dei flussi di rete responsabili della maggior parte delle ritrasmissioni, dei timeout di ritrasmissione e del volume di dati trasferiti.

Tieni traccia dei migliori oratori nel tuo ambiente Amazon EKS

Molti team utilizzano EKS come base per le proprie piattaforme, rendendolo il punto focale dell'attività di rete di un ambiente applicativo. Utilizzando le funzionalità di monitoraggio della rete di questa funzionalità, puoi monitorare quali carichi di lavoro sono responsabili della maggior parte del traffico (misurato in base al volume di dati) all'interno del cluster AZs, oltre al traffico verso destinazioni esterne all'interno AWS (DynamoDB e S3) e oltre AWS il cloud (Internet o on-premise). Inoltre, è possibile monitorare le prestazioni di ciascuno di questi flussi in base alle ritrasmissioni, ai timeout di ritrasmissione e ai dati trasferiti.

Funzionalità

1. Metriche delle prestazioni: questa funzionalità consente di acquisire le metriche di sistema relative alla rete per pod e nodi di lavoro direttamente dall'agente Network Flow Monitor (NFM) in esecuzione nel cluster EKS.
2. Mappa dei servizi: questa funzionalità visualizza dinamicamente l'intercomunicazione tra i carichi di lavoro nel cluster, consentendo di divulgare rapidamente le metriche chiave (ritrasmissioni - RT, timeout di ritrasmissione - RTO e dati trasferiti - DT) associate ai flussi di rete tra pod comunicanti.
3. Tabella dei flussi: con questa tabella, puoi monitorare i migliori oratori nei carichi di lavoro Kubernetes del tuo cluster da tre diverse angolazioni: visualizzazione del AWS servizio, visualizzazione del cluster e visualizzazione esterna. Per ogni vista, puoi vedere le ritrasmissioni, i timeout di ritrasmissione e i dati trasferiti tra il pod di origine e la sua destinazione.
   - AWS visualizzazione del servizio: mostra i migliori oratori ai AWS servizi (DynamoDB e S3)
   - Visualizzazione cluster: mostra i migliori oratori all'interno del cluster (da est ← a → ovest)
   - Vista esterna: mostra i migliori oratori alle destinazioni esterne al cluster all'esterno AWS

Nozioni di base

Per iniziare, abilita Container Network Observability nella console EKS per un cluster nuovo o esistente. Ciò automatizzerà la creazione delle dipendenze di Network Flow Monitor (NFM) (risorse Scope and Monitor). Inoltre, dovrai installare il componente aggiuntivo Network Flow Monitor Agent. In alternativa, puoi installare queste dipendenze utilizzando EKS APIs (per il componente aggiuntivo) `AWS CLI`, NFM APIs o Infrastructure as Code (come Terraform). Una volta stabilite queste dipendenze, puoi configurare lo strumento di monitoraggio preferito per raccogliere le metriche delle prestazioni di rete per pod e nodi di lavoro dall'agente NFM. Per visualizzare l'attività di

rete e le prestazioni dei carichi di lavoro, è possibile accedere alla console EKS nella scheda «Rete» del dashboard di osservabilità del cluster.

Quando si utilizza Network Flow Monitor in EKS, è possibile mantenere il flusso di lavoro e lo stack tecnologico di osservabilità esistenti, sfruttando al contempo una serie di funzionalità aggiuntive che consentono di comprendere e ottimizzare ulteriormente il livello di rete del proprio ambiente EKS. Puoi saperne di più sui prezzi di Network Flow Monitor qui.

Prerequisiti e note importanti

1. Come accennato in precedenza, se abilitate Container Network Observability dalla console EKS, le dipendenze delle risorse NFM sottostanti (Scope e Monitor) verranno create automaticamente per vostro conto e sarete guidati attraverso il processo di installazione del componente aggiuntivo EKS per NFM.
2. Se desideri abilitare questa funzionalità utilizzando Infrastructure as Code (IaC) come Terraform, dovrai definire le seguenti dipendenze nel tuo IaC: NFM Scope, NFM Monitor, componente aggiuntivo EKS per NFM. Inoltre, dovrai concedere le autorizzazioni pertinenti al componente aggiuntivo EKS utilizzando Pod Identity o IAM roles for service accounts (IRSA).
3. È necessario eseguire una versione minima di 1.1.0 per il componente aggiuntivo EKS dell'agente NFM.
4. È necessario utilizzare la versione 6.21.0 o successiva di Terraform AWS Provider per il supporto delle risorse di Network Flow Monitor.

Autorizzazioni IAM richieste

Componente aggiuntivo EKS per l'agente NFM

Puoi utilizzare la policy `CloudWatchNetworkFlowMonitorAgentPublishPolicy`AWS gestita con Pod Identity. Questa policy contiene le autorizzazioni per l'agente NFM a inviare report di telemetria (metriche) a un endpoint Network Flow Monitor.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "networkflowmonitor:Publish"
      ],
```

```
      "Resource" : "*"
    }
  ]
}
```

Osservabilità della rete dei container nella console EKS

Le seguenti autorizzazioni sono necessarie per abilitare la funzionalità e visualizzare la mappa dei servizi e la tabella dei flussi nella 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": "*"
    }
  ]
}
```

Utilizzo di AWS CLI, API EKS e API 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})"
```

Utilizzo dell'infrastruttura come codice (IaC)

Terraform

Se utilizzi Terraform per gestire la tua infrastruttura AWS cloud, puoi includere le seguenti configurazioni di risorse per abilitare l'osservabilità della rete di contenitori per il tuo cluster.

Ambito 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 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"
  }
}
```

Componente aggiuntivo EKS per NFM

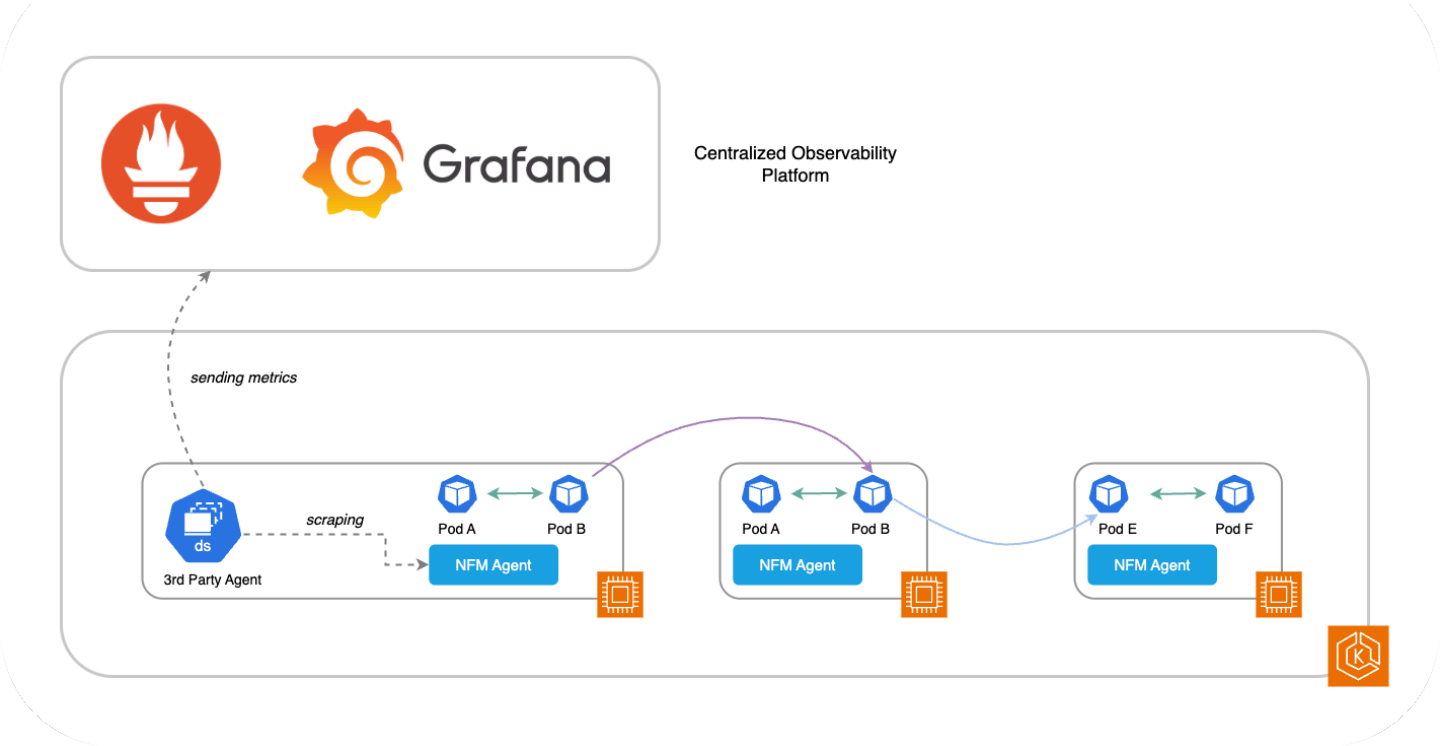
```
resource "aws_eks_addon" "example" {
  cluster_name                = aws_eks_cluster.example.name
  addon_name                  = "aws-network-flow-monitoring-agent"
}
```

Come funziona?

Metriche delle prestazioni

Parametri del sistema

Se utilizzi strumenti di terze parti (3P) per monitorare il tuo ambiente EKS (come Prometheus e Grafana), puoi acquisire le metriche di sistema supportate direttamente dall'agente Network Flow Monitor. Queste metriche possono essere inviate allo stack di monitoraggio per ampliare la misurazione delle prestazioni di rete del sistema a livello di pod e nodo di lavoro. Le metriche disponibili sono elencate nella tabella, in Metriche di sistema supportate.



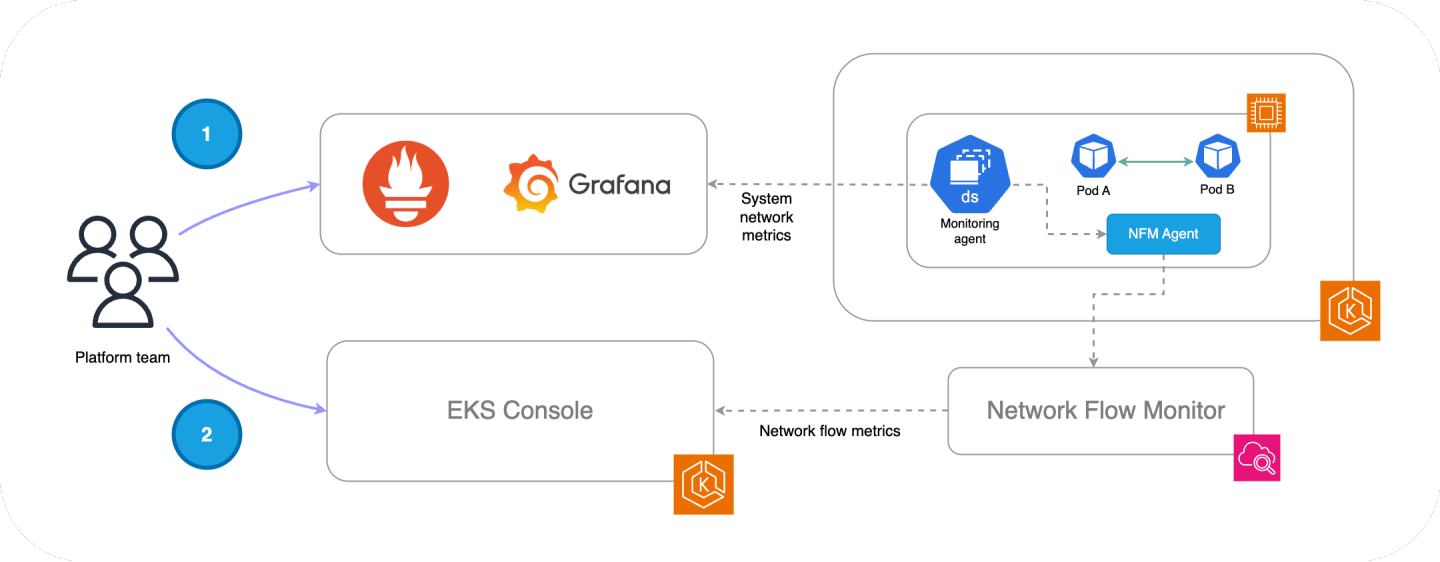
Per abilitare queste metriche, sovrascrivete le seguenti variabili di ambiente utilizzando le variabili di configurazione durante il processo di installazione (vedi: containers/ -advanced-configuration/): https://aws.amazon.com/blogs/ amazon-eks-add-ons

```
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]
```

Metriche del livello di flusso

Inoltre, Network Flow Monitor acquisisce i dati del flusso di rete insieme alle metriche del livello di flusso: ritrasmissioni, timeout di ritrasmissione e dati trasferiti. Questi dati vengono elaborati da Network Flow Monitor e visualizzati nella console EKS per far emergere il traffico nell'ambiente del cluster e il relativo rendimento in base a queste metriche del livello di flusso.

Il diagramma seguente illustra un flusso di lavoro in cui entrambi i tipi di metriche (sistema e livello di flusso) possono essere sfruttati per ottenere una maggiore intelligenza operativa.



1. Il team della piattaforma può raccogliere e visualizzare le metriche di sistema nel proprio stack di monitoraggio. Una volta attivati gli avvisi, possono rilevare anomalie di rete o problemi che influiscono sui pod o sui nodi di lavoro utilizzando le metriche di sistema fornite dall'agente NFM.
2. Come passaggio successivo, i team della piattaforma possono sfruttare le visualizzazioni native della console EKS per restringere ulteriormente l'ambito delle indagini e accelerare la risoluzione dei problemi in base alle rappresentazioni dei flussi e alle metriche associate.

Nota importante: l'acquisizione delle metriche di sistema dall'agente NFM e il processo con cui l'agente NFM invia le metriche a livello di flusso al backend NFM sono processi indipendenti.

Metriche di sistema supportate

Nota importante: le metriche di sistema vengono esportate in formato. OpenMetrics

Nome parametro	Tipo	Dimensioni	Description
ingress_flow	Gauge	instance_id, iface, pod, namespace, nodo	Conteggio del flusso TCP in ingresso () TcpPassiveOpens

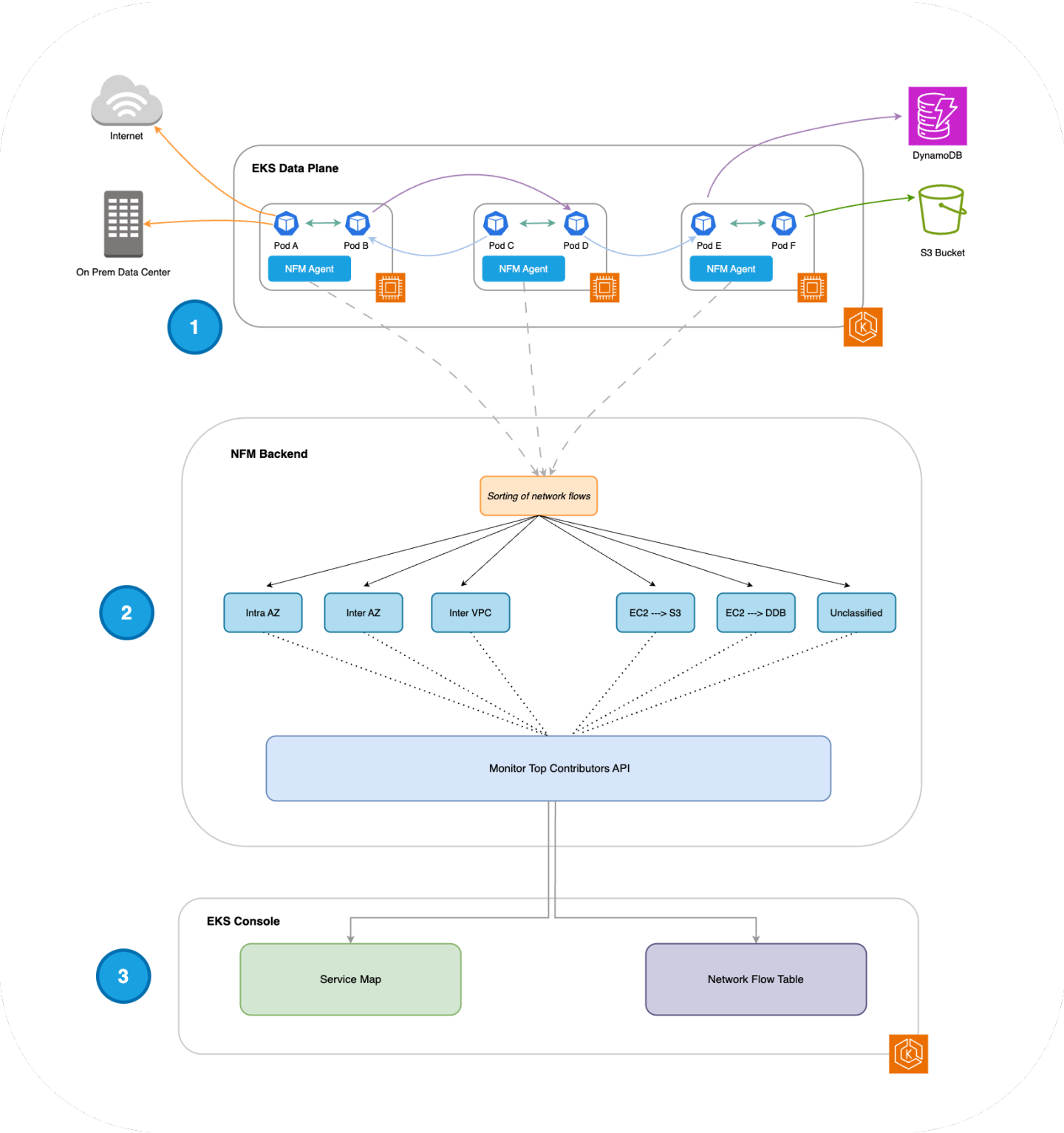
Nome parametro	Tipo	Dimensioni	Description
Flusso_uscita	Gauge	instance_id, iface, pod, namespace, node	Conteggio del flusso TCP in uscita () TcpActiveOpens
pacchetti_in ingresso	Gauge	instance_id, iface, pod, namespace, nodo	Numero di pacchetti in ingresso (delta)
pacchetti_in uscita	Gauge	instance_id, iface, pod, namespace, node	Numero di pacchetti in uscita (delta)
ingress_bytes	Gauge	instance_id, iface, pod, namespace, nodo	Conteggio dei byte in ingresso (delta)
egress_bytes	Gauge	instance_id, iface, pod, namespace, nodo	Conteggio dei byte in uscita (delta)
bw_in_allowance_exceeded	Gauge	instance_id, eni, nodo	Pacchetti queued/dropped dovuti al limite di larghezza di banda in entrata
bw_out_allowance_exceeded	Gauge	instance_id, eni, nodo	Pacchetti queued/dropped dovuti al limite di larghezza di banda in uscita
pps_allowance_exceeded	Gauge	instance_id, eni, nodo	Pacchetti queued/dropped dovuti al limite PPS bidirezionale

Nome parametro	Tipo	Dimensioni	Description
conntrack_allowance_exceeded	Gauge	instance_id, eni, nodo	Pacchetti persi a causa del limite di tracciamento della connessione
linklocal_allowance_exceeded	Gauge	instance_id, eni, nodo	I pacchetti sono caduti a causa del limite PPS del servizio proxy locale

Metriche del livello di flusso supportate

Nome parametro	Tipo	Description
Ritrasmissioni TCP	Contatore	Numero di volte in cui un mittente invia nuovamente un pacchetto perso o danneggiato durante la trasmissione.
Timeout di ritrasmissione TCP	Contatore	Numero di volte in cui un mittente ha avviato un periodo di attesa per determinare se un pacchetto è andato perso durante il transito.
Dati (byte) trasferiti	Contatore	Volume di dati trasferiti tra un'origine e una destinazione per un determinato flusso.

Mappa dei servizi e tabella dei flussi



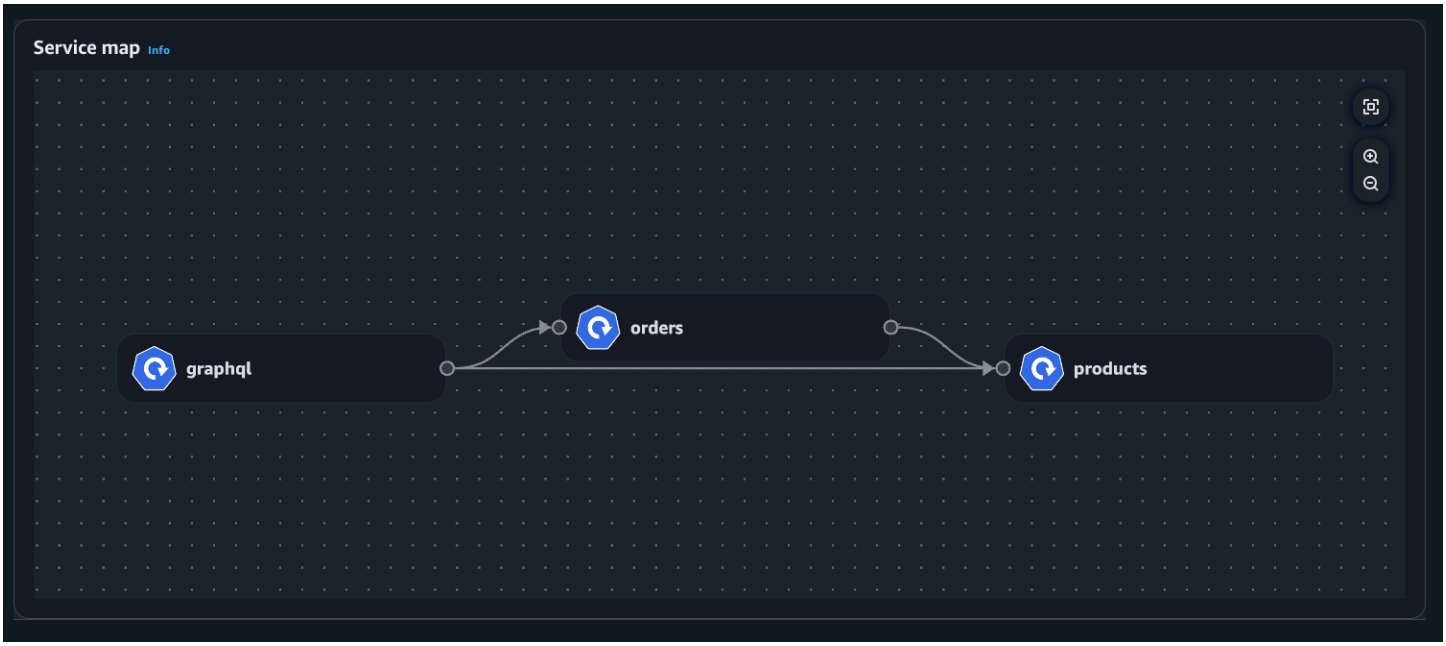
1. Una volta installato, l'agente Network Flow Monitor viene eseguito DaemonSet su ogni nodo di lavoro e raccoglie i primi 500 flussi di rete (in base al volume di dati trasferiti) ogni 30 secondi.

2. Questi flussi di rete sono suddivisi nelle seguenti categorie: Intra AZ, Inter AZ, EC2 → S3, → EC2 DynamoDB (DDB) e Unclassified. A ogni flusso sono associate 3 metriche: ritrasmissioni, timeout di ritrasmissione e dati trasferiti (in byte).
   - Intra AZ: flussi di rete tra pod nella stessa AZ
   - Inter AZ: la rete scorre tra pod diversi AZs
   - EC2 → S3: la rete scorre dai pod a S3
   - EC2 → DDB: flussi di rete dai pod al DDB
   - Non classificato: la rete scorre dai pod a Internet o in locale
3. I flussi di rete dell'API Network Flow Monitor Top Contributors vengono utilizzati per potenziare le seguenti esperienze nella console EKS:
   - Mappa dei servizi: visualizzazione dei flussi di rete all'interno del cluster (Intra AZ e Inter AZ).
   - Tabella dei flussi: presentazione in tabella dei flussi di rete all'interno del cluster (Intra AZ e Inter AZ), dai pod ai AWS servizi (EC2 → S3 e EC2 → DDB) e dai pod alle destinazioni esterne (non classificate).

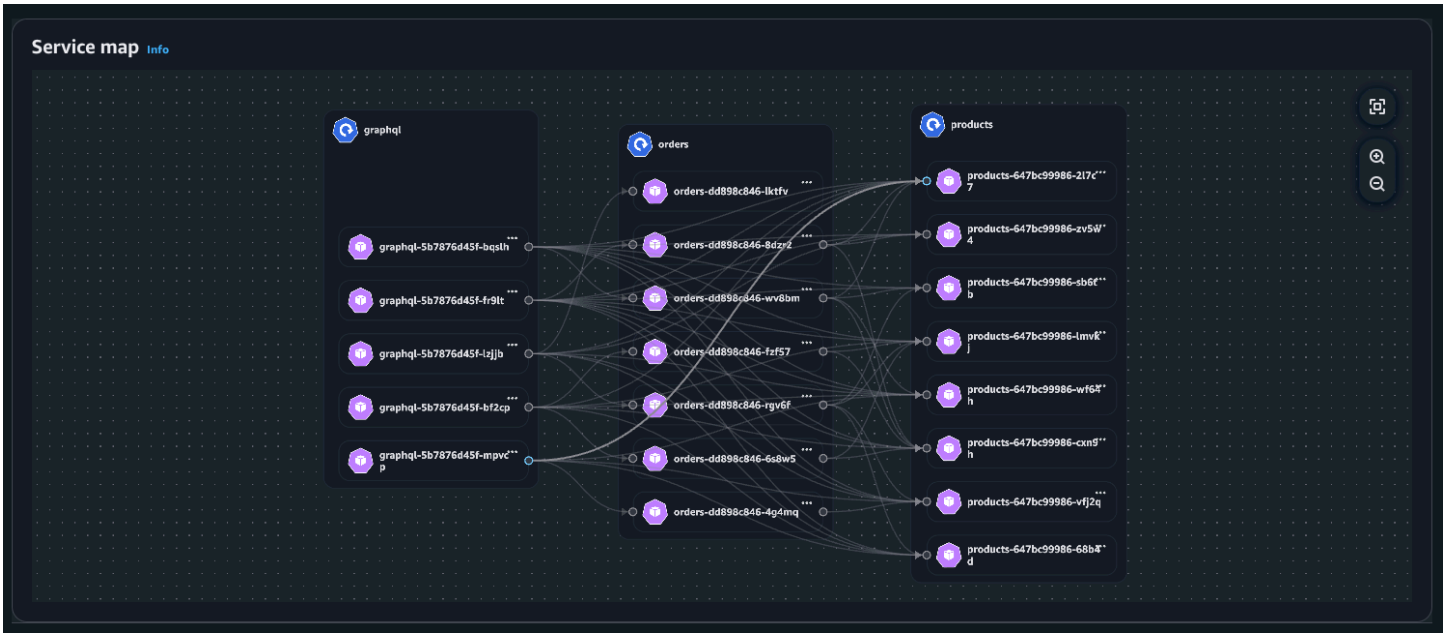
I flussi di rete estratti dall'API Top Contributors hanno un ambito temporale di 1 ora e possono includere fino a 500 flussi per categoria. Per quanto riguarda la mappa dei servizi, ciò significa che è possibile reperire e presentare fino a 1000 flussi dalle categorie di flusso Intra AZ e Inter AZ in un intervallo di tempo di 1 ora. Per quanto riguarda la tabella dei flussi, ciò significa che è possibile reperire e presentare fino a 3000 flussi di rete da tutte le 6 categorie di flussi di rete in un intervallo di tempo di 2 ore.

Esempio: mappa dei servizi

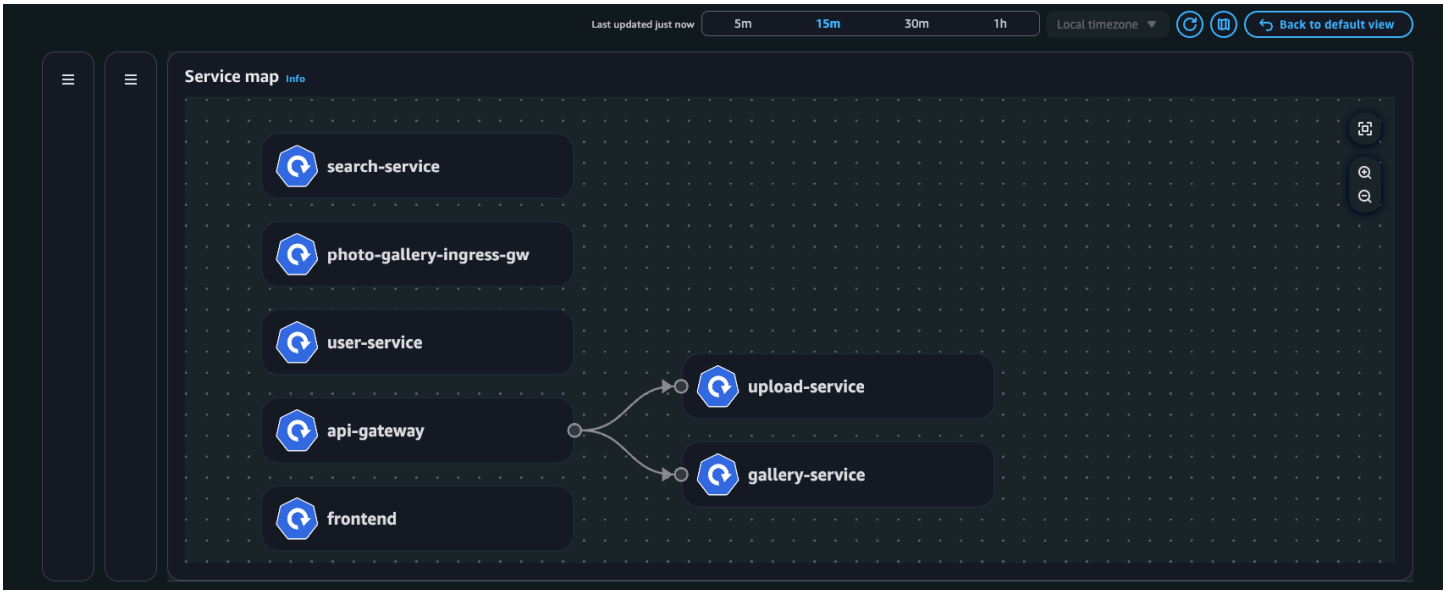
Visualizzazione della distribuzione



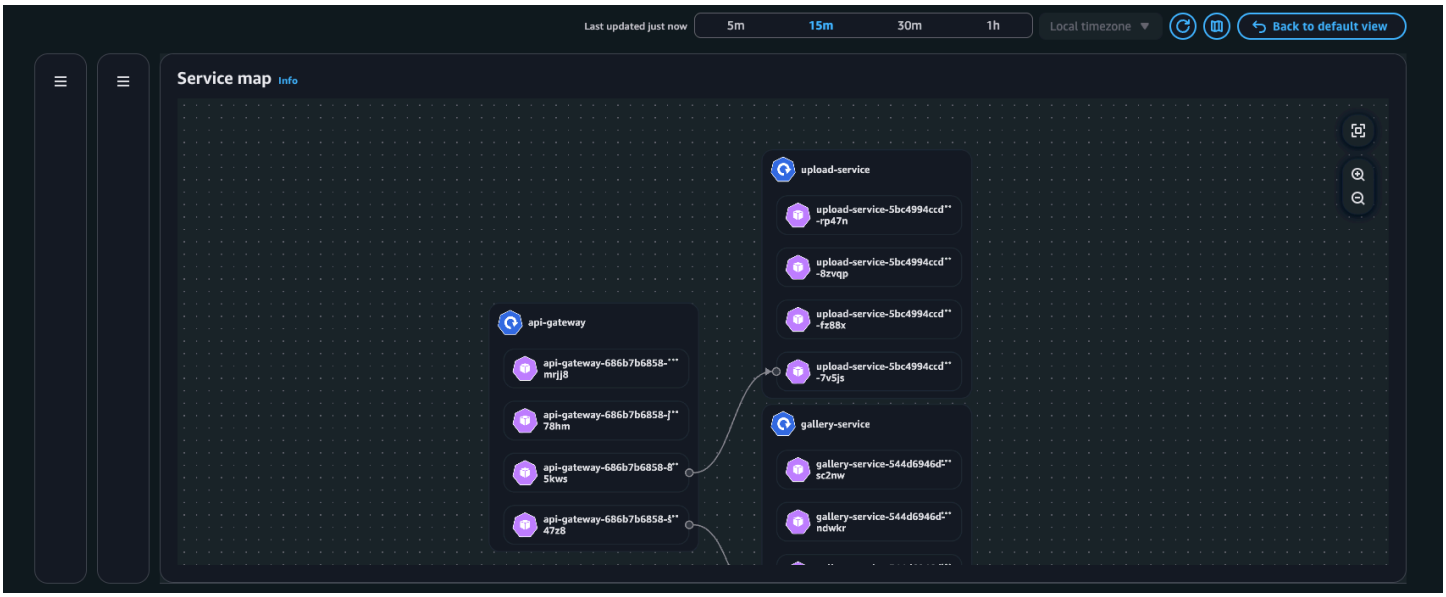
Vista Pod



Vista di distribuzione

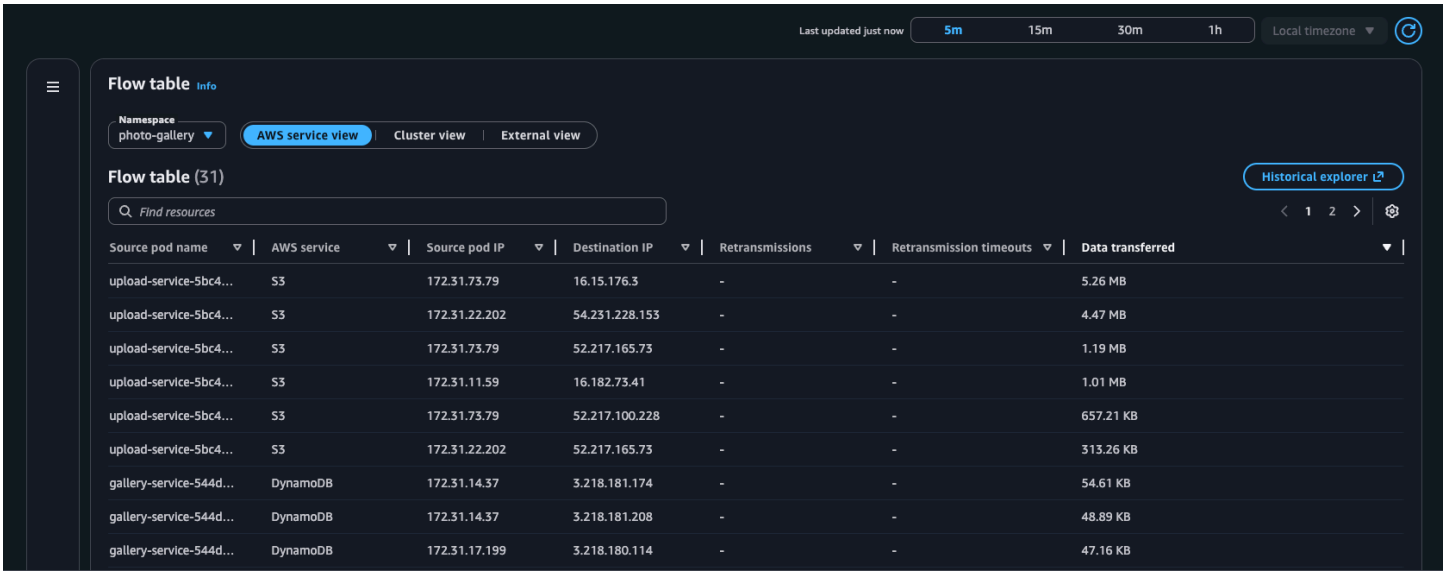


Vista Pod

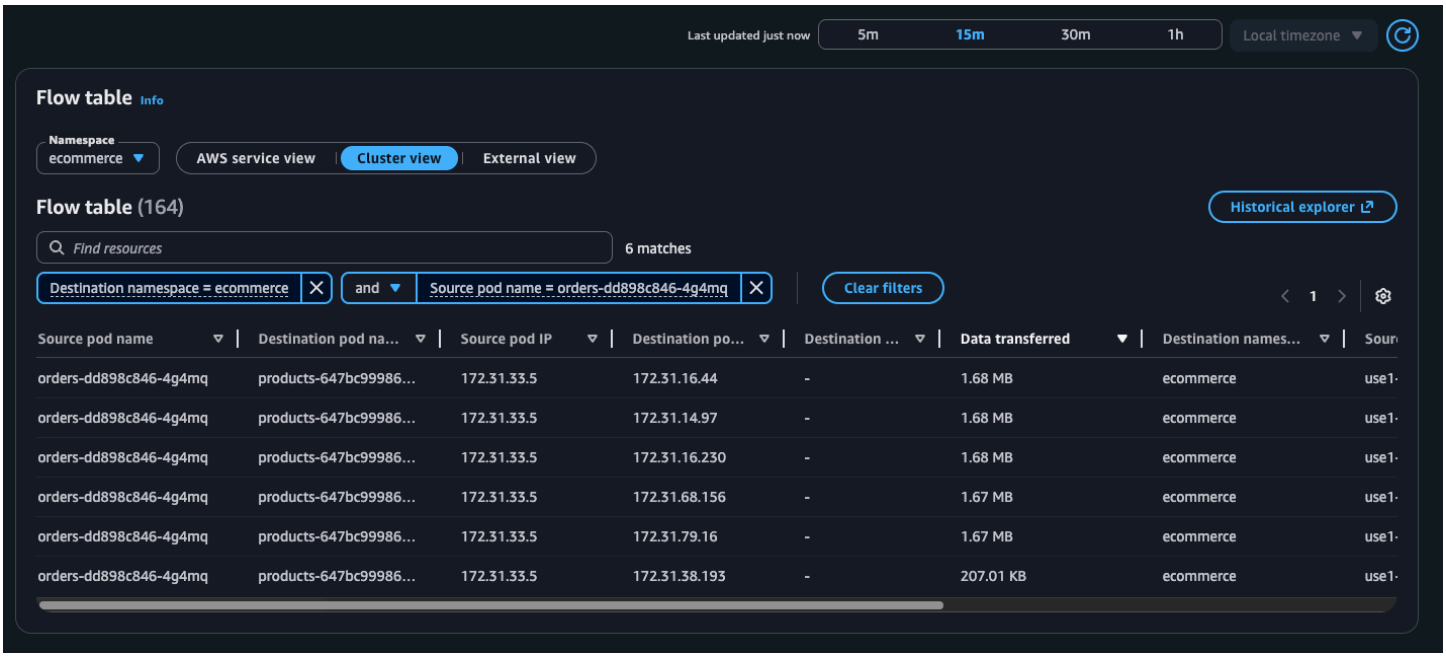


Esempio: tabella di flusso

AWS visualizzazione del servizio



Vista a grappolo



Considerazioni e limitazioni

- L'osservabilità della rete di contenitori in EKS è disponibile solo nelle regioni in cui è supportato Network Flow Monitor.
- Le metriche di sistema supportate sono in OpenMetrics formato e possono essere estratte direttamente dall'agente Network Flow Monitor (NFM).

- Per abilitare l'osservabilità della rete dei container in EKS utilizzando Infrastructure as Code (IaC) come Terraform, è necessario definire e creare queste dipendenze nelle configurazioni: ambito NFM, monitor NFM e agente NFM.
- Network Flow Monitor supporta fino a circa 5 milioni di flussi al minuto. Si tratta di circa 5.000 EC2 istanze (nodi di lavoro EKS) con l'agente Network Flow Monitor installato. L'installazione di agenti su più di 5000 istanze può influire sulle prestazioni di monitoraggio fino a quando non sarà disponibile capacità aggiuntiva.
- È necessario eseguire una versione minima di 1.1.0 per il componente aggiuntivo EKS dell'agente NFM.
- È necessario utilizzare la versione 6.21.0 o successiva di Terraform AWS Provider per il supporto delle risorse di Network Flow Monitor.
- Per arricchire i flussi di rete con i metadati dei pod, i pod devono funzionare nel proprio spazio dei nomi di rete isolato, non nello spazio dei nomi della rete host.

Monitoraggio delle metriche del cluster con Prometheus

Prometheus è un database di monitoraggio e di serie temporali che esegue lo scraping degli endpoint. Offre la possibilità di eseguire query, aggregare e archiviare i dati raccolti. È possibile utilizzarlo anche per gli avvisi e l'aggregazione degli avvisi. Questo argomento spiega come configurare Prometheus come un'opzione gestita oppure open source. Il monitoraggio dei parametri del piano di controllo di Amazon EKS è un caso d'uso comune.

Amazon Managed Service for Prometheus è un servizio di monitoraggio e avviso compatibile con Prometheus che semplifica il monitoraggio di applicazioni e infrastrutture containerizzate su larga scala. È un servizio completamente gestito che dimensiona automaticamente l'importazione, l'archiviazione, le query e gli avvisi dei parametri. Si integra anche con i servizi di sicurezza di AWS per consentire un accesso rapido e sicuro ai tuoi dati. È possibile utilizzare il linguaggio di query open source ProMQL per fare una query e creare avvisi relativi ai parametri. Inoltre, è possibile utilizzare l'alert manager nel servizio gestito da Amazon per Prometheus per configurare le regole di avviso per gli avvisi critici. Quindi, è possibile inviare questi avvisi critici come notifiche a un argomento Amazon SNS.

Esistono diverse opzioni per utilizzare Prometheus con Amazon EKS:

- È possibile attivare le metriche Prometheus quando si crea per la prima volta un cluster Amazon EKS o è possibile creare il proprio scraper Prometheus personalizzato per i cluster esistenti. Entrambe queste opzioni sono trattate in questo argomento.

- È possibile implementare Prometheus utilizzando Helm. Per ulteriori informazioni, consulta the section called "Implementare con Helm".
- È possibile visualizzare le metriche grezze del piano di controllo in formato Prometheus. Per ulteriori informazioni, consulta the section called "Piano di controllo (control-plane)".

Passaggio 1: attivazione delle metriche Prometheus

> **Important**
>
> Le risorse del servizio gestito da Amazon per Prometheus non rientrano nel ciclo di vita del cluster e devono essere gestite indipendentemente dal cluster. Quando si elimina il cluster, assicurarsi di eliminare anche tutti gli scraper applicabili per bloccare i relativi costi. Per ulteriori informazioni, consulta Trova ed elimina scraper nella Guida per l'utente del servizio gestito da Amazon per Prometheus.

Prometheus rileva e raccoglie i parametri dal cluster tramite un modello basato su pull chiamato scraping. Gli scraper sono configurati per raccogliere dati dall'infrastruttura del cluster e dalle applicazioni containerizzate. Quando attivi l'opzione di invio delle metriche Prometheus, il servizio gestito da Amazon per Prometheus fornisce uno scraper privo di agente completamente gestito.

Se non hai ancora creato il cluster, è possibile attivare l'opzione per inviare i le metriche a Prometheus al momento della creazione del cluster per la prima volta. Nella console Amazon EKS, questa opzione si trova nel passaggio Configura osservabilità della creazione di un nuovo cluster. Per ulteriori informazioni, consulta the section called "Creazione di un cluster ".

Se si dispone già di un cluster esistente, è possibile creare il proprio scraper Prometheus. Per eseguire questa operazione nella console Amazon EKS, accedi alla scheda Osservabilità del cluster e scegli il pulsante Aggiungi scraper. Se preferisci farlo con l'API AWS o AWS CLI, consulta Creare uno scraper nella Guida per l'utente del servizio gestito da Amazon per Prometheus.

Le seguenti opzioni sono disponibili quando si crea lo scraper con la console Amazon EKS.

Alias scraper

(Facoltativo) Inserisci un alias univoco per lo scraper.

Destinazione

Scegli un workspace Amazon Managed Service for Prometheus. un'area di lavoro è uno spazio logico dedicato all'archiviazione e all'interrogazione dei parametri di Prometheus. Con questo workspace, sarai in grado di visualizzare i le metriche Prometheus di tutti gli account che vi hanno accesso. L'opzione Crea nuovo workspace consente ad Amazon EKS di creare un workspace per tuo conto utilizzando l'alias del workspace che fornisci. Con l'opzione Seleziona workspace esistente, puoi selezionare un workspace esistente da un elenco a discesa. Per ulteriori informazioni sui workspace, consulta Gestione di workspace nella Guida per l'utente di Amazon Managed Service for Prometheus.

Accesso al servizio

Questa sezione riassume le autorizzazioni concesse per l'invio delle metriche Prometheus:

- Consentire ad Amazon Managed Service for Prometheus di descrivere il cluster Amazon EKS sottoposto a scraping
- Consentire la scrittura remota nel workspace gestito da Amazon per Prometheus

Se `AmazonManagedScraperRole` esiste già, viene utilizzato dallo scraper. Scegli il link `AmazonManagedScraperRole` per visualizzare i dettagli dell'autorizzazione. Se `AmazonManagedScraperRole` non esiste ancora, scegli il link Visualizza dettagli di autorizzazione per vedere le autorizzazioni specifiche che concedi inviando le metriche Prometheus.

Sottoreti

Modificare le sottoreti che sono ereditate dallo scraper. Se è necessario aggiungere un'opzione di sottoreti disattivata, torna al passaggio di creazione del cluster Specifica la rete.

Configurazione dello scraper

Modifica la configurazione dello scraper in formato YAML secondo le esigenze. A tale scopo, utilizza il modulo o carica un file YAML sostitutivo. Per ulteriori informazioni, consulta Configurazione dello scraper nella Guida per l'utente di Amazon Managed Service for Prometheus.

Amazon Managed Service for Prometheus si riferisce allo scraper agentless creato insieme al cluster come raccoglitore gestito da AWS. Per ulteriori informazioni sugli agenti di raccolta gestiti da AWS, consulta Imposta metriche con raccoglitori gestiti da AWS nella Guida per l'utente del servizio gestito da Amazon per Prometheus.

> **Important**
>
> - Se crei uno scraper Prometheus utilizzando AWS CLI o AWS API, è necessario modificarne la configurazione per fornire allo scraper le autorizzazioni interne al cluster. Per ulteriori informazioni, consulta Configurazione del cluster Amazon EKS nella Guida per l'utente di Amazon Managed Service for Prometheus.
> - Se disponi di uno scraper Prometheus creato prima dell'11 novembre 2024 che utilizza `aws-auth ConfigMap` anziché le voci di accesso, devi aggiornarlo per accedere a metriche aggiuntive dal piano di controllo del cluster Amazon EKS. Per la configurazione aggiornata, consulta Configurazione manuale di Amazon EKS per l'accesso allo scraper nella Guida per l'utente del servizio gestito da Amazon per Prometheus.

Passaggio 2: utilizzo delle metriche Prometheus

Per ulteriori informazioni su come utilizzare le metriche Prometheus dopo averli attivati per il cluster, consulta la Guida per l'utente del servizio gestito da Amazon per Prometheus.

Passaggio 3: gestione degli scraper Prometheus

Per gestire gli scraper, scegli la scheda Osservabilità nella console Amazon EKS. Una tabella mostra un elenco di scraper per il cluster, incluse informazioni come l'ID, l'alias, lo stato e la data di creazione dello scraper. È possibile aggiungere altri scraper, modificarli, eliminarli o visualizzare ulteriori informazioni sugli scraper correnti.

Per visualizzare ulteriori dettagli sullo scraper, scegli un link relativo all'ID dello scraper. Ad esempio, è possibile visualizzare ARN, ambiente, ID dell'area di lavoro, ruolo IAM, configurazione e informazioni sulla rete. È possibile utilizzare l'ID scraper come input per operazioni dell'API del servizio gestito da Amazon per Prometheus come `DescribeScraper`, `UpdateScraper` e `DeleteScraper`. Per ulteriori informazioni sull'uso dell'API Prometheus, consulta Riferimento delle API del servizio gestito da Amazon per Prometheus.

Implementare Prometheus utilizzando Helm

In alternativa all'utilizzo del Servizio gestito da Amazon per Prometheus, è possibile implementare Prometheus nel cluster con Helm. Se Helm è già stato installato, è possibile controllarne la versione con il comando `helm version`. Helm è un programma di gestione del pacchetto per cluster

Kubernetes. Per ulteriori informazioni su Helm e su come installarlo, consultare the section called "Implementazione di app con Helm".

Dopo aver configurato Helm per il cluster Amazon EKS, è possibile utilizzarlo per implementare Prometheus attraverso i seguenti passaggi.

1. Creare un namespace Prometheus.

```
kubectl create namespace prometheus
```

2. Aggiungere il repository del grafico `prometheus-community`.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

3. Implementare Prometheus.

```
helm upgrade -i prometheus prometheus-community/prometheus \
    --namespace prometheus \
    --set alertmanager.persistence.storageClass="gp2" \
    --set server.persistentVolume.storageClass="gp2"
```

> **Note**
>
> Se si riceve l'errore `Error: failed to download "stable/prometheus" (hint: running helm repo update may help)` durante l'esecuzione di questo comando, eseguire `helm repo update prometheus-community`, quindi provare a eseguire nuovamente il comando della fase 2.

Se si riceve l'errore `Error: rendered manifests contain a resource that already exists`, eseguire `helm uninstall` *`your-release-name`* `-n` *`namespace`* , quindi provare a eseguire nuovamente il comando della fase 3.

4. Verificare che tutti i pod nel namespace `prometheus` siano nello stato `READY`.

```
kubectl get pods -n prometheus
```

Di seguito viene riportato un output di esempio.

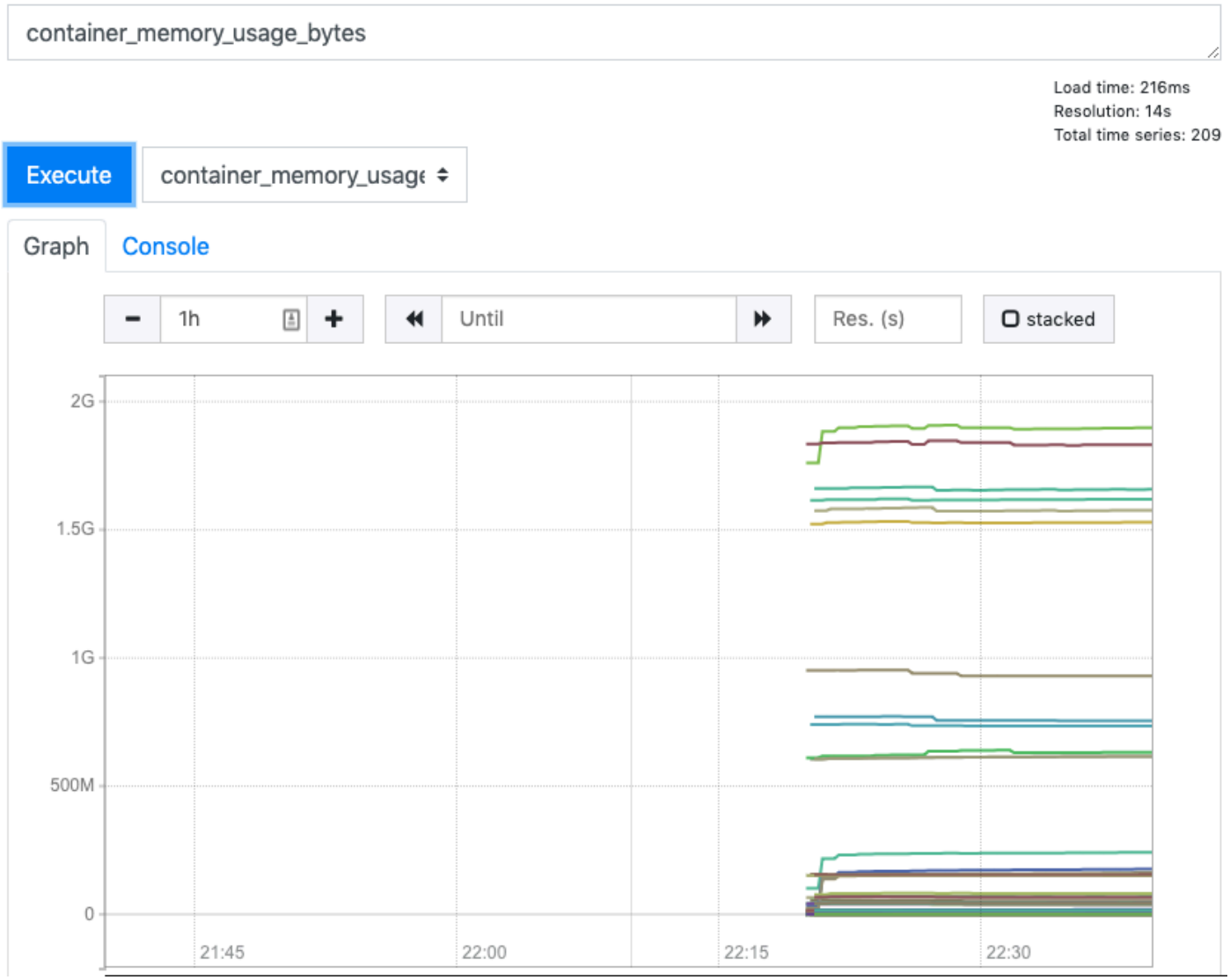
```
NAME                                             READY   STATUS    RESTARTS   AGE
```

```
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
```

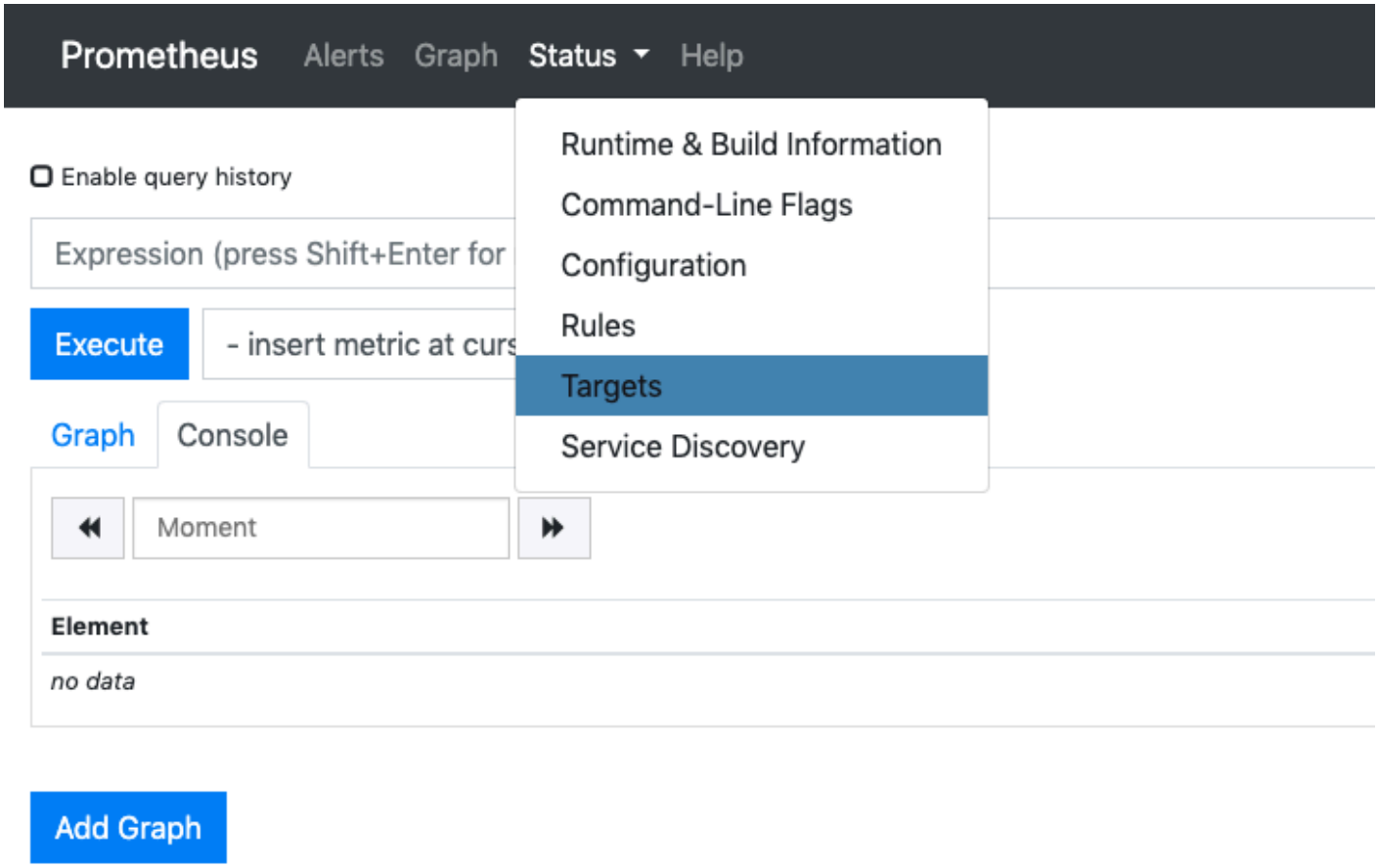
5. Utilizzare `kubectl` per eseguire l'inoltro della porta della console Prometheus al computer locale.

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. Indirizzare un browser web a `http://localhost:9090` per visualizzare la console Prometheus.
7. Scegliere un parametro dal menu - insert metric at cursor, quindi selezionare Execute (Esegui). Scegliere la scheda Graph (Grafico) per visualizzare il parametro nel tempo. L'immagine che segue mostra `container_memory_usage_bytes` nel tempo.



8. Dalla barra di navigazione superiore, scegliere Status (Stato), quindi Targets (Destinazioni).



Vengono visualizzati tutti gli endpoint Kubernetes che sono connessi a Prometheus utilizzando il rilevamento di servizi.

Recupero delle metriche grezze del piano di controllo in formato Prometheus

Il piano di controllo Kubernetes espone una serie di metriche che sono rappresentate in un formato Prometheus. Questi parametri sono utili per il monitoraggio e l'analisi. Sono esposti internamente tramite endpoint metrici ed è possibile accedervi senza dover implementare completamente Prometheus. Tuttavia, l'implementazione più semplice di Prometheus consente di analizzare le metriche nel tempo.

Per visualizzare il risultato delle metriche grezze, sostituisci `endpoint` ed esegui il comando riportato.

```
kubectl get --raw endpoint
```

Questo comando consente di inoltrare qualsiasi endpoint e restituisce la risposta non elaborata. I diversi parametri sono elencati per riga e ogni riga include un nome di parametro, tag e un valore.

```
metric_name{tag="value"[,...]} value
```

Recuperare le metriche dal server API

L'endpoint del server API generale è esposto sul piano di controllo Amazon EKS. Questo endpoint è utile principalmente per esaminare una metrica specifica.

```
kubectl get --raw /metrics
```

Di seguito viene riportato un output di esempio:

```
[...]
# 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
```

Questo output non elaborato restituisce in modo completo ciò che il server API espone.

Recupera le metriche del piano di controllo con **`metrics.eks.amazonaws.com`**

Per i cluster con versione Kubernetes `1.28` e successive, Amazon EKS espone anche i parametri relativi al gruppo di API `metrics.eks.amazonaws.com`. Queste metriche includono componenti del piano di controllo come `kube-scheduler` e `kube-controller-manager`.

> **Note**
>
> Se disponi di una configurazione webhook che potrebbe bloccare la creazione della nuova risorsa `APIService` `v1.metrics.eks.amazonaws.com` nel cluster, la funzionalità degli endpoint delle metriche potrebbe non essere disponibile. È possibile verificarlo nel log di verifica `kube-apiserver` cercando la parola chiave `v1.metrics.eks.amazonaws.com`.

Recuperare le metriche **`kube-scheduler`**

Utilizza il codice seguente per recuperare le metriche `kube-scheduler`.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics"
```

Di seguito viene riportato un output di esempio:

```
# 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
[...]
```

Recuperare le metriche **`kube-controller-manager`**

Utilizza il codice seguente per recuperare le metriche `kube-controller-manager`.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics"
```

Di seguito viene riportato un output di esempio:

```
[...]
```

```
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
[...]
```

Comprendere le metriche di gestione del sistema di pianificazione e del controller

La tabella seguente descrive le metriche di gestione del sistema di pianificazione e del controller rese disponibili per lo scraping in stile Prometheus. Per ulteriori informazioni relative a queste metriche, consulta Riferimento delle metriche Kubernetes nella documentazione Kubernetes.

Parametro	Componente del piano di controllo	Descrizione
scheduler_pending_pods	pianificatore	Il numero di pod in attesa di essere pianificati su un nodo per l'esecuzione.
scheduler_schedule_attempts_total	pianificatore	Il numero di tentativi effettuati per pianificare i pod.
scheduler_preemption_attempts_total	pianificatore	Il numero di tentativi effettuati dal sistema di pianificazione per i pod con priorità più alta eliminando quelli con priorità inferiore.
scheduler_preemption_victims	pianificatore	Il numero di pod che sono stati selezionati per l'espulsione per

Parametro	Componente del piano di controllo	Descrizione
		fare spazio ai pod con priorità più alta.
scheduler_pod_scheduling_attempts	pianificatore	Il numero di tentativi di pianificare correttamente un pod.
scheduler_scheduling_attempt_duration_seconds	pianificatore	Indica la velocità o la lentezza con cui lo strumento di pianificazione riesce a trovare un luogo adatto per l'esecuzione di un pod in base a vari fattori come la disponibilità delle risorse e le regole di pianificazione.
scheduler_pod_scheduling_sli_duration_seconds	pianificatore	La latenza end-to-end per un pod in fase di pianificazione, dal momento in cui il pod entra nella coda di pianificazione. Ciò potrebbe comportare più tentativi di pianificazione.
cronjob_controller_job_creation_skew_duration_seconds	gestore del controller	Il tempo che intercorre tra la pianificazione dell'esecuzione di un cronjob e la creazione del processo corrispondente.
workqueue_depth	gestore del controller	La profondità corrente della coda.
workqueue_adds_total	gestore del controller	Il numero totale di aggiunte gestite dalla coda di lavoro.

Parametro	Componente del piano di controllo	Descrizione
workqueue_queue_duration_seconds	gestore del controller	Il tempo in secondi in cui un elemento rimane nella coda di lavoro prima di essere richiesto.
workqueue_work_duration_seconds	gestore del controller	Il tempo in secondi impiegato per l'elaborazione di un elemento dalla coda di lavoro.

Implementare uno scraper Prometheus per acquisire metriche in modo coerente

Per implementare uno scraper Prometheus per eseguire lo scraping delle metriche in modo coerente, utilizza la seguente configurazione:

```
---
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
```

L'autorizzazione che segue è richiesta per consentire al pod di accedere al nuovo endpoint metrico.

```
{
  "effect": "allow",
  "apiGroups": [
    "metrics.eks.amazonaws.com"
  ],
  "resources": [
    "kcm/metrics",
    "ksh/metrics"
  ],
  "verbs": [
    "get"
  ] },
```

Per applicare una patch al ruolo, puoi utilizzare il seguente comando.

```
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"]
```

```
    }
  }
]'
```

Quindi puoi visualizzare la dashboard di Prometheus effettuando il proxy della porta dello scraper Prometheus verso la porta locale.

```
kubectl port-forward pods/prom-pod 9090:9090
```

Per il tuo cluster Amazon EKS, le principali metriche del piano di controllo Kubernetes sono inserite anche nelle metriche di Amazon CloudWatch sotto il namespace `AWS/EKS`. Per visualizzarli, apri la console CloudWatch e seleziona Tutte le metriche dal riquadro di navigazione a sinistra. Nella pagina di selezione Metriche, scegli il namespace `AWS/EKS` e una dimensione delle metriche per il tuo cluster.

Monitora i dati del cluster con Amazon CloudWatch

Amazon CloudWatch è un servizio di monitoraggio che raccoglie metriche e log dalle tue risorse cloud. CloudWatch fornisce gratuitamente alcuni parametri di base di Amazon EKS quando si utilizza un nuovo cluster di versione `1.28` o superiore. Tuttavia, quando si utilizza CloudWatch Observability Operator come componente aggiuntivo di Amazon EKS, è possibile ottenere funzionalità di osservabilità avanzate.

Metriche di base in Amazon CloudWatch

Per i cluster con versione Kubernetes `1.28` e successive, ottieni metriche CloudWatch vendute gratuitamente nel namespace. `AWS/EKS` La seguente tabella fornisce un elenco delle metriche di base disponibili per le versioni supportate. Ogni metrica presente nell'elenco ha una frequenza di un minuto.

Nome parametro	Description
`apiserver_flowcontrol_current_executing_seats`	Il numero di postazioni attualmente in uso per l'esecuzione delle richieste API. L'allocazione dei posti è determinata dalla configurazione priority_level e flow_schema nella funzionalità Priority and Fairness dell'API Kubernetes.

Nome parametro	Description
	Unità: numero Statistiche valide: somma
`scheduler_schedule_attempts_total`	Il numero totale di tentativi da parte pianificatore di pianificare i pod nel cluster per un determinato periodo. Questa metrica aiuta a monitorare il carico di lavoro del pianificatore e può indicare la pressione della pianificazione o potenziali problemi con il posizionamento dei pod. Unità: numero Statistiche valide: somma
`scheduler_schedule_attempts_SCHEDULED`	Il numero di tentativi riusciti da parte pianificatore di pianificare i pod sui nodi nel cluster per un determinato periodo. Unità: numero Statistiche valide: somma
`scheduler_schedule_attempts_UNSCHEDULABLE`	Il numero di tentativi di pianificazione dei pod che non è stato possibile pianificare per un determinato periodo a causa di vincoli validi, quali CPU o memoria insufficienti su un nodo. Unità: numero Statistiche valide: somma

Nome parametro	Description
scheduler_schedule_attempts_ERROR	Il numero di tentativi di pianificazione dei pod che non sono riusciti in un determinato periodo a causa di un problema interno del pianificatore stesso, ad esempio problemi di connettività del server API. Unità: numero Statistiche valide: somma
scheduler_pending_pods	Il numero totale di pod in sospeso che il pianificatore del cluster deve pianificare per un determinato periodo. Unità: numero Statistiche valide: somma
scheduler_pending_pods_ACTIVEQ	Il numero di pod in sospeso in ActiveQ che attendono di essere pianificati nel cluster in un determinato periodo. Unità: numero Statistiche valide: somma
scheduler_pending_pods_UNSCHEDULABLE	Il numero di pod in sospeso che il pianificatore ha tentato di pianificare senza successo e che vengono mantenuti in uno stato non pianificabile per un nuovo tentativo. Unità: numero Statistiche valide: somma

Nome parametro	Description
`scheduler_pending_pods_BACKOFF`	Il numero di Pod in sospeso in `backoffQ` in uno stato di backoff che sono in attesa della scadenza del periodo di backoff. Unità: numero Statistiche valide: somma
`scheduler_pending_pods_GATED`	Il numero di pod in sospeso che sono attualmente in attesa in uno stato chiuso, in quanto non possono essere pianificati finché non soddisfano le condizioni richieste. Unità: numero Statistiche valide: somma
`apiserver_request_total`	Il numero di richieste HTTP inviate su tutti i server API del cluster. Unità: numero Statistiche valide: somma
`apiserver_request_total_4XX`	Il numero di richieste HTTP effettuate a tutti i server API del cluster che hanno generato codici di stato `4XX` (errore client). Unità: numero Statistiche valide: somma

Nome parametro	Description
apiserver_request_total_429	Il numero di richieste HTTP inviate a tutti i server API del cluster che hanno generato il codice di stato 429, che si verifica quando i client superano le soglie di limitazione della velocità. Unità: numero Statistiche valide: somma
apiserver_request_total_5XX	Il numero di richieste HTTP effettuate a tutti i server API del cluster che hanno generato codici di stato 5XX (errore server). Unità: numero Statistiche valide: somma
apiserver_request_total_LIST_PODS	Il numero di richieste Pod LIST inviate a tutti i server API del cluster. Unità: numero Statistiche valide: somma
apiserver_request_duration_seconds_PUT_P99	Il 99° percentile della latenza per le richieste PUT calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste PUT. Unità: secondi Statistiche valide: media

Nome parametro	Description
apiserver_request_duration_seconds_PATCH_P99	Il 99° percentile della latenza per le richieste PATCH calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste PATCH. Unità: secondi Statistiche valide: media
apiserver_request_duration_seconds_POST_P99	Il 99° percentile della latenza per le richieste POST calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste POST. Unità: secondi Statistiche valide: media
apiserver_request_duration_seconds_GET_P99	Il 99° percentile della latenza per le richieste GET calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste GET. Unità: secondi Statistiche valide: media

Nome parametro	Description
`apiserver_request_duration_seconds_LIST_P99`	Il 99° percentile della latenza per le richieste `LIST` calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste `LIST`. Unità: secondi Statistiche valide: media
`apiserver_request_duration_seconds_DELETE_P99`	Il 99° percentile della latenza per le richieste `DELETE` calcolato da tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste `DELETE`. Unità: secondi Statistiche valide: media
`apiserver_current_inflight_requests_MUTATING`	Il numero di richieste di mutazione (`POST`, `PUT`, `DELETE`, `PATCH`) attualmente in elaborazione su tutti i server API del cluster. Questa metrica rappresenta le richieste in corso per le quali l'elaborazione non è ancora stata completata. Unità: numero Statistiche valide: somma

Nome parametro	Description
`apiserver_current_inflight_requests_READONLY`	Il numero di richieste di sola lettura (`GET`, `LIST`) attualmente in elaborazione su tutti i server API del cluster. Questa metrica rappresenta le richieste in corso per le quali l'elaborazione non è ancora stata completata. Unità: numero Statistiche valide: somma
`apiserver_admission_webhook_request_total`	Il numero di richieste webhook di ammissione inviate su tutti i server API nel cluster. Unità: numero Statistiche valide: somma
`apiserver_admission_webhook_request_total_ADMIT`	Il numero di richieste di mutazione webhook di ammissione inviate su tutti i server API nel cluster. Unità: numero Statistiche valide: somma
`apiserver_admission_webhook_request_total_VALIDATING`	Il numero di richieste webhook di ammission e di convalida inviate su tutti i server API nel cluster. Unità: numero Statistiche valide: somma

Nome parametro	Description
`apiserver_admission_webhook_rejection_count`	Il numero di richieste webhook di ammissione inviate su tutti i server API del cluster che sono state rifiutate. Unità: numero Statistiche valide: somma
`apiserver_admission_webhook_rejection_count_ADMIT`	Il numero di richieste di mutazione webhook di ammissione inviate su tutti i server API del cluster che sono state rifiutate. Unità: numero Statistiche valide: somma
`apiserver_admission_webhook_rejection_count_VALIDATING`	Il numero di richieste webhook di ammission e di convalida inviate su tutti i server API del cluster che sono state rifiutate. Unità: numero Statistiche valide: somma
`apiserver_admission_webhook_admission_duration_seconds`	Il 99° percentile della latenza per le richieste webhook di ammissione di terze parti calcolato sulla base di tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste webhook di ammission e di terze parti. Unità: secondi Statistiche valide: media

Nome parametro	Description
`apiserver_admission_webhook_admission_duration_seconds_ADMIT_P99`	Il 99° percentile della latenza per le richieste webhook di mutazione di ammissione di terze parti calcolato sulla base di tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste webhook di mutazione di ammissione di terze parti. Unità: secondi Statistiche valide: media
`apiserver_admission_webhook_admission_duration_seconds_VALIDATING_P99`	Il 99° percentile della latenza per le richieste webhook di validazione di ammissione di terze parti calcolato sulla base di tutte le richieste su tutti i server API nel cluster. Rappresenta il tempo di risposta al di sotto del quale viene completato il 99% di tutte le richieste webhook di validazione di ammissione di terze parti. Unità: secondi Statistiche valide: media
`apiserver_storage_size_bytes`	La dimensione fisica in byte del file del database di storage etcd usato dai server API del cluster. Questa metrica rappresenta lo spazio su disco effettivo allocato per lo storage. Unità: byte Statistiche valide: massimo

Operatore di CloudWatch osservabilità di Amazon

Amazon CloudWatch Observability raccoglie log, parametri e dati di tracciamento in tempo reale. Li invia ad Amazon CloudWatch e AWS X-Ray. Puoi installare questo componente aggiuntivo per abilitare sia CloudWatch Application Signals che CloudWatch Container Insights con una migliore osservabilità per Amazon EKS. In questo modo puoi monitorare lo stato e le prestazioni dell'infrastruttura e delle applicazioni containerizzate. Amazon CloudWatch Observability Operator è progettato per installare e configurare i componenti necessari.

Amazon EKS supporta CloudWatch Observability Operator come componente aggiuntivo di Amazon EKS. Il componente aggiuntivo consente l'uso di Container Insights sui nodi di lavoro Linux e Windows nel cluster. Per abilitare Container Insights su Windows, la versione del componente aggiuntivo Amazon EKS deve essere `1.5.0` o superiore. Attualmente, CloudWatch Application Signals non è supportato su Amazon EKS Windows.

Gli argomenti seguenti descrivono come iniziare a utilizzare CloudWatch Observability Operator per il tuo cluster Amazon EKS.

- Per istruzioni sull'installazione di questo componente aggiuntivo, consulta Installa l' CloudWatch agente con il componente aggiuntivo Amazon CloudWatch Observability EKS o il grafico Helm nella Amazon User Guide. CloudWatch
- Per ulteriori informazioni su CloudWatch Application Signals, consulta Application Signals nella Amazon CloudWatch User Guide.
- Per ulteriori informazioni su Container Insights, consulta Using Container Insights nella Amazon CloudWatch User Guide.

Invia i registri del piano di controllo ai CloudWatch registri

La registrazione del piano di controllo di Amazon EKS fornisce log di audit e diagnostica direttamente dal piano di controllo di Amazon CloudWatch EKS ai log del tuo account. Questi log consentono di semplificare la protezione e l'esecuzione dei cluster. Puoi selezionare i tipi di log esatti di cui hai bisogno e i log vengono inviati come flussi di log a un gruppo per ogni cluster Amazon EKS in cui opera. CloudWatch Puoi utilizzare i filtri di CloudWatch abbonamento per eseguire analisi in tempo reale sui log o per inoltrarli ad altri servizi (i log saranno codificati in Base64 e compressi con il formato gzip). Per ulteriori informazioni, consulta Amazon CloudWatch logging.

> **Note**
>
> I log del piano di controllo di Amazon EKS vengono consegnati a CloudWatch Logs in pochi minuti. Tuttavia, la consegna dei log è la soluzione migliore.

É possibile iniziare a utilizzare il piano di controllo Amazon EKS scegliendo quali tipi di log che si desidera abilitare per ogni cluster Amazon EKS nuovo o esistente. Puoi abilitare o disabilitare ogni tipo di registro per cluster utilizzando la Console di gestione AWS AWS CLI (versione `1.16.139` o successiva) o tramite l'API Amazon EKS. Se abilitato, i log vengono inviati automaticamente dal cluster Amazon CloudWatch EKS ai log nello stesso account.

Quando si utilizza la registrazione del piano di controllo Amazon EKS, viene addebitato il prezzo Amazon EKS standard per ogni cluster eseguito. Ti vengono addebitati i costi standard di inserimento e archiviazione dei dati di CloudWatch Logs per tutti i log inviati a Logs dai tuoi cluster. CloudWatch Ti viene inoltre addebitato il costo di tutte AWS le risorse, come le istanze Amazon EC2 o i volumi Amazon EBS, di cui effettui il provisioning come parte del cluster.

Sono disponibili i seguenti tipi di registro del piano di controllo del cluster. Ogni tipo di registro corrisponde a un componente del piano di controllo Kubernetes. Per ulteriori informazioni su questi componenti, consultare Kubernetes Components nella documentazione Kubernetes.

Server API**`api`** ()

Il server API del cluster è il componente del piano di controllo (control-plane) che espone l'API Kubernetes. Se abiliti i log del server API quando avvii il cluster o poco dopo, questi log includono i flag del server API utilizzati per l'avvio del server API. Per ulteriori informazioni, consulta kube-apiserver e la policy di audit nella documentazione Kubernetes.

Audit**`audit`** ()

I log di audit Kubernetes forniscono un record dei singoli utenti, amministratori o componenti di sistema che interessano il cluster. Per ulteriori informazioni, consultare Auditing nella documentazione Kubernetes.

Autenticatore**`authenticator`** ()

I log dell'autenticatore sono univoci per Amazon EKS. Questi registri rappresentano il componente del piano di controllo utilizzato da Amazon EKS per l'autenticazione Role Based Access Control (RBAC) Kubernetes utilizzando credenziali IAM. Per ulteriori informazioni, consulta Gestione dei cluster.

Gestore controller**`controllerManager`** ()

Il gestore controller gestisce i loop di controllo di base inviati con Kubernetes. Per ulteriori informazioni, consultare kube-controller-manager nella documentazione Kubernetes.

Pianificatore**`scheduler`** ()

Il componente pianificatore gestisce quando e dove eseguire i pod nel cluster. Per ulteriori informazioni, consultare kube-scheduler nella documentazione Kubernetes.

Abilitare o disabilitare i log del piano di controllo (control-plane)

Per impostazione predefinita, i log del piano di controllo del cluster non vengono inviati a Logs. CloudWatch È necessario abilitare ogni tipo di registro singolarmente per inviare i log per il cluster. CloudWatch Le tariffe di inserimento dei log, archiviazione, archiviazione e scansione dei dati si applicano ai log del piano di controllo abilitati. Per ulteriori informazioni, consultare Prezzi di CloudWatch .

Per aggiornare la configurazione di registrazione del piano di controllo (control-plane), Amazon EKS richiede fino a cinque indirizzi IP disponibili in ciascuna sottorete. Quando abiliti un tipo di registro, i log vengono inviati con un livello di dettaglio dei log di 2.

È possibile abilitare o disabilitare i log del piano di controllo (control-plane) con la Console di gestione AWS o AWS CLI.

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegliere il nome del cluster per visualizzare le informazioni sul cluster.
3. Scegli la scheda Osservabilità.
4. Nella sezione Registrazione del piano di controllo, scegli Gestisci registrazione.
5. Per ogni singolo tipo di log, scegli se il tipo di log deve essere attivato o disattivato. Per impostazione predefinita, i tipi di log sono disattivati.
6. Scegliere Salva le modifiche per terminare.

AWS CLI

1. Controlla la tua versione AWS CLI con il seguente comando.

```
aws --version
```

Se la tua versione AWS CLI è precedente alla`1.16.139`, devi prima eseguire l'aggiornamento alla versione più recente. Per installare o aggiornare la AWS CLI, consulta Installazione dell'interfaccia a riga di AWS comando nella Guida per l'utente dell'interfaccia a riga di AWS comando.

2. Aggiorna la configurazione di esportazione dei log del piano di controllo del cluster con il seguente comando AWS CLI. Sostituisci *`my-cluster`* con il nome del cluster e specifica i valori desiderati per l'accesso all'endpoint.

> **Note**
>
> Il comando seguente invia tutti i tipi di log disponibili a CloudWatch Logs.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

Di seguito viene riportato un output di esempio.

```
{
    "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. Monitorare lo stato di aggiornamento della configurazione del log con il comando seguente utilizzando il nome del cluster e l'ID di aggiornamento restituiti dal comando precedente. L'aggiornamento è completo quando lo stato viene visualizzato come `Successful`.

```
aws eks describe-update \
    --region region-code\
    --name my-cluster \
    --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9
```

Di seguito viene riportato un output di esempio.

```
{
    "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": []
    }
}
```

Visualizzare i log del piano di controllo (control-plane) del cluster

Dopo aver abilitato uno qualsiasi dei tipi di log del piano di controllo per il tuo cluster Amazon EKS, puoi visualizzarli sulla CloudWatch console.

Per ulteriori informazioni sulla visualizzazione, l'analisi e la gestione dei log in CloudWatch, consulta la Amazon CloudWatch Logs User Guide.

1. Apri la CloudWatch console. Questo link apre la console e mostra i gruppi di log disponibili correnti e li filtra con il prefisso `/aws/eks`.
2. Scegliere il cluster per il quale si intende visualizzare i log. Il formato del nome del gruppo di log è `/aws/eks/`*`my-cluster`*`/cluster`.

3. Scegliere il flusso di log da visualizzare. L'elenco seguente descrive il formato del nome del flusso di log per ogni tipo di registro.

> **Note**
>
> Al crescere della quantità di dati del flusso di log, i nomi del flusso di log vengono ruotati. Quando esistono più flussi di log per un determinato tipo di log, puoi visualizzare l'ultimo flusso di log cercando il nome del flusso di log con Last event time (Ora ultimo evento) più recente.

- Log del componente server API Kubernetes (**`api`**) – `kube-apiserver-`*`1234567890abcdef01234567890abcde`*
- Audit (**`audit`**) – `kube-apiserver-audit-`*`1234567890abcdef01234567890abcde`*
- Autenticatore (**`authenticator`**) – `authenticator-`*`1234567890abcdef01234567890abcde`*
- Gestore controller (**`controllerManager`**) – `kube-controller-manager-`*`1234567890abcdef01234567890abcde`*
- Pianificatore (**`scheduler`**) – `kube-scheduler-`*`1234567890abcdef01234567890abcde`*

4. Esamina gli eventi del flusso di log.

   Ad esempio, dovresti vedere i flag iniziali del server API del cluster quando visualizzi la parte iniziale di `kube-apiserver-`*`1234567890abcdef01234567890abcde`* .

> **Note**
>
> Se i log del server API non vengono visualizzati all'inizio del flusso di log, è probabile che il file di log del server API sia stato ruotato sul server prima di abilitare la registrazione di log del server API sul server. I file di log che vengono ruotati prima dell'attivazione della registrazione del server API non possono essere esportati in. CloudWatch

Tuttavia, è possibile creare un nuovo cluster con la stessa versione di Kubernetes e abilitare la registrazione del server API quando si crea il cluster. I cluster con la stessa versione della piattaforma hanno gli stessi flag abilitati, quindi i flag devono corrispondere a quelli del nuovo cluster. Al termine

della visualizzazione dei flag relativi al nuovo cluster CloudWatch, è possibile eliminare il nuovo cluster.

Registra le chiamate API come eventi AWS CloudTrail

Amazon EKS è integrato con AWS CloudTrail. CloudTrail è un servizio che fornisce un registro delle operazioni eseguite da un utente, un ruolo o un servizio AWS in Amazon EKS. CloudTrail acquisisce tutte le chiamate API per Amazon EKS come eventi. Questi includono le chiamate della console Amazon EKS e le chiamate del codice alle operazioni API Amazon EKS.

Se crei un percorso, puoi abilitare la distribuzione continua di eventi CloudTrail in un bucket Amazon S3, inclusi gli eventi per Amazon EKS. Se non configuri un trail, puoi comunque visualizzare gli eventi più recenti nella console di CloudTrail in Cronologia eventi. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare diversi dettagli su una richiesta. Ad esempio, puoi determinare quando è stata effettuata una richiesta ad Amazon EKS, l'indirizzo IP da cui è stata eseguita e l'autore.

Per ulteriori informazioni su CloudTrail, consulta AWS CloudTrail User Guide.

Argomenti

- Visualizzazione dei riferimenti utili per AWS CloudTrail
- Analisi delle voci del file di log AWS CloudTrail
- Visualizzare i parametri per i gruppi Amazon EC2 Auto Scaling

Visualizzazione dei riferimenti utili per AWS CloudTrail

Al momento della creazione del tuo account AWS, CloudTrail è abilitato sul tuo account AWS. Quando si verifica un'attività in Amazon EKS, questa viene registrata in un evento CloudTrail insieme ad altri eventi di servizio AWS nella Cronologia eventi. È possibile visualizzare, cercare e scaricare gli eventi recenti nell'account AWS. Per ulteriori informazioni, consulta Visualizzazione di eventi mediante la cronologia eventi di CloudTrail.

Per una registrazione continua degli eventi nell'account AWS, inclusi gli eventi per Amazon EKS, creare un percorso. Un percorso consente a CloudTrail di distribuire i file di log in un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le Regioni AWS. Il percorso registra gli eventi da tutte le regioni AWSnella partizione AWS e distribuisce i file di log nel bucket Simple Storage Service (Amazon S3) specificato. Inoltre, è possibile configurare

altri servizi AWS per analizzare con maggiore dettaglio e utilizzare i dati evento raccolti nei log CloudTrail. Per ulteriori informazioni, consulta le risorse seguenti.

- Panoramica della creazione di un percorso
- Servizi e integrazioni CloudTrail supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di registro CloudTrail da più Regioni e Ricezione di file di log CloudTrail da più account

Tutte le operazioni Amazon EKS vengono registrate da CloudTrail e sono documentate nella Documentazione di riferimento delle API di Amazon EKS. Ad esempio, le chiamate alle sezioni CreateCluster, ListClusters e DeleteCluster generano voci nei file di log CloudTrail.

Ogni evento o voce di log include informazioni sul tipo di identità IAM che ha effettuato la richiesta e sulle credenziali utilizzate. Se sono state utilizzate credenziali temporanee, la voce mostra il modo in cui tali credenziali sono state ottenute.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

Analisi delle voci del file di log AWS CloudTrail

Un trail è una configurazione che consente la distribuzione di eventi come i file di log in un bucket Amazon S3 specificato. I file di log di CloudTrail possono contenere una o più voci di log. Un evento rappresenta una singola richiesta da un'origine e include informazioni sull'operazione richiesta, come data e ora dell'operazione e i parametri della richiesta. I file di log CloudTrail non sono una traccia di stack ordinata delle chiamate API pubbliche e di conseguenza non devono apparire in base a un ordine specifico.

L'esempio seguente mostra una voce di registro di CloudTrail che illustra l'operazione `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"
```

```
}
```

Voci di registro per ruoli collegati ai servizi Amazon EKS

I ruoli collegati ai servizi Amazon EKS effettuano chiamate API alle risorse AWS. Verranno visualizzate le voci di registro CloudTrail con `username: AWSServiceRoleForAmazonEKS` e `username: AWSServiceRoleForAmazonEKSNodegroup` per chiamate effettuate dai ruoli collegati ai servizi Amazon EKS. Per ulteriori informazioni su Amazon EKS e sui ruoli collegati ai servizi, consultare the section called "Ruoli collegati ai servizi".

Nell'esempio seguente è illustrata una voce di log CloudTrail che illustra un'operazione `DeleteInstanceProfile` eseguita dal ruolo collegato ai servizi `AWSServiceRoleForAmazonEKSNodegroup`, annotato in `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"
}
```

Visualizzare i parametri per i gruppi Amazon EC2 Auto Scaling

Nei gruppi di nodi gestiti da Amazon EKS i parametri dei gruppi di Amazon EC2 Auto Scaling sono abilitati per impostazione predefinita senza costi aggiuntivi. Il gruppo Auto Scaling invia dati campionati ad Amazon CloudWatch ogni minuto. Questi parametri possono essere perfezionati in base al nome dei gruppi Auto Scaling. Con questi parametri, si ottiene una visibilità quasi continua nella cronologia del gruppo Auto Scaling che alimenta i gruppi di nodi gestiti, ad esempio le modifiche delle dimensioni del gruppo nel tempo. I parametri di un gruppo Auto Scaling sono disponibili nella console di Amazon CloudWatch o in quella di Auto Scaling. Per ulteriori informazioni, consultare la pagina Monitor CloudWatch metrics for your Auto Scaling groups and instances.

Abilitando la raccolta dei parametri di un gruppo Auto Scaling, sarà possibile monitorare la scalabilità dei gruppi di nodi gestiti. I parametri di un gruppo con scalabilità automatica riportano la dimensione minima, massima e desiderata di un gruppo con scalabilità automatica. Puoi creare un allarme se il numero di nodi in un gruppo di nodi scende al di sotto della dimensione minima, fatto che indicherebbe un gruppo di nodi non integro. Il monitoraggio delle dimensioni dei gruppi di nodi è utile anche per regolare il numero massimo in modo che il piano dati non esaurisca la capacità.

Se si preferisce non raccogliere questi parametri, è possibile disabilitarli tutti o solo alcuni. Ad esempio, è possibile farlo per evitare il rumore nelle dashboard CloudWatch. Per ulteriori informazioni, consultare la pagina Amazon CloudWatch metrics for Amazon EC2 Auto Scaling.

Invio di dati metrici e di tracciamento con l'operatore ADOT

Amazon EKS supporta ora l'utilizzo di Console di gestione AWS, AWS CLI e l'API di Amazon EKS per installare e gestire l'operatore AWS Distro for OpenTelemetry (ADOT). Ciò rende più semplice

la possibilità per le applicazioni in esecuzione su Amazon EKS di inviare parametri e dati di traccia a vari servizi di monitoraggio, ad esempio Amazon CloudWatch, Prometheus e X-Ray.

Per ulteriori informazioni, consulta Getting Started with AWS Distro for OpenTelemetry using EKS Add-Ons nella documentazione di AWS Distro per OpenTelemetry.

Visualizza i dati aggregati sulle risorse del cluster con il pannello di controllo EKS

image::images/eks-dashboard.png[screenshot of account level cluster metrics]

Cos'è il pannello di controllo Amazon EKS?

La dashboard di Amazon EKS offre una visibilità consolidata sui cluster Kubernetes in più regioni e account. AWS AWS Con questo pannello di controllo puoi:

- Tieni traccia dei cluster pianificati per end-of-support gli aggiornamenti automatici entro i prossimi 90 giorni.
- Progettare i costi del piano di controllo EKS per i cluster con supporto esteso.
- Esaminare i cluster con approfondimenti che richiedono attenzione prima dell'aggiornamento.
- Identificare i gruppi di nodi gestiti che eseguono versioni AMI specifiche.
- Monitorare la distribuzione del tipo di supporto del cluster (standard rispetto al supporto esteso).

La dashboard EKS si integra con EKS Cluster Insights per far emergere problemi con i cluster, come l'uso di Kubernetes obsoleti. APIs Per ulteriori informazioni, consulta the section called "Approfondimenti sui cluster".

> **Note**
>
> Il pannello di controllo EKS non è in tempo reale e si aggiorna ogni 12 ore. Per il monitoraggio in tempo reale dei cluster, consulta Monitoraggio dei cluster

In che modo la dashboard utilizza AWS Organizations?

La dashboard di Amazon EKS richiede l'integrazione di AWS Organizations per la funzionalità. Sfrutta AWS Organizations per raccogliere in modo sicuro le informazioni sui cluster tra gli account.

Questa integrazione offre gestione e governance centralizzate man mano che l'infrastruttura AWS si espande.

Se AWS Organizations non è abilitato per la tua infrastruttura, consulta la AWS Organizations User Guide per le istruzioni di configurazione.

Accesso multi-account e tra regioni

La dashboard EKS può visualizzare le risorse del cluster in qualsiasi account membro dell' AWS organizzazione. Per generare un elenco di AWS account nell'organizzazione, consulta Esportazione dei dettagli per tutti gli account di un'organizzazione.

La regione AWS us-east-1 genera il dashboard. Per visualizzare il pannello di controllo, devi accedere a questa regione. La dashboard aggrega i dati tra AWS le regioni, ma non include le `GovCloud` regioni della Cina.

Termini chiave

- AWS Organizzazione: una struttura di gestione unificata per più AWS account.
- Account di gestione: l'account principale che controlla l' AWS organizzazione.
- Account membro: qualsiasi account all'interno dell'organizzazione ad eccezione dell'account di gestione.
- Amministratore delegato: account membro a cui sono state concesse autorizzazioni amministrative specifiche per più account. All'interno dell'account di gestione, è possibile selezionare un account amministratore delegato per AWS servizio.
- Accesso attendibile: autorizzazione per il pannello di controllo EKS ad accedere alle informazioni del cluster tra gli account dell'organizzazione.
- Service-Linked Role (SLRs): un tipo unico di ruolo IAM direttamente collegato a un servizio. AWS Il pannello di controllo EKS usa un SLR per leggere le informazioni sugli account e sulle organizzazioni.
- Per ulteriori informazioni, vedere Terminologia e concetti nella AWS Organizations User Guide.

Panoramica generale

1. Accedi all'account di gestione della tua AWS organizzazione.

- I passaggi per accedere all'account di gestione dipendono da come hai configurato l' AWS organizzazione. Ad esempio, puoi accedere all'account di gestione tramite AWS Identity Center o Okta.

2. Abilita l'accesso attendibile tramite la console EKS.
3. Assegna un amministratore delegato utilizzando il suo ID AWS account.
4. Passa all'account amministratore delegato.
5. Accedi alla console EKS migliorata con visibilità a livello di organizzazione.

Abilita la dashboard EKS utilizzando la console AWS

> **Important**
>
> È necessario accedere all'account di gestione della propria AWS organizzazione per abilitare la dashboard EKS.

Accedi alle impostazioni del pannello di controllo EKS

1. Conferma quanto segue:
   a. AWS Organizations è abilitata e configurata.
   b. Hai effettuato l'accesso all'account di gestione dell'organizzazione.
   c. Stai visualizzando la console AWS di gestione nella regione us-east-1.
2. Passa alla console EKS.
3. Nella barra laterale sinistra, apri Impostazioni del pannello di controllo.

Configurazione dell'accesso al pannello di controllo Amazon EKS

1. Trova l'ID dell' AWS AWS account a cui desideri consentire la visualizzazione della dashboard EKS.
   a. Questo passaggio è facoltativo ma consigliato. In caso contrario, puoi accedere al pannello di controllo solo dall'account di gestione. Come best practice, dovresti limitare gli accessi all'account di gestione.
2. Fai clic su Abilitazione dell'accesso attendibile.
   a. Ora puoi vedere il pannello di controllo dall'account di gestione.

3. Fai clic su Registra amministratore delegato e inserisci l'ID account dell' AWS account che utilizzerai per visualizzare la dashboard.
   a. Ora puoi visualizzare il pannello di controllo dall'account amministratore delegato o dall'account di gestione.

Per informazioni sulle autorizzazioni necessarie per abilitare il pannello di controllo, consulta Policy IAM minime richieste.

Visualizzazione del pannello di controllo EKS

1. Accedi all'account amministratore delegato (consigliato) o all'account di gestione.
2. Accedi alla regione us-east-1.
3. Vai al servizio EKS e seleziona pannello di controllo dalla barra laterale sinistra.

> **Note**
>
> Controlla le autorizzazioni IAM necessarie per visualizzare il pannello di controllo EKS.

Configura il pannello di controllo

Puoi configurare la visualizzazione del pannello di controllo e filtrare le risorse.

Risorse disponibili

- Cluster: visualizza informazioni aggregate relative allo stato e alla posizione dei cluster EKS.
  - Cluster con problemi di integrità.
  - Cluster su EKS Extended Support.
  - Suddivisione dei cluster sulla base della versione di Kubernetes.
- Gruppi di nodi gestiti: esamina i gruppi e le EC2 istanze di nodi gestiti.
  - Gruppi di nodi per tipo di AMI, come Amazon Linux o Bottlerocket.
  - Problemi di integrità dei gruppi di nodi.
  - Distribuzione dei tipi di istanza
- Componenti aggiuntivi: scopri quali componenti aggiuntivi Amazon EKS hai installato e qual è il loro stato.

- Numero di installazioni per componente aggiuntivo.
- Componenti aggiuntivi con problemi di integrità.
- Distribuzione della versione per componente aggiuntivo.

Visualizzazioni disponibili

- Vista grafico
  - Una vista widget personalizzabile che presenta grafici e visualizzazioni della risorsa selezionata.
  - Le modifiche alla vista grafico, come la rimozione di un widget, sono visibili a tutti gli utenti del pannello di controllo EKS.
- Vista risorsa
  - Una vista a elenco della risorsa selezionata, che supporta i filtri.
- Vista mappa
  - Mostra la distribuzione geografica della risorsa selezionata.

Filtrare il pannello di controllo EKS

Puoi filtrare il pannello di controllo EKS per:

- AWS Account
- Unità organizzativa, definita da AWS Organizations
- AWS Regione

Disattiva la dashboard EKS utilizzando la AWS console

1. Conferma quanto segue:
   a. AWS Organizations è abilitata e configurata.
   b. Hai effettuato l'accesso all'account di gestione dell'organizzazione.
   c. Stai visualizzando la console AWS di gestione nella regione us-east-1.
2. Passa alla console EKS.
3. Nella barra laterale sinistra, apri Impostazioni del pannello di controllo.
4. Fai clic su Disabilita accesso attendibile.

Risoluzione dei problemi del pannello di controllo EKS

Problema di attivazione del pannello di controllo EKS

- È necessario accedere all'account di gestione di un'organizzazione. AWS
  - Se non hai un' AWS organizzazione, creane una. Scopri come Creare e configurare un'organizzazione.
  - Se il tuo AWS account è già membro di un' AWS organizzazione, identifica l'amministratore dell'organizzazione.
- È necessario accedere all' AWS account con autorizzazioni IAM sufficienti per creare e aggiornare le risorse AWS Organizations.

Problema nella visualizzazione del pannello di controllo EKS

- Devi aver effettuato l'accesso a uno dei seguenti account: AWS
  - L'account di gestione dell' AWS organizzazione
  - Un account amministratore delegato, identificato nelle impostazioni del pannello di controllo EKS dell'account di gestione.
- Se hai recentemente abilitato il pannello di controllo EKS, tieni presente che la compilazione iniziale dei dati può richiedere fino a 12 ore.
- Prova a riattivare il pannello di controllo usando la CLI, inclusa la creazione del ruolo collegato al servizio.

I widget del pannello di controllo si spostano in modo imprevisto

- La dashboard EKS salva la visualizzazione configurabile del widget a livello di AWS account. Se modifichi la visualizzazione del widget, le modifiche verranno visualizzate da altre persone che utilizzano lo stesso AWS account.

Configurare l'integrazione di EKS Dashboard con AWS Organizations

Questa sezione fornisce step-by-step istruzioni per configurare l'integrazione di EKS Dashboard con AWS Organizations. Imparerai come abilitare e disabilitare l'accesso attendibile tra i servizi, nonché come registrare e cancellare gli account di amministratore delegato. Ogni attività di configurazione può essere eseguita utilizzando la AWS console o la AWS CLI.

Abilitazione dell'accesso attendibile

L'accesso attendibile autorizza il pannello di controllo EKS ad accedere in modo sicuro alle informazioni sui cluster di tutti gli account della tua organizzazione.

Utilizzo della console AWS

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Accedi alla console EKS nella regione Stati Uniti orientali 1.
3. Nella barra laterale sinistra, seleziona Impostazioni del pannello di controllo.
4. Fai clic su Abilitazione dell'accesso attendibile.

> **Note**
>
> Quando abiliti l'accesso attendibile tramite la console EKS, il sistema crea automaticamente il ruolo collegato al servizio `AWSServiceRoleForAmazonEKSDashboard`. Questa creazione automatica non si verifica se si abilita l'accesso affidabile utilizzando la AWS CLI o la console AWS Organizations.

Utilizzo della AWS CLI

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Esegui i comandi seguenti:

```
aws iam create-service-linked-role --aws-service-name dashboard.eks.amazonaws.com
aws organizations enable-aws-service-access --service-principal eks.amazonaws.com
```

Disabilitare l'accesso attendibile

La disabilitazione dell'accesso attendibile revoca l'autorizzazione del pannello di controllo EKS ad accedere alle informazioni sui cluster negli account dell'organizzazione.

Utilizzo della AWS console

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Accedi alla console EKS nella regione Stati Uniti orientali 1.

3. Nella barra laterale sinistra, seleziona Impostazioni del pannello di controllo.
4. Fai clic su Disabilita accesso attendibile.

Utilizzo della AWS CLI

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Esegui il comando seguente:

```
aws organizations disable-aws-service-access --service-principal eks.amazonaws.com
```

Abilitazione di un account di amministratore delegato

Un amministratore delegato è un account membro a cui è stata concessa l'autorizzazione ad accedere al pannello di controllo EKS.

Utilizzo della AWS console

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Accedi alla console EKS nella regione Stati Uniti orientali 1.
3. Nella barra laterale sinistra, seleziona Impostazioni del pannello di controllo.
4. Fai clic su Registrazione degli amministratori delegati.
5. Inserisci l'ID account dell' AWS account che desideri scegliere come amministratore delegato.
6. Conferma la registrazione.

Utilizzo della AWS CLI

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Esegui il seguente comando, sostituendo `123456789012` con l'ID del tuo account:

```
aws organizations register-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

Disabilitare un account di amministratore delegato

La disabilitazione di un amministratore delegato rimuove l'autorizzazione dell'account ad accedere al pannello di controllo EKS.

Utilizzo della AWS console

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Accedi alla console EKS nella regione Stati Uniti orientali 1.
3. Nella barra laterale sinistra, seleziona Impostazioni del pannello di controllo.
4. Trova l'amministratore delegato nell'elenco.
5. Fai clic su Annulla registrazione vicino all'account che vuoi rimuovere come amministratore delegato.

Utilizzo della AWS CLI

1. Accedi all'account di gestione della tua AWS organizzazione.
2. Esegui il seguente comando, sostituendolo `123456789012` con l'ID dell'account dell'amministratore delegato.

```
aws organizations deregister-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

Sono richieste policy IAM minime

Questa sezione descrive le politiche IAM minime richieste per consentire l'accesso affidabile e delegare un amministratore per l'integrazione di EKS Dashboard con Organizations AWS .

Policy per consentire l'accesso attendibile

Per abilitare l'accesso affidabile tra EKS Dashboard e AWS Organizations, sono necessarie le seguenti autorizzazioni:

```
{
    "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"
        }
    ]
}
```

Policy per la delega di un amministratore

Per registrare o annullare la registrazione di un amministratore delegato per il pannello di controllo EKS, servono le seguenti autorizzazioni:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "organizations:RegisterDelegatedAdministrator",
                "organizations:DeregisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": "*"
        }
    ]
}
```

Policy per la visualizzazione del pannello di controllo 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**
>
> Queste policy devono essere collegate al principale IAM (utente o ruolo) nell'account di gestione dell' AWS organizzazione. Gli account dei membri non possono consentire l'accesso attendibile o delegare amministratori.

Migliora EKS con AWS servizi integrati

Oltre ai servizi descritti in altre sezioni, Amazon EKS collabora con altri AWS servizi per fornire soluzioni aggiuntive. Questo argomento identifica alcuni degli altri servizi che utilizzano Amazon EKS per aggiungere funzionalità o i servizi utilizzati da Amazon EKS per eseguire le attività.

Argomenti

- Backup dei cluster EKS con AWS Backup
- Creare risorse Amazon EKS con AWS CloudFormation
- Connect ai repository Git con AWS CodeConnections
- Analizza gli eventi di sicurezza su EKS con Amazon Detective
- Rileva minacce con Amazon GuardDuty
- Avvia cluster EKS a bassa latenza con AWS Local Zones
- Valutare la resilienza dei cluster EKS con AWS Resilience Hub
- Gestisci i segreti delle applicazioni con AWS Secrets Manager
- Centralizza e analizza i dati di sicurezza EKS con Security Lake
- Abilitare una connettività sicura tra cluster con Amazon VPC Lattice

Backup dei cluster EKS con AWS Backup

AWS Backup supporta i backup dei cluster Amazon EKS, incluso lo stato del cluster Kubernetes e lo storage persistente collegato al cluster EKS tramite una dichiarazione di volume persistente (volumi EBS, file system EFS e bucket S3). Un backup Amazon EKS creerà un punto di ripristino composito, in cui verrà creato un punto di ripristino secondario per ogni risorsa di cui è stato eseguito il backup.

- Come eseguire il backup delle risorse: Guida introduttiva a AWS Backup
- Creazione di backup per tipo di risorsa: Amazon EKS Backups
- Come ripristinare i cluster Amazon EKS: Amazon EKS Backups restore

Per iniziare tramite la console EKS, la console di AWS Backup o la CLI, assicurati che il tuo ruolo IAM disponga delle seguenti autorizzazioni:

- Queste sono disponibili nella politica gestita di AWS Backup. AWSBackupServiceRolePolicyForBackup Contiene le autorizzazioni necessarie per il backup del cluster Amazon EKS e dello storage persistente EBS ed EFS.
- Se il tuo cluster EKS contiene un bucket S3, dovrai assicurarti che le seguenti politiche e prerequisiti per il tuo bucket S3 vengano aggiunte e abilitate come documentato:
- AWSBackupServiceRolePolicyForBackup S3
- Prerequisiti per i backup S3

Assicurati che i tuoi cluster EKS abbiano le seguenti impostazioni:

- Modalità di autorizzazione del cluster EKS impostata su API o API_AND _CONFIG_MAP for AWS Backup per creare voci di accesso per accedere al cluster EKS.

Creare risorse Amazon EKS con AWS CloudFormation

Amazon EKS è integrato con AWS CloudFormation, un servizio che consente di modellare e configurare le risorse AWS in modo da dedicare meno tempo alla creazione e alla gestione delle risorse e dell'infrastruttura. # possibile creare un modello che descriva tutte le risorse AWS desiderate, ad esempio un cluster Amazon EKS. In questo modo, sarà AWS CloudFormation a gestire il provisioning e la configurazione delle risorse.

Quando si utilizza AWS CloudFormation, è possibile riutilizzare il modello per configurare le risorse Amazon EKS in modo coerente e continuo. È sufficiente descrivere le risorse una volta e quindi allestire le stesse risorse più volte in più regioni e account AWS.

Modelli Amazon EKS e AWS CloudFormation

necessario conoscere i modelli AWS CloudFormation, prima di poter effettuare il provisioning e la configurazione delle risorse per Amazon EKS e per i servizi correlati. I modelli sono file di testo formattati in JSON o YAML. Questi modelli descrivono le risorse di cui si vuole effettuare il provisioning negli stack AWS CloudFormation. Se non si ha familiarità con JSON o YAML, è possibile usare AWS CloudFormation Designer per iniziare a utilizzare i modelli AWS CloudFormation. Per ulteriori informazioni, consultare la pagina What is AWS CloudFormation Designer? della Guida per l'utente di AWS CloudFormation.

Amazon EKS supporta la creazione di cluster e gruppi di nodi in AWS CloudFormation. Per ulteriori informazioni, inclusi esempi di modelli JSON e YAML per le risorse Amazon EKS, consultare la pagina Amazon EKS resource type reference della Guida per l'utente di AWS CloudFormation.

Ulteriori informazioni su AWS CloudFormation

Per ulteriori informazioni su AWS CloudFormation, consultare le seguenti risorse:

- AWS CloudFormation
- Guida per l'utente di AWS CloudFormation
- Guida per l'utente dell'interfaccia a riga di comando di AWS CloudFormation

Connect ai repository Git con AWS CodeConnections

AWS CodeConnectionsfornisce un modo sicuro per connettere i AWS servizi a repository di codice sorgente di terze parti. AWS CodeConnections supporta GitHub GitLab, Bitbucket e altri provider. Per saperne di più e iniziare, consulta Lavorare con le connessioni.

Utilizzare AWS CodeConnections con Argo CD

Quando si utilizza la funzionalità EKS per Argo CD, è possibile scegliere di AWS CodeConnections abilitare l'autenticazione sicura nei repository Git senza gestire credenziali di lunga durata o token di accesso personali. AWS CodeConnections gestisce il flusso di OAuth autenticazione e gestisce la connessione al provider Git, fornendo un approccio sicuro e gestibile all'accesso ai GitOps repository e ai manifesti delle applicazioni archiviati in provider Git di terze parti.

Prerequisiti

- Creazione di un cluster Amazon EKS con funzionalità Argo CD
- Una connessione creata nel AWS CodeConnections tuo provider Git
- Autorizzazioni IAM configurate per Argo CD per utilizzare la connessione

Per configurare l'accesso al CodeConnections repository di Argo CD

1. Crea una connessione nella console: CodeConnections
   a. Apri la CodeConnections console.

b. Scegli Crea connessione.

c. Seleziona il tuo provider (GitHubo Bitbucket) e segui il flusso di autenticazione. GitLab

d. Nota l'ARN di connessione da utilizzare nella configurazione del CD Argo.

2. Assicurati che il ruolo di funzionalità di Argo CD disponga delle autorizzazioni per utilizzare la connessione con una politica basata sulle risorse:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection",
        "codeconnections:GetConnection"
      ],
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-
id"
    }
  ]
}
```

3. Configura Argo CD in modo che faccia riferimento alla CodeConnections risorsa quando aggiungi un repository, usando l'endpoint della risorsa come URL del CodeConnections repository. La funzionalità Argo CD utilizza la connessione per autenticarsi al provider Git senza richiedere credenziali di lunga durata.

Considerazioni sull'utilizzo con Argo CD CodeConnections

Quando si utilizza AWS CodeConnections con il CD EKS Capability for Argo, tenete presente quanto segue:

- La CodeConnections connessione deve trovarsi nella stessa AWS regione del cluster EKS
- Il ruolo di funzionalità di Argo CD deve disporre di autorizzazioni `codeconnections:UseConnection` e `codeconnections:GetConnection`
- CodeConnections gestisce automaticamente il OAuth flusso e il ciclo di vita delle credenziali

Per ulteriori informazioni sulla configurazione dell'accesso al repository con Argo CD, consulta. the section called “Configura i repository”

Analizza gli eventi di sicurezza su EKS con Amazon Detective

Amazon Detective consente di analizzare, esaminare e identificare rapidamente la causa principale degli esiti di sicurezza o delle attività sospette. Detective raccoglie automaticamente i dati di registro dalle tue AWS risorse. Utilizza quindi il machine learning, l'analisi statistica e la teoria dei grafi per generare visualizzazioni che consentono di condurre indagini sulla sicurezza più rapide ed efficaci. Le aggregazioni di dati, i riepiloghi e il contesto predefiniti di Detective facilitano e velocizzano l'analisi e la determinazione della natura e dell'estensione dei possibili problemi di sicurezza. Per ulteriori informazioni, consulta la Guida per l'utente di Amazon Detective.

Detective organizza Kubernetes e i AWS dati in risultati come:

- Dettagli del cluster Amazon EKS, tra cui l'identità IAM che ha creato il cluster e il ruolo di servizio del cluster. Puoi esaminare l'attività dell'API Kubernetes AWS e di queste identità IAM con Detective.
- Dettagli del container, come l'immagine e il contesto di sicurezza. Puoi esaminare i dettagli relativi ai pod terminati.
- Attività API di Kubernetes, incluse le tendenze generali di attività API e i dettagli su chiamate API specifiche. Ad esempio, puoi mostrare il numero di chiamate API di Kubernetes riuscite e non riuscite emesse in un intervallo di tempo selezionato. Inoltre, la sezione sulle chiamate API appena osservate potrebbe essere utile per identificare attività sospette.

I log di audit di Amazon EKS sono un pacchetto di origini dati facoltativo che può essere aggiunto al grafico del comportamento di Detective. Puoi visualizzare i pacchetti di origine facoltativi disponibili e il rispettivo stato dal tuo account. Per ulteriori informazioni, consulta Log di audit di Amazon EKS per Detective nella Guida per l'utente di Amazon Detective.

Uso di Amazon Detective con Amazon EKS

Prima di poter esaminare i risultati, Detective deve essere abilitato per almeno 48 ore nella stessa AWS regione in cui si trova il cluster. Per ulteriori informazioni, consulta le sezioni Configurazione di Amazon Detective nella Guida per l'utente di Amazon Detective.

1. Apri la console Detective all'indirizzo https://console.aws.amazon.com/detective/.
2. Seleziona Cerca nel riquadro di navigazione a sinistra.
3. Seleziona Scegli il tipo, quindi seleziona il Cluster EKS.

4. Inserisci il nome o l'ARN del cluster e quindi scegli Cerca.
5. Nei risultati della ricerca, seleziona il nome del cluster del quale desideri visualizzare l'attività. Per ulteriori informazioni su ciò che puoi visualizzare, consulta Overall Kubernetes API activity involving an Amazon EKS cluster nella Guida per l'utente di Amazon Detective.

Rileva minacce con Amazon GuardDuty

Amazon GuardDuty è un servizio di rilevamento delle minacce che monitora continuamente gli account, i container, i carichi di lavoro e i dati nell'ambiente AWS. Grazie a modelli di machine learning (ML) e funzionalità di rilevamento di anomalie e minacce, GuardDuty monitora continuamente origini di log e attività di runtime diverse per identificare e dare priorità ai potenziali rischi per la sicurezza e alle attività dannose nell'ambiente.

Tra le altre funzionalità, GuardDuty offre le due funzionalità seguenti che rilevano potenziali minacce ai cluster EKS: Protezione EKS e Monitoraggio del runtime.

> **Note**
>
> Novità: la modalità automatica di Amazon EKS si integra con GuardDuty.

Protezione EKS

Questa funzionalità fornisce una copertura di rilevamento delle minacce per aiutarti a proteggere i cluster Amazon EKS monitorando i log di audit Kubernetes associati. I log di audit Kubernetes acquisiscono azioni sequenziali all'interno del cluster, comprese attività di utenti, applicazioni che utilizzano l'API Kubernetes e il piano di controllo. GuardDuty è in grado di rilevare che le API chiamate per manomettere potenzialmente le risorse in un cluster Kubernetes sono state invocate da un utente non autenticato.

Quando attivi Protezione EKS, GuardDuty sarà in grado di accedere ai log di audit di Amazon EKS solo per il rilevamento continuo delle minacce. Se GuardDuty identifica una potenziale minaccia per il cluster, genera un esito del log di controllo Kubernetes associato di un tipo specifico. Per ulteriori informazioni sui tipi di esito disponibili nei log di audit di Kubernetes, consultare Kubernetes audit logs finding types nella Guida per l'utente di Amazon GuardDuty.

Per maggiori informazioni, consultare EKS Protection nella Guida per l'utente di Amazon GuardDuty.

Monitoraggio del runtime

Questa funzionalità monitora e analizza gli eventi a livello di sistema operativo, di rete e dei file per aiutarti a rilevare potenziali minacce in carichi di lavoro AWS specifici all'interno dell'ambiente.

Quando abiliti Monitoraggio del runtime e installi l'agente GuardDuty nei tuoi cluster Amazon EKS, GuardDuty inizia a monitorare gli eventi di runtime associati a questo cluster. Tieni presente che l'agente GuardDuty e Monitoraggio del runtime non sono disponibili per Amazon EKS Hybrid Nodes, quindi Monitoraggio del runtime non è disponibile per gli eventi di runtime che si verificano sui nodi ibridi. Se GuardDuty identifica una potenziale minaccia per il cluster, genera un esito associato al Monitoraggio del runtime. Ad esempio, una minaccia può iniziare compromettendo un singolo container che esegue un'applicazione Web vulnerabile. Questa applicazione Web potrebbe disporre delle autorizzazioni di accesso ai container e ai carichi di lavoro sottostanti. In questo scenario, le credenziali configurate in modo errato potrebbero portare a un accesso più ampio all'account e ai dati archiviati al suo interno.

Per configurare Monitoraggio del runtime, installa l'agente GuardDuty nel cluster come componente aggiuntivo di Amazon EKS. Per ulteriori informazioni sul componente aggiuntivo, consultare the section called " AWS componenti aggiuntivi".

Per ulteriori informazioni, consulta Runtime Monitoring nella Guida per l'utente di Amazon GuardDuty.

Avvia cluster EKS a bassa latenza con AWS Local Zones

Una zona locale AWS è un'estensione di una Regione AWS in prossimità geografica degli utenti. Le zone locali hanno le loro connessioni a Internet e supportano AWS Direct Connect. Le risorse create in una zona locale possono servire gli utenti locali con comunicazioni a latenza molto bassa. Per ulteriori informazioni, consulta AWS Local Zones User Guide e Local Zones nella Guida per l'utente di Amazon EC2.

Amazon EKS supporta alcune risorse nelle zone locali, tra cui i gruppi di nodi gestiti, i nodi Amazon EC2 autogestiti, i volumi Amazon EBS e gli Application Load Balancer (ALB). Si consiglia di prendere in considerazione quanto segue nel momento in cui si utilizzano le zone locali come parte del cluster Amazon EKS.

- Non puoi creare nodi Fargate nelle zone locali con Amazon EKS.
- Il piano di controllo Kubernetes gestito da Amazon EKS viene sempre eseguito nella Regione AWS. Il piano di controllo Kubernetes gestito da Amazon EKS non può essere eseguito nella zona

locale. Poiché le Local Zones vengono visualizzate come sottoreti all'interno del VPC, Kubernetes considera le risorse della zona locale come parte di tale sottorete.

- Il cluster Amazon EKS Kubernetes comunica con le istanze Amazon EC2 eseguite nella Regione o zona locale AWS utilizzando l'interfaccia di rete elastiche gestita Amazon EKS. Per ulteriori informazioni sull'architettura di rete Amazon EKS, consulta Configurazione delle reti.
- A differenza delle sottoreti regionali, Amazon EKS non può inserire interfacce di rete nelle sottoreti delle zone locali. Ciò significa che non è necessario specificare sottoreti della zona locale quando si crea il cluster. Tuttavia, puoi avere nodi worker in diverse zone locali multiple connesse allo stesso cluster.

Valutare la resilienza dei cluster EKS con AWS Resilience Hub

AWS Resilience Hub valuta la resilienza di un cluster Amazon EKS analizzandone l'infrastruttura. AWS Resilience Hub utilizza la configurazione relativa al controllo degli accessi basato sul ruolo (RBAC) di Kubernetes per valutare i carichi di lavoro Kubernetes implementati nel cluster. Per ulteriori informazioni, consulta Enabling AWS Resilience Hub access to your Amazon EKS cluster nella Guida per l'utente di AWS Resilience Hub.

Gestisci i segreti delle applicazioni con AWS Secrets Manager

AWS Secrets Manager ti aiuta a gestire, accedere e ruotare credenziali, chiavi API e altri segreti durante il loro ciclo di vita. Con Secrets Manager, puoi proteggere e gestire i segreti utilizzati per accedere alle risorse in AWS, su servizi di terze parti e in locale. Per ulteriori informazioni, consulta la Guida per l'utente di AWS Secrets Manager.

Quando si utilizza il CD EKS Capability for Argo, Secrets Manager offre un modo sicuro per archiviare e recuperare le credenziali del repository Git senza codificare i dati sensibili nella configurazione e nelle risorse del CD Argo. Questa integrazione è particolarmente utile per gestire i token di accesso ai repository privati e le chiavi SSH utilizzate da Argo CD per sincronizzare le applicazioni dai repository Git.

Usare AWS Secrets Manager con Argo CD

Quando si utilizza il CD EKS Capability for Argo, è possibile memorizzare le credenziali del repository Git in Secrets Manager e configurare Argo CD per recuperarle. Questo approccio è più sicuro rispetto all'archiviazione delle credenziali direttamente nella configurazione di Argo CD o all'utilizzo di token di accesso personali di lunga durata.

Prerequisiti

- Un cluster Amazon EKS con funzionalità Argo CD abilitata
- Credenziali del repository Git archiviate in AWS Secrets Manager
- Autorizzazioni IAM configurate per Argo CD per accedere a Secrets Manager

Per configurare Argo CD per utilizzare Secrets Manager per le credenziali del repository

1. Memorizza le tue credenziali Git in Secrets Manager. Ad esempio, per memorizzare un token di accesso GitHub personale:

```
aws secretsmanager create-secret \
  --name argocd/github-token \
  --secret-string '{"username":"git","password":"ghp_xxxxxxxxxxxx"}'
```

2. Assicurati che il ruolo di funzionalità di Argo CD disponga delle autorizzazioni per recuperare il segreto:

```
{
  "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**
>
> L'autorizzazione di decrittografia KMS è richiesta perché Secrets Manager crittografa tutti i segreti utilizzando KMS. AWS La condizione limita la decrittografia ai soli segreti con il prefisso. `argocd/` Se si utilizza la chiave AWS gestita predefinita per Secrets Manager, questa autorizzazione è sufficiente. Per le chiavi KMS gestite dal cliente, aggiorna il `Resource` campo con la chiave ARN specifica.

3. Configura Argo CD per utilizzare le credenziali di Secrets Manager. Per informazioni sulla sincronizzazione dei segreti da Secrets Manager con i segreti di Kubernetes a cui Argo CD può fare riferimento, consulta Secret Management nella documentazione di Argo CD.
4. Crea una configurazione del repository Argo CD che faccia riferimento all'ARN segreto:

```
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
```

Per ulteriori informazioni sulla configurazione dell'accesso al repository con Argo CD, vedere. the section called “Configura i repository”

Centralizza e analizza i dati di sicurezza EKS con Security Lake

Amazon Security Lake è un servizio di sicurezza di data lake completamente gestito che consente di centralizzare i dati di sicurezza provenienti da varie origini, incluso Amazon EKS. Integrando Amazon

EKS con Security Lake, puoi ottenere informazioni più approfondite sulle attività eseguite sulle tue risorse Kubernetes e migliorare il livello di sicurezza dei tuoi cluster Amazon EKS.

> **Note**
>
> Per ulteriori informazioni sull'uso di Security Lake con Amazon EKS e sulla configurazione delle origini dati, consultare la documentazione di Amazon Security Lake.

Vantaggi dell'utilizzo di Security Lake con Amazon EKS

Dati di sicurezza centralizzati: Security Lake raccoglie e centralizza automaticamente i dati di sicurezza dai cluster Amazon EKS, insieme ai dati di altri servizi, provider SaaS AWS , fonti locali e fonti di terze parti. Ciò fornisce una visione completa della posizione di sicurezza dell'intera organizzazione.

Formato dati standardizzato: Security Lake converte i dati raccolti nel formato Open Cybersecurity Schema Framework (OCSF), che è uno schema open source standard. Questa normalizzazione consente un'analisi e un'integrazione più semplici con altri strumenti e servizi di sicurezza.

Rilevamento delle minacce migliorato: analizzando i dati di sicurezza centralizzati, inclusi i log del piano di controllo di Amazon EKS, è possibile rilevare attività potenzialmente sospette all'interno dei cluster Amazon EKS in modo più efficace. Questo aiuta a identificare e rispondere tempestivamente agli incidenti di sicurezza.

Gestione semplificata dei dati: Security Lake gestisce il ciclo di vita dei dati di sicurezza con impostazioni di conservazione e replica personalizzabili. Ciò semplifica le attività di gestione dei dati e garantisce la conservazione dei dati necessari per scopi di conformità e controllo.

Abilitare Security Lake per Amazon EKS

1. Abilita la registrazione del piano di controllo di Amazon EKS per i tuoi cluster EKS. Per istruzioni dettagliate, consulta Enabling and disabling control plane logs.
2. Add Amazon EKS Audit Logs as a source in Security Lake. Security Lake inizierà quindi a raccogliere informazioni approfondite sulle attività eseguite sulle risorse Kubernetes in esecuzione nei cluster EKS.
3. Configure retention and replication settings per i dati di sicurezza in Security Lake in base alle tue esigenze.

4. Utilizza i dati OCSF normalizzati archiviati in Security Lake per la risposta agli incidenti, l'analisi della sicurezza e l'integrazione con altri servizi o strumenti di terze parti. AWS Ad esempio, puoi generare informazioni sulla sicurezza dai dati di Amazon Security Lake utilizzando Amazon OpenSearch Ingestion.

Analisi dei log EKS in Security Lake

Security Lake normalizza gli eventi del log EKS nel formato OCSF, semplificando l'analisi e la correlazione dei dati con altri eventi di sicurezza. Puoi utilizzare vari strumenti e servizi, come Amazon Athena, Amazon o strumenti di analisi della sicurezza di terze parti, per interrogare e visualizzare i dati normalizzati. QuickSight

Per ulteriori informazioni sulla mappatura OCSF per gli eventi di registro EKS, consulta https://github.com/ocsf/examples/tree/main/mappings/markdown/AWS/v1.1.0/EKS Audit Logs [riferimento alla mappatura] nel repository OCSF. GitHub

Abilitare una connettività sicura tra cluster con Amazon VPC Lattice

Amazon VPC Lattice è un servizio di rete di applicazioni completamente gestito integrato direttamente nell'infrastruttura di rete AWS che puoi utilizzare per connettere, proteggere e monitorare i tuoi servizi su più account e cloud privati virtuali (VPC). Con Amazon EKS, puoi sfruttare Amazon VPC Lattice tramite l'uso del Controller API Gateway AWS, un'implementazione dell'API Gateway Kubernetes. Utilizzando Amazon VPC Lattice, è possibile configurare la connettività tra cluster con semantica Kubernetes standard in modo semplice e coerente. Per iniziare a usare Amazon VPC Lattice con Amazon EKS, consulta la Guida per l'utente del controller API Gateway AWS.

Connettere un cluster Kubernetes a una Console di gestione Amazon EKS con Amazon EKS Connector

È possibile usare Amazon EKS Connector per registrare e connettere qualsiasi cluster Kubernetes conforme a AWS e visualizzarlo nella console Amazon EKS. Una volta connesso, è possibile visualizzare lo stato, la configurazione e i carichi di lavoro del cluster nella console Amazon EKS. È possibile utilizzare questa funzionalità per visualizzare i cluster connessi nella console Amazon EKS, ma non è possibile gestirli. Amazon EKS Connector richiede un agente che è un progetto open source su GitHub. Per altri contenuti tecnici, incluse le domande frequenti e la risoluzione dei problemi, consultare the section called "Risoluzione dei problemi del connettore EKS".

Amazon EKS Connector connette i seguenti tipi di cluster Kubernetes ad Amazon EKS.

- Cluster Kubernetes on-premise
- Cluster autogestiti in esecuzione su Amazon EC2
- Cluster gestiti da altri provider cloud

Considerazioni su Amazon EKS Connector

Prima di utilizzare Amazon EKS Connector, attenersi a quanto segue:

- Prima di connettere il cluster ad Amazon EKS, è necessario disporre dei privilegi amministrativi per il cluster Kubernetes.
- Il cluster Kubernetes deve avere nodi (worker) Linux a 64 bit (x86) presenti prima della connessione. I nodi worker ARM non sono supportati.
- È necessario disporre di nodi (worker) nel cluster Kubernetes che abbiano accesso in uscita agli endpoint di Systems Manager `ssm.` e `ssmmessages.`. Per ulteriori informazioni, consultare Systems Manager endpoints nei Riferimenti generali di AWS.
- Per impostazione predefinita, è possibile connettere fino a 10 cluster per regione. È possibile richiedere un aumento tramite la console Service Quotas. Per ulteriori informazioni, consultare Richiesta di un aumento di quota.
- Solo le API `RegisterCluster`, `ListClusters`, `DescribeCluster` e `DeregisterCluster` di Amazon EKS sono supportate per i cluster Kubernetes esterni.
- È necessario disporre delle autorizzazioni seguenti per registrare un cluster:

- eks:RegisterCluster
- ssm:CreateActivation
- ssm:DeleteActivation
- iam:PassRole

- È necessario disporre delle autorizzazioni seguenti per annullare la registrazione di un cluster:
  - eks:DeregisterCluster
  - ssm:DeleteActivation
  - ssm:DeregisterManagedInstance

Ruoli IAM richiesti per Amazon EKS Connector

L'utilizzo di Amazon EKS Connector richiede i seguenti due ruoli IAM:

- Il ruolo Amazon EKS Connectore collegato al servizio viene creato al momento della prima registrazione del cluster.
- È indispensabile creare il ruolo IAM dell'agente Amazon EKS Connector. Per informazioni dettagliate, vedi the section called "Ruolo IAM di Connector".

Per abilitare l'autorizzazione alla visualizzazione del cluster e del carico di lavoro per i principali IAM, è necessario applicare il `eks-connector` e i ruoli del cluster Amazon EKS Connector nel cluster. Seguire la procedura descritta alla pagina Grant access to view Kubernetes cluster resources on an Amazon EKS console.

Connettere un cluster Kubernetes esterno alla Console di gestione Amazon EKS

È possibile collegare un cluster Kubernetes esterno ad Amazon EKS utilizzando più metodi, come illustrato nel seguente processo. Questo processo prevede due passaggi: registrare il cluster con Amazon EKS e installare l'agente `eks-connector` nel cluster.

> **Important**
>
> È necessario completare il secondo passaggio entro 3 giorni dal completamento del primo passaggio, ossia prima che la registrazione scada.

Considerazioni

È possibile utilizzare i manifesti YAML durante l'installazione dell'agente. In alternativa, è possibile utilizzare Helm se si registra il cluster con la Console di gestione AWS o l'Interfaccia della linea di comando AWS. Tuttavia, non è possibile utilizzare Helm per installare l'agente se si registra il cluster con `eksctl`.

Prerequisiti

- Assicurati di avere creato il ruolo agente del connettore Amazon EKS. Seguire i passaggi descritti alla pagina Creating the Amazon EKS connector agent role.
- È necessario disporre delle autorizzazioni seguenti per registrare un cluster:
  - `eks:RegisterCluster`
  - `ssm:CreateActivation`
  - `ssm:DeleteActivation`
  - `iam:PassRole`

Fase 1: Registrazione del cluster

Per registrare un cluster in Amazon EKS Connector, è possibile utilizzare uno dei seguenti strumenti:

- the section called "CLI AWS"
- the section called "Console di gestione AWS"
- the section called "`eksctl`"

CLI AWS

1. La AWS CLI deve essere installata. Per installarla o aggiornarla, consultare la pagina Installing the AWS CLI.
2. Per la configurazione del plug-in, specificare il ruolo IAM dell'agente Amazon EKS Connector. Per ulteriori informazioni, consulta the section called "Ruoli IAM richiesti per 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
```

Di seguito viene riportato un output di esempio:

```
{
    "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"
    }
}
```

I valori `aws-region`, `activationId`, e `activationCode` verranno utilizzati nel prossimo passaggio.

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli Add cluster (Aggiungi cluster) e seleziona Register (Registra) per visualizzare la pagina di configurazione.
3. Nella pagina Configura cluster compilare i campi riportati di seguito:
   - Nome: un nome univoco per il cluster.
   - Provider: scegli per visualizzare l'elenco a discesa dei provider del cluster Kubernetes. Se non si conosce il provider specifico, selezionare Altro.
   - Ruolo plug-in EKS: selezionare il ruolo da utilizzare per la connessione del cluster.
4. Selezionare Registra cluster.
5. Viene visualizzata la pagina Panoramica del cluster. Se desideri utilizzare il grafico Helm, copia il comando `helm install` e prosegui con il passaggio successivo. Se desideri utilizzare il manifesto YAML, scegli Scarica file YAML per scaricare il file manifesto sull'unità locale.

> **Important**
>
> Questa è l'unica occasione per copiare il comando `helm install` o eseguire il download di questo file. Non uscire da questa pagina, poiché il link non sarà accessibile e sarà necessario annullare la registrazione del cluster e riavviare i passaggi dall'inizio.

Il comando o il file manifesto può essere utilizzato una sola volta per il cluster registrato. Se si eliminano risorse dal cluster Kubernetes, è necessario registrare nuovamente il cluster e ottenere un nuovo file manifesto.

Continua con la fase successiva per applicare il file manifesto al cluster Kubernetes.

`eksctl`

1. Deve essere installato `eksctl` versione `0.68` o successiva. Per installarlo o aggiornarlo, consulta la pagina the section called "Crea cluster (eksctl)".
2. Registra il cluster fornendo un nome, un provider e una regione.

```
eksctl register cluster --name my-cluster --provider my-provider --region region-code
```

Output di esempio:

```
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
```

In questo modo vengono creati file sul computer locale. Questi file devono essere applicati al cluster esterno entro 3 giorni; in caso contrario, la registrazione scade.

3. In un terminale in grado di accedere al cluster, applica il file `eks-connector-binding.yaml`:

```
kubectl apply -f eks-connector-binding.yaml
```

Passaggio 2: installazione dell'agente `eks-connector`

Per installare l'agente `eks-connector`, utilizzare uno dei seguenti strumenti:

- the section called "Helm"
- the section called "yaml"

Helm

> **Note**
>
> Se il cluster è stato registrato con `eksctl`, utilizzare il metodo con manifesto YAML anziché il metodo con grafico Helm.

1. Se nel passaggio precedente è stata utilizzata la AWS CLI, sostituire i valori di `ACTIVATION_CODE` e `ACTIVATION_ID` nel seguente comando rispettivamente con i valori `activationId` e `activationCode`. Sostituire `aws-region` con la Regione AWS utilizzata nella fase precedente. Dopodiché, esegui il comando seguente per installare l'agente `eks-connector` nel cluster in fase di registrazione:

```
$ 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 nel passaggio precedente hai utilizzato la Console di gestione AWS, utilizza il comando copiato in tale passaggio inserendovi questi valori.

2. Verifica l'integrità dell'implementazione del `eks-connector` installato e attendi che lo stato del cluster registrato in Amazon EKS diventi `ACTIVE`.

yaml

Completare la connessione applicando il file manifesto Amazon EKS Connector al proprio cluster Kubernetes. A tale scopo, è necessario utilizzare i metodi descritti in precedenza. Se il manifesto non viene applicato entro tre giorni, la registrazione di Amazon EKS Connector scade. Se la connessione al cluster scade, la registrazione del cluster deve essere annullata prima di poterlo connettere nuovamente.

1. Eseguire il download del file YAML di Amazon EKS Connector.

```
curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/eks-connector/latest/eks-connector.yaml
```

2. Modifica il file YAML del connettore Amazon EKS per sostituire tutti i riferimenti di `%AWS_REGION%`, `%EKS_ACTIVATION_ID%` e `%EKS_ACTIVATION_CODE%` con i valori `aws-region`, `activationId`, e `activationCode` dell'output del passaggio precedente.

   Il comando di esempio seguente può sostituire questi valori.

```
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**
   >
   > Assicurarsi che il codice di attivazione sia in formato base64.

3. In un terminale dotato di accesso al cluster, è possibile applicare il file manifesto aggiornato eseguendo il seguente comando:

```
kubectl apply -f eks-connector.yaml
```

4. Dopo aver applicato al cluster Kubernetes il manifesto e i file YAML dell'associazione dei ruoli di Amazon EKS Connector, verificare che il cluster sia connesso.

```
aws eks describe-cluster \
     --name "my-first-registered-cluster" \
```

```
    --region AWS_REGION
```

L'output deve includere `status=ACTIVE`.

5. (Facoltativo) Assegna tag al cluster. Per ulteriori informazioni, consulta the section called "Assegnazione di tag alle risorse ".

Passaggi successivi

In caso di problemi con questi passaggi, consulta la pagina the section called "Risoluzione dei problemi del connettore EKS".

Per concedere ai principali IAM aggiuntivi l'accesso alla console Amazon EKS al fine di visualizzare le risorse Kubernetes in un cluster connesso, consultare the section called "Garantire l'accesso ai cluster".

Concedere l'accesso per visualizzare le risorse del cluster Kubernetes in una console Amazon EKS

Concedere ai principali IAM l'accesso alla console Amazon EKS per visualizzare informazioni sulle risorse Kubernetes in esecuzione sul cluster connesso.

Prerequisiti

Il principale IAM utilizzato per accedere alla Console di gestione AWS deve soddisfare i seguenti requisiti:

- Deve disporre dell'autorizzazione IAM `eks:AccessKubernetesApi`.
- L'account di servizio Amazon EKS Connector dovrebbe essere in grado di rappresentare il principale IAM nel cluster. Ciò consente ad Amazon EKS Connector di mappare il principale IAM a un utente Kubernetes.

Per creare e applicare il ruolo del cluster Amazon EKS Connector

1. Eseguire il download del modello di ruolo del cluster `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. Modificare il file YAML del modello di ruolo del cluster. Sostituire i riferimenti di `%IAM_ARN%` con il nome della risorsa Amazon (ARN) del principale IAM.
3. Applicare il ruolo del cluster Amazon EKS Connector YAML al cluster Kubernetes.

```
kubectl apply -f eks-connector-clusterrole.yaml
```

Affinché un principale IAM possa visualizzare le risorse Kubernetes nella console Amazon EKS, deve essere associato a un `role` o `clusterrole` Kubernetes con le autorizzazioni necessarie per leggere le risorse. Per ulteriori informazioni, consulta Utilizzo dell'autorizzazione RBAC nella documentazione di Kubernetes.

Per configurare un principale IAM per accedere al cluster connesso

1. È possibile eseguire il download di questi file manifesto di esempio per creare, rispettivamente, un `clusterrole` e un'`clusterrolebinding` oppure un `role` e un'`rolebinding`:

   Visualizzare le risorse Kubernetes in tutti i namespace

   - Il ruolo del cluster `eks-connector-console-dashboard-full-access-clusterrole` consente di accedere a tutti gli spazi dei nomi e alle risorse visualizzabili nella console. É possibile modificare il nome di `role`, `clusterrole` e il loro legame corrispondente prima di applicarlo al cluster. Utilizzare il seguente comando per eseguire il download di un file di esempio.

     ```
     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
     ```

   Visualizzare le risorse Kubernetes in un namespace specifico

   - Lo spazio dei nomi in questo file è `default`, quindi se si desidera specificare uno spazio dei nomi diverso, modificare il file prima di applicarlo al cluster. Utilizzare il seguente comando per eseguire il download di un file di esempio.

     ```
     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. Modificare il file YAML con accesso completo o con accesso limitato per sostituire i riferimenti di `%IAM_ARN%` con il nome della risorsa Amazon (ARN) del principale IAM.

3. Applicare i file YAML ad accesso completo o limitato al cluster Kubernetes. Sostituire il valore del file YAML con il proprio valore.

```
kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
```

Per visualizzare le risorse Kubernetes nel cluster connesso, consulta the section called "Accesso alle risorse di cluster". I dati relativi ad alcuni tipi di risorse nella scheda Risorse non sono disponibili per i cluster collegati.

Annullare la registrazione di un cluster Kubernetes dalla console Amazon EKS

Al termine dell'utilizzo di un cluster connesso, è possibile annullarne la registrazione. Una volta annullata la registrazione, il cluster non sarà più visibile nella console Amazon EKS.

È necessario disporre delle autorizzazioni seguenti per chiamare l'API di deregisterCluster:

- `eks:DeregisterCluster`
- `ssm:DeleteActivation`
- `ssm:DeregisterManagedInstance`

Questo processo prevede due passaggi: annullare la registrazione del cluster con Amazon EKS e disinstallare l'agente eks-connector nel cluster.

Annullare la registrazione del cluster Kubernetes

Per annullare la registrazione di un cluster da Amazon EKS Connector, è possibile utilizzare uno dei seguenti strumenti:

- the section called "CLI AWS"
- the section called "Console di gestione AWS"
- the section called "`eksctl`"

CLI AWS

1. La AWS CLI deve essere installata. Per installarla o aggiornarla, consultare la pagina Installing the AWS CLI.
2. Assicurati di avere creato il ruolo agente del connettore Amazon EKS.
3. Annullare la registrazione del cluster connesso.

```
aws eks deregister-cluster \
    --name my-cluster \
    --region region-code
```

Console di gestione AWS

1. Aprire la Console Amazon EKS.
2. Scegli Cluster.
3. Nella pagina Cluster, selezionare il cluster connesso e selezionare Annulla registrazione.
4. Confermare che si desidera annullare la registrazione del cluster.

`eksctl`

1. Installare la versione `0.68` o successiva di `eksctl`. Per installarlo o aggiornarlo, consulta la pagina the section called "Crea cluster (eksctl)".
2. Assicurati di avere creato il ruolo agente del connettore Amazon EKS.
3. Annullare la registrazione del cluster connesso:

```
eksctl deregister cluster --name my-cluster
```

Eliminare le risorse del cluster Kubernetes

Per disinstallare l'agente `eks-connector`, utilizzare uno dei seguenti strumenti:

- the section called "helm"
- the section called "yaml"

helm

Per disinstallare l'agente, esegui il comando seguente.

```
helm -n eks-connector uninstall eks-connector
```

yaml

1. Eliminare il file YAML di Amazon EKS Connector dal cluster Kubernetes.

   ```
   kubectl delete -f eks-connector.yaml
   ```

2. Se sono stati creati `clusterrole` o `clusterrolebindings` per fornire accesso al cluster a principali IAM aggiuntivi, assicurarsi di eliminarli dal cluster Kubernetes.

Risoluzione dei problemi del connettore Amazon EKS

In questo argomento sono descritti alcuni degli errori più comuni che si possono verificare durante l'utilizzo del connettore Amazon EKS e sono riportate le istruzioni per risolverli in maniera temporanea o definitiva.

Risoluzione dei problemi di base

In questa sezione sono descritti i passaggi per la diagnosi dei problemi del connettore Amazon EKS.

Verifica dello stato di Amazon EKS Connector

Per controllare lo stato del connettore Amazon EKS, inserisci:

```
kubectl get pods -n eks-connector
```

Ispezione dei log del connettore Amazon EKS

Il pod di Amazon EKS Connector è composto da tre container. Per recuperare i registri completi per tutti questi container in modo da poterli ispezionare, eseguire i seguenti comandi:

- `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
```

Ottenimento del nome effettivo del cluster

I cluster Amazon EKS sono identificati in modo univoco da `clusterName` all'interno di un singolo account AWS e una regione AWS. Se si hanno più cluster connessi in Amazon EKS, è possibile verificare quale cluster Amazon EKS è registrato con il cluster Kubernetes corrente. Per fare ciò, inserisci quanto segue per individuare il `clusterName` del cluster corrente.

```
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/"
```

Comandi vari

I seguenti comandi sono utili per recuperare le informazioni necessarie per risolvere i problemi.

- Utilizzare il seguente comando per raccogliere immagini utilizzate dai pod in Amazon EKS Connector.

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}"
 | tr -s '[[:space:]]' '\n'
```

- Usa il comando seguente per determinare i nomi dei nodi su cui è in esecuzione il connettore Amazon EKS.

```
kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" | tr -s
 '[[:space:]]' '\n'
```

- Per ottenere le versioni client e server Kubernetes esegui il comando riportato.

```
kubectl version
```

- Usa il comando seguente per ottenere informazioni sui tuoi nodi.

```
kubectl get nodes -o wide --show-labels
```

Problema di Helm: 403 Forbidden

Se hai riscontrato il seguente errore durante l'esecuzione dei comandi di installazione di 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
```

Puoi eseguire la riga seguente per risolvere il problema:

```
docker logout public.ecr.aws
```

Errore della console: il cluster è bloccato nello stato Pending (In sospeso)

Se il cluster rimane bloccato nello stato `Pending` nella console Amazon EKS dopo averlo registrato, è possibile che il connettore Amazon EKS non abbia ancora connesso correttamente il cluster a AWS. Per un cluster registrato, lo stato `Pending` implica che la connessione non è stata stabilita correttamente. Per risolvere il problema, assicurati di aver applicato il manifesto al cluster Kubernetes di destinazione. Se l'hai applicato al cluster ma il cluster rimane ancora nello stato `Pending`, lo statefulset `eks-connector` potrebbe non essere integro. Per risolvere questo problema, consulta the section called "I pod di Amazon EKS Connector stanno eseguendo un ciclo di arresto anomalo" in questo argomento.

Errore della console: User system:serviceaccount:eks-connector:eks-connector non può impersonare la risorsa utenti nel gruppo API nell'ambito del cluster

Il connettore Amazon EKS utilizza la rappresentazione dell'utente di Kubernetes per agire per conto dei principali IAM da Console di gestione AWS. A ogni principale che accede all'API di Kubernetes dall'account del servizio AWS `eks-connector` deve essere concessa l'autorizzazione a impersonare l'utente Kubernetes corrispondente con un'ARN IAM come nome utente Kubernetes. Negli esempi seguenti, l'ARN IAM viene mappato a un utente Kubernetes.

- L'utente IAM *`john`* dall'account AWS *`111122223333`* è mappato a un utente Kubernetes. Le best practice IAM consigliano di concedere le autorizzazioni ai ruoli anziché agli utenti.

  ```
  arn:aws:iam::111122223333:user/john
  ```

- L'*`amministratore`* del ruolo IAM dall'account AWS *`111122223333`* è mappato a un utente Kubernetes:

  ```
  arn:aws:iam::111122223333:role/admin
  ```

  Il risultato è un ARN del ruolo IAM invece dell'ARN di sessione AWS STS.

Per le istruzioni per configurare `ClusterRole` e `ClusterRoleBinding` per concedere il privilegio dell'account di servizio `eks-connector` per impersonare l'utente mappato, consulta the section called "Garantire l'accesso al cluster". Accertati che nel modello `%IAM_ARN%` sia sostituito con l'ARN IAM del principale IAM della Console di gestione AWS.

Errore della console: […] non consentito: L'utente […] non può elencare la risorsa […] nel gruppo API nell'ambito del cluster

Consideriamo il seguente problema. Amazon EKS Connector ha impersonato con successo il principale IAM della richiedente Console di gestione AWS nel cluster Kubernetes di destinazione. Tuttavia, il principale impersonato non dispone dell'autorizzazione RBAC per le operazioni API di Kubernetes.

Per risolvere questo problema, esistono due metodi per concedere le autorizzazioni ad altri utenti. Se in precedenza hai installato eks-connector tramite un grafico Helm, puoi concedere facilmente l'accesso agli utenti eseguendo il seguente comando. Sostituisci i valori `userARN1` e `userARN2` con un elenco delle ARN dei ruoli IAM a cui consentire l'accesso per visualizzare le risorse Kubernetes:

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
    --reuse-values \
    --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

In alternativa, in qualità di amministratore del cluster, concedi il livello appropriato di privilegi RBAC ai singoli utenti Kubernetes. Per maggiori informazioni ed esempi, consulta the section called "Garantire l'accesso ai cluster".

Errore della console: Amazon EKS non è in grado di comunicare con il server API cluster Kubernetes. Per una connessione corretta, il cluster deve trovarsi nello stato ACTIVE (ATTIVO). Riprova tra qualche minuto.

Se il servizio Amazon EKS non può comunicare con il connettore Amazon EKS nel cluster di destinazione, il motivo potrebbe essere uno dei seguenti:

- Il connettore Amazon EKS nel cluster di destinazione non è integro.
- Connettività scadente o connessione interrotta tra il cluster di destinazione e la regione AWS.

Per risolvere questo problema, controlla Amazon EKS Connector logs (Log del connettore Amazon EKS). Se non visualizzi un errore per il connettore Amazon EKS, ritenta la connessione dopo qualche minuto. Se per il cluster di destinazione si verifica regolarmente una latenza elevata o una connettività intermittente, prendi in considerazione la possibilità di registrare nuovamente il cluster in una regione AWS più vicina a te.

I pod di Amazon EKS Connector stanno eseguendo un ciclo di arresto anomalo

Esistono molti motivi che possono causare il passaggio di un pod del connettore Amazon EKS nello stato `CrashLoopBackOff`. Questo problema riguarda probabilmente il container `connector-init`. Controlla lo stato del pod del connettore Amazon EKS.

```
kubectl get pods -n eks-connector
```

Di seguito viene riportato un output di esempio:

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:CrashLoopBackOff   1          7s
```

Se l'output è simile a quello precedente, consulta la pagina the section called "Ispezione dei log del connettore Amazon EKS" per risolvere il problema.

Impossibile avviare eks-connector: InvalidActivation

Quando avvii il connettore Amazon EKS per la prima volta, il connettore registra un `activationId` e un `activationCode` con Amazon Web Services. La registrazione potrebbe non riuscire, causando un arresto anomalo del container `connector-init` con un errore simile al seguente.

```
F1116 20:30:47.261469       1 init.go:43] failed to initiate eks-connector:
 InvalidActivation:
```

Per risolvere questo problema, prendere in considerazione le seguenti cause e le correzioni consigliate:

- La registrazione potrebbe non essere riuscita perché nel file manifesto mancano `activationId` e `activationCode`. In questo caso, accertati che siano i valori corretti restituiti dall'operazione API `RegisterCluster` e che `activationCode` sia presente nel file manifesto. `activationCode` è aggiunto ai segreti Kubernetes, per cui deve essere codificato in `base64`. Per ulteriori informazioni, consulta the section called "Fase 1: Registrazione del cluster".
- La registrazione potrebbe non essere riuscita perché l'attivazione è scaduta. Ciò accade perché, per motivi di sicurezza, devi attivare il connettore Amazon EKS entro tre giorni dalla registrazione del cluster. Per risolvere il problema, assicurati di aver applicato il manifesto del connettore Amazon EKS al cluster Kubernetes di destinazione prima della data e dell'ora di scadenza. Per verificare la data di scadenza dell'attivazione, richiama l'operazione API `DescribeCluster`.

  ```
  aws eks describe-cluster --name my-cluster
  ```

  Nella risposta di esempio seguente, la data e l'ora di scadenza vengono registrate come `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 `activationExpiry` è passato, annulla la registrazione del cluster e registralo di nuovo. In questo modo si genera una nuova attivazione.

Nel nodo del cluster manca la connettività in uscita

Per il corretto funzionamento, il connettore Amazon EKS richiede la connettività in uscita a diversi endpoint AWS. Non è possibile connettere un cluster privato senza connettività in uscita a una regione AWS di destinazione. Per risolvere il problema, è necessario aggiungere la connettività in uscita necessaria. Per informazioni sui requisiti del connettore, consultare the section called "Considerazioni su Amazon EKS Connector".

I pod del connettore Amazon EKS sono in stato **`ImagePullBackOff`**

Se esegui il comando `get pods` e i pod sono nello stato `ImagePullBackOff`, non possono funzionare correttamente. Se i pod del connettore Amazon EKS sono nello stato `ImagePullBackOff`, non possono funzionare correttamente. Controlla lo stato dei pod del connettore Amazon EKS.

```
kubectl get pods -n eks-connector
```

Di seguito viene riportato un output di esempio:

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:ImagePullBackOff   0          4s
```

Il file manifesto predefinito del connettore Amazon EKS fa riferimento alle immagini di Amazon ECR Public Gallery. È possibile che il cluster Kubernetes di destinazione non possa estrarre immagini da Amazon ECR Public Gallery. Risolvi il problema di estrazione delle immagini di Amazon ECR Public Gallery o valuta il mirroring delle immagini nel registro del container privato di tua scelta.

Domande frequenti sui connettori AWS

D: Come funziona la tecnologia alla base del connettore Amazon EKS?

R: Il connettore Amazon EKS è basato sull'agente AWS System Manager (Systems Manager). Il connettore Amazon EKS è eseguito come `StatefulSet` nel tuo cluster . Stabilisce una connessione e applica un proxy alla comunicazione tra il server API del cluster e Amazon Web Services. Ciò è necessario per visualizzare i dati del cluster nella console Amazon EKS fino a quando non si disconnette il cluster da AWS. L'agente Systems Manager è un progetto open source. Per ulteriori informazioni su questo progetto, consultare la pagina del progetto GitHub.

D: Dispongo di un cluster Kubernetes on-premise a cui desidero connettermi. Devo aprire le porte firewall per connetterlo?

R: No, non è necessario aprire alcuna porta firewall. Il cluster Kubernetes richiede solo la connessione in uscita a regioni AWS. I servizi AWS non accedono mai alle risorse nella rete on-premises. Amazon EKS Connector viene eseguito sul cluster e avvia la connessione a AWS. Al termine della registrazione del cluster, AWS emette comandi su Amazon EKS Connector solo dopo aver avviato un'azione dalla console Amazon EKS che richiede informazioni dal server API Kubernetes sul cluster.

D: Quali dati vengono inviati dal cluster a AWS tramite Amazon EKS Connector?

R: Il connettore Amazon EKS invia le informazioni tecniche necessarie per la registrazione del cluster su AWS. Invia inoltre metadati di cluster e carichi di lavoro per le funzionalità della console Amazon EKS richieste dai clienti. Il connettore Amazon EKS raccoglie o invia questi dati solo se avvii un'azione dalla console Amazon EKS o dall'API Amazon EKS che richiede l'invio dei dati a AWS. Per impostazione predefinita, oltre al numero di versione di Kubernetes, AWS non memorizza dati. Memorizza i dati solo se lo autorizzi.

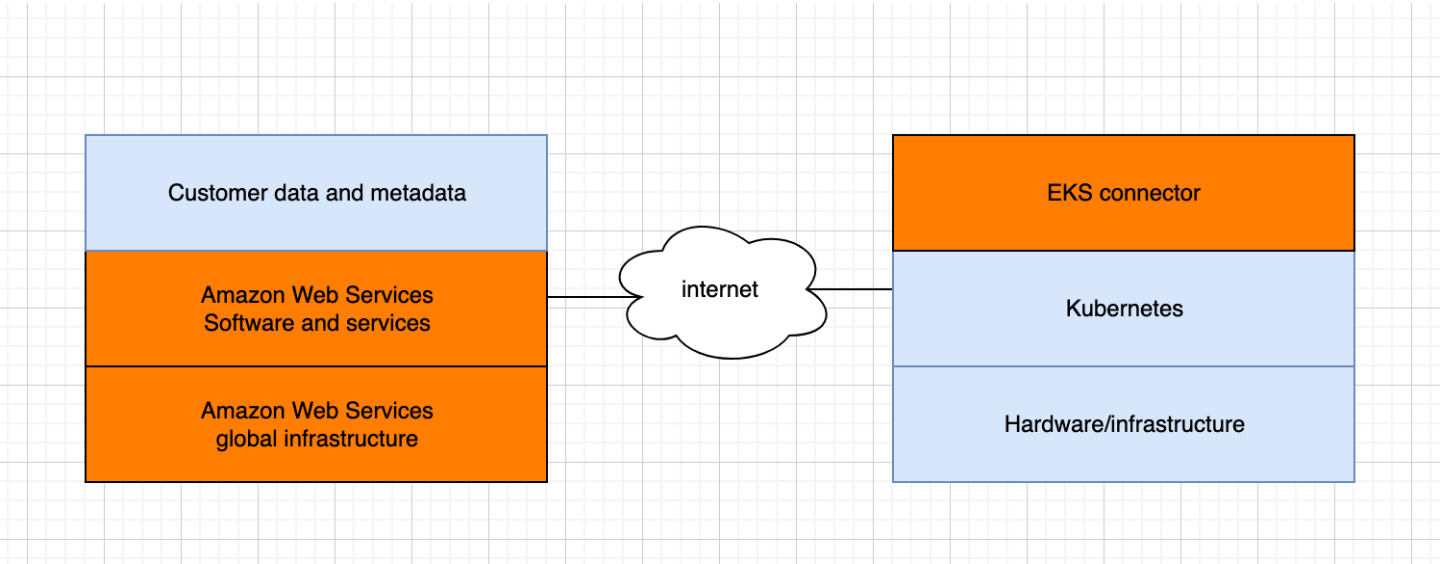
D: È possibile collegare un cluster al di fuori di una regione AWS?

R: Sì, puoi connettere un cluster da qualsiasi posizione ad Amazon EKS. Inoltre, il servizio Amazon EKS può essere collocato in qualsiasi regione AWS commerciale pubblica AWS. Funziona con

una connessione di rete valida dal cluster alla regione di AWS destinazione. È preferibile scegliere la regione AWS più vicina alla posizione del tuo cluster per ottimizzare le prestazioni dell'UI. Ad esempio, se hai un cluster in esecuzione a Tokyo, connetti il cluster alla regione AWS di Tokyo (cioè la regione `ap-northeast-1` AWS) per bassa latenza. È possibile connettere un cluster da qualsiasi posizione ad Amazon EKS in una qualsiasi delle regioni AWS commerciali pubbliche, ad eccezione delle regioni AWS Cina o GovCloud.

Comprensione della sicurezza in Amazon EKS Connector

Amazon EKS Connector è un componente open source che viene eseguito sul cluster Kubernetes. Questo cluster può trovarsi al di fuori dell'ambiente AWS, sollevando alcune considerazioni in materia di responsabilità di sicurezza. Il diagramma seguente illustra tale configurazione. Il colore arancione indica le responsabilità di AWS e il colore blu le responsabilità del cliente:



Questo argomento illustra le differenze nel modello di responsabilità quando il cluster connesso è esterno ad AWS.

Responsabilità di AWS

- Gestire, creare e implementare Amazon EKS Connector, un componente open source che viene eseguito sul cluster Kubernetes di un cliente e comunica con AWS.
- Mantenere la sicurezza delle comunicazioni a livello di trasporto e di applicazione tra il cluster Kubernetes connesso e i servizi AWS.

Responsabilità del cliente

- Garantire la sicurezza specifica del cluster Kubernetes, in particolare seguendo le linee guida riportate di seguito:
  - I segreti di Kubernetes devono essere adeguatamente crittografati e protetti.
  - L'accesso allo spazio dei nomi `eks-connector` deve essere bloccato.
- Configurazione delle autorizzazioni relative al controllo degli accessi basato sul ruolo (RBAC) per gestire l'accesso del principale IAM da AWS. Per istruzioni, consultare the section called "Garantire l'accesso ai cluster".
- Installazione e aggiornamento di Amazon EKS Connector.
- Manutenzione dell'hardware, del software e dell'infrastruttura che supportano il cluster Kubernetes collegato.
- Protezione degli account AWS (ad esempio, tramite la salvaguardia delle credenziali dell'utente root).

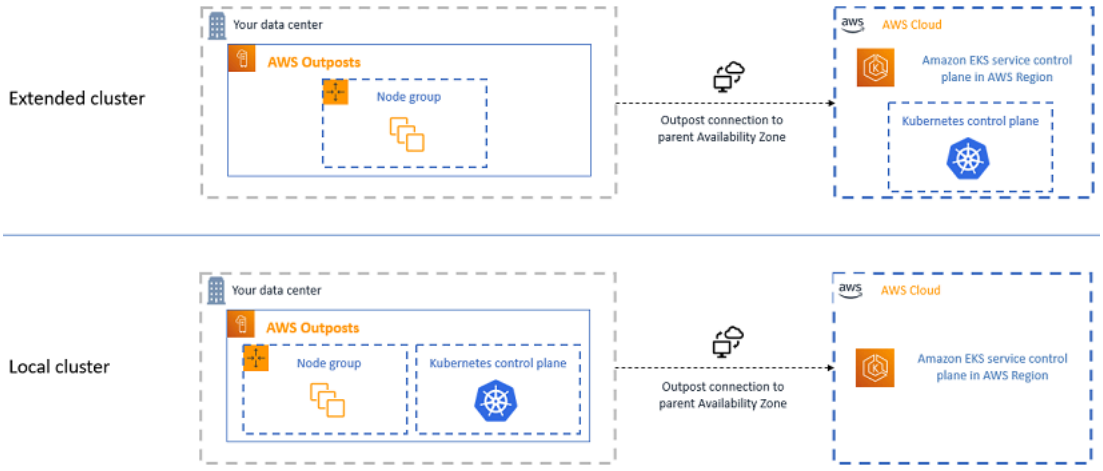
Implementazione di Amazon EKS on-premises con AWS Outposts

Puoi utilizzare Amazon EKS per eseguire applicazioni Kubernetes on-premises su AWS Outposts. Puoi implementare Amazon EKS su Outposts nei seguenti modi:

- Cluster estesi: esegui il piano di controllo (control-plane) di Kubernetes in una Regione AWS e i nodi sull'Outpost.
- Cluster locali: esegui il piano di controllo (control-plane) di Kubernetes e i nodi sull'Outpost.

Per entrambe le opzioni di implementazione, il piano di controllo (control-plane) di Kubernetes è completamente gestito da AWS. Puoi utilizzare le stesse API, gli stessi strumenti e le stesse console di Amazon EKS che utilizzi nel cloud per creare ed eseguire Amazon EKS su Outposts.

Il diagramma seguente illustra queste opzioni di implementazione.



Quando utilizzare ciascuna opzione di implementazione

Sia i cluster locali sia quelli estesi sono opzioni di implementazione generiche e possono essere utilizzati per una vasta gamma di applicazioni.

Con i cluster locali puoi eseguire l'intero cluster Amazon EKS in locale su Outposts. Questa opzione consente di ridurre il rischio di tempi di inattività delle applicazioni che possono derivare da disconnessioni temporanee del cloud dalla rete. Queste disconnessioni di rete possono essere causate da interruzioni della fibra o da eventi meteorologici. Poiché l'intero cluster Amazon EKS viene eseguito in locale su Outposts, le applicazioni rimangono disponibili. Durante le disconnessioni del

cloud dalla rete è possibile eseguire operazioni del cluster. Per ulteriori informazioni, consulta the section called "Preparazione per le disconnessioni". Se ti preoccupa la qualità della connessione di rete dai tuoi Outpost alla Regione AWS principale e hai bisogno di una disponibilità elevata durante le disconnessioni di rete, ti consigliamo di utilizzare l'opzione di implementazione del cluster locale.

I cluster estesi consentono di risparmiare capacità sul tuo Outpost perché il piano di controllo (control-plane) di Kubernetes viene eseguito nella Regione AWS principale. Questa opzione può essere più adatta se sei in grado di investire in una connettività di rete affidabile e ridondante dal tuo Outpost alla Regione AWS. La qualità della connessione di rete è fondamentale per questa opzione. Il modo in cui Kubernetes gestisce le disconnessioni di rete tra il piano di controllo (control-plane) Kubernetes e i nodi potrebbe comportare tempi di inattività delle applicazioni. Per ulteriori informazioni sul funzionamento di Kubernetes, consulta Scheduling, Preemption, and Eviction nella documentazione di Kubernetes.

Confronto tra le opzioni di implementazione

La tabella seguente riporta le differenze tra le due opzioni.

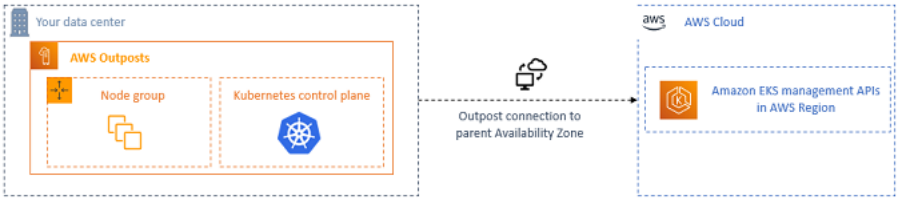
Funzionalità	Cluster esteso	Cluster locale
Posizione del piano di controllo Kubernetes	Regione AWS	Outpost
Account del piano di controllo (control-plane) di Kubernetes	Account AWS	Il tuo account
Disponibilità regionale	Consulta Service endpoints	Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (California settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Seoul), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda), Europa (Londra),

Funzionalità	Cluster esteso	Cluster locale
		Medio Oriente (Bahrein) e Sud America (San Paolo)
Versioni secondarie di Kubernetes	eks/latest/userguide/kubernetes-versions.html [Versioni di Amazon EKS supportate, type="documentation"].	eks/latest/userguide/kubernetes-versions.html [Versioni di Amazon EKS supportate, type="documentation"].
Versioni della piattaforma	Consulta Versioni della piattaforma di EKS	Consulta la sezione the section called "Versioni della piattaforma EKS"
Fattori di forma dell'Outpost	Rack dell'Outpost	Rack dell'Outpost
Interfacce utente	Console di gestione AWS, AWS CLI, API Amazon EKS, `eksctl`, AWS CloudFormation e Terraform	Console di gestione AWS, AWS CLI, API Amazon EKS, `eksctl`, AWS CloudFormation e Terraform
Policy gestite	AmazonEKSClusterPolicy e the section called "AWS politica gestita: Amazon EKSService RolePolicy"	AmazonEKSLocalOutpostClusterPolicy e the section called "AWS politica gestita: Amazon EKSLocal OutpostServiceRolePolicy"
VPC e sottoreti del cluster	Consulta la sezione the section called "Requisiti del VPC e delle sottoreti"	Consulta la sezione the section called "Creazione di un VPC e delle sottoreti"
Accesso all'endpoint del cluster	Pubblico o privato o entrambi	Solo privati
Autenticazione del server API Kubernetes	AWS Identity and Access Management (IAM) e OIDC	IAM e certificati `x.509`
Tipi di nodo	Solo autogestito	Solo autogestito

Funzionalità	Cluster esteso	Cluster locale
Tipi di elaborazione dei nodi	Amazon EC2 on demand	Amazon EC2 on demand
Tipi di archiviazione dei nodi	gp2 di Amazon EBS e SSD NVMe locali	gp2 di Amazon EBS e SSD NVMe locali
AMI ottimizzate per Amazon EKS	Amazon Linux, Windows e Bottlerocket	Solo Amazon Linux
Versioni IP	Solo IPv4	Solo IPv4
Componenti aggiuntivi	Componenti aggiuntivi Amazon EKS o autogestiti	Solo componenti aggiuntivi autogestiti
Interfaccia di rete del container predefinita	Plug-in CNI di Amazon VPC per Kubernetes	Plug-in CNI di Amazon VPC per Kubernetes
Registri del piano di controllo Kubernetes	Amazon CloudWatch Logs	Amazon CloudWatch Logs
Sistema di bilanciamento del carico	Utilizza AWS Load Balancer Controller per eseguire il provisioning solo di Applicati on Load Balancer (nessun Network Load Balancer)	Utilizza AWS Load Balancer Controller per eseguire il provisioning solo di Applicati on Load Balancer (nessun Network Load Balancer)
Crittografia a busta dei segreti	Consulta la sezione the section called "Abilitare la crittografia dei segreti"	Non supportato
Ruoli IAM per gli account di servizio	Consulta la sezione the section called "Credenziali con IRSA"	Non supportato
Risoluzione dei problemi	Consulta la sezione Risoluzio ne dei problemi	Consulta la sezione the section called "Risoluzione dei problemi dei cluster"

Creazione di cluster Amazon EKS locali su AWS Outposts per un'elevata disponibilità

I cluster locali consentono di eseguire l'intero cluster Amazon EKS in locale su AWS Outposts. Ciò consente di ridurre il rischio di tempi di inattività delle applicazioni che possono derivare da disconnessioni temporanee del cloud dalla rete. Queste disconnessioni di rete possono essere causate da interruzioni della fibra o da eventi meteorologici. Poiché l'intero cluster Kubernetes viene eseguito localmente su Outposts, le applicazioni rimangono disponibili. Durante le disconnessioni del cloud dalla rete è possibile eseguire operazioni del cluster. Per ulteriori informazioni, consulta the section called "Preparazione per le disconnessioni". Il diagramma seguente mostra un'implementazione di un cluster locale.



I cluster locali sono generalmente disponibili per l'uso con i rack di Outposts.

Regioni AWS supportate

Puoi creare cluster locali nelle seguenti Regioni AWS: Stati Uniti orientali (Ohio), Stati Uniti orientali (Virginia settentrionale), Stati Uniti occidentali (California settentrionale), Stati Uniti occidentali (Oregon), Asia Pacifico (Seoul), Asia Pacifico (Singapore), Asia Pacifico (Sydney), Asia Pacifico (Tokyo), Canada (Centrale), Europa (Francoforte), Europa (Irlanda), Europa (Londra), Medio Oriente (Bahrein) e Sud America (San Paolo). Per informazioni dettagliate sulle funzionalità supportate, consulta la sezione the section called "Confronto tra le opzioni di implementazione".

Implementa un cluster Amazon EKS su Outposts AWS

Questo argomento offre una panoramica degli elementi da prendere in considerazione quando si esegue un cluster locale su un Outpost. L'argomento fornisce anche istruzioni su come implementare un cluster locale su un Outpost.

> **Important**
>
> - Queste considerazioni non sono trattate nella documentazione correlata di Amazon EKS. Se altri argomenti della documentazione di Amazon EKS sono in conflitto con le considerazioni qui riportate, segui le considerazioni qui.
> - Queste considerazioni sono soggette a modifiche e potrebbero cambiare frequentemente. Pertanto, ti consigliamo di rivedere regolarmente questo argomento.
> - Molte considerazioni sono diverse da quelle relative alla creazione di un cluster sul cloud. AWS

- I cluster locali supportano solo i rack dell'Outpost. Un singolo cluster locale può essere eseguito su più dell'Outpost fisici che comprendono un unico Outpost logico. Un singolo cluster locale non può essere eseguito su più Outpost logici. Ogni Outpost logico ha un singolo ARN dell'Outpost.
- I cluster locali eseguono e gestiscono il piano di controllo (control-plane) di Kubernetes nel tuo account sull'Outpost. Non puoi eseguire carichi di lavoro sulle istanze del piano di controllo (control-plane) di Kubernetes né modificare i relativi componenti. Questi nodi sono gestiti dal servizio Amazon EKS. Le modifiche al piano di controllo (control-plane) di Kubernetes non persistono tra un'operazione di gestione automatica di Amazon EKS e l'altra, come l'applicazione di patch.
- I cluster locali supportano i componenti aggiuntivi autogestiti e i gruppi di nodi Amazon Linux autogestiti. I componenti aggiuntivi: plug-in per la Container Network Interface (CNI) di Amazon VPC per Kubernetes, kube-proxy e CoreDNS vengono installati automaticamente nei cluster locali.
- I cluster locali richiedono l'utilizzo di Amazon EBS su Outposts. Il tuo Outpost deve avere a disposizione Amazon EBS per l'archiviazione del piano di controllo (control-plane) di Kubernetes. Gli Outpost supportano solo i volumi `gp2` di Amazon EBS.
- I `PersistentVolumes` Kubernetes sostenuti da Amazon EBS sono supportati tramite il driver CSI per Amazon EBS.
- Le istanze del piano di controllo (control-plane) dei cluster locali sono configurate in una topologia in stack ad alta disponibilità. Due delle tre istanze del piano di controllo (control-plane) devono essere sempre integre per mantenere il quorum. Se il quorum viene perso, contatta l' AWS assistenza, poiché saranno necessarie alcune azioni sul lato del servizio per abilitare le nuove istanze gestite.

Prerequisiti

- Familiarità con le opzioni di distribuzione di Outposts, Select Instance Types e Placement Groups per i cluster Amazon EKS AWS su Outposts in base a considerazioni sulla capacità e requisiti e considerazioni VPC.
- Un Outpost esistente. Per ulteriori informazioni, consulta What is AWS Outposts.
- Lo strumento a riga di comando `kubectl` è installato sul computer o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.
- Un principale IAM (utente o ruolo) con autorizzazioni `create` e `describe` per un cluster Amazon EKS. Per ulteriori informazioni, consultare the section called "Creazione di un cluster Kubernetes locale su un Outpost" e the section called "Elencare o descrivere tutti i cluster".

Quando viene creato un cluster Amazon EKS locale, il principale IAM che crea il cluster viene aggiunto in modo permanente. Il principale viene aggiunto specificamente alla tabella di autorizzazione RBAC Kubernetes come amministratore. Questa entità dispone di autorizzazioni `system:masters`. L'identità di questa entità non è visibile nella configurazione del cluster, per cui è fondamentale prendere nota dell'entità che ha creato il cluster, in modo da non eliminarla. Inizialmente, solo il principale che ha creato il server può effettuare chiamate al server API Kubernetes utilizzando `kubectl`. Se utilizzi la console per creare il cluster, assicurati che le stesse credenziali IAM siano presenti nella catena di credenziali AWS SDK quando `kubectl` esegui i comandi sul cluster. Dopo aver creato il cluster, è possibile concedere l'accesso ad altri principali IAM al cluster.

Creazione di un cluster locale Amazon EKS.

Puoi creare un cluster locale con i seguenti strumenti descritti in questa pagina:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

Puoi anche usare la AWS CLI, l'API Amazon EKS AWS SDKs, AWS CloudFormationo Terraform per creare cluster su Outposts.

`eksctl`

Per creare un cluster con **`eksctl`**

1. Installa la versione `0.215.0` o successiva dello strumento da riga di `eksctl` comando sul tuo dispositivo o. AWS CloudShell Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Copia i seguenti contenuti sul tuo dispositivo. Sostituisci i valori seguenti, quindi esegui il comando modificato per creare il file `outpost-control-plane.yaml`:
   - Sostituiscilo *`region-code`* con la AWS regione supportata in cui desideri creare il cluster.
   - Sostituisci *`my-cluster`* con un nome da assegnare al cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
   - Sostituisci *`vpc-ExampleID1`* e *`subnet-ExampleID1`* con il IDs VPC e la sottorete esistenti. Il VPC e la sottorete devono soddisfare i requisiti in Crea un VPC e sottoreti per i cluster Amazon EKS su Outposts. AWS
   - Sostituiscilo con *`uniqueid`* l'ID del tuo Outpost.
   - Sostituisci *`m5.large`* con un tipo di istanza disponibile sul tuo Outpost. Prima di scegliere un tipo di istanza, consulta la sezione the section called "Considerazioni sulla capacità". Vengono implementate tre istanze del piano di controllo. Questo numero non può essere modificato.

```
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
```

Per un elenco completo di tutte le opzioni disponibili, consulta AWS Outposts Support e Config file schema nella documentazione di `eksctl`.

3. Crea il cluster utilizzando il file di configurazione creato nel passaggio precedente. `eksctl` crea un VPC e una sottorete sul tuo Outpost in cui implementare il cluster.

```
eksctl create cluster -f outpost-control-plane.yaml
```

Il provisioning del cluster richiede diversi minuti. Durante la creazione del cluster, vengono visualizzate diverse righe di output. L'ultima riga di output è simile alla seguente riga di esempio.

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

> **Tip**
>
> Per visualizzare la maggior parte delle opzioni che è possibile specificare durante la creazione di un cluster con `eksctl`, utilizza il comando `eksctl create cluster --help`. Per visualizzare tutte le opzioni disponibili, puoi utilizzare un file `config`.

Per ulteriori informazioni, consulta Uso dei file config e lo Schema dei file config nella documentazione di `eksctl`. Gli esempi di file di configurazione sono disponibili su GitHub.

Il comando `eksctl` ha creato automaticamente una voce di accesso per il principale IAM (utente o ruolo) che ha creato il cluster e ha concesso a quest'ultimo autorizzazioni di amministratore per gli oggetti Kubernetes sul cluster. Se non desideri che il creatore del cluster disponga di un accesso di amministratore agli oggetti Kubernetes sul cluster, aggiungi il seguente testo al file di configurazione precedente: `bootstrapClusterCreatorAdminPermissions: false` (allo stesso livello di `metadata`, `vpc` e `outpost`). Se hai aggiunto l'opzione, dopo la creazione del cluster devi creare una voce di accesso per almeno un principale IAM, altrimenti nessun principale IAM avrà accesso agli oggetti Kubernetes sul cluster.

Console di gestione AWS

Per creare il cluster con la Console di gestione AWS

1. Hai bisogno di un VPC e una sottorete esistenti che soddisfano i requisiti di Amazon EKS. Per ulteriori informazioni, consulta the section called "Creazione di un VPC e delle sottoreti".
2. Se disponi già di un ruolo IAM del cluster locale o intendi creare il cluster con `eksctl`, puoi ignorare questo passaggio. Per impostazione predefinita, `eksctl` crea un ruolo per te.
   a. Per creare un file JSON della policy di attendibilità IAM, esegui il comando seguente.

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

   b. Crea il ruolo IAM del cluster Amazon EKS. Per creare un ruolo IAM, è necessario assegnare l'azione `iam:CreateRole` (autorizzazione) al principale IAM che sta creando il ruolo.

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

c. Allega la policy gestita di Amazon EKS denominata Amazon EKSLocal OutpostClusterPolicy al ruolo. Per allegare una policy IAM a un principale IAM, è necessario assegnare al principale che sta allegando la policy una delle azioni IAM (autorizzazioni) seguenti: `iam:AttachUserPolicy` o `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

3. Aprire la Console Amazon EKS.
4. Nella parte superiore dello schermo della console, assicurati di aver selezionato una AWS regione supportata.
5. Seleziona Aggiungi cluster e quindi Crea.
6. Nella pagina Configure cluster (Configura cluster), inserisci o seleziona i valori per i campi seguenti:
   - Posizione del piano di controllo Kubernetes: scegli Outposts. AWS
   - Outpost ID (ID Outpost): scegli l'ID dell'Outpost su cui vuoi creare il tuo piano di controllo.
   - Instance type (Tipo di istanza): seleziona un tipo di istanza. Vengono visualizzati solo i tipi di istanza disponibili nel tuo Outpost. Nell'elenco a discesa, ogni tipo di istanza descrive per quanti nodi è consigliato il tipo di istanza. Prima di scegliere un tipo di istanza, consulta la sezione the section called "Considerazioni sulla capacità". Tutte le repliche vengono implementate utilizzando lo stesso tipo di istanza. Non è possibile modificare il tipo di istanza dopo la creazione del cluster. Vengono implementate tre istanze del piano di controllo. Questo numero non può essere modificato.
   - Nome: un nome per il cluster. Deve essere univoca nel tuo account. AWS Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster. Il nome deve essere univoco all'interno AWS della regione e AWS dell'account in cui stai creando il cluster.
   - Versione Kubernetes: scegli la versione di Kubernetes da utilizzare per il cluster. È preferibile selezionare la versione più recente, a meno che non occorra utilizzare una versione precedente.

- Ruolo del servizio cluster: scegli il ruolo IAM del cluster Amazon EKS creato in un passaggio precedente per consentire al piano di controllo Kubernetes di gestire le risorse. AWS
- Accesso di amministratore del cluster Kubernetes: se desideri che il principale IAM (ruolo o utente) che crea il cluster disponga dell'accesso di amministratore agli oggetti Kubernetes sul cluster, accetta l'impostazione predefinita (Consenti). Amazon EKS crea una voce di accesso per il principale IAM e concede le autorizzazioni di amministratore del cluster per la voce di accesso. Per ulteriori informazioni sulle voci di accesso, consulta la sezione the section called "Voci di accesso".

  Se desideri che un principale IAM diverso da quello che crea il cluster disponga dell'accesso di amministratore agli oggetti del cluster Kubernetes, scegli l'opzione Non consentire. Dopo la creazione del cluster, qualsiasi principale IAM che disponga delle autorizzazioni IAM per creare voci di accesso può aggiungerne una per tutti i principali IAM che necessitano dell'accesso agli oggetti del cluster Kubernetes. Per ulteriori informazioni sulle autorizzazioni IAM richieste, consulta Actions defined by Amazon Elastic Kubernetes Service nella Documentazione di riferimento per l'autorizzazione ai servizi. Se scegli l'opzione Non consentire e non crei alcuna voce di accesso, nessun principale IAM avrà accesso agli oggetti Kubernetes sul cluster.
- Tag: (facoltativo) aggiunge eventuali tag al cluster. Per ulteriori informazioni, consulta the section called "Assegnazione di tag alle risorse ". Quando hai finito con questa pagina, seleziona Avanti.

7. Nella pagina Specifica reti, seleziona i valori dei campi riportati di seguito:
   - VPC: scegli un VPC esistente. Il VPC deve disporre di un numero sufficiente di indirizzi IP da mettere a disposizione per il cluster, i nodi e le altre risorse Kubernetes da creare. Il VPC deve soddisfare i requisiti elencati in VPC requirements and considerations.
   - Sottoreti: per impostazione predefinita, tutte le sottoreti disponibili nel VPC specificato nel campo precedente sono preselezionate. Le sottoreti scelte devono soddisfare i requisiti elencati in Subnet requirements and considerations.
   - Gruppi di sicurezza: (facoltativo) specifica uno o più gruppi di sicurezza da associare alle interfacce di rete create da Amazon EKS. Amazon EKS crea automaticamente un gruppo di sicurezza che consente la comunicazione tra il cluster e il VPC. Amazon EKS associa questo gruppo di sicurezza, e altri gruppi eventualmente scelti dall'utente, alle interfacce di rete create. Per ulteriori informazioni sul gruppo di sicurezza del cluster creato da Amazon EKS, consulta the section called "Requisiti relativi al gruppo di sicurezza". Puoi modificare le regole nel gruppo di sicurezza del cluster creato da Amazon EKS. Se vuoi aggiungere i tuoi gruppi di sicurezza, non potrai modificare i gruppi scelti dopo la creazione del cluster. Per consentire agli host on-premises di comunicare con l'endpoint del cluster, è necessario consentire il traffico in entrata

dal gruppo di sicurezza del cluster. Per i cluster che non dispongono di una connessione internet in ingresso e in uscita (noti anche come cluster privati), è necessario effettuare una delle seguenti operazioni:

- Aggiungi il gruppo di sicurezza associato agli endpoint VPC richiesti. Per ulteriori informazioni sugli endpoint richiesti, consulta la sezione Accesso tramite sottorete the section called "Utilizzo di endpoint VPC dell'interfaccia" ai servizi. AWS
- Modifica il gruppo di sicurezza creato da Amazon EKS per consentire il traffico proveniente dal gruppo di sicurezza associato agli endpoint VPC. Quando hai finito con questa pagina, seleziona Avanti.

8. Nella pagina Configura osservabilità, puoi scegliere facoltativamente le opzioni Parametri e Registrazione del piano di controllo che desideri attivare. Per impostazione predefinita, i tipi di log sono disattivati.
   - Per ulteriori informazioni sull'opzione relativa alle metriche di Prometheus, consulta the section called "Passaggio 1: attivazione delle metriche Prometheus".
   - Per ulteriori informazioni sulle opzioni Registrazione del piano di controllo, consulta the section called "Registrazione dei log del piano di controllo (control-plane)". Quando hai finito con questa pagina, seleziona Avanti.
9. Nella pagina Rivedi e crea, controlla le informazioni che hai inserito o selezionato nelle pagine precedenti. Se devi apportare modifiche, seleziona Edit (Modifica). Al termine della configurazione, seleziona Create (Crea). Durante il provisioning del cluster, nel campo Stato viene visualizzato il messaggio CREAZIONE.

   Il provisioning del cluster richiede diversi minuti.

Visualizzazione del cluster locale Amazon EKS

1. Dopo aver creato il cluster, puoi visualizzare le istanze del piano EC2 di controllo Amazon che sono state create.

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?
Key==`Name`]|[0].Value}' | grep my-cluster-control-plane
```

   Di seguito viene riportato un output di esempio.

```
"Name": "my-cluster-control-plane-id1"
"Name": "my-cluster-control-plane-id2"
```

```
"Name": "my-cluster-control-plane-id3"
```

Ogni istanza è contaminata dal taint `node-role.eks-local.amazonaws.com/control-plane` per evitare che i carichi di lavoro vengano pianificati sulle istanze del piano di controllo. Per ulteriori informazioni sui taint, consulta Taints and Tolerations nella documentazione di Kubernetes. Amazon EKS monitora continuamente lo stato dei cluster locali. Eseguiamo operazioni di gestione automatiche, come applicazioni di patch di sicurezza e riparazioni di istanze non integre. Quando i cluster locali vengono disconnessi dal cloud, eseguiamo una serie di operazioni per garantire che il cluster venga ripristinato a uno stato di integrità al momento della riconnessione.

2. Se hai creato il cluster utilizzando `eksctl`, puoi ignorare questo passaggio. `eksctl` lo completerà automaticamente per tuo conto. Abilita `kubectl` per consentire la comunicazione con il cluster aggiungendo un nuovo contesto al file `config` `kubectl`. Per istruzioni su come creare e aggiornare il file, consulta the section called "Accedere al cluster con kubectl".

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

   Di seguito viene riportato un output di esempio.

   ```
   Added new context arn:aws: eks:region-code:111122223333:cluster/my-cluster to /home/
   username/.kube/config
   ```

3. Per connetterti al server API Kubernetes del cluster locale, devi disporre dell'accesso al gateway locale per la sottorete o connetterti dall'interno del VPC. Per ulteriori informazioni sulla connessione di un rack Outpost alla rete locale, consulta Come funzionano i gateway locali per i rack nella Guida per l'utente di Outposts AWS . Se utilizzi l'instradamento VPC diretto e la sottorete dell'Outpost ha un instradamento al gateway locale, gli indirizzi IP privati delle istanze del piano di controllo (control-plane) di Kubernetes vengono trasmessi automaticamente sulla rete locale. L'endpoint del server API Kubernetes del server locale è ospitato in Amazon Route 53 (Route 53). L'endpoint del servizio API può essere risolto dai server DNS pubblici negli indirizzi IP privati dei server dell'API Kubernetes.

   Le istanze del piano di controllo (control-plane) di Kubernetes dei cluster locali sono configurate con interfacce di rete elastiche statiche con indirizzi IP privati fissi che non cambiano durante il ciclo di vita del cluster. Le macchine che interagiscono con il server API Kubernetes potrebbero non avere connettività a Route 53 durante le disconnessioni di rete. In tal caso, si consiglia di configurare `/etc/hosts` con gli indirizzi IP privati statici per continuare le operazioni. Ti

consigliamo inoltre di configurare i server DNS locali e di collegarli al tuo Outpost. Per ulteriori informazioni, consulta la documentazione di AWS Outposts. Esegui il comando riportato per confermare che è stata stabilita la comunicazione con il cluster.

```
kubectl get svc
```

Di seguito viene riportato un output di esempio.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

4. (Facoltativo) Verifica l'autenticazione del cluster locale quando si trova in uno stato disconnesso dal cloud. AWS Per istruzioni, consulta the section called "Preparazione per le disconnessioni".

Risorse interne

Amazon EKS crea le seguenti risorse sul tuo cluster. Le risorse sono destinate all'uso interno di Amazon EKS. Per garantire il corretto funzionamento del cluster, non modificare o alterare queste risorse.

- I seguenti pod mirror:
  - `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`*
- I seguenti componenti aggiuntivi autogestiti:
  - `kube-system/coredns`
  - `kube-system/ kube-proxy` (non viene creato finché non aggiungi il primo nodo)
  - `kube-system/aws-node` (non viene creato finché non si aggiunge il primo nodo). I cluster locali utilizzano il plug-in per la CNI di Amazon VPC per Kubernetes per le reti di cluster. Non modificare la configurazione per le istanze del piano di controllo (pod denominati `aws-node-controlplane-*`). Ci sono variabili di configurazione che consentono di modificare il valore predefinito quando il plugin crea nuove interfacce di rete. Per ulteriori informazioni, consulta la documentazione su GitHub.

- I seguenti servizi:
  - `default/kubernetes`
  - `kube-system/kube-dns`
- Una `PodSecurityPolicy` denominata `eks.system`
- Una `ClusterRole` denominata `eks:system:podsecuritypolicy`
- Una `ClusterRoleBinding` denominata `eks:system`
- Oltre al gruppo di sicurezza del cluster, Amazon EKS crea un gruppo di sicurezza denominato nel tuo AWS account`eks-local-internal-do-not-use-or-edit-`*`cluster-name`*`-`*`uniqueid`* . Questo gruppo di sicurezza consente al traffico di fluire liberamente tra i componenti Kubernetes in esecuzione sulle istanze del piano di controllo (control-plane).

Fasi successive consigliate:

- Concedi al responsabile IAM che ha creato il cluster le autorizzazioni necessarie per visualizzare le risorse Kubernetes nel Console di gestione AWS
- Concedere alle entità IAM l'accesso al cluster. Per consentire alle entità di visualizzare le risorse Kubernetes nella console Amazon EKS, concedi le relative autorizzazioni richieste.
- Configurare la registrazione per il cluster
- Acquisisci familiarità con ciò che accade durante le disconnessioni dalla rete.
- Aggiungere nodi al cluster
- Prendi in considerazione la possibilità di impostare un piano di backup per `etcd`. Amazon EKS non supporta il backup e il ripristino automatici di `etcd` per i cluster locali. Per ulteriori informazioni, consulta Backing up an etcd cluster nella documentazione di Kubernetes. Le due opzioni principali sono l'utilizzo di `etcdctl` per automatizzare l'acquisizione di snapshot o l'utilizzo del backup del volume di archiviazione di Amazon EBS.

Scopri le versioni della piattaforma Kubernetes e Amazon EKS per Outposts AWS

Le versioni della piattaforma cluster locale rappresentano le funzionalità del cluster Amazon EKS su AWS Outposts. Le versioni includono i componenti che vengono eseguiti sul piano di controllo (control-plane) Kubernetes, i cui flag del server API Kubernetes sono abilitati. Includono anche la versione corrente della patch di Kubernetes. Ogni versione secondaria Kubernetes dispone di una o più versioni della piattaforma associate. Le versioni della piattaforma per le diverse versioni

secondarie di Kubernetes sono indipendenti. Le versioni della piattaforma per i cluster locali e i cluster Amazon EKS nel cloud sono indipendenti.

Quando è disponibile una nuova versione secondaria di Kubernetes per i cluster locali, ad esempio `1.31`, la versione della piattaforma iniziale per tale versione secondaria di Kubernetes inizia da `eks-local-outposts.1`. Tuttavia, Amazon EKS rilascia periodicamente nuove versioni della piattaforma per abilitare nuove impostazioni del piano di controllo Kubernetes e fornire soluzioni per problemi relativi alla sicurezza.

Quando nuove versioni della piattaforma del cluster locale diventano disponibili per una versione secondaria:

- Il numero di versione della piattaforma viene incrementato (`eks-local-outposts.n+1`).
- Amazon EKS aggiorna automaticamente tutti i cluster locali esistenti alla versione della piattaforma più recente per la versione secondaria di Kubernetes corrispondente. Gli aggiornamenti automatici delle versioni della piattaforma esistenti vengono implementati in modo incrementale. Il processo di rollout consiste nella sostituzione delle istanze gestite del piano di controllo (control-plane) Kubernetes in esecuzione su Outpost, una alla volta, fino a quando tutte le 3 non vengono sostituite da istanze nuove.
- Il processo di sostituzione delle istanze del piano di controllo (control-plane) Kubernetes verrà arrestato in caso di rischio di interruzione del servizio. Amazon EKS cercherà di sostituire un'istanza del piano di controllo (control-plane) Kubernetes solo nel caso in cui le altre 2 siano integre e soddisfino tutte le condizioni di disponibilità come nodo del cluster.
- Il completamento del rollout di una versione della piattaforma richiede in genere meno di 30 minuti. Se un cluster rimane attivo `UPDATING` per un periodo di tempo prolungato, consulta the section called "Risoluzione dei problemi dei cluster" e chiedi aiuto a AWS Support. Non terminare mai manualmente le istanze del piano di controllo Kubernetes a meno che non venga richiesto da Support. AWS
- Amazon EKS potrebbe pubblicare un nuovo nodo AMI con una versione patch corrispondente. Tutte le versioni delle patch sono compatibili tra il piano di controllo e il nodo Kubernetes per una singola versione secondaria di Kubernetes. AMIs

Le nuove versioni della piattaforma non introducono modifiche in conflitto né causano interruzioni del servizio.

I cluster locali vengono sempre creati con la versione della piattaforma disponibile più recente (`eks-local-outposts.n`) per la versione Kubernetes specificata.

Le versioni della piattaforma correnti e recenti sono descritte nelle tabelle qui di seguito.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, è possibile iscriversi al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/outposts/eks-outposts-platform-versions.adoc.atom
```

Versione **1.31** di Kubernetes

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma `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`.

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
`1.31.14`	`eks-local-outposts.8`	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. kube-proxy aggiornato a. `v1.31.14` AWS IAM `v0.7.8` Authenticator aggiornato a. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.4`. Versione Bottleroc	23 dicembre 2025

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
		ket aggiornata a `v1.52.0`.	
`1.31.12`	`eks-local-outposts.5`	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. kube-proxy aggiornato a. `v1.31.10` AWS IAM `v0.7.4` Authentic ator aggiornato a. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.2`. Versione Bottleroc ket aggiornata a `v1.47.0`.	3 ottobre 2025

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
1.31.9	eks-local-outposts.4	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. kube-proxy aggiornato a. v1.31.9 AWS IAM v0.7.2 Authenticator aggiornato a. Plugin Amazon VPC CNI per Kubernetes aggiornato alla versione Bottlerocket aggiornata a. v1.20.0 v1.43.0	9 agosto 2025
1.31.7	eks-local-outposts.3	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. kube-proxy aggiornato a. v1.31.9 AWS IAM v0.7.1 Authenticator aggiornato a. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.5. Versione Bottlerocket aggiornata a v1.40.0.	19 giugno 2025

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
1.31.6	eks-local-outposts.2	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. Versione Bottlerocket aggiornata a v1.36.0.	24 aprile 2025
1.31.6	eks-local-outposts.1	Rilascio iniziale di Kubernetes versione v1.31 per cluster Amazon EKS locali su Outposts.	9 aprile 2025

Versione **1.30** di Kubernetes

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 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.

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
1.30.14	eks-local-outposts.10	Nuova versione della piattaforma con	23 dicembre 2025

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
		correzioni e miglioramenti di sicurezza. AWS IAM Authenticator aggiornato a. `v0.7.8` Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.4`. Versione Bottlerocket aggiornata a `v1.52.0`.	
`1.30.14`	`eks-local-outposts.7`	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. kube-proxy aggiornato a. `v1.30.14` AWS IAM `v0.7.4` Authenticator aggiornato a. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.2`. Versione Bottlerocket aggiornata a `v1.47.0`.	3 ottobre 2025

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
1.30.13	eks-local-outposts.6	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. kube-proxy aggiornato a. v1.30.13 AWS IAM v0.7.2 Authentic ator aggiornato a. Plug-in CNI di Amazon VPC per Kubernetes aggiornat o a v1.20.0. Versione Bottleroc ket aggiornata a v1.43.0.	09 agosto 2025
1.30.11	eks-local-outposts.5	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. kube-proxy aggiornato a. v1.30.11 AWS IAM v0.7.1 Authentic ator aggiornato a. Plugin Amazon VPC CNI per Kubernete s aggiornato alla versione Bottleroc ket aggiornata a. v1.19.5 v1.40.0	19 giugno 2025

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
1.30.10	eks-local-outposts.4	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. Versione Bottlerocket aggiornata a v1.36.0.	24 aprile 2025
1.30.10	eks-local-outposts.3	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. kube-proxy aggiornato a. v1.30.10 AWS IAM v0.6.29 Authenticator aggiornato a. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.2. CoredNS aggiornato a. v1.11.4 AWS Cloud Controller Manager aggiornato a. v1.30.8 Versione Bottlerocket aggiornata a a v1.34.0.	27 marzo 2025

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
1.30.7	eks-local-outposts.2	Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. kube-proxy aggiornato a. v1.30.7 AWS IAM v0.6.28 Authenticator aggiornato a. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.0. Versione di Bottlerocket aggiornata a v1.29.0.	10 gennaio 2025
1.30.5	eks-local-outposts.1	Rilascio iniziale di Kubernetes versione v1.30 per cluster Amazon EKS locali su Outposts.	13 novembre 2024

Versione **1.29** di Kubernetes

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 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.

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
1.29.15	eks-local-outposts.13	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. AWS IAM Authenticator aggiornato a v0.7.8 Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.20.4. Versione Bottlerocket aggiornata a v1.52.0.	23 dicembre 2025
v1.29.15	eks-local-outposts.10	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. AWS IAM Authenticator aggiornato a v0.7.4 Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.20.2. Versione Bottlerocket aggiornata a v1.47.0.	3 ottobre 2025
v1.29.15	eks-local-outposts.9	Nuova versione della piattaforma con	9 agosto 2025

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
		correzioni e miglioramenti di sicurezza. AWS IAM Authenticator aggiornato a. `v0.7.2` Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.20.0`. Versione Bottlerocket aggiornata a `v1.43.0`.	
`v1.29.15`	`eks-local-outposts.8`	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. kube-proxy aggiornato a. `v1.29.15` AWS IAM `v0.7.1` Authenticator aggiornato a. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a `v1.19.5`. Versione Bottlerocket aggiornata a `v1.40.0`.	19 giugno 2025

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
v1.29.14	eks-local-outposts.7	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. Versione Bottlerocket aggiornata a v1.36.0.	24 marzo 2025
v1.29.14	eks-local-outposts.6	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. kube-proxy aggiornato a v1.29.14. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.2. CoredNS aggiornato a. v1.11.4 AWS Cloud Controller Manager aggiornato a. v1.29.8 Versione Bottlerocket aggiornata a a v1.34.0.	27 marzo 2025

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
v1.29.11	eks-local-outposts.5	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. kube-proxy aggiornato a v1.29.11. Plug-in CNI di Amazon VPC per Kubernetes aggiornato a v1.19.0. Immagine CoreDNS aggiornata a v1.11.3. Versione di Bottlerocket aggiornata a v1.29.0.	10 gennaio 2025
1.29.9	eks-local-outposts.4	Nuova versione della piattaforma con correzioni e miglioramenti alla sicurezza. kube-proxy aggiornato a. v1.29.9 AWS IAM v0.6.26 Authenticator aggiornato a. Versione di Bottlerocket aggiornata a v1.26.0.	8 novembre 2024

Versione di Kubernetes	Versione della piattaforma Amazon EKS	Note di rilascio	Data di rilascio
1.29.6	eks-local-outposts.3	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. Versione di Bottlerocket aggiornata a v1.22.0.	22 ottobre 2024
1.29.6	eks-local-outposts.2	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. Versione di Bottlerocket aggiornata a v1.21.0.	27 agosto 2024
1.29.6	eks-local-outposts.1	Rilascio iniziale di Kubernetes versione v1.29 per cluster Amazon EKS locali su Outposts.	20 agosto 2024

Creazione di VPC e sottoreti per i cluster Amazon EKS su AWS Outposts

Durante la creazione di un cluster locale, in genere si specificano un VPC e almeno una sottorete privata in esecuzione su Outposts. Questo argomento offre una panoramica dei requisiti e delle considerazioni relativi al VPC e alle sottoreti utilizzate con il cluster locale.

Considerazioni e requisiti relativi al VPC

Durante la creazione di un cluster locale, il VPC specificato deve soddisfare i requisiti e le considerazioni seguenti:

- Assicurarsi che il VPC disponga di un numero sufficiente di indirizzi IP per il cluster locale, i nodi e le altre risorse Kubernetes da creare. Se il VPC da utilizzare non dispone di un numero sufficiente di indirizzi IP, aumentane il numero. Tale operazione può essere effettuata associando i blocchi di instradamento interdominio senza classi (CIDR) secondari al VPC. L'associazione di blocchi di CIDR privati (RFC 1918) e pubblici (non RFC 1918) può avvenire prima o dopo la creazione del cluster. Il riconoscimento del blocco CIDR associato a un VPC da parte del cluster può richiedere fino a cinque ore.
- Il VPC non può avere prefissi IP o intervalli CIDR IPv6 assegnati. A causa di queste limitazioni, le informazioni riportate alla pagina Assign more IP addresses to Amazon EKS nodes with prefixes e the section called "IPv6" non sono applicabili al VPC.
- Il VPC ha un nome host DNS e una risoluzione DNS abilitati. Senza queste funzionalità la creazione del cluster locale ha esito negativo, quindi dovrai abilitare le funzionalità e creare nuovamente il cluster. Per ulteriori informazioni, consulta Attributi DNS nel VPC nella Guida per l'utente di Amazon VPC.
- Per accedere al cluster locale tramite la rete locale, il VPC deve essere associato alla tabella di instradamento del gateway locale dell'Outpost. Per ulteriori informazioni, consultare la pagina VPC associations della Guida per l'utente di AWS Outposts.

Considerazioni e requisiti relativi alle sottoreti

Durante la creazione del cluster, è necessario specificare almeno una sottorete privata. Se si specificano più sottoreti, le istanze del piano di controllo (control-plane) Kubernetes sono distribuite uniformemente tra le sottoreti. Se viene specificata più di una sottorete, le sottoreti devono essere presenti sullo stesso Outpost. Tuttavia, per comunicare tra loro le sottoreti devono avere gli instradamenti corretti e le autorizzazioni del gruppo di sicurezza. Quando crei un cluster locale, le sottoreti specificate devono soddisfare i requisiti seguenti:

- Le sottoreti si trovano tutte sullo stesso Outpost logico.
- Le sottoreti devono avere complessivamente almeno tre indirizzi IP disponibili per le istanze del piano di controllo (control-plane) Kubernetes. Se sono specificate tre sottoreti, ogni sottorete deve avere almeno un indirizzo IP disponibile. Se sono specificate due sottoreti, ogni sottorete deve avere almeno due indirizzi IP disponibili. Se è specificata una sottorete, la sottorete deve avere almeno tre indirizzi IP disponibili.
- Le sottoreti hanno un istradamento al gateway locale del rack dell'Outpost per accedere al server API Kubernetes sulla rete locale. Se le sottoreti non dispongono di un instradamento per il gateway

locale del rack dell'Outpost, è necessario comunicare con il server API Kubernetes dall'interno del VPC.

- Le sottoreti devono utilizzare la denominazione basata sull'indirizzo IP. La denominazione basata sulle risorse di Amazon EC2 non è supportata da Amazon EKS.

Accesso della sottorete ai servizi AWS

Le sottoreti private del cluster locale su Outposts devono essere in grado di comunicare con i servizi AWS Regionali. Ciò è possibile utilizzando un gateway NAT per l'accesso a Internet in uscita o, se si desidera mantenere privato tutto il traffico all'interno del VPC, tramite gli endpoint VPC di interfaccia.

Utilizzo di un gateway NAT

Le sottoreti private del cluster locale su Outposts devono avere una tabella di routing associata con un instradamento a un gateway NAT in una sottorete pubblica che si trova nella zona di disponibilità parent dell'Outpost. La sottorete pubblica deve disporre di una route a un gateway Internet. Il gateway NAT consente l'accesso a Internet in uscita e impedisce le connessioni in entrata non richieste da Internet alle istanze sull'Outpost.

Utilizzo di endpoint VPC dell'interfaccia

Se le sottoreti private del cluster locale su Outposts non dispongono di una connessione Internet in uscita o se si desidera mantenere tutto il traffico privato all'interno del VPC, è necessario creare i seguenti endpoint VPC ed endpoint gateway in una sottorete Regionale prima di creare il cluster.

Endpoint	Endpoint type (Tipo di endpoint)
`com.amazonaws.` *`region-code`* `.ssm`	Interfaccia
`com.amazonaws.` *`region-code`* `.ssmmessages`	Interfaccia
`com.amazonaws.` *`region-code`* `.ec2messages`	Interfaccia
`com.amazonaws.` *`region-code`* `.ec2`	Interfaccia
`com.amazonaws.` *`region-code`* `.secretsmanager`	Interfaccia

Endpoint	Endpoint type (Tipo di endpoint)
com.amazonaws. *region-code* .logs	Interfaccia
com.amazonaws. *region-code* .sts	Interfaccia
com.amazonaws. *region-code* .ecr.api	Interfaccia
com.amazonaws. *region-code* .ecr.dkr	Interfaccia
com.amazonaws. *region-code* .s3	Gateway

Gli endpoint devono soddisfare i seguenti requisiti:

- Creato in una sottorete privata situata nella zona di disponibilità parent dell'Outpost
- Abilitazione dei nomi DNS privati
- Disponi di un gruppo di sicurezza collegato che consente il traffico HTTPS in ingresso dall'intervallo CIDR della sottorete privata dell'Outpost.

La creazione di endpoint comporta dei costi. Per ulteriori informazioni, consulta Prezzi di AWS PrivateLink. Se i pod hanno bisogno di accedere ad altri servizi AWS, sarà necessario creare degli endpoint aggiuntivi. Per un elenco completo degli endpoint, consultare la pagina AWS services that integrate with AWS PrivateLink.

Crea un VPC

È possibile creare un VPC che soddisfi i requisiti precedenti utilizzando uno dei seguenti modelli AWS CloudFormation:

- Modello 1: questo modello crea un VPC con una sottorete privata nell'Outpost e una sottorete pubblica nella Regione AWS. La sottorete privata ha un instradamento verso Internet tramite un gateway NAT che si trova nella sottorete pubblica nella Regione AWS. Questo modello può essere utilizzato per creare un cluster locale in una sottorete con accesso a Internet in uscita.
- Modello 2: questo modello crea un VPC con una sottorete privata sull'Outpost e il set minimo di endpoint VPC necessari per creare un cluster locale in una sottorete che non dispone di accesso a Internet in ingresso o in uscita (nota anche come sottorete privata).

Preparazione di cluster Amazon EKS locali su AWS Outposts per le disconnessioni di rete

Se la tua rete locale ha perso la connettività con il cloud AWS, puoi continuare a utilizzare il cluster Amazon EKS locale su un Outpost. Questo argomento illustra come preparare il cluster locale per le disconnessioni di rete e le considerazioni correlate.

- I cluster locali consentono la stabilità e la continua operatività durante le disconnessioni di rete temporanee e non pianificate. AWS Outposts rimane un'offerta completamente connessa che funge da estensione del cloud AWS nel tuo data center. In caso di disconnessioni dalla rete tra l'Outpost e il cloud AWS, ti consigliamo di provare a ripristinare la connessione. Per le relative istruzioni, consulta AWS Outposts rack network troubleshooting checklist nella Guida per l'utente di AWS Outposts. Per informazioni su come risolvere i problemi con i cluster locali, consulta the section called "Risoluzione dei problemi dei cluster".
- Gli Outpost emettono un parametro `ConnectedStatus` che puoi utilizzare per monitorare lo stato di connettività del tuo Outpost. Per ulteriori informazioni, consulta Outposts Metrics nella Guida per l'utente di AWS Outposts.
- I cluster locali utilizzano IAM come meccanismo di autenticazione predefinito tramite AWS Identity and Access Management authenticator for Kubernetes. IAM non è disponibile durante le disconnessioni di rete. Pertanto, i cluster locali supportano un meccanismo di autenticazione alternativo basato sui certificati `x.509` che puoi utilizzare per connetterti al cluster durante le disconnessioni dalla rete. Per informazioni su come ottenere e utilizzare un certificato `x.509` per il tuo cluster, consulta the section called "Autenticazione al cluster locale durante una disconnessione dalla rete".
- Se non riesci ad accedere a Route 53 durante le disconnessioni dalla rete, prendi in considerazione l'utilizzo dei server DNS locali nel tuo ambiente on-premises. Le istanze del piano di controllo (control-plane) di Kubernetes utilizzano indirizzi IP statici. Puoi configurare gli host che utilizzi per connetterti al cluster con il nome host e gli indirizzi IP dell'endpoint anziché utilizzare i server DNS locali. Per ulteriori informazioni, consulta DNS nella Guida per l'utente di AWS Outposts.
- Se prevedi un aumento del traffico delle applicazioni durante le disconnessioni dalla rete, puoi allocare capacità di elaborazione di riserva nel tuo cluster quando sei connesso al cloud. Le istanze Amazon EC2 sono incluse nel prezzo di AWS Outposts. Pertanto, l'esecuzione di istanze di riserva non influisce sui costi di utilizzo di AWS.
- Durante le disconnessioni dalla rete, per consentire le operazioni di creazione, aggiornamento e dimensionamento dei carichi di lavoro, le immagini di container dell'applicazione devono

essere accessibili sulla rete locale e il cluster deve disporre di una capacità sufficiente. I cluster locali non ospitano un registro dei container per tuo conto. Se i pod in precedenza sono stati eseguiti su tali nodi, le immagini di container vengono memorizzate nella cache dei nodi. Se in genere estrai le immagini dei container dell'applicazione da Amazon ECR nel cloud, prendi in considerazione l'esecuzione di una cache o di un registro locale. Una cache o un registro locale è utile se hai bisogno di creare, aggiornare e dimensionare le risorse del carico di lavoro durante le disconnessioni dalla rete.

- I cluster locali utilizzano Amazon EBS come classe di archiviazione predefinita per i volumi persistenti e il driver Amazon EBS CSI per gestire il ciclo di vita dei volumi persistenti di Amazon EBS. Durante le disconnessioni di rete, i pod supportati da Amazon EBS non possono essere creati, aggiornati o dimensionati. Questo perché queste operazioni richiedono chiamate all'API Amazon EBS nel cloud. Se stai implementando carichi di lavoro stateful su cluster locali e hai bisogno di creare, aggiornare o dimensionare le operazioni durante le disconnessioni dalla rete, prendi in considerazione l'utilizzo di un meccanismo di archiviazione alternativo.
- Gli snapshot di Amazon EBS non possono essere creati o eliminati se AWS Outposts non può accedere alle API pertinenti nella Regione AWS (ad esempio le API per Amazon EBS o Amazon S3).
- Quando integri ALB (ingresso) con Gestione certificati AWS (ACM), i certificati vengono inviati e archiviati nella memoria dell'istanza AWS Outposts ALB Compute. Il terminale TLS corrente continuerà a funzionare in caso di disconnessione dalla Regione AWS. La modifica delle operazioni in questo contesto non riuscirà (ad esempio nuove definizioni di ingresso, nuove operazioni dell'API su certificati ACM, dimensionamento del calcolo ALB o rotazione dei certificati). Per ulteriori informazioni, consulta Troubleshooting managed certificate renewal nella Guida per l'utente di Gestione certificati AWS.
- I registri del piano di controllo (control-plane) di Amazon EKS vengono memorizzati nella cache locale sulle istanze del piano di controllo (control-plane) di Kubernetes durante le disconnessioni dalla rete. Dopo la riconnessione, i log vengono inviati a CloudWatch Logs nella Regione AWS principale. Per monitorare il cluster in locale utilizzando l'endpoint dei parametri del server API Kubernetes o Fluent Bit per i log, puoi utilizzare Prometheus, Grafana e le soluzioni partner di Amazon EKS.
- Se stai utilizzando AWS Load Balancer Controller su Outposts per il traffico delle applicazioni, i pod esistenti anticipati da AWS Load Balancer Controller continueranno a ricevere traffico durante le disconnessioni di rete. I nuovi pod creati durante le disconnessioni dalla rete non ricevono traffico fino a quando l'Outpost non ristabilisce la connessione con il cloud AWS. Puoi impostare il numero

di repliche per le tue applicazioni mentre effettui la connessione al cloud AWS in base alle tue esigenze di dimensionamento durante le disconnessioni di rete.

- Il plug-in della CNI di Amazon VPC per Kubernetes utilizza per impostazione predefinita la modalità IP secondaria. È configurato con `WARM_ENI_TARGET=1`, che consente al plug-in di mantenere disponibile "un'interfaccia di rete completamente elastica" degli indirizzi IP disponibili. Puoi modificare i valori `WARM_ENI_TARGET`, `WARM_IP_TARGET` e `MINIMUM_IP_TARGET` in base alle tue esigenze di scalabilità durante uno stato di disconnessione. Per ulteriori informazioni, consulta il file readme per il plug-in su GitHub. Per un elenco del numero massimo di pod supportati da ciascun tipo di istanza, consulta il file eni-max-pods.txt su GitHub.

Autenticazione al cluster locale durante una disconnessione dalla rete

AWS Identity and Access Management (IAM) non è disponibile durante le disconnessioni di rete. Non puoi autenticarti nel cluster locale utilizzando le credenziali IAM durante una disconnessione. Tuttavia, durante la disconnessione, potrai connetterti al cluster tramite la rete locale utilizzando i certificati `x509`. È necessario scaricare e archiviare un certificato `X509` client da utilizzare durante le disconnessioni. In questo argomento viene illustrato come creare e utilizzare il certificato per autenticarsi nel cluster quando è disconnesso.

1. Creare una richiesta di firma del certificato.
   a. Genera una richiesta di firma del certificato.

```
openssl req -new -newkey rsa:4096 -nodes -days 365 \
    -keyout admin.key -out admin.csr -subj "/CN=admin"
```

   b. Crea una richiesta di firma del certificato in 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. Crea una richiesta di firma del certificato utilizzando `kubectl`.

```
kubectl create -f admin-csr.yaml
```

3. Controlla lo stato della richiesta di firma del certificato.

```
kubectl get csr admin-csr
```

   Di seguito viene riportato un output di esempio:

```
NAME       AGE   REQUESTOR                      CONDITION
admin-csr  11m   kubernetes-admin               Pending
```

   Kubernetes ha creato la richiesta di firma del certificato.

4. Approva la richiesta di firma del certificato.

```
kubectl certificate approve admin-csr
```

5. Controlla nuovamente lo stato della richiesta di firma del certificato per l'approvazione.

```
kubectl get csr admin-csr
```

   Di seguito viene riportato un output di esempio:

```
NAME       AGE   REQUESTOR                     CONDITION
admin-csr  11m   kubernetes-admin              Approved
```

6. Recupera e verifica il certificato.

   a. Recupera il certificato.

```
kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode >
 admin.crt
```

   b. Verifica il certificato.

```
cat admin.crt
```

7. Crea un'associazione di ruoli del cluster per un utente `admin`.

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
    --user=admin --group=system:masters
```

8. Genera un kubeconfig con ambito utente per uno stato disconnesso.

   Puoi generare un file `kubeconfig` utilizzando il certificati `admin` scaricati. Sostituisci *`my-cluster`* e *`apiserver-endpoint`* nei comandi seguenti.

```
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. Visualizza il file `kubeconfig`.

```
kubectl get nodes --kubeconfig admin.kubeconfig
```

10. Se disponi di servizi già in produzione sul tuo Outpost, salta questo passaggio. Se Amazon EKS è l'unico servizio in esecuzione sul tuo Outpost e quest'ultimo non è attualmente in produzione, puoi simulare una disconnessione di rete. Prima di entrare in produzione con il cluster locale, puoi simulare una disconnessione per assicurarti di riuscire ad accedere al cluster quando è in uno stato disconnesso.

    a. Applica le regole del firewall sui dispositivi di rete che connettono il tuo Outpost alla Regione AWS. Questo disconnette il link del servizio dell'Outpost. Non puoi creare nuove istanze. Le istanze attualmente in esecuzione perdono la connettività con la Regione AWS e internet.

b. Puoi testare la connessione al cluster locale durante una disconnessione tramite il certificato x509. Assicurati di modificare la kubeconfig con il admin.kubeconfig che hai creato in una fase precedente. Sostituisci *my-cluster* con il nome del cluster locale.

```
kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
```

Se riscontri problemi con i cluster locali mentre sono disconnessi, ti consigliamo di inviare una richiesta di supporto.

Selezione dei tipi di istanze e dei gruppi di posizionamento per i cluster Amazon EKS su AWS Outposts in base alle considerazioni sulla capacità

Questo argomento fornisce assistenza per la selezione del tipo di istanza del piano di controllo (control-plane) di Kubernetes e (facoltativamente) per l'uso di gruppi di posizionamento per soddisfare requisiti di alta disponibilità per il tuo cluster Amazon EKS locale su un Outpost.

Prima di selezionare un tipo di istanza (m5, c5 o r5) da utilizzare per il piano di controllo (control-plane) di Kubernetes del cluster locale su Outposts, devi verificare i tipi di istanza disponibili nella configurazione dell'Outpost. Dopo aver identificato i tipi di istanza disponibili, devi selezionare la dimensione dell'istanza (large, xlarge o 2xlarge) in base al numero di nodi richiesti dai carichi di lavoro. La tabella seguente fornisce consigli per la selezione della dimensione dell'istanza.

> **Note**
>
> Le dimensioni delle istanze devono essere assegnate ai tuoi Outposts. Assicurati di disporre di una capacità sufficiente per tre istanze della dimensione disponibile negli Outposts per tutta la durata del cluster locale. Per un elenco dei tipi di istanza Amazon EC2 disponibili, consulta le sezioni relative al calcolo e all'archiviazione in Funzionalità dei rack AWS Outposts.

Numero di nodi	Dimensione dell'istanza del piano di controllo (control-plane) Kubernetes
1-20	large
21-100	xlarge

Numero di nodi	Dimensione dell'istanza del piano di controllo (control-plane) Kubernetes
101-250	`2xlarge`
251-500	`4xlarge`

L'archiviazione per il piano di controllo (control-plane) di Kubernetes richiede 246 GB di spazio di archiviazione Amazon EBS per cluster locale per soddisfare gli IOPS richiesti per `etcd`. Quando viene creato il cluster, i volumi Amazon EBS vengono allocati automaticamente per tuo conto.

Collocamento del piano di controllo (control-plane)

Se non specifichi un gruppo di posizionamento con la proprietà `OutpostConfig.ControlPlanePlacement.GroupName`, le istanze Amazon EC2 fornite per il tuo piano di controllo (control-plane) di Kubernetes non ricevono alcuna specifica applicazione del collocamento dell'hardware nella capacità sottostante disponibile sul tuo Outpost.

Puoi utilizzare gruppi di collocamento per soddisfare i requisiti di alta disponibilità del tuo cluster Amazon EKS locale su un Outpost. Specificando un gruppo di posizionamento durante la creazione del cluster, influisci sul collocamento delle istanze del piano di controllo (control-plane) di Kubernetes. Le istanze sono distribuite su un hardware sottostante indipendente (rack o host), riducendo al minimo l'impatto di istanze correlate in caso di guasti hardware.

Il tipo di spread che puoi configurare dipende dal numero di rack Outpost esistenti nella tua implementazione.

- Implementazioni con uno o due rack fisici in un unico Outpost logico: occorrono almeno tre host configurati con il tipo di istanza scelto per le tue istanze del piano di controllo (control plane) di Kubernetes. Un gruppo di posizionamento degli spread che utilizza lo spread a livello di host garantisce che tutte le istanze del piano di controllo (control-plane) di Kubernetes vengano eseguite su host distinti all'interno dei rack sottostanti disponibili nella tua implementazione Outpost.
- Implementazioni con tre o più rack fisici in un unico Outpost logico: occorrono almeno tre host configurati con il tipo di istanza scelto per le tue istanze del piano di controllo (control plane) di Kubernetes. Un gruppo di posizionamento degli spread che utilizza lo spread a livello di rack garantisce che tutte le istanze del piano di controllo (control-plane) di Kubernetes vengano eseguite su rack distinti nella tua implementazione Outpost. In alternativa, puoi utilizzare il gruppo di collocamento spread a livello di host come descritto nell'opzione precedente.

Sei responsabile della creazione del gruppo di collocamento desiderato. Specifichi il gruppo di collocamento quando richiami l'API `CreateCluster`. Per ulteriori informazioni, sui gruppi di posizionamento e per le istruzioni per crearli, consulta Placement Groups nella Guida per l'utente di Amazon EC2.

- Quando viene specificato un gruppo di collocamento, deve essere disponibile una capacità slot sull'Outpost per creare correttamente un cluster Amazon EKS locale. La capacità varia a seconda se utilizzi il tipo di spread host o rack. Se la capacità è insufficiente, il cluster rimane nello stato `Creating`. Puoi controllare `Insufficient Capacity Error` nel campo di integrità della risposta dell'API DescribeCluster. Devi liberare capacità per l'avanzamento del processo di creazione.
- Durante gli aggiornamenti della piattaforma e della versione del cluster locale Amazon EKS, le istanze del piano di controllo (control-plane) di Kubernetes dal tuo cluster vengono sostituite da nuove istanze utilizzando una strategia di aggiornamento in sequenza. Durante questo processo di sostituzione, ogni istanza del piano di controllo (control plane) termina, liberando il rispettivo slot. Al suo posto viene fornita una nuova istanza aggiornata. L'istanza aggiornata potrebbe essere collocata nello slot che è stato rilasciato. Se lo slot viene utilizzato da un'altra istanza non correlata e la capacità rimanente non soddisfa i requisiti di topologia spread richiesti, il cluster rimane nello stato `Updating`. Puoi controllare il rispettivo `Insufficient Capacity Error` nel campo di integrità della risposta dell'API DescribeCluster. Devi liberare la capacità per l'avanzamento del processo di aggiornamento e la ridefinizione dei livelli di alta disponibilità precedenti.
- In ogni Regione AWS puoi creare un massimo di 500 gruppi di posizionamento per account. Per ulteriori informazioni, consulta General rules and limitations nella Guida per l'utente di Amazon EC2.

Risolvi i problemi relativi ai cluster Amazon EKS locali su Outposts AWS

Questo argomento illustra alcuni errori comuni che si potrebbero verificare durante l'utilizzo dei cluster locali e il modo in cui risolverli. I cluster locali siano simili ai cluster Amazon EKS sul cloud, ma esistono delle differenze nel modo in cui sono gestiti da Amazon EKS.

> **Important**
>
> Non terminare mai alcuna istanza del `Kubernetes` piano di controllo del cluster locale EKS gestita in esecuzione su Outpost a meno che non venga esplicitamente richiesto da Support. AWS La terminazione di queste istanze comporta un rischio per la disponibilità dei servizi del cluster locale, inclusa la perdita del cluster locale nel caso in cui più istanze vengano

terminate contemporaneamente. Le istanze del `Kubernetes` piano di controllo del cluster locale EKS sono identificate dal tag sulla console dell'istanza. `eks-local:controlplane-name` EC2

Funzionamento dell'API

I cluster locali vengono creati tramite l'API Amazon EKS, ma vengono eseguiti in modo asincrono. Ciò significa che le richieste all'API Amazon EKS vengono restituite immediatamente per i cluster locali. Tuttavia, queste richieste potrebbero avere esito positivo, anticipare l'errore a causa di errori di convalida degli input oppure fallire e riportare errori di convalida descrittivi. Questo funzionamento è simile a quello dell'API Kubernetes.

I cluster locali non effettuano la transizione a uno stato `FAILED`. Amazon EKS prova a riconciliare lo stato del cluster con lo stato desiderato richiesto dall'utente in modo continuo. Di conseguenza, un cluster locale può rimanere a lungo nello stato `CREATING`, fino a quando il problema di base non viene risolto.

Descrizione del campo di integrità del cluster

I problemi relativi ai cluster locali possono essere scoperti utilizzando il comando describe-cluster Amazon EKS AWS CLI. I problemi relativi ai cluster locali vengono evidenziati dal campo `cluster.health` della risposta del comando `describe-cluster`. Il messaggio contenuto in questo campo include un codice di errore, un messaggio descrittivo e una risorsa correlata. IDs Queste informazioni sono disponibili solo tramite l'API e la AWS CLI di Amazon EKS. Nell'esempio seguente, sostituiscilo *`my-cluster`* con il nome del tuo cluster locale.

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

Di seguito viene riportato un output di esempio.

```
{
    "issues": [
        {
            "code": "ConfigurationConflict",
            "message": "The instance type 'm5.large' is not supported in Outpost 'my-
outpost-arn'.",
            "resourceIds": [
                "my-cluster-arn"
            ]
```

```
        }
    ]
}
```

Se il problema non è risolvibile, potrebbe essere necessario eliminare il cluster locale e crearne uno nuovo. Ad esempio, ciò potrebbe verificarsi nel caso in cui si provi ad allocare un cluster con un tipo di istanza che non è disponibile sul tuo Outpost. La tabella seguente include gli errori più comuni relativi allo stato di integrità.

Scenario di errore	Codice	Messaggio	ResourceIds
Non è stato possibile trovare le sottoreti fornite.	`ResourceNotFound`	`The subnet ID` *`subnet-id`* `does not exist`	Tutte le sottoreti fornite IDs
Le sottoreti fornite non appartengono allo stesso VPC.	`ConfigurationConflict`	`Subnets specified must belong to the same VPC`	Tutte le sottoreti fornite IDs
Alcune delle sottoreti fornite non appartengono all'Outpost specificato.	`ConfigurationConflict`	`Subnet` *`subnet-id`* `expected to be in` *`outpost-arn`* `, but is in` *`other-outpost-arn`*	ID della sottorete problematico
Alcune delle sottoreti fornite non appartengono ad alcun Outpost.	`ConfigurationConflict`	`Subnet` *`subnet-id`* `is not part of any Outpost`	ID della sottorete problematico
Alcune sottoreti fornite non dispongono di indirizzi liberi sufficienti per creare interfacce di rete elastiche per le istanze del piano di controllo (control-plane).	`ResourceLimitExceeded`	`The specified subnet does not have enough free addresses to satisfy the request.`	ID della sottorete problematico

Scenario di errore	Codice	Messaggio	ResourceIds
Il tipo di istanza del piano di controllo (control-plane) specificato non è supportato sul tuo Outpost.	`ConfigurationConflict`	`The instance type type is not supported in Outpost outpost-arn`	ARN del cluster
Hai terminato un' EC2 istanza Amazon del piano di controllo o l'operazione `run-instance` è stata completata, ma lo stato ha rilevato modifiche a. `Terminated` Ciò può verificarsi per un periodo di tempo dopo la riconnessione di Outpost e gli errori interni di Amazon EBS causano il fallimento di un flusso di lavoro EC2 interno di Amazon.	`InternalFailure`	`EC2 instance state "Terminated" is unexpected`	ARN del cluster
Capacità insufficiente nell'Outpost. Ciò può verificarsi anche quando viene creato un cluster se un Outpost è disconnesso dalla regione. AWS	`ResourceLimitExceeded`	`There is not enough capacity on the Outpost to launch or start the instance.`	ARN del cluster

Scenario di errore	Codice	Messaggio	ResourceIds
Il tuo account supera la quota del gruppo di sicurezza	`ResourceLimitExceeded`	Messaggio di errore restituito dall' EC2 API Amazon	ID del VPC di destinazione
Il tuo account supera la quota dell'interfaccia di rete elastica	`ResourceLimitExceeded`	Messaggio di errore restituito dall' EC2 API Amazon	ID della sottorete di destinazione
Le istanze del piano di controllo non erano raggiungibili tramite Systems Manager AWS . Per la risoluzione, consulta la sezione the section called “Le istanze del piano di controllo non sono raggiungibili tramite Systems Manager AWS”.	`ClusterUnreachable`	Le istanze del piano di controllo di Amazon EKS non sono raggiungibili tramite SSM. Verifica la configurazione SSM e di rete e fai riferimento alla documentazione per la risoluzione dei problemi di EKS su Outposts.	EC2 Istanza Amazon IDs
Si è verificato un errore durante la raccolta dei dettagli per un gruppo di sicurezza gestito o un'interfaccia di rete elastica.	Basato sul codice di errore EC2 del client Amazon.	Messaggio di errore restituito dall' EC2 API Amazon	Tutti i gruppi di sicurezza gestiti IDs

Scenario di errore	Codice	Messaggio	ResourceIds
Si è verificato un errore durante l'autorizzazione o la revoca delle regole di ingresso dei gruppi di sicurezza. Questo vale per i gruppi di sicurezza sia del cluster sia del piano di controllo.	Basato sul codice di errore EC2 del client Amazon.	Messaggio di errore restituito dall' EC2 API Amazon	ID gruppo di sicurezza problematico
Si è verificato un errore durante l'eliminazione di un'interfaccia di rete elastica per un'istanza del piano di controllo	Basato sul codice di errore EC2 del client Amazon.	Messaggio di errore restituito dall' EC2 API Amazon	ID dell'interfaccia di rete elastica problematica

La tabella seguente elenca gli errori di altri AWS servizi presentati nel campo relativo allo stato della `describe-cluster` risposta.

Codice EC2 di errore Amazon	Codice del problema di integrità del cluster	Description
`AuthFailure`	`AccessDenied`	Questo problema può verificarsi per una serie di motivi. Di solito si verificano se un tag utilizzato dal servizio per definire la policy dei ruoli collegati al servizio viene rimosso accidentalmente dal piano di controllo. In tal caso, Amazon EKS non può più

Codice EC2 di errore Amazon	Codice del problema di integrità del cluster	Description
		gestire e monitorare queste AWS risorse.
`UnauthorizedOperation`	`AccessDenied`	Questo problema può verificarsi per una serie di motivi. Di solito si verificano se un tag utilizzato dal servizio per definire la policy dei ruoli collegati al servizio viene rimosso accidentalmente dal piano di controllo. In tal caso, Amazon EKS non può più gestire e monitorare queste AWS risorse.
`InvalidSubnetID.NotFound`	`ResourceNotFound`	Questo errore si verifica quando non viene trovato l'ID di sottorete per le regole di ingresso di un gruppo di sicurezza.
`InvalidPermission.NotFound`	`ResourceNotFound`	Questo errore si verifica quando le autorizzazioni per le regole di ingresso di un gruppo di sicurezza non sono corrette.
`InvalidGroup.NotFound`	`ResourceNotFound`	Questo errore si verifica quando non viene trovato il gruppo delle regole di ingresso di un gruppo di sicurezza.

Codice EC2 di errore Amazon	Codice del problema di integrità del cluster	Description
`InvalidNetworkInterfaceID.NotFound`	`ResourceNotFound`	Questo errore si verifica quando non viene trovato l'ID dell'interfaccia di rete per le regole di ingresso di un gruppo di sicurezza.
`InsufficientFreeAddressesInSubnet`	`ResourceLimitExceeded`	Questo errore si verifica quando viene superata la quota di risorse della sottorete.
`InsufficientCapacityOnOutpost`	`ResourceLimitExceeded`	Questo errore si verifica quando viene superata la quota di capacità dell'outpost.
`NetworkInterfaceLimitExceeded`	`ResourceLimitExceeded`	Questo errore si verifica quando viene superata la quota dell'interfaccia di rete elastica.
`SecurityGroupLimitExceeded`	`ResourceLimitExceeded`	Questo errore si verifica quando viene superata la quota del gruppo di sicurezza.

Codice EC2 di errore Amazon	Codice del problema di integrità del cluster	Description
`VcpuLimitExceeded`	`ResourceLimitExceeded`	Ciò si verifica quando si crea un' EC2 istanza Amazon in un nuovo account. L'errore potrebbe essere simile al seguente: "`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`	Amazon EC2 restituisce questo codice di errore se il tipo di istanza specificato non è supportato su Outpost.
Tutti gli altri errori	`InternalFailure`	Nessuno

Impossibile creare o modificare i cluster

I cluster locali richiedono autorizzazioni e policy diverse rispetto ai cluster Amazon EKS ospitati nel cloud. Quando un cluster non riesce a creare e produce un `InvalidPermissions` errore, ricontrolla che al ruolo del cluster che stai utilizzando sia associata la policy EKSLocal OutpostClusterPolicy gestita da Amazon. Tutte le altre chiamate API richiedono lo stesso set di autorizzazioni dei cluster Amazon EKS sul cloud.

Il cluster è bloccato nello stato **`CREATING`**

La quantità di tempo necessaria per creare un cluster locale varia a seconda di diversi fattori. Questi fattori includono la configurazione della rete, dell'Outpost e del cluster. In generale, viene creato un cluster locale che passa allo stato `ACTIVE` entro 15-20 minuti. Se un cluster locale rimane nello stato `CREATING`, puoi richiamare `describe-cluster` per informazioni sulla causa nel campo di output `cluster.health`.

I problemi più comuni sono i seguenti:

- Il cluster non può connettersi all'istanza del piano di controllo dalla AWS regione in cui si trova Systems Manager. Puoi eseguire una verifica chiamando `aws ssm start-session --target` *`instance-id`* da un host bastione nella regione. Se il comando non funziona, controlla se Systems Manager è in esecuzione sull'istanza del piano di controllo (control-plane). Oppure, un'altra soluzione alternativa consiste nell'eliminare il cluster e quindi ricrearlo.
- Le istanze del piano di controllo (control-plane) non vengono create a causa delle autorizzazioni della chiave KMS per i volumi EBS. Con le chiavi KMS gestite dall'utente per i volumi EBS crittografati, le istanze del piano di controllo (control-plane) termineranno se la chiave non è accessibile. Se le istanze vengono terminate, passa a una chiave KMS AWS gestita o assicurati che la politica delle chiavi gestite dagli utenti conceda le autorizzazioni necessarie per il ruolo del cluster.
- Le istanze del piano di controllo di Systems Manager potrebbero non avere accesso a Internet. Verifica se la sottorete fornita durante la creazione del cluster dispone di un gateway NAT e un VPC con un gateway Internet. Usa VPC Reachability Analyzer per verificare se l'istanza del control plane può raggiungere il gateway Internet. Per ulteriori informazioni, consulta la sezione Getting started with VPC Reachability Analyzer (Nozioni di base su VPC Reachability Analyzer).
- L'ARN del ruolo fornito è privo di alcune policy. Verifica se la policy AWS gestita: Amazon EKSLocal OutpostClusterPolicy è stata rimossa dal ruolo. Ciò può verificarsi anche se uno AWS CloudFormation stack non è configurato correttamente.
- Tutte le sottoreti fornite devono essere associate allo stesso Outpost ed essere in grado di raggiungersi tra loro. Quando vengono specificate più sottoreti durante la creazione del cluster, Amazon EKS prova a distribuire le istanze del piano di controllo su più sottoreti.
- I gruppi di sicurezza gestiti da Amazon EKS vengono applicati all'interfaccia di rete elastica. Tuttavia, altri elementi di configurazione, come le regole del firewall NACL, potrebbero essere in conflitto con le regole per l'interfaccia di rete elastica.

La configurazione del DNS del VPC e della sottorete non è configurata correttamente o è assente

Consulta Creare un VPC e sottoreti per i cluster Amazon EKS su Outposts. AWS

Il cluster è bloccato nello stato **`UPDATING`**

Amazon EKS aggiorna automaticamente tutti i cluster locali esistenti alla versione della piattaforma più recente per la versione secondaria di Kubernetes corrispondente. Per ulteriori informazioni sulle versioni della piattaforma, consulta the section called "Versioni della piattaforma EKS".

Durante l'implementazione automatica di una versione della piattaforma, lo stato di un cluster cambia in `UPDATING`. Il processo di aggiornamento consiste nella sostituzione di tutte le istanze del piano di controllo (control-plane) Kubernetes con nuove istanze contenenti le patch di sicurezza e le correzioni di bug più recenti rilasciate per la rispettiva versione secondaria di Kubernetes. In generale, il processo di aggiornamento della piattaforma di un cluster locale viene completato in meno di 30 minuti e il cluster torna allo stato `ACTIVE`. Se un cluster locale rimane nello stato `UPDATING` per un periodo di tempo prolungato, puoi chiamare `describe-cluster` per controllare le informazioni sulla causa nel campo di output `cluster.health`.

Amazon EKS garantisce che almeno due istanze del piano di controllo (control-plane) Kubernetes su tre siano nodi del cluster integri e operativi, al fine di mantenere la disponibilità del cluster locale e prevenire l'interruzione del servizio. Se un cluster locale è bloccato in uno stato `UPDATING`, di solito è perché c'è qualche problema di infrastruttura o di configurazione che impedisce di garantire la disponibilità minima delle due istanze nel caso in cui il processo continui. Pertanto, il processo di aggiornamento arresta l'avanzamento per proteggere l'interruzione del servizio del cluster locale.

È importante risolvere i problemi di un cluster locale bloccato nello stato `UPDATING` e risolvere la causa principale in modo che il processo di aggiornamento possa essere completato e ripristinare il cluster locale nello stato `ACTIVE` con l'elevata disponibilità di tre istanze del piano di controllo (control-plane) Kubernetes.

Non terminare alcuna `Kubernetes` istanza di cluster locale EKS gestita su Outposts a meno che non venga esplicitamente richiesto da Support. AWS Ciò è particolarmente importante per i cluster locali bloccati nello stato `UPDATING`, perché esiste un'alta probabilità che altri nodi del piano di controllo (control-plane) non siano completamente integri e la terminazione dell'istanza sbagliata potrebbe causare l'interruzione del servizio e il rischio di perdita dei dati del cluster locale.

I problemi più comuni sono i seguenti:

- Una o più istanze del piano di controllo (control-plane) non sono in grado di connettersi a System Manager a causa di una modifica della configurazione di rete avvenuta dopo la prima creazione del cluster locale. Puoi eseguire una verifica chiamando `aws ssm start-session --target instance-id` da un host bastione nella regione. Se il comando non funziona, controlla se Systems Manager è in esecuzione sull'istanza del piano di controllo (control-plane).
- Impossibile creare nuove istanze del piano di controllo (control-plane) a causa delle autorizzazioni chiave KMS per i volumi EBS. Con le chiavi KMS gestite dall'utente per i volumi EBS crittografati, le istanze del piano di controllo (control-plane) termineranno se la chiave non è accessibile. Se le istanze vengono terminate, passa a una chiave KMS gestita o assicurati che la politica delle chiavi AWS gestite dagli utenti conceda le autorizzazioni necessarie per il ruolo del cluster.
- Le istanze del piano di controllo (control-plane) di Systems Manager potrebbero aver perso l'accesso a internet. Verifica se la sottorete fornita durante la creazione del cluster dispone di un gateway NAT e un VPC con un gateway Internet. Usa VPC Reachability Analyzer per verificare se l'istanza del control plane può raggiungere il gateway Internet. Per ulteriori informazioni, consulta la sezione Getting started with VPC Reachability Analyzer (Nozioni di base su VPC Reachability Analyzer). Se le reti private non dispongono di una connessione internet in uscita, assicurati che tutti gli endpoint VPC e gli endpoint gateway richiesti siano ancora presenti nella sottorete regionale del cluster (consulta the section called "Accesso della sottorete ai servizi AWS").
- L'ARN del ruolo fornito è privo di alcune policy. Verifica se la policy AWS gestita: Amazon non EKSLocal OutpostClusterPolicy è stata rimossa dal ruolo.
- Una delle nuove istanze del piano di controllo (control-plane) Kubernetes potrebbe aver subito un errore di bootstrap imprevisto. Invia un ticket al centro di Supporto AWS per ulteriori indicazioni sulla risoluzione dei problemi e sulla raccolta dei log in questo caso eccezionale.

Impossibile unire i nodi a un cluster

- Problemi relativi alle AMI:
  - Stai usando un'AMI incompatibile. È supportato solo Amazon Linux 2 AMIs ottimizzato per Amazon EKS (`amazon-linux-2`,`amazon-linux-2-gpu`,`amazon-linux-2-arm64`). Se tenti di unire AL2 023 nodi a EKS LocalClusters su AWS Outposts, i nodi non riescono a unirsi al cluster. Per ulteriori informazioni, consulta Creare nodi Amazon Linux su AWS Outposts.
  - Se hai usato un AWS CloudFormation modello per creare i tuoi nodi, assicurati che non usasse un'AMI non supportata.
- Manca l' AWS IAM Authenticator`ConfigMap`: se manca, devi crearlo. Per ulteriori informazioni, consulta the section called "Applica la `aws-auth ConfigMap` al cluster".

- È stato utilizzato un gruppo di sicurezza non corretto: assicurati di utilizzare `eks-cluster-sg-`*`cluster-name`*`-`*`uniqueid`* per il gruppo di sicurezza dei nodi worker. Il gruppo di sicurezza selezionato viene modificato AWS CloudFormation per consentire un nuovo gruppo di sicurezza ogni volta che viene utilizzato lo stack.
- A seguito dei passaggi del VPC di collegamento privato imprevisto: vengono passati dati CA (`--b64-cluster-ca`) o API Endpoint (`--apiserver-endpoint`) errati.

Raccolta di registri

Quando un Outpost viene disconnesso dalla AWS regione a cui è associato, è probabile che il cluster Kubernetes continui a funzionare normalmente. Tuttavia, se il cluster non funziona correttamente, segui la procedura di risoluzione dei problemi in Preparare i cluster Amazon EKS locali su AWS Outposts per le disconnessioni di rete. Se riscontri altri problemi, contatta l' AWS assistenza. AWS Support può aiutarti a scaricare ed eseguire uno strumento di raccolta dei log. In questo modo, puoi raccogliere i log dalle istanze del piano di controllo del cluster Kubernetes e inviarli all'assistenza AWS Support per ulteriori indagini.

Le istanze del piano di controllo non sono raggiungibili tramite Systems Manager AWS

Quando le istanze del piano di controllo di Amazon EKS non sono raggiungibili tramite AWS Systems Manager (Systems Manager), Amazon EKS visualizza il seguente errore per il 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.
```

Per risolvere questo problema, assicurati che il tuo VPC e le sottoreti soddisfino i requisiti in Creare un VPC e sottoreti per i cluster Amazon EKS su AWS Outposts e di aver completato i passaggi descritti nella Configurazione di Session Manager nella Systems Manager User Guide. AWS

Crea nodi Amazon Linux su AWS Outposts

> **Important**
>
> Amazon EKS Local Clusters on Outposts supporta solo nodi creati dai seguenti Amazon Linux 2023 ottimizzati per Amazon EKS: AMIs
>
> - Amazon Linux 2023 standard (`amazon-linux-2023/x86_64/standard`)

- Nvidia Amazon Linux 2023 accelerato () `amazon-linux-2023/x86_64/nvidia`
- Neurone accelerato Amazon Linux 2023 () `amazon-linux-2023/x86_64/neuron`

AWS ha terminato il supporto per EKS AL2 ottimizzato e AL2 accelerato AMIs, a partire dal 26 novembre 2025. Sebbene tu possa continuare a utilizzare EKS AL2 AMIs dopo la data end-of-support (EOS) (26 novembre 2025), EKS non rilascerà più nuove versioni o aggiornamenti di Kubernetes AL2 AMIs, incluse versioni minori, patch e correzioni di bug dopo questa data. Consulta questo articolo per ulteriori informazioni sulla deprecazione. AL2

In questo argomento viene descritto come avviare gruppi con scalabilità automatica di nodi Amazon Linux su un Outpost che si registrano con il cluster Amazon EKS. Il cluster può essere sul AWS Cloud o su un Outpost.

- Un Outpost esistente. Per ulteriori informazioni, consulta What is AWS Outposts.
- Un cluster Amazon EKS esistente. Per distribuire un cluster sul AWS cloud, consulta. the section called "Creazione di un cluster " Per implementare un cluster su un Outpost, consulta la sezione the section called "Esecuzione di cluster locali".
- Supponiamo di creare i nodi in un cluster sul AWS Cloud e di disporre di sottoreti nella AWS regione in cui sono abilitati Outposts AWS , Wavelength AWS o Local Zones. AWS Quindi, le sottoreti non devono essere state passate al momento della creazione del cluster. Se stai creando i nodi in un cluster su un Outpost, durante la creazione del cluster devi avere passato una sottorete Outpost.
- (Consigliato per i cluster sul AWS cloud) Il plug-in CNI Amazon VPC per Kubernetes è configurato con il proprio ruolo IAM a cui è associata la politica IAM necessaria. Per ulteriori informazioni, consulta the section called "Configurare per IRSA". I cluster locali non supportano i ruoli IAM per gli account di servizio.

Puoi creare un gruppo di nodi Amazon Linux autogestito con `eksctl` o Console di gestione AWS (con un AWS CloudFormation modello). Puoi anche utilizzare Terraform.

Puoi creare un gruppo di nodi autogestito per il cluster locale con i seguenti strumenti descritti in questa pagina:

- the section called "`eksctl`"
- the section called "Console di gestione AWS"

Important

- Il gruppo di nodi autogestito include EC2 istanze Amazon nel tuo account. Queste istanze non vengono aggiornate in automatico quando l'utente o Amazon EKS aggiorna la versione del piano di controllo (control-plane) per tuo conto. Nella console, un gruppo di nodi autogestito non riceve alcuna indicazione di aggiornamento. È possibile visualizzare la versione di `kubelet` installata su un nodo selezionando il nodo nella finestra di dialogo Nodi nella scheda Panoramica del cluster per determinare quali nodi devono essere aggiornati. È necessario aggiornare manualmente i nodi. Per ulteriori informazioni, consulta the section called "Metodi di aggiornamento".
- I certificati utilizzati da kubelet sui nodi autogestiti vengono emessi con una scadenza di un anno. Per impostazione predefinita, la rotazione dei certificati non è abilitata (vedi: https:// kubernetes. io/docs/reference/config-api/kubelet-config.v1beta1/# kubelet-config-k 8 s-io-v 1beta1-KubeletConfiguration), questo significa che se hai un nodo autogestito attivo per più di un anno, non sarà più in grado di autenticarsi nell'API Kubernetes.
- Come best practice, consigliamo ai clienti di aggiornare regolarmente i propri gruppi di nodi autogestiti per ricevere CVEs patch di sicurezza dalle più recenti AMI ottimizzate per Amazon EKS. L'aggiornamento delle AMI utilizzate nei gruppi di nodi autogestiti attiva anche la ricreazione dei nodi e assicura che non si verifichino problemi dovuti a certificati kubelet scaduti.
- In alternativa, puoi anche abilitare la rotazione dei certificati client (vedi: https://kubernetes). io/docs/tasks/tls/certificate-rotation/) durante la creazione dei gruppi di nodi autogestiti per assicurarsi che i certificati kubelet vengano rinnovati man mano che il certificato corrente si avvicina alla scadenza.

`eksctl`

Per avviare nodi Linux autogestiti utilizzando **`eksctl`**

1. Installa la versione `0.215.0` o successiva dello strumento da riga di `eksctl` comando installato sul tuo dispositivo o. AWS CloudShell Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.
2. Se il tuo cluster è sul AWS cloud e la policy IAM gestita da Amazoneks_CNI_Policy è collegata al ruolo IAM del nodo Amazon EKS, ti consigliamo di assegnarlo a un ruolo IAM da associare invece all'account del servizio Kubernetes. `aws-node` Per ulteriori informazioni, consulta the section

called "Configurare per IRSA". Se il cluster è sull'Outpost, la policy deve essere collegata al ruolo del tuo nodo.

3. Il comando seguente crea un gruppo di nodi in un cluster esistente. Il cluster deve essere stato creato utilizzando `eksctl`. Sostituisci *`al-nodes`* con un nome per il gruppo di nodi. Il nome del gruppo di nodi non può contenere più di 63 caratteri. Deve iniziare con una lettera o un numero, ma può anche includere trattini e caratteri di sottolineatura. Sostituisci *`my-cluster`* con il nome del cluster. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della regione e dell'account in cui stai creando il cluster. AWS AWS Se il cluster si trova su un Outpost, sostituisci *`id`* con l'ID di una sottorete dell'Outpost. Se il cluster esiste nel AWS cloud, sostituiscilo *`id`* con l'ID di una sottorete che non hai specificato al momento della creazione del cluster. Sostituisci i valori di esempio con i tuoi valori. Per impostazione predefinita, i nodi vengono creati con la stessa versione Kubernetes del piano di controllo.

   Sostituisci *`instance-type`* con un tipo di istanza disponibile sul tuo Outpost.

   Sostituiscilo *`my-key`* con il nome della tua coppia di EC2 chiavi Amazon o della tua chiave pubblica. Questa chiave viene utilizzata per eseguire il SSH nei nodi dopo il loro avvio. Se non disponi già di una coppia di EC2 chiavi Amazon, puoi crearne una in Console di gestione AWS. Per ulteriori informazioni, consulta le coppie di EC2 chiavi Amazon nella Amazon EC2 User Guide.

   Crea il tuo gruppo di nodi con il comando seguente.

```
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 il tuo cluster è distribuito sul AWS cloud:

   - Il gruppo di nodi implementato può assegnare gli indirizzi `IPv4` ai pod da un altro intervallo CIDR rispetto a quello dell'istanza. Per ulteriori informazioni, consulta the section called "Rete personalizzata".
   - Il gruppo di nodi che implementi non richiede l'accesso a internet in uscita. Per ulteriori informazioni, consulta the section called "Cluster privati".

Per un elenco completo di tutte le impostazioni predefinite e le opzioni disponibili, consulta AWS Outposts Support nella documentazione di `eksctl`.

- Se i nodi non riescono a unirsi al cluster, consulta the section called "Impossibile aggiungere i nodi al cluster" la sezione Risoluzione dei problemi con i cluster e i nodi Amazon EKS e Risoluzione dei problemi dei cluster Amazon EKS locali su Outposts. the section called "Impossibile unire i nodi a un cluster" AWS
- Di seguito viene riportato un output di esempio. Durante la creazione dei nodi vengono generate diverse righe. Una delle ultime righe di output è simile alla seguente riga di esempio.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

4. (Facoltativo) implementare un'applicazione di esempio per testare il cluster e i nodi Linux.

Console di gestione AWS

Fase 1: avvia i nodi Linux autogestiti utilizzando la Console di gestione AWS

1. Scarica l'ultima versione del modello. AWS CloudFormation

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2025-11-24/
amazon-eks-outpost-nodegroup.yaml
```

2. Apri la AWS CloudFormation console.
3. Scegli Crea stack e quindi seleziona Con nuove risorse (standard).
4. In Specifica modello, seleziona Carica un file di modello e Scegli file. Seleziona il file `amazon-eks-nodegroup.yaml` scaricato in un passaggio precedente, quindi seleziona Next (Successivo).
5. Nella pagina Specifica i dettagli dello stack, immetti i parametri seguenti e scegli Successivo:
   - Nome dello stack: scegli un nome per lo stack. AWS CloudFormation Ad esempio, è possibile chiamarlo *`al-nodes`*. Il nome può contenere solo caratteri alfanumerici (con distinzione tra lettere maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfanumerico e non può avere una lunghezza superiore a 100 caratteri. Il nome deve essere univoco all'interno della AWS regione e AWS dell'account in cui stai creando il cluster.
   - ApiServerEndpoint: Inserisci l'endpoint del server API Kubernetes, visibile nella console EKS o tramite API. DescribeCluster

- ClusterName: Inserisci il nome del tuo cluster. Se questo nome non corrisponde al nome del cluster, i nodi non possono unirsi al cluster.
- ClusterId: Inserisci l'id assegnato al cluster dal servizio EKS. Visibile tramite DescribeCluster API. Se questo ID non corrisponde al tuo ID del cluster, i tuoi nodi non possono unirsi al cluster.
- CertificateAuthority: Inserisci la stringa codificata in base64 dell'autorità di certificazione Kubernetes. Visibile nella console EKS o tramite API. DescribeCluster
- ServiceCidr: Accedi al CIDR di Kubernetes Services. Visibile nella console EKS o tramite API. DescribeCluster
- ClusterControlPlaneSecurityGroup: scegli il SecurityGroupsvalore dall' AWS CloudFormation output che hai generato quando hai creato il tuo VPC.

  Nella procedura seguente viene illustrata un'operazione per recuperare il gruppo applicabile.

  i. Aprire la Console Amazon EKS.

  ii. Scegli il nome del cluster.

  iii. Scegli la scheda Reti.

  iv. Utilizza il valore dei gruppi di sicurezza aggiuntivi come riferimento quando selezioni dall'ClusterControlPlaneSecurityGroupelenco a discesa.
- NodeGroupName: inserisci un nome per il tuo gruppo di nodi. Questo nome può essere utilizzato in seguito per identificare il gruppo di nodi con dimensionamento automatico creato per i tuoi nodi.
- NodeAutoScalingGroupMinSize: Inserisci il numero minimo di nodi su cui il gruppo Auto Scaling del nodo può scalare.
- NodeAutoScalingGroupDesiredCapacity: Inserisci il numero di nodi desiderato su cui scalare quando viene creato lo stack.
- NodeAutoScalingGroupMaxSize: Inserisci il numero massimo di nodi su cui il gruppo Auto Scaling del nodo può scalare orizzontalmente.
- NodeInstanceType: scegli un tipo di istanza per i tuoi nodi. Se il cluster è in esecuzione sul AWS cloud, per ulteriori informazioni, consultathe section called "Tipi di istanza Amazon EC2". Se il cluster è in esecuzione su un Outpost, puoi selezionare solo un tipo di istanza disponibile sul tuo Outpost.
- NodeImageIdSSMParam: precompilato con il parametro Amazon EC2 Systems Manager di una recente AMI ottimizzata per Amazon EKS per una versione variabile di Kubernetes. Se si desidera utilizzare una diversa versione secondaria di Kubernetes supportata da Amazon EKS,

è possibile sostituire *1.XX* con una versione supportata differente. Si consiglia di specificare la stessa versione Kubernetes del cluster.

Per utilizzare un'AMI accelerata ottimizzata per Amazon EKS, aggiorna il *NodeImageIdSSMParam* valore al parametro SSM desiderato. Scopri come recuperare l'AMI EKS IDs da SSM qui.

> **Note**
>
> I nodi Amazon EKS AMIs sono basati su Amazon Linux. Tieni traccia degli eventi di sicurezza o di privacy per Amazon Linux in Amazon Linux security center scegliendo la scheda per la versione desiderata. Puoi anche abbonarti al feed RSS applicabile. Gli eventi di sicurezza e privacy includono una panoramica del problema, quali sono i pacchetti interessati e come aggiornare le istanze per risolvere il problema.

- NodeImageId: (Facoltativo) Se utilizzi un'AMI personalizzata (anziché un'AMI ottimizzata per Amazon EKS), inserisci un ID AMI del nodo per la tua AWS regione. Se specifichi un valore qui, questo sostituisce tutti i valori nel NodeImageIdSSMParamcampo.
- NodeVolumeSize: Specificate la dimensione del volume root per i nodi, in GiB.
- NodeVolumeType: Specificate un tipo di volume root per i nodi.
- KeyName: inserisci il nome di una coppia di chiavi Amazon EC2 SSH che puoi usare per connetterti tramite SSH ai tuoi nodi dopo il loro avvio. Se non disponi già di una coppia di EC2 chiavi Amazon, puoi crearne una in Console di gestione AWS. Per ulteriori informazioni, consulta le coppie di EC2 chiavi Amazon nella Amazon EC2 User Guide.

> **Note**
>
> Se non fornisci una key pair qui, la creazione dello AWS CloudFormation stack fallisce.

- Disabilita IMDSv1: per impostazione predefinita, ogni nodo supporta Instance Metadata Service versione 1 (IMDSv1) e. IMDSv2 È possibile disabilitare IMDSv1. Per impedire l'utilizzo di nodi e Pod futuri nel gruppo di nodi IMDSv1, imposta Disable su IMDSv1 true. Per ulteriori informazioni su IMDS, consulta Configurazione del servizio di metadati dell'istanza. Per ulteriori informazioni sulle relative limitazioni dell'accesso ai nodi, consulta Limita l'accesso al profilo dell'istanza assegnato al nodo (worker).
- VpcId: inserisci l'ID per il VPC che hai creato. Prima di scegliere un VPC, consulta VPC requirements and considerations.

- Subnets: (Sottoreti) se il cluster è su un Outpost, scegli almeno una sottorete privata nel VPC. Prima di scegliere le sottoreti, consulta Considerazioni e requisiti relativi alla sottorete. È possibile visualizzare le sottoreti private aprendo ogni collegamento relativo alla sottorete dalla scheda Reti del cluster.

6. Seleziona le opzioni desiderate nella pagina Configura opzioni dello stack, quindi scegli Next (Avanti).
7. Seleziona la casella di controllo a sinistra di Riconosco che AWS CloudFormation potrebbe creare risorse IAM. , quindi scegli Create stack.
8. Al termine della creazione dello stack, selezionalo nella console e scegli Output.
9. Registra il NodeInstanceRoleper il gruppo di nodi che è stato creato. Ciò sarà utile quando configurerai i nodi di Amazon EKS.

Fase 2: abilitazione dell'aggiunta di nodi al cluster

1. Verifica se disponi già di una `ConfigMap` per `aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Se ti viene mostrata una `ConfigMap` per `aws-auth`, aggiornala se necessario.
   a. Apri `ConfigMap` per la modifica.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Aggiungi una nuova voce `mapRoles`, se necessario. Impostate il `rolearn` NodeInstanceRolevalore sul valore registrato nella procedura precedente.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

   c. Salva il file ed esci dall'editor di testo.

3. Se hai ricevuto un messaggio di errore che indica "`Error from server (NotFound): configmaps "aws-auth" not found`", applica lo `ConfigMap` di stock.

   a. Scarica la mappa di configurazione.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-auth-cm.yaml
   ```

   b. Nel `aws-auth-cm.yaml` file, impostate `rolearn` il NodeInstanceRolevalore sul valore registrato nella procedura precedente. Per eseguire questa operazione, utilizza un editor di testo o sostituisci *`my-node-instance-role`* tramite il comando seguente:

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
   ```

   c. Applica la configurazione. L'esecuzione di questo comando potrebbe richiedere alcuni minuti.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

4. Guarda lo stato dei nodi e attendi che raggiungano lo stato `Ready`.

   ```
   kubectl get nodes --watch
   ```

   Inserisci `Ctrl+C` per tornare a un prompt dello shell (interprete di comandi).

   > **Note**
   >
   > Se ricevi qualsiasi altro errore di tipo di risorsa o autorizzazione, consulta la sezione the section called "Accesso negato o non autorizzato (`kubectl`)" nell'argomento relativo alla risoluzione dei problemi.

   Se i nodi non riescono a unirsi al cluster, consulta the section called "Impossibile aggiungere i nodi al cluster" la sezione Risoluzione dei problemi con i cluster e i nodi Amazon EKS e Risoluzione dei problemi dei cluster Amazon EKS locali su Outposts. the section called "Impossibile unire i nodi a un cluster" AWS

5. Installa il driver CSI per Amazon EBS. Per ulteriori informazioni, consulta Installazione su. GitHub Nella sezione Set up driver permission (Configura autorizzazione del driver), assicurati di seguire le istruzioni per l'opzione Using IAM instance profile (Utilizzo del profilo dell'istanza

IAM). È necessario utilizzare la classe di archiviazione gp2. La classe di archiviazione gp3 non è supportata.

Per creare una classe di archiviazione gp2 nel cluster, completa i seguenti passaggi.

a. Per creare il file `gp2-storage-class.yaml`, emetti il seguente comando:

```
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. Applica il file manifesto al cluster.

```
kubectl apply -f gp2-storage-class.yaml
```

6. (Solo nodi GPU) Se hai scelto un tipo di istanza GPU e un'AMI accelerata ottimizzata per Amazon EKS, devi applicare il plug-in del dispositivo NVIDIA per Kubernetes sul tuo cluster. DaemonSet Sostituiscilo *vX.X.X* con la versione s-device-pluginNVIDIA/K8 desiderata prima di eseguire il comando seguente.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Fase 3: operazioni aggiuntive

1. (Facoltativo) implementare un'applicazione di esempio per testare il cluster e i nodi Linux.
2. Se il cluster è implementato su un Outpost, salta questo passaggio. Se il cluster è distribuito sul AWS Cloud, le seguenti informazioni sono opzionali. Se la policy IAM gestita AmazonEKS_CNI_Policy è collegata al ruolo IAM del nodo Amazon EKS, si consiglia, invece, di

assegnarla a un ruolo IAM associato all'account di servizio Kubernetes `aws-node`. Per ulteriori informazioni, consulta the section called "Configurare per IRSA".

Panoramica dell’intelligenza artificiale (IA) e del machine learning (ML) su Amazon EKS

Amazon Elastic Kubernetes Service (EKS) è una piattaforma Kubernetes gestita che consente alle organizzazioni di implementare, gestire e scalare carichi di lavoro di IA e machine learning (ML) con flessibilità e controllo senza precedenti. Basato sull'ecosistema open source Kubernetes, EKS ti consente di sfruttare le tue competenze Kubernetes esistenti, integrandosi perfettamente con strumenti e servizi open source. AWS

Che tu stia addestrando modelli su larga scala, eseguendo inferenze online in tempo reale o implementando applicazioni di intelligenza artificiale generativa, EKS offre le prestazioni, la scalabilità e l'efficienza dei costi richieste dai tuoi progetti. AI/ML

Perché scegliere EKS per IA/ML?

EKS è una piattaforma Kubernetes gestita che ti aiuta a implementare e gestire carichi di lavoro complessi. AI/ML Basata sull'ecosistema open source Kubernetes, si integra con i AWS servizi, fornendo il controllo e la scalabilità necessari per progetti avanzati. Per i team alle prime armi con le AI/ML implementazioni, le competenze Kubernetes esistenti vengono trasferite direttamente, consentendo un'orchestrazione efficiente di più carichi di lavoro.

EKS supporta tutto, dalla personalizzazione del sistema operativo alla scalabilità del calcolo, e la sua base open source promuove la flessibilità tecnologica, preservando la scelta per le future decisioni sull’infrastruttura. La piattaforma offre le prestazioni e le opzioni di ottimizzazione richieste dai carichi di lavoro, supportando funzionalità come: AI/ML

- Controllo completo del cluster per ottimizzare costi e configurazioni senza astrazioni nascoste
- Latenza inferiore al secondo per carichi di lavoro di inferenza in tempo reale in produzione
- Personalizzazioni avanzate come strategie multiistanza e multi-cloud e GPUs ottimizzazione a livello di sistema operativo
- Capacità di centralizzare i carichi di lavoro utilizzando EKS come orchestratore unificato tra le pipeline AI/ML

Casi d'uso principali

Amazon EKS offre una piattaforma robusta per un'ampia gamma di AI/ML carichi di lavoro, supportando varie tecnologie e modelli di implementazione:

- Inferenza in tempo reale (online): EKS consente previsioni immediate sui dati in entrata, come il rilevamento delle frodi, con una latenza inferiore al secondo utilizzando strumenti come TorchServeTriton Inference Server e istanze Amazon Inf1 e KServeInf2. EC2 Questi carichi di lavoro traggono vantaggio dal dimensionamento dinamico con Karpenter e KEDA, sfruttando al contempo Amazon EFS per lo sharding dei modelli tra i pod. Amazon ECR Pull Through Cache (PTC) accelera gli aggiornamenti dei modelli, mentre i volumi di dati Bottlerocket con volumi ottimizzati per Amazon EBS garantiscono un accesso rapido ai dati.
- Formazione generale sui modelli: le organizzazioni sfruttano EKS per addestrare modelli complessi su set di dati di grandi dimensioni per periodi prolungati utilizzando Kubeflow Training Operator, Ray Serve e Torch Distributed Elastic su istanze Amazon EC2 P4d e Amazon Trn1. EC2 Questi carichi di lavoro sono supportati dalla pianificazione in batch con strumenti come Volcano, Yunikorn e Kueue. Amazon EFS consente la condivisione dei checkpoint del modello e Amazon S3 gestisce il import/export modello con politiche del ciclo di vita per la gestione delle versioni.
- Pipeline di generazione potenziata da recupero dati (RAG): EKS gestisce chatbot di assistenza clienti e applicazioni simili integrando processi di recupero e generazione. Questi carichi di lavoro utilizzano spesso strumenti come Argo Workflows e Kubeflow per l'orchestrazione, database vettoriali come Pinecone, Weaviate o Amazon ed espongono le OpenSearch applicazioni agli utenti tramite l'Application Load Balancer Controller (LBC). NVIDIA NIM ottimizza l'utilizzo della GPU, mentre Prometheus e Grafana monitorano l'utilizzo delle risorse.
- Implementazione di modelli di intelligenza artificiale generativa: le aziende implementano servizi di creazione di contenuti in tempo reale su EKS, come la generazione di testo o immagini, utilizzando Ray Serve, vLLM e Triton Inference Server su Amazon G5 e acceleratori Inferentia. EC2 Queste implementazioni ottimizzano le prestazioni e l'utilizzo della memoria per modelli su larga scala. JupyterHubconsente lo sviluppo iterativo, Gradio fornisce semplici interfacce web e il driver S3 Mountpoint CSI consente di montare bucket S3 come file system per accedere a file di modello di grandi dimensioni.
- Inferenza in batch (offline): le organizzazioni elaborano set di dati di grandi dimensioni in modo efficiente tramite processi pianificati con AWS Batch o Volcano. Questi carichi di lavoro utilizzano spesso istanze Inf1 e Inf2 per chip AWSInferentia, EC2 istanze EC2 Amazon G4dn per NVIDIA GPUs T4 o istanze CPU c5 e c6i, massimizzando l'utilizzo delle risorse durante le ore non di punta per le attività di analisi. AWS I driver Neuron SDK e NVIDIA GPU ottimizzano le prestazioni,

abilitando al contempo la condivisione della GPU. MIG/TS Le soluzioni di storage includono Amazon S3 e Amazon EFS e FSx for Lustre, con driver CSI per varie classi di storage. La gestione dei modelli sfrutta strumenti come Kubeflow Pipelines, Argo Workflows e Ray Cluster, mentre il monitoraggio è gestito da Prometheus, Grafana e strumenti di monitoraggio dei modelli personalizzati.

Casi di studio

I clienti scelgono Amazon EKS per vari motivi, come l'ottimizzazione dell'utilizzo della GPU o l'esecuzione di carichi di lavoro di inferenza in tempo reale con latenza inferiore al secondo, come dimostrato nei seguenti casi di studio. Per un elenco di tutti i casi di studio relativi ad Amazon EKS, consulta AWS Customer Success Stories.

- Unitary elabora 26 milioni di video al giorno utilizzando l'IA per la moderazione dei contenuti, richiedendo un'inferenza ad alto throughput e bassa latenza e ha ottenuto una riduzione dell'80% dei tempi di avvio dei container, garantendo una risposta rapida agli eventi di scalabilità in base alle fluttuazioni del traffico.
- Miro, la piattaforma di collaborazione visiva che supporta 70 milioni di utenti in tutto il mondo, ha registrato una riduzione dell'80% dei costi di elaborazione rispetto ai precedenti cluster Kubernetes autogestiti.
- Synthesia, che offre ai clienti la creazione di video con IA generativa come servizio per consentire ai clienti di creare video realistici a partire da prompt di testo, ha ottenuto un miglioramento di 30 volte nel throughput per l'addestramento dei modelli ML.
- Harri, che fornisce tecnologia HR per il settore alberghiero, ha raggiunto una riduzione orizzontale più rapida del 90% in risposta ai picchi di domanda e ha ridotto i costi di elaborazione del 30% eseguendo la migrazione ai processori AWS Graviton.
- Ada Support, una società di automazione del servizio clienti basata sull'IA, ha ottenuto una riduzione del 15% dei costi di elaborazione insieme a un aumento del 30% dell'efficienza nell'elaborazione.
- Snorkel AI, che consente alle aziende di creare e adattare modelli di base e modelli linguistici di grandi dimensioni, ha ottenuto risparmi sui costi di oltre il 40% implementando meccanismi di scalabilità intelligenti per le risorse GPU.

Inizia a usare il machine learning su EKS

Per iniziare a pianificare e utilizzare le piattaforme e i carichi di lavoro di Machine Learning su EKS sul AWS cloud, vai alla the section called "Risorse" sezione.

Running real-time online inference workloads on Amazon EKS

> **Tip**
>
> Registrati ai prossimi AI/ML workshop Amazon EKS.

Questa sezione è progettata per aiutarti a distribuire e gestire carichi di lavoro di inferenza online in tempo reale su Amazon Elastic Kubernetes Service (EKS). Troverai indicazioni sulla creazione di cluster ottimizzati con nodi accelerati da GPU, sull'integrazione di AWS servizi per lo storage e la scalabilità automatica, sull'implementazione di modelli di esempio per la convalida e su considerazioni architettoniche chiave come il disaccoppiamento delle attività di CPU e GPU, la selezione dei tipi di istanza appropriati e la garanzia di un'esposizione a bassa latenza degli endpoint di inferenza. AMIs

Argomenti

- Guida alle best practice per la configurazione dei cluster per l'inferenza in tempo reale su Amazon EKS
- Avvio rapido: inferenza LLM ad alto rendimento con vLLM su Amazon EKS

Guida alle best practice per la configurazione dei cluster per l'inferenza in tempo reale su Amazon EKS

> **Tip**
>
> Registrati ai prossimi AI/ML workshop Amazon EKS.

Introduzione

Questa guida offre una guida pratica per configurare un cluster Amazon Elastic Kubernetes Service (EKS) ottimizzato per carichi di lavoro di inferenza online in tempo reale, incorporando le migliori

pratiche curate da esperti. AWS Utilizza un'autorevole architettura EKS Quickstart, un set curato di driver, tipi di istanze e configurazioni in linea con le migliori pratiche per modelli, acceleratori e scalabilità. AWS Questo approccio consente di aggirare il compito di selezionare le impostazioni del cluster, consentendo di ottenere rapidamente un cluster funzionale e preconfigurato. Lungo il percorso, distribuiremo esempi di carichi di lavoro per convalidare la configurazione, spiegheremo i concetti architettonici chiave (come il disaccoppiamento delle attività legate alla CPU dai calcoli ad uso intensivo della GPU), risponderemo a domande comuni (ad esempio, perché scegliere Bottlerocket AMI?) AL2023 e delineeremo i passaggi successivi per estendere le funzionalità del cluster.

Progettata specificamente per gli ingegneri di Machine Learning (ML) e Artificial Intelligence (AI), gli amministratori di piattaforma, gli operatori e gli data/AI specialisti che non conoscono l'ecosistema AWS ed EKS, questa guida presuppone la familiarità con Kubernetes ma nessuna esperienza precedente con EKS. Infatti, ha l'obiettivo di aiutare a comprendere i passaggi e i processi necessari per rendere operativi i carichi di lavoro di inferenza online in tempo reale. La guida illustra gli elementi essenziali della creazione di un cluster di inferenza a nodo singolo, tra cui il provisioning di risorse GPU, l'integrazione dello storage per gli artefatti del modello, l'abilitazione dell'accesso sicuro ai AWS servizi e l'esposizione degli endpoint di inferenza. Nel complesso, si concentra sul design resiliente e a bassa latenza per applicazioni rivolte agli utenti come il rilevamento delle frodi, i chatbot in tempo reale e l'analisi del sentiment nei sistemi di feedback dei clienti.

In questa guida, ci concentriamo esclusivamente sulla configurazione di un punto di partenza fondamentale e prescrittivo utilizzando le istanze G5 di EC2. Se stai cercando configurazioni di cluster o flussi di lavoro specifici per AWS Inferenza, consulta i nostri workshop su. end-to-end the section called "Configura cluster di inferenza con Inferentia" the section called "Risorse"

Prima di iniziare

Prima di iniziare, esegui le seguenti attività:

- Configura il tuo ambiente per Amazon EKS
- Installa la versione più recente di eksctl
- Installa Helm
- (Opzionale) Installa Docker
- (Opzionale) Installa la CLI NGC

Architecture

L'inferenza online in tempo reale si riferisce al processo di utilizzo di un modello di machine learning addestrato per generare previsioni oppure output sui flussi di dati in entrata con una latenza minima. Ad esempio, consente il rilevamento delle frodi in tempo reale, la classificazione delle immagini o la generazione di grafici di conoscenza in risposta agli input degli utenti. L'architettura di un sistema di inferenza online in tempo reale offre previsioni di machine learning a bassa latenza nelle applicazioni rivolte agli utenti, disaccoppiando la gestione del traffico web legato alla CPU dai calcoli di intelligenza artificiale a uso intensivo di GPU. Questo processo si trova in genere all'interno di un ecosistema applicativo più ampio e spesso include servizi di backend, frontend, vettoriali e di modello, con particolare attenzione a componenti specializzati per consentire scalabilità indipendente, sviluppo parallelo e resilienza ai guasti. L'isolamento delle attività di inferenza su hardware GPU dedicato e l'utilizzo di interfacce come e WebSockets garantiscono un'elevata concorrenza, un'elaborazione rapida di APIs modelli come i trasformatori e le interazioni degli utenti tramite il frontend. Tieni presente che, sebbene svolgano spesso un ruolo importante nei sistemi di inferenza in tempo reale, i database vettoriali e le pipeline di generazione potenziata da recupero dati (RAG) non sono trattati in questa guida. Un'architettura tipica spesso include almeno:

- Servizio frontend: funge da interfaccia utente, gestisce la logica lato client, rende contenuti dinamici e facilita le interazioni in tempo reale, comunica con il servizio di backend per avviare richieste di inferenza e visualizzare i risultati, spesso avviando richieste al servizio di backend che lo utilizza WebSockets per gli aggiornamenti in streaming o APIs per lo scambio di dati strutturati. Questo servizio in genere non richiede un sistema di bilanciamento del carico dedicato, in quanto può essere ospitato su reti di distribuzione dei contenuti (CDNs) come AWS CloudFront per le risorse statiche o fornito direttamente dai server Web, con scalabilità gestita tramite gruppi di auto-scaling se necessario per contenuti dinamici.
- Servizio di backend: funge da orchestratore dell'applicazione, gestendo logiche di business come l'autenticazione degli utenti, la convalida dei dati e il coordinamento dei servizi (ad esempio, tramite gli endpoint o per le connessioni persistenti). APIs RESTful WebSockets Comunica con il servizio di inferenza, scala indipendentemente su multi-core CPUs e RAM per gestire un traffico Web elevato senza fare affidamento su GPUs e spesso richiede un sistema di bilanciamento del carico (come Application Load Balancer o Network Load AWS Balancer) per distribuire le richieste in entrata su più istanze, specialmente in scenari ad alta concorrenza. Un controller di ingresso può gestire ulteriormente l'accesso esterno e le regole di routing per migliorare la sicurezza e la modellazione del traffico.
- Servizio di inferenza: funge da base per i calcoli di intelligenza artificiale, in esecuzione GPUs con una VRAM sufficiente (ad esempio, 8-12 GB per modelli come DistilBert) per eseguire

incorporamenti vettoriali, estrazione della conoscenza e inferenza del modello (ad esempio, esposto APIs per richieste in batch o WebSockets per lo streaming in tempo reale) utilizzando modelli personalizzati o open source. Questo isolamento previene i conflitti di dipendenza, consente di aggiornare i modelli senza tempo di inattività e agevola il dimensionamento orizzontale con bilanciamento del carico per più richieste simultanee. Per esporre in modo efficace il servizio modello, in genere si basa su un sistema di bilanciamento del carico per distribuire i carichi di lavoro legati alla GPU tra istanze replicate, mentre una risorsa o un controller in ingresso (come ALB Ingress Controller in AWS) gestisce il routing esterno, la terminazione SSL e l'inoltro basato sul percorso per garantire un accesso sicuro ed efficiente senza sovraccaricare le persone GPUs.

Panoramica soluzione

I sistemi di inferenza online in tempo reale richiedono un'architettura resiliente e ad alte prestazioni in grado di fornire una latenza estremamente bassa gestendo al contempo esplosioni di traffico imprevedibili e ad alto volume. Questa panoramica della soluzione spiega come i seguenti AWS componenti interagiscono nel cluster Amazon EKS che creeremo per garantire che il nostro cluster sia in grado di ospitare e gestire modelli di machine learning che forniscono previsioni immediate su dati in tempo reale con un ritardo minimo per gli utenti finali.

- Istanze Amazon G5 EC2: per attività di inferenza a uso intensivo di GPU, utilizziamo i tipi di istanze G5 EC2 g5.xlarge e g5.2xlarge, che dispongono di una (1) singola GPU NVIDIA A10G con 24 GB di memoria (ad esempio, 8 miliardi di parametri in FP16). Queste GPU, basate sull'architettura NVIDIA Ampere, sono alimentate da GPU NVIDIA A10G Tensor Core e processori AMD EPYC di seconda generazione, supportano 4-8 vCPU, larghezza di banda della rete fino a 10 Gbps e 250-450 GB di archiviazione SSD NVMe locale, garantendo un rapido spostamento dei dati e potenza di calcolo per modelli complessi. Per questo motivo, sono ideali per attività di inferenza a bassa latenza e ad alto throughput. La scelta di un tipo di istanza EC2 è specifica dell'applicazione, dipende dal modello (ad esempio, immagine, video, modello di testo) e dai requisiti di latenza e throughput effettivo. Ad esempio, se utilizzi un modello di immagine e/o video, potresti voler utilizzare le istanze EC2 P5 per una latenza ottimale in tempo reale. Ti consigliamo di iniziare con le istanze G5 EC2 in quanto forniscono un buon punto di partenza per iniziare subito a funzionare, per poi valutare se sono la soluzione giusta per i tuoi carichi di lavoro attraverso test di benchmark delle prestazioni. Per casi più avanzati, prendi in considerazione le istanze G6 EC2.
- Istanze Amazon EC2 M7g: per attività che richiedono un uso intensivo della CPU come la pre-elaborazione dei dati, la gestione delle richieste API, l'hosting del controller Karpenter, componenti aggiuntivi e altri componenti di sistema, utilizziamo il tipo di istanza EC2 M7g m5.xlarge. Le istanze M7g sono istanze basate su ARM che offrono 4 vCPUs, 16 GB di memoria, una larghezza di

banda di rete fino a 12,5 Gbps ed è alimentata da processori Graviton3. AWS La scelta di un tipo di istanza EC2 è specifica dell'applicazione e dipende dai requisiti di calcolo, memoria e scalabilità del carico di lavoro. Per carichi di lavoro ottimizzati per il calcolo, potresti prendere in considerazione le istanze C7g EC2, che utilizzano anch'esse processori Graviton3 ma sono ottimizzate per prestazioni di calcolo più elevate rispetto alle istanze M7g per determinati casi d'uso. In alternativa, le istanze C8g EC2 più recenti (se disponibili) offrono prestazioni di elaborazione migliori fino al 30% rispetto alle istanze C7g. Consigliamo di iniziare con le istanze M7g EC2 per la loro efficienza in termini di costi e la compatibilità con un'ampia gamma di carichi di lavoro (ad esempio, server di applicazioni, microservizi, server di gioco, archivi dati di medie dimensioni) per poi valutare se sono la soluzione giusta per i propri carichi di lavoro attraverso test di benchmark delle prestazioni.

- Driver CSI Mountpoint di Amazon S3: per carichi di lavoro su istanze a GPU singola in cui più pod condividono una GPU (ad esempio, più pod pianificati sullo stesso nodo per utilizzare le relative risorse GPU), utilizziamo il driver CSI Mountpoint di S3 per ottimizzare l'utilizzo della memoria, essenziale per attività come l'inferenza del modello di grandi dimensioni in configurazioni sensibili ai costi e a bassa complessità. Infatti, espone i bucket Amazon S3 come un file system simile a POSIX disponibile per il cluster Kubernetes, che consente ai pod di inferenza di leggere gli artefatti del modello (ad esempio, il peso) direttamente nella memoria senza doverli prima scaricare e di inserire i set di dati utilizzando operazioni standard sui file. Inoltre, S3 ha una capacità di archiviazione praticamente illimitata e accelera i carichi di lavoro di inferenza ad alta intensità di dati. La scelta di un driver CSI di archiviazione è specifica dell'applicazione e dipende dai requisiti di throughput e dalla latenza del carico di lavoro. Sebbene il driver CSI FSx for OpenZFS offra una latenza inferiore al millisecondo per volumi persistenti condivisi casuali I/O o completamente conformi a POSIX tra i nodi, consigliamo di iniziare con il driver CSI Mountpoint S3 grazie alla sua scalabilità, ai costi inferiori per set di dati di grandi dimensioni e all'integrazione integrata con lo storage di oggetti gestito da S3 per modelli di inferenza ad alta intensità di lettura (ad esempio, input del modello di streaming), quindi valuta se è la soluzione giusta per il tuo lavoro si carica attraverso test di benchmark delle prestazioni.
- EKS Pod Identity Agent: per consentire l'accesso ai AWS servizi, utilizziamo EKS Pod Identity Agent, che utilizza un unico servizio principale e facilita le associazioni di ruoli IAM a livello di pod all'interno del cluster Amazon EKS. Pod Identity di EKS offre un'alternativa semplificata al tradizionale approccio IAM Roles for Service Accounts (IRSA) utilizzando un unico principale del servizio (pods.eks.amazonaws.com) anziché affidarsi a singoli provider OIDC per ogni cluster, il che semplifica l'assegnazione delle autorizzazioni. Inoltre, consente il riutilizzo dei ruoli su più cluster e supporta funzionalità avanzate come IAM role session tags e Target IAM roles.
- Agente di monitoraggio dei nodi di EKS: per garantire la disponibilità e l'affidabilità continue dei servizi di inferenza, utilizziamo l'agente di monitoraggio dei nodi di EKS con la riparazione

automatica, che rileva e sostituisce automaticamente i nodi non integri riducendo al minimo il tempo di inattività. Monitora continuamente i nodi per problemi hardware, kernel, di rete e di storage utilizzando controlli di integrità avanzati (ad esempio,). KernelReady NetworkingReady Per i nodi GPU, rileva i guasti specifici dell'acceleratore e avvia una correzione adeguata isolando i nodi non integri, attendendo 10 minuti per la risoluzione dei problemi transitori della GPU e sostituendo i nodi dopo 30 minuti in caso di guasti persistenti.

- AMI Bottlerocket: per fornire una base sicura per il cluster EKS, utilizziamo l'AMI Bottlerocket, che include solo i componenti essenziali necessari per eseguire i container e offre tempi di avvio minimi per una scalabilità rapida. La scelta di un nodo AMI è specifica dell'applicazione e dipende dai requisiti di personalizzazione, sicurezza e scalabilità del carico di lavoro. Sebbene l'AL2023 AMI offra una maggiore flessibilità per le installazioni e le personalizzazioni a livello di host (ad esempio, specificando una directory cache dedicata in una configurazione PV/PVC senza nodi aggiuntivi), consigliamo di iniziare con l'AMI Bottlerocket per il suo ingombro ridotto e l'ottimizzazione integrata per carichi di lavoro containerizzati (ad esempio microservizi, server di inferenza, scalabile APIs), quindi di valutare se è la soluzione giusta per i tuoi carichi di lavoro attraverso test di benchmark delle prestazioni.
- AWS Load Balancer Controller (LBC): per esporre gli endpoint di inferenza in tempo reale, utilizziamo il Load AWS Balancer Controller, che fornisce e gestisce automaticamente Application Load Balancer () per il traffico e Network Load Balancer (ALBs) per il HTTP/HTTPS TCP/UDP traffico basato sulle risorse Kubernetes Ingress and Service, abilitando l'integrazione di modelli di inferenza con client esterni. NLBs Inoltre, supporta funzionalità come il routing basato sul percorso per distribuire le richieste di inferenza su più pod o nodi, garantendo la scalabilità durante i picchi di traffico e riducendo al minimo la latenza attraverso ottimizzazioni AWS native come il multiplexing delle connessioni e i controlli di integrità.

1. Creazione di un cluster EKS

In questa fase, creiamo un cluster con nodi CPU e un gruppo di nodi gestito utilizzando un modello eksctl basato su tecnologia. AWS CloudFormation ClusterConfig L'inizializzazione del cluster con soli nodi CPU ci consente di utilizzare Karpenter esclusivamente per gestire nodi che richiedono un uso intensivo di CPU e nodi GPU per un'allocazione ottimizzata delle risorse utilizzando Karpenter, che creeremo nei passaggi successivi. NodePools Per supportare i nostri carichi di lavoro di inferenza in tempo reale, forniamo al cluster l'AMI EKS Bottlerocket, EKS Node Monitoring Agent, EKS Pod Identity Agent, il driver CSI Mountpoint S3, il Load AWS Balancer Controller (LBC) e i driver kube-proxy, vpc-cni e coredns. Le istanze m7g.xlarge verranno utilizzate per le attività del sistema CPU, tra cui hosting del controller Karpenter, componenti aggiuntivi e altri componenti di sistema.

Per impostazione predefinita, `eksctl` creerà un VPC dedicato per il cluster con un intervallo CIDR di `192.168.0.0/16`. Il VPC include tre sottoreti pubbliche e tre sottoreti private, ciascuna distribuita su tre diverse zone di disponibilità (o due AZs nella `us-east-1` regione), il metodo ideale per implementare i carichi di lavoro Kubernetes. Il modello implementa anche un gateway Internet, che fornisce l'accesso a Internet alle sottoreti pubbliche tramite percorsi predefiniti nelle relative tabelle di routing e un singolo gateway NAT in una delle sottoreti pubbliche, con percorsi predefiniti nelle tabelle di routing delle sottoreti private che indirizzano il traffico in uscita attraverso il gateway NAT per l'accesso a Internet. Per ulteriori informazioni su questa configurazione, consulta Deploy Nodes to Private Subnets.

Verifica le credenziali

Verifica se le tue credenziali AWS CLI sono valide e possono autenticarsi con i servizi: AWS

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

In caso di successo, la CLI restituirà i dettagli sulla tua AWS identità (UserId, Account e Arn).

Verifica la disponibilità delle istanze

I tipi di istanze G5 sono disponibili in tutte le regioni. Controlla la regione più vicina. Esempio:

```
aws ec2 describe-instance-types --instance-types g5.xlarge g5.2xlarge --region us-
east-1
```

in caso di successo, il tipo di istanza G5 è disponibile nella regione specificata.

L'AMI Bottlerocket non è disponibile in tutte le regioni. Verifica tramite il recupero di un ID AMI Bottlerocket per la regione più vicina. Esempio:

```
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
```

in caso di successo, l'AMI Bottlerocket è disponibile nella regione specificata.

Prepara un ambiente

Innanzitutto, imposta le seguenti variabili di ambiente in una nuova finestra di terminale. Nota: assicurati di sostituire i segnaposto di esempio con i tuoi valori univoci, tra cui il nome del cluster, la regione desiderata, la versione di rilascio di Karpenter e la versione di Kubernetes.

> **Tip**
>
> Alcune variabili (come `${AWS_REGION}` e `${K8S_VERSION}`) vengono definite all'inizio del blocco e utilizzate nei comandi successivi per garantire coerenza ed evitare ripetizioni. Assicurati di eseguire i comandi in sequenza in modo che questi valori vengano esportati correttamente e siano disponibili per l'utilizzo nelle definizioni successive.

```
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/')"
```

Crea i ruoli e le policy richiesti

Karpenter necessita di ruoli e policy IAM specifici (ad esempio, il ruolo IAM del controller Karpenter, il profilo dell'istanza e le policy) per gestire le istanze EC2 come nodi worker Kubernetes. Utilizza questi ruoli per eseguire azioni come l'avvio e la chiusura delle istanze EC2, l'etichettatura delle risorse e l'interazione con altri servizi. AWS Crea i ruoli e le policy di Karpenter utilizzando cloudformation.yaml di 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}"
```

L' AWS LBC necessita dell'autorizzazione per fornire e gestire sistemi di AWS bilanciamento del carico, ad esempio per la creazione ALBs di risorse per Ingress o per servizi di questo tipo. NLBs `LoadBalancer` Specificheremo questa policy di autorizzazione durante la creazione del cluster. Durante la creazione del cluster, creeremo l'account di servizio con eksctl in. ClusterConfig Crea la policy IAM 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 il driver CSI Mountpoint S3 è installato, i suoi DaemonSet pod sono configurati per utilizzare un account di servizio per l'esecuzione. Il driver CSI Mountpoint per Mountpoint S3 richiede l'autorizzazione per interagire con il bucket Amazon S3 che creerai più avanti in questa guida. Specificheremo questa policy di autorizzazione durante la creazione del cluster. Durante la creazione del cluster, creeremo l'account di servizio con eksctl in. ClusterConfig Crea la policy IAM di 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}/*\"]}]}"
```

Nota: se esiste già un ruolo con questo nome, assegna al ruolo un nome diverso. Il ruolo che creiamo in questo passaggio è specifico per il tuo cluster e il tuo bucket S3.

Creazione del cluster

In questo modello, eksctl crea automaticamente un account di servizio Kubernetes per Pod Identity EKS, l'agente di monitoraggio dei nodi, CoreDNS, Kubeproxy, il plug-in CNI del VPC. Ad oggi, il driver CSI Mountpoint di S3 non è disponibile per Pod Identity EKS, quindi creiamo un IRSA (IAM Roles for Service Account) e un endpoint OIDC. Inoltre, creiamo un account di servizio per il AWS Load Balancer Controller (LBC). Per l'accesso ai nodi Bottlerocket, eksctl collega automaticamente Amazon SSMManaged InstanceCore for Bottlerocket per consentire sessioni shell sicure tramite SSM.

Nello stesso terminale in cui imposti le variabili di ambiente, esegui il seguente blocco di comandi per creare il 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
```

Il processo richiede alcuni minuti. Se desideri monitorare lo stato, consulta la console. AWS CloudFormation

2. Verifica dello stato del nodo del cluster e del pod

Eseguiamo alcuni controlli dell'integrità per assicurarci che il cluster sia pronto. Quando il comando precedente viene completato, visualizza i tipi di istanze e verifica che tutti i nodi del sistema CPU abbiano raggiunto lo stato `Ready` con il comando seguente:

```
kubectl get nodes -L node.kubernetes.io/instance-type
```

L'output previsto dovrebbe essere simile al seguente:

```
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
```

Verifica tutte le associazioni Pod Identity e il modo in cui associano un ruolo a un account di servizio in un namespace nel cluster con il comando seguente:

```
eksctl get podidentityassociation --cluster ${EKS_CLUSTER_NAME} --region ${AWS_REGION}
```

L'output dovrebbe mostrare i ruoli IAM per Karpenter («karpenter») e AWS LBC (» «). aws-load-balancer-controller

Verifica che siano disponibili: DaemonSets

```
kubectl get daemonsets -n kube-system
```

L'output previsto dovrebbe essere simile al seguente:

```
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
```

Verifica che tutti i componenti aggiuntivi siano installati nel cluster:

```
eksctl get addons --cluster ${EKS_CLUSTER_NAME} --region ${AWS_REGION}
```

L'output previsto dovrebbe essere simile al seguente:

```
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. Installa Karpenter

Installa il controller Karpenter sui nodi worker della CPU (`cpu-worker`) per ottimizzare i costi e conservare le risorse della GPU. Lo installeremo nel namespace "kube-system" e specificheremo l'account di servizio "karpenter" che abbiamo definito durante la creazione del cluster. Inoltre, questo comando configura il nome del cluster e una coda di interruzione delle istanze spot per i nodi della CPU. Karpenter utilizzerà IRSA per assumere questo ruolo 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
```

L'output previsto dovrebbe essere simile al seguente:

```
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
```

Verifica che Karpenter sia in esecuzione:

```
kubectl get pods -n kube-system -l app.kubernetes.io/name=karpenter
```

L'output previsto dovrebbe essere simile al seguente:

```
NAME                         READY   STATUS    RESTARTS   AGE
karpenter-555895dc-865bc   1/1     Running   0          5m58s
karpenter-555895dc-j7tk9   1/1     Running   0          5m58s
```

4. Configurazione Karpenter NodePools

In questo passaggio, configuriamo CPU e GPU Karpenter che si escludono a vicenda. NodePools Il `limits` campo nelle NodePool specifiche limita le risorse totali massime (ad esempio, CPU, memoria GPUs) che ciascuna di esse NodePool può consumare su tutti i nodi a cui è stato assegnato il provisioning, impedendo il provisioning aggiuntivo dei nodi in caso di superamento di questi limiti. Sebbene NodePools supporti ampie categorie di istanze (ad esempio,g)`c`, la specificazione di tipi di istanze, tipi di capacità e limiti di risorse specifici consente di stimare più facilmente i costi per i carichi di lavoro on-demand. In questi casi NodePools, utilizziamo una serie diversificata di tipi di istanze all'interno della famiglia di istanze G5. Ciò consente a Karpenter di selezionare automaticamente il tipo di istanza più appropriato in base alle richieste di risorse dei pod, ottimizzando l'utilizzo delle risorse e rispettando i limiti totali. NodePool Per ulteriori informazioni, consulta Creazione. NodePools

Configura la GPU NodePool

In questo caso NodePool, stabiliamo dei limiti di risorse per gestire il provisioning di nodi con funzionalità GPU. Questi limiti sono progettati per limitare le risorse totali su tutti i nodi del pool, consentendo un massimo di 10 istanze in totale. Ogni istanza può essere g5.xlarge (4 v, CPUs 16 GiB di memoria, 1 GPU) o g5.2xlarge (8 vCPUs, 32 GiB di memoria, 1 GPU), purché la CPUs v totale non superi 80, la memoria totale non superi 320 GiB e il totale non superi 10. GPUs Ad esempio, il pool può fornire 10 istanze g5.2xlarge (80 v, CPUs 320 GiB, 10) o 10 istanze g5.xlarge (40 v, 160 GiB, 10 GPUs) o una combinazione come 5 g5.xlarge e 5 g5.2xlarge (60 vCPUs, 240 GiB, GPUs 10), garantendo la flessibilità in base alle richieste del carico di lavoro rispettando i vincoli delle risorse. CPUs GPUs

Inoltre, specifichiamo l'ID della variante Nvidia dell'AMI Bottlerocket. Infine, abbiamo stabilito una policy di interruzione per rimuovere i nodi vuoti dopo 30 minuti (`consolidateAfter: 30m`) e abbiamo impostato una durata massima dei nodi di 30 giorni (`expireAfter: 720h`) per ottimizzare i costi e mantenere lo stato dei nodi per le attività che richiedono un uso intensivo della GPU. Per ulteriori informazioni, consulta Disabilitare il consolidamento di Karpenter per carichi di lavoro sensibili alle interruzioni e Use ttlSecondsAfter Finished to Auto Clean-Up Kubernetes Jobs.

```
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
```

L'output previsto dovrebbe essere simile al seguente:

```
nodepool.karpenter.sh/gpu-a10g-inference-g5 created
ec2nodeclass.karpenter.k8s.aws/gpu-a10g-inference-ec2 created
```

Verifica che sia stato creato e funzionante: NodePool

```
kubectl get nodepool gpu-a10g-inference-g5 -o yaml
```

Cerca `status.conditions` «mi piace`ValidationSucceeded: True`» e `Ready: True` verifica che NodePool sia sano. `NodeClassReady: True`

Configura la CPU NodePool

In questo caso NodePool, abbiamo fissato dei limiti per supportare circa 50 istanze, allineandoci a un carico di lavoro moderato della CPU (ad esempio, 100-200 pod) e alle quote vCPU AWS tipiche (ad esempio 128-1152). I limiti vengono calcolati presupponendo che si NodePool debba scalare fino a 50 istanze m7.xlarge: CPU (4 v CPUs per istanza × 50 istanze = 200 v) CPUs e memoria (16 GiB per istanza × 50 istanze = 800 GiB). Questi limiti sono progettati per limitare le risorse totali su tutti i nodi del pool, consentendo fino a 50 istanze da m7g.xlarge (ciascuna con 4 v e CPUs 16 GiB di memoria), purché la v totale non superi 200 e la memoria totale CPUs non superi 800 GiB.

Inoltre, specifichiamo l'ID della variante standard dell'AMI Bottlerocket. Infine, abbiamo stabilito una policy di interruzione per rimuovere i nodi vuoti dopo 60 minuti (`consolidateAfter: 60m`) e abbiamo impostato una durata massima dei nodi di 30 giorni (`expireAfter: 720h`) per ottimizzare i costi e mantenere lo stato dei nodi per le attività che richiedono un uso intensivo della GPU. Per ulteriori informazioni, consulta Disabilitare il consolidamento di Karpenter per carichi di lavoro sensibili alle interruzioni e Use Finished to Auto Clean-Up Kubernetes Jobs. ttlSecondsAfter

```
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
```

L'output previsto dovrebbe essere simile al seguente:

```
nodepool.karpenter.sh/cpu-inference-m7gxlarge created
ec2nodeclass.karpenter.k8s.aws/cpu-inference-m7gxlarge-ec2 created
```

Verifica che sia stato creato e funzionante: NodePool

```
kubectl get nodepool cpu-inference-m7gxlarge -o yaml
```

Cerca `status.conditions` «mi piace`ValidationSucceeded: True`» e `Ready: True` verifica che NodePool sia sano. `NodeClassReady: True`

5. Implementa un Pod GPU per esporre una GPU

È necessario il plug-in per dispositivi di Nvidia per consentire a Kubernetes di esporre i dispositivi GPU al cluster Kubernetes. In genere, è necessario distribuire il plug-in come un file DaemonSet; tuttavia, l'AMI Bottlerocket preinstalla il plug-in come parte dell'AMI. Ciò significa che quando si utilizza Bottlerocket AMIs, non è necessario implementare il plug-in del dispositivo Nvidia. DaemonSet Per ulteriori informazioni, consulta Kubernetes Device Plugin to expose. GPUs

Implementa un pod di esempio

Karpenter agisce in modo dinamico: effettua il provisioning dei nodi GPU quando un carico di lavoro (pod) richiede risorse GPU. Per verificare che i pod siano in grado di richiedere e utilizzare GPUs, implementa un pod che richieda la `nvidia.com/gpu` risorsa nei suoi limiti (ad esempio,).

`nvidia.com/gpu: 1` Per ulteriori informazioni su queste etichette, consulta Schedule workloads with GPU requirements using Well-Known labels.

```
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
```

L'output previsto dovrebbe essere simile al seguente:

```
pod/gpu-ndivia-smi created
```

Aspetta un minuto e controlla se il Pod ha lo stato «In sospeso»ContainerCreating, «In esecuzione» e quindi «Completato»:

```
kubectl get pod gpu-nvidia-smi -w
```

Verifica che il nodo del pod appartenga alla GPU: NodePool

```
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"
```

L'output previsto dovrebbe essere simile al seguente:

```
Name                              Nodepool
ip-192-168-83-245.ec2.internal    gpu-a10g-inference-g5
```

Controlla i log del pod:

```
kubectl logs gpu-nvidia-smi
```

L'output previsto dovrebbe essere simile al seguente:

```
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. (Facoltativo) Prepara e carica gli artefatti del modello per l'implementazione

In questa fase, implementerai il servizio di un modello per la classificazione delle immagini in tempo reale, iniziando con il caricamento dei pesi dei modelli in un bucket Amazon S3. A titolo dimostrativo, stiamo utilizzando il modello di visione open source GPUNet-0 di NVIDIA GPUNet, che supporta l'inferenza a bassa latenza sulle immagini utilizzando NVIDIA e TensorRT. GPUs Questo modello è preconfigurato ImageNet, ci consente di classificare al volo gli oggetti nelle foto o nei flussi video ed è considerato un modello piccolo con 11,9 milioni di parametri.

Configurare l'ambiente

Per scaricare i pesi del modello GPUNet -0 In questo passaggio, è necessario accedere al catalogo NGC di NVIDIA e al Docker installato sul computer locale. Segui questa procedura per configurare un account gratuito e la CLI NGC:

- Crea un account NGC gratuito e genera una chiave API dalla dashboard NGC (Icona utente > Configurazione > Genera chiave API > Genera chiave personale > Catalogo NGC).
- Scarica e installa NGC CLI Linux/macOS/Windows () e configura la CLI utilizzando:. `ngc config set` Inserisci la tua chiave API quando richiesto, poi imposta org su `nvidia` e premi Invio per accettare i valori predefiniti per gli altri. Se l'operazione viene eseguita correttamente, dovresti vedere qualcosa come: `Successfully saved NGC configuration to /Users/your-username/.ngc/config`.

Verifica le autorizzazioni dell'account di servizio

Prima di iniziare, controlla le autorizzazioni dell'account del servizio Kubernetes:

```
kubectl get serviceaccount s3-csi-driver-sa -n kube-system -o yaml
```

Durante la creazione del cluster, abbiamo collegato la CSIDriver policy S3 a un ruolo IAM e annotato l'account del servizio («s3-»). csi-driver-sa I pod del driver CSI Mountpoint di S3 ereditano le

autorizzazioni del ruolo IAM quando interagiscono con S3. L'output previsto dovrebbe essere simile al seguente:

```
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
```

Aggiungi una tolleranza

Il driver S3 CSI funziona come un su tutti i nodi. DaemonSet I pod utilizzano il driver CSI su quei nodi per montare i volumi S3. Per consentirne la programmazione sui nostri nodi GPU che presentano difetti, aggiungi una tolleranza a: 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"}}]'
```

L'output previsto dovrebbe essere simile al seguente:

```
daemonset.apps/s3-csi-node patched
```

Carica i pesi dei modelli su S3

In questo passaggio, creerai un bucket Amazon S3, scaricherai i pesi del modello GPUNet -0 da NVIDIA GPU Cloud (NGC) e li caricherai nel bucket. La nostra applicazione accederà a questi pesi in fase di runtime per scopi di inferenza.

Crea un bucket Amazon S3:

```
aws s3 mb s3://${S3_BUCKET_NAME} --region ${AWS_REGION}
```

Abilita il controllo delle versioni S3 per il bucket, per evitare che eliminazioni e sovrascritture accidentali causino una perdita di dati immediata e permanente:

```
aws s3api put-bucket-versioning --bucket ${S3_BUCKET_NAME} --versioning-configuration
 Status=Enabled
```

Applica una regola del ciclo di vita al bucket per eliminare le versioni degli oggetti sovrascritte o eliminate 14 giorni dopo che sono diventate non attuali e per rimuovere i marcatori di eliminazione scaduti e i caricamenti incompleti in più parti dopo 7 giorni. Per ulteriori informazioni, consulta Examples of S3 Lifecycle configurations.

```
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}}]}'
```

Scarica i pesi del modello -0 da NGC. GPUNet Ad esempio, su macOS:

```
ngc registry model download-version nvidia/dle/gpunet_0_pyt_ckpt:21.12.0_amp --dest ~/
downloads
```

> **Note**
>
> Potresti dover modificare questo comando di download per il sistema operativo in uso. Affinché questo comando funzioni su un sistema Linux, probabilmente è necessario creare la directory come parte del comando (ad esempio, `mkdir ~/downloads`).

L'output previsto dovrebbe essere simile al seguente:

```
{
  "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]"
}
```

Rinomina il file del checkpoint in modo che corrisponda alla denominazione prevista nel codice dell'applicazione nei passaggi successivi (non è necessaria alcuna estrazione, poiché si tratta di un checkpoint PyTorch *.pth.tar standard contenente il dizionario dello stato del modello):

```
mv ~/downloads/gpunet_0_pyt_ckpt_v21.12.0_amp/0.65ms.pth.tar gpunet-0.pth
```

Abilita AWS Common Runtime nella AWS CLI per ottimizzare il throughput di S3:

```
aws configure set s3.preferred_transfer_client crt
```

Carica i pesi del modello nel bucket S3:

```
aws s3 cp gpunet-0.pth s3://${S3_BUCKET_NAME}/gpunet-0.pth
```

L'output previsto dovrebbe essere simile al seguente:

```
upload: ./gpunet-0.pth to s3://eks-rt-inference-models-us-east-1-1752722786/
gpunet-0.pth
```

Crea il servizio del modello

In questo passaggio, configurerai un'applicazione web FastAPI per la classificazione delle immagini con accelerazione GPU utilizzando il modello di visione -0. GPUNet L'applicazione scarica i pesi dei modelli da Amazon S3 in fase di esecuzione, recupera l'architettura del modello dal repository NVIDIA per la memorizzazione nella cache e scarica le etichette delle classi tramite HTTP. ImageNet L'applicazione include la preelaborazione, la trasformazione e l'esposizione di due endpoint: un GET root per il controllo dell'integrità e un endpoint `/predict` POST che accetta l'URL di un'immagine.

Utilizziamo il modello utilizzando FastAPI con PyTorch caricamento dei pesi da Amazon S3 in fase di esecuzione in una configurazione containerizzata per la prototipazione rapida e l'implementazione di Kubernetes. Per altri metodi come il batching ottimizzato o i motori ad alto throughput, consulta Serving ML Models.

Creazione dell'applicazione

Crea una directory per i file dell'applicazione, ad esempio `model-testing`, quindi modifica le directory al suo interno e aggiungi il codice seguente a un nuovo file denominato `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))
```

Crea un Dockerfile

Il seguente Dockerfile crea un'immagine del contenitore per la nostra applicazione utilizzando il modello del repository NVIDIA Deep Learning Examples for Tensor Cores GPUNet . GitHub

Riduciamo le dimensioni dell'immagine del contenitore utilizzando una PyTorch base di sola esecuzione, installando solo pacchetti essenziali con pulizia della cache, prememorizzazione del codice del modello ed evitando di «caricare» pesi nell'immagine del contenitore per consentire operazioni di estrazione e aggiornamento più rapide. Per ulteriori informazioni, consulta Reducing Container Image Sizes.

Nella stessa directory di `app.py`, crea il `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"]
```

Eseguire il test dell'applicazione

Dalla stessa directory della tua `app.py` e`Dockerfile`, crea l'immagine del contenitore per l'applicazione di inferenza, mirata all'architettura: AMD64

```
docker build --platform linux/amd64 -t gpunet-inference-app .
```

Imposta le variabili di ambiente per le tue AWS credenziali e, facoltativamente, un token di sessione. AWS Esempio:

```
export AWS_REGION="us-east-1"
export AWS_ACCESS_KEY_ID=ABCEXAMPLESCUJFEIELSMUHHAZ
export AWS_SECRET_ACCESS_KEY=123EXAMPLEMZREoQXr8XkiicsOgWDQ5TpUsq0/Z
```

Esegui il contenitore localmente, inserendo AWS le credenziali come variabili di ambiente per l'accesso a S3. Esempio:

```
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
```

L'output previsto dovrebbe essere simile al seguente:

```
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)
```

In una nuova finestra del terminale, verifica l'endpoint di inferenza inviando un esempio di richiesta POST con un URL di immagine pubblica come parametro di query:

```
curl -X POST "http://localhost:8080/predict?image_url=http://images.cocodataset.org/
test-stuff2017/000000024309.jpg"
```

L'output previsto dovrebbe essere una risposta JSON con le prime 5 previsioni, simile a questa (le etichette e le probabilità effettive possono variare leggermente in base alla precisione dell'immagine e del modello):

```
{"predictions":[{"label":"desk","probability":0.28885871171951294},
{"label":"laptop","probability":0.24679335951805115},
{"label":"notebook","probability":0.08539070934057236},
{"label":"library","probability":0.030645888298749924},
{"label":"monitor","probability":0.029896063730120 66}]}
```

Esci dall'applicazione usando "Ctrl + C".

Invia il container ad Amazon ECR

In questa fase, carichiamo l'immagine del contenitore per il servizio modello GPUNet -0 su Amazon Elastic Container Registry (ECR), rendendola disponibile per la distribuzione su Amazon EKS. Questo processo prevede la creazione di un nuovo repository ECR per archiviare l'immagine, l'autenticazione con ECR e poi l'etichettatura e l'invio dell'immagine del container nel nostro registro.

Innanzitutto, torna alla directory in cui hai impostato le variabili di ambiente all'inizio di questa guida. Esempio:

```
cd ..
```

Crea un repository in Amazon ECR:

```
aws ecr create-repository --repository-name gpunet-inference-app --region ${AWS_REGION}
```

Accedi ad 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
```

L'output previsto dovrebbe essere simile al seguente:

```
Login Succeeded
```

Tagga l'immagine:

```
docker tag gpunet-inference-app:latest ${AWS_ACCOUNT_ID}.dkr.ecr.
${AWS_REGION}.amazonaws.com/gpunet-inference-app:latest
```

Invia l'immagine al tuo repository Amazon ECR:

```
docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/gpunet-inference-
app:latest
```

Per il completamento di questa fase finale possono essere necessari alcuni minuti.

7. (Facoltativo) Esponi il servizio del modello

In questa fase, esporrai il tuo servizio di modello di inferenza in tempo reale esternamente su Amazon EKS utilizzando il Load AWS Balancer Controller (LBC). Questo comporta la configurazione dell'LBC, il montaggio dei pesi del modello da Amazon S3 come volume persistente utilizzando il driver CSI Mountpoint di S3, la distribuzione di un pod applicativo accelerato da GPU, la creazione di un servizio e un ingresso per il provisioning di un Application Load Balancer (ALB) e il test dell'endpoint.

Innanzitutto, verifica l'associazione Pod Identity per AWS LBC, confermando che l'account del servizio sia collegato correttamente al ruolo IAM richiesto:

```
eksctl get podidentityassociation --cluster ${EKS_CLUSTER_NAME} --namespace kube-system
 --service-account-name aws-load-balancer-controller
```

L'output previsto dovrebbe essere simile al seguente:

```
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
```

Assegna tag a un gruppo di sicurezza del cluster

Il AWS Load Balancer Controller supporta solo un singolo gruppo di sicurezza con la chiave del tag `karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"` per la selezione del gruppo di sicurezza di Karpenter. Durante la creazione di un cluster con eksctl, il gruppo di sicurezza del cluster predefinito (che ha il tag `"kubernetes.io/cluster/<cluster-name>: owned"`) non viene automaticamente etichettato con tag `karpenter.sh/discovery`. Questo tag è essenziale per permettere a Karpenter di scoprire e collegare questo gruppo di sicurezza ai nodi che fornisce. Il collegamento di questo gruppo di sicurezza garantisce la compatibilità con il AWS Load Balancer Controller (LBC), consentendogli di gestire automaticamente le regole del traffico in entrata per i servizi esposti tramite Ingress, come il servizio modello in questi passaggi.

Individua l'ID VPC del cluster:

```
CLUSTER_VPC_ID="$(aws eks describe-cluster --name ${EKS_CLUSTER_NAME} --query
 cluster.resourcesVpcConfig.vpcId --output text)"
```

Esporta il gruppo di sicurezza predefinito per il 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)"
```

Aggiungi il tag `karpenter.sh/discovery` al gruppo di sicurezza del cluster predefinito. Ciò consentirà ai nostri selettori di CPU e EC2 NodeClass GPU di utilizzarlo:

```
aws ec2 create-tags --resources ${CLUSTER_SG_ID} --tags Key=karpenter.sh/
discovery,Value=${EKS_CLUSTER_NAME}
```

Verifica che il tag sia stato aggiunto:

```
aws ec2 describe-security-groups --group-ids ${CLUSTER_SG_ID} --query
 "SecurityGroups[].Tags"
```

Tra i risultati, dovresti vedere quanto segue con il tag e il nome del cluster. Esempio:

```
{
  "Key": "karpenter.sh/discovery",
  "Value": "eks-rt-inference-us-east-1"
}
```

Configurazione del AWS Load Balancer Controller (LBC)

L' AWS LBC è essenziale per gestire il traffico in ingresso ai AI/ML carichi di lavoro su Amazon EKS, garantendo l'accesso agli endpoint di inferenza o alle pipeline di elaborazione dei dati. Integrandosi con AWS Application Load Balancers (ALB) e Network Load Balancers (NLB), LBC indirizza dinamicamente il traffico verso applicazioni containerizzate, come quelle che eseguono modelli linguistici di grandi dimensioni, modelli di visione artificiale o servizi di inferenza in tempo reale. Poiché abbiamo già creato l'account del servizio e la Pod Identity Association durante la creazione del cluster, impostiamo `serviceAccount.name` in modo che corrisponda a quanto definito nella nostra configurazione del cluster (`aws-load-balancer-controller`).

AWS Aggiungi l'archivio di grafici Helm di proprietà di eks-charts:

```
helm repo add eks https://aws.github.io/eks-charts
```

Aggiorna i tuoi repository Helm locali con i grafici più recenti:

```
helm repo update eks
```

Implementa AWS LBC utilizzando Helm, specificando il nome del cluster EKS e facendo riferimento all'account di servizio creato in precedenza:

```
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
```

L'output previsto dovrebbe essere simile al seguente:

```
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!
```

Monta il modello in un volume persistente

In questa fase, monterai i pesi dei modelli dal tuo bucket Amazon S3 utilizzando PersistentVolume un (PV) supportato dal driver CSI Mountpoint for Amazon S3. Questo consente ai pod Kubernetes di accedere agli oggetti S3 come file locali, eliminando i download a uso intensivo di risorse verso container di archiviazione temporanei su pod o container init, ideali per modelli di pesi di grandi dimensioni da più gigabyte.

Il PV monta l'intera root del bucket (nessun percorso specificato in `volumeAttributes`), supporta l'accesso simultaneo in sola lettura da più pod ed espone file come il peso dei modelli (`/models/gpunet-0.pth`) all'interno del container per l'inferenza. Ciò garantisce che il fallback "download" nella nostra applicazione (`app.py`) non si attivi perché il file esiste tramite il mount. Disaccoppiando

il modello dall'immagine del container, viene consentito l'accesso condiviso e gli aggiornamenti indipendenti delle versioni del modello senza ricostruire l'immagine.

Crea il (PV) PersistentVolume

Crea una risorsa PersistentVolume (PV) per montare il bucket S3 contenente i pesi del modello, abilitando l'accesso in sola lettura a più pod senza scaricare file in fase di esecuzione:

```
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
```

Crea il PersistentVolumeClaim (PVC)

Create un PersistentVolumeClaim (PVC) da collegare al PV, richiedendo l'accesso in sola lettura ai dati del modello S3 montato:

```
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
```

Distribuzione dell'applicazione

Implementa l'applicazione di inferenza come implementazione Kubernetes, montando il volume persistente supportato da S3 per l'accesso al modello, applicando i selettori e le tolleranze dei nodi GPU e impostando le variabili di ambiente per il percorso del modello. Questa implementazione imposta il percorso del modello (var amb di "`/models/gpunet-0.pth`"), quindi la nostra applicazione (in `app.py`) utilizzerà questo percorso per impostazione predefinita. Con il montaggio del volume di implementazione su `/models` (sola lettura), il download del modello non si attiva se il file è già presente tramite il 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
```

Karpenter impiegherà alcuni minuti per eseguire il provisioning di un nodo GPU se non è già disponibile. Verifica che il pod di inferenza sia nello stato "Running":

```
kubectl get pods -l app=gpunet-inference-app
```

L'output previsto dovrebbe essere simile al seguente:

```
NAME                                READY   STATUS    RESTARTS   AGE
gpunet-inference-app-5d4b6c7f8-abcde        1/1     Running   0          2m
```

Esponi il servizio con Ingress e bilanciatore del carico

Crea un servizio ClusterIP per esporre l'implementazione dell'inferenza internamente nel cluster EKS, mirando alla porta dell'applicazione:

```
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
```

Crea una risorsa Ingress per fornire un Application Load Balancer (ALB) con accesso a Internet tramite AWS LBC, indirizzando il traffico esterno al servizio di inferenza:

```
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
```

Application Load Balancer (ALB) completerà il provisioning nell'arco di alcuni minuti. Monitora lo stato delle risorse Ingress per confermare che sia stato eseguito il provisioning dell'ALB:

```
kubectl get ingress gpunet-model-ingress
```

L'output previsto dovrebbe essere simile al seguente (con il campo ADDRESS compilato):

```
NAME                    CLASS    HOSTS    ADDRESS
 PORTS   AGE
gpunet-model-ingress    alb      *        k8s-default-gpunetmo-183de3f819-516310036.us-
east-1.elb.amazonaws.com   80      6m58s
```

Estrai ed esporta il nome host dell'ALB dallo stato Ingress per utilizzarlo nei test successivi:

```
export ALB_HOSTNAME=$(kubectl get ingress gpunet-model-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

Prova il servizio del modello

Convalida l'endpoint di inferenza esposto inviando una richiesta POST con un URL di immagine di esempio (ad esempio, dal set di dati COCO), simulando la previsione in tempo reale:

```
curl -X POST "http://${ALB_HOSTNAME}/predict?image_url=http://images.cocodataset.org/
test-stuff2017/000000024309.jpg"
```

L'output previsto dovrebbe essere una risposta JSON con le prime 5 previsioni, simile a questa (le etichette e le probabilità effettive possono variare leggermente in base alla precisione dell'immagine e del modello):

```
{"predictions":[{"label":"desk","probability":0.2888975441455841},
{"label":"laptop","probability":0.246435031294827},
{"label":"notebook","probability":0.08554483205080032},
{"label":"library","probability":0.030612602829933167},
{"label":"monitor","probability":0.02989672815084457}]}
```

Facoltativamente, puoi continuare a testare altre immagini in una nuova richiesta POST. Esempio:

```
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
```

Conclusioni

In questa guida, configurerai un cluster Amazon EKS ottimizzato per carichi di lavoro di inferenza in tempo reale accelerati da GPU. Hai effettuato il provisioning di un cluster con istanze G5 EC2, installato il driver CSI Mountpoint S3, EKS Pod Identity Agent, EKS Node Monitoring Agent,

BottlerocketAMI, Load AWS Balancer Controller (LBC) e Karpenter per gestire CPU e GPU. NodePools Hai utilizzato il NVIDIA Device Plugin per abilitare la pianificazione delle GPU e configurato S3 con un e per l'accesso al modello. PersistentVolume PersistentVolumeClaim Hai convalidato la configurazione distribuendo un pod GPU di esempio, configurando l'accesso al modello NVIDIA GPUNet-0 su Amazon S3, abilitando l'inizializzazione del pod ed esponendo il servizio di inferenza tramite Application Load Balancer. Per utilizzare appieno il cluster, configura l'agente di monitoraggio dei nodi di EKS con riparazione automatica. Assicurati di condurre test di benchmark, tra cui valutazioni delle prestazioni, della latenza e del throughput della GPU per ottimizzare i tempi di risposta. Per ulteriori informazioni, consulta Utilizzo degli strumenti di monitoraggio e osservabilità per i carichi di lavoro. AI/ML

Eliminazione

Per evitare di incorrere in addebiti futuri, è necessario eliminare manualmente lo CloudFormation stack associato per eliminare tutte le risorse create durante questa guida, inclusa la rete VPC.

Elimina lo CloudFormation stack usando il flag con eksctl: `--wait`

```
eksctl delete cluster --region ${AWS_REGION} --name ${EKS_CLUSTER_NAME} --wait
```

Al momento del completamento, dovresti vedere il seguente messaggio di risposta:

```
2025-07-29 13:03:55 [#]  all cluster resources were deleted
```

Elimina il bucket Amazon S3 creato durante questa guida utilizzando la console Amazon S3.

Avvio rapido: inferenza LLM ad alto rendimento con vLLM su Amazon EKS

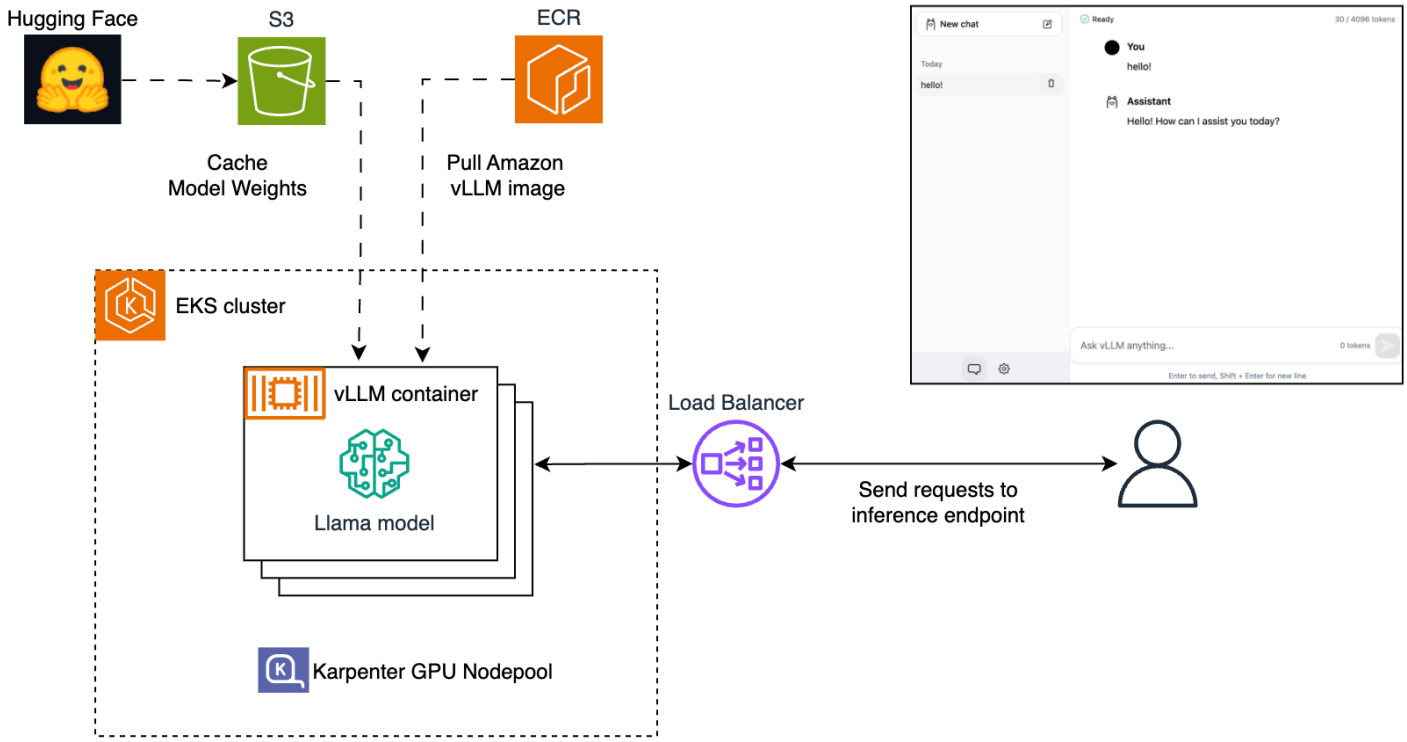
Introduzione

Questa guida rapida fornisce una procedura dettagliata per la distribuzione di Large Language Models () LLMs su Amazon EKS utilizzando VLLM e GPUs per applicazioni di inferenza in tempo reale basate su testo.

La soluzione sfrutta Amazon EKS per l'orchestrazione dei container e VLLM per un servizio di modelli efficiente, consentendoti di creare applicazioni AI scalabili con accelerazione GPU e server di inferenza ad alto rendimento. Il modello Llama 3.1 8B Instruct viene utilizzato a scopo dimostrativo, ma è possibile implementare qualsiasi altro LLM supportato da vLLM (consulta la documentazione di VLLm per un elenco dei modelli supportati). Per testare l'inferenza LLM, utilizziamo un'applicazione

chatbot di esempio basata sul progetto. nextjs-vllm-ui Infine, utilizziamo GuideLM per confrontare e ottimizzare i parametri di configurazione vLLM per ottimizzare le prestazioni di inferenza.

Architettura VLLm su EKS



Una volta completata questa procedura, disporrete di un endpoint di inferenza VLLm ottimizzato per la velocità effettiva e la bassa latenza e sarete in grado di interagire con un modello Llama tramite un'applicazione di chat frontend, dimostrando un tipico caso d'uso per gli assistenti di chatbot e altre applicazioni basate su LLM.

Per ulteriori indicazioni e risorse di implementazione avanzate, consulta la nostra guida alle migliori pratiche EKS per i carichi di lavoro e l'intelligenza artificiale pronta per la produzione sui grafici di inferenza EKS. AI/ML

Prima di iniziare

Prima di iniziare, assicurati di avere:

- Un cluster Amazon EKS con i seguenti componenti principali: pool di nodi Karpenter con famiglia di EC2 istanze G5 o G6, NVIDIA Device Plugin installato sui nodi di lavoro abilitati alla GPU e il driver S3 Mountpoint CSI installato. Per creare questa configurazione di base, segui i passaggi indicati, fino al completamento del passaggio #4. the section called "Crea cluster"

- Un account Hugging Face. Per registrarti, consulta https://huggingface.co/login.

Configurazione dello storage modello con Amazon S3

Archivia in modo efficiente file LLM di grandi dimensioni in Amazon S3 per separare lo storage dalle risorse di elaborazione. Questo approccio semplifica gli aggiornamenti dei modelli, riduce i costi e semplifica la gestione nelle configurazioni di produzione. S3 gestisce file di grandi dimensioni in modo affidabile, mentre l'integrazione con Kubernetes tramite il driver CSI Mountpoint consente ai pod di accedere a modelli come l'archiviazione locale, senza bisogno di lunghi download all'avvio. Segui questi passaggi per creare un bucket S3, caricare un LLM e montarlo come volume nel tuo contenitore di servizi di inferenza.

Su EKS sono disponibili anche altre soluzioni di storage per la memorizzazione nella cache dei modelli, come EFS e FSx Lustre. Per ulteriori informazioni, consulta EKS Best Practices.

Impostazione delle variabili di ambiente

Crea un nome univoco per un nuovo bucket Amazon S3 che creeremo più avanti in questa guida. Una volta creato, usa lo stesso nome di bucket per tutti i passaggi. Esempio:

```
MY_BUCKET_NAME=model-store-$(date +%s)
```

Definisci le variabili di ambiente e memorizzale in un file:

```
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
```

Carica le variabili di ambiente nell'ambiente della shell. Se chiudi l'ambiente di shell corrente e ne apri uno nuovo, assicurati di rifornire le variabili di ambiente usando lo stesso comando:

```
source .env-quickstart-vllm
```

Crea un bucket S3 per archiviare i file del modello

Crea un bucket S3 per archiviare i file del modello:

```
aws s3 mb s3://${BUCKET_NAME} --region ${AWS_REGION}
```

Scarica il modello da Hugging Face

Hugging Face è uno dei principali hub di modelli per l'accesso ai modelli LLM. Per scaricare il modello Llama, dovrai accettare la licenza del modello e configurare l'autenticazione tramite token:

1. Accetta la licenza del modello Llama 3.1 8B Instruct all'indirizzo https://huggingface. co/meta-llama/Llama-3.1-8B-Instruct.
2. Genera un token di accesso (vai al tuo profilo > Impostazioni > Token di accesso, quindi crea un nuovo token usando il tipo di token Read).

Imposta una variabile di ambiente con il tuo token Hugging Face:

```
export HF_TOKEN=your_token_here
```

Installa il pacchetto pip3 se non è già installato nel tuo ambiente. Comando di esempio in Amazon Linux 2023:

```
sudo dnf install -y python3-pip
```

Installa la CLI Hugging Face:

```
pip install huggingface-hub
```

Scarica il modello Llama-3.1-8B-Instruct da Hugging Face (~15 GB) `--exclude` usando il flag per saltare il formato precedente e scaricare solo i file in formato safetensors ottimizzato, che riduce PyTorch le dimensioni di download mantenendo la piena compatibilità con i motori di inferenza più diffusi:

```
huggingface-cli download meta-llama/Meta-Llama-3.1-8B-Instruct \
  --exclude "original/*" \
  --local-dir ./llama-3.1-8b-instruct \
  --token $HF_TOKEN
```

Verifica i file scaricati:

```
$ ls llama-3.1-8b-instruct
```

L'output previsto dovrebbe essere simile al seguente:

```
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
```

Carica i file del modello

Abilita AWS Common Runtime (CRT) per migliorare le prestazioni di trasferimento S3. Il client di trasferimento basato su CRT offre velocità di trasmissione e affidabilità migliorate per operazioni su file di grandi dimensioni:

```
aws configure set s3.preferred_transfer_client crt
```

Carica il modello:

```
aws s3 cp ./llama-3.1-8b-instruct s3://$BUCKET_NAME/llama-3.1-8b-instruct \
  --recursive
```

L'output previsto dovrebbe essere simile al seguente:

```
...
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
```

Configura le autorizzazioni S3 Mountpoint CSI

Il driver S3 Mountpoint CSI consente l'integrazione nativa tra Kubernetes e S3, consentendo ai pod di accedere direttamente ai file del modello come se fossero archiviazione locale, eliminando la necessità di copie locali durante l'avvio del contenitore.

Crea una policy IAM per consentire al punto di montaggio S3 di leggere dal tuo bucket 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}/*\"]}]}"
```

Trova il nome del ruolo IAM utilizzato dal driver CSI S3 Mountpoint controllando le annotazioni dell'account del servizio S3 CSI Driver:

```
ROLE_NAME=$(kubectl get serviceaccount s3-csi-driver-sa -n kube-system -o
 jsonpath='{.metadata.annotations.eks\.amazonaws\.com/role-arn}' | cut -d'/' -f2)
```

Allega la tua policy IAM al ruolo CSI di S3 Mountpoint:

```
aws iam attach-role-policy \
 --role-name ${ROLE_NAME} \
 --policy-arn arn:aws:iam::${AWS_ACCOUNT_ID}:policy/S3BucketAccess-${BUCKET_NAME}
```

Se S3 Mountpoint CSI non è installato nel cluster, segui la procedura di distribuzione riportata in. the section called “Crea cluster”

Monta il bucket S3 come volume Kubernetes

Crea un Persistent Volume (PV) e un Persistent Volume Claim (PVC) per fornire l'accesso in sola lettura al bucket S3 su più pod di inferenza. La modalità di ReadOnlyMany accesso garantisce l'accesso simultaneo ai file del modello, mentre il driver CSI gestisce il montaggio del bucket 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
```

Configurazione dell'infrastruttura GPU

Nodi del cluster

Stiamo usando il cluster EKS creato inthe section called "Crea cluster". Questo cluster include pool di nodi Karpenter in grado di fornire ai nodi abilitati alla GPU uno spazio di archiviazione sufficiente per scaricare l'immagine del contenitore VLLm. Se utilizzi un cluster EKS personalizzato, assicurati che possa avviare nodi abilitati alla GPU.

Selezione dell'istanza

La corretta selezione delle istanze per l'inferenza LLM richiede che la memoria GPU disponibile sia sufficiente per caricare i pesi del modello. I pesi dei modelli per Llama 3.1 8B Instruct sono di circa 16 GB (dimensione dei file del modello .safetensor), pertanto è necessario fornire almeno questa quantità di memoria al processo vllm per caricare il modello.

EC2 Le istanze Amazon G5 con A10G GPUs e le istanze G6 con L4 forniscono GPUs entrambe 24 GB di VRAM per GPU, sufficienti per caricare EC2 pesi Llama 3.1 8B Instruct. Se stai implementando

un modello con pesi maggiori, prendi in considerazione l'utilizzo di un tipo di istanza multi-GPU o di una configurazione multinodo.

Driver per dispositivi NVIDIA

I driver NVIDIA forniscono l'ambiente di runtime necessario per consentire ai container di accedere alle risorse GPU in modo efficiente. Consente l'allocazione e la gestione delle risorse GPU all'interno di Kubernetes, rendendole disponibili come risorse pianificabili. GPUs

Il nostro cluster utilizza EKS Bottlerocket AMIs, che include tutti i driver e i plug-in dei dispositivi NVIDIA necessari su tutti i nodi abilitati alla GPU, garantendo l'accessibilità immediata della GPU per carichi di lavoro containerizzati senza configurazioni aggiuntive. Se utilizzi altri tipi di nodi EKS, devi assicurarti che siano installati tutti i driver e i plugin necessari.

Testa l'infrastruttura GPU

Metti alla prova le funzionalità della GPU del tuo cluster eseguendo i passaggi seguenti per assicurarti che i pod possano accedere alle risorse GPU NVIDIA e pianificare correttamente sui nodi abilitati alla GPU.

Implementa un pod di test 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
```

Controlla i log dei pod per verificare che siano elencati i dettagli della GPU, analogamente a quanto riportato di seguito (non necessariamente lo stesso modello di 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
   |
+-----------------------------------------------------------------------------------------
+
```

Questo output mostra che i pod possono accedere correttamente alle risorse della GPU.

IMPORTANTE: questo pod utilizza una configurazione NodeSelector che si allinea ai pool di nodi Karpenter. the section called "Crea cluster" Se utilizzi pool di nodi diversi, assicurati che il pod corrisponda di conseguenza a NodeSelector e Tolerations.

Distribuisci Inference Container

Lo stack di servizi determina sia le prestazioni che le capacità di scalabilità dell'infrastruttura di inferenza. vLLM si è affermata come una soluzione leader per le implementazioni di produzione. L'architettura di vLLM fornisce il batching continuo per l'elaborazione dinamica delle richieste, ottimizzazioni del kernel per un'inferenza più rapida e una gestione efficiente della memoria GPU tramite. PagedAttention Queste funzionalità, combinate con un'API REST pronta per la produzione e il supporto per i formati di modello più diffusi, lo rendono una scelta ottimale per implementazioni di inferenza ad alte prestazioni.

Seleziona l'immagine del Deep Learning Container AWS

AWS Deep Learning Containers (DLCs) forniscono ambienti preottimizzati con aggiornamenti di sicurezza, compatibilità AWS dell'infrastruttura e configurazioni ottimizzate dei driver. Ciò riduce la complessità di implementazione e il sovraccarico di manutenzione, garantendo al contempo la disponibilità della produzione.

Per questa implementazione, utilizzeremo il AWS DLC per VLLm 0.9, che include le librerie Nvidia e configurazioni ottimizzate delle prestazioni della GPU specificamente ottimizzate per l'inferenza del modello di trasformatore su istanze GPU. AWS

```
image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/vllm:0.9-gpu-py312-ec2
```

Applica i manifesti di VLM Kubernetes

Esistono diversi modi per implementare vLLm in EKS. Questa guida illustra l'implementazione di VLLM utilizzando una distribuzione Kubernetes, un modo semplice e nativo di Kubernetes per iniziare. Per opzioni di implementazione avanzate, consulta i documenti VLLm e i blueprint AI on EKS.

Definisci i parametri di implementazione tramite i manifesti di Kubernetes per controllare l'allocazione delle risorse, il posizionamento dei nodi, i controlli sullo stato di salute, l'esposizione del servizio, ecc. Configura la distribuzione per eseguire un pod abilitato alla GPU utilizzando l'immagine del Deep Learning Container per vLLM. AWS Imposta parametri ottimizzati per l'inferenza LLM ed esponi l'endpoint compatibile con VLLM OpenAPI tramite il servizio Load Balancer: AWS

```
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
```

Verifica che il `Ready  1/1` pod VLm sia in stato:

```
kubectl get pod -l app=vllm-inference-app -w
```

Output previsto:

```
NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
vllm-inference-app-65df5fddc8-5kmjm   1/1     1            1           5m
```

Potrebbero essere necessari alcuni minuti mentre l'immagine del contenitore viene estratta e VLLm carica i file del modello nella memoria della GPU. Procedi solo quando il pod è pronto e disponibile.

Esponi il servizio

Esponi l'endpoint di inferenza localmente tramite il port forwarding di Kubernetes per lo sviluppo e il test locali. Lascia questo comando in esecuzione in una finestra di terminale separata:

```
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
```

Il AWS Load Balancer Controller crea automaticamente un Network Load Balancer che espone l'endpoint del servizio VLLm all'esterno. Recupera l'endpoint NLB eseguendo:

```
NLB=$(kubectl get service vllm-inference-svc -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

Devi installare AWS Load Balancer Controller? Segui le fasi di implementazione riportate inthe section called " AWS Controller Load Balancer".

Esecuzione dell'inferenza

Convalida il pod di inferenza

Convalida la funzionalità del contenitore di inferenza localmente tramite la porta inoltrata. Invia una richiesta di connessione e assicurati che la risposta includa il codice 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
```

Verifica le capacità di inferenza e convalida la connettività esterna inviando una richiesta di completamento all'LLM tramite l'endpoint 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
  }'
```

Questo endpoint segue il formato API OpenAI, rendendolo compatibile con le applicazioni esistenti e fornendo parametri di generazione configurabili come lunghezza e temperatura di risposta per il controllo della diversità di output.

Esegui l'app chatbot

A scopo dimostrativo, questa guida esegue un'applicazione chatbot di esempio che utilizza Project nextjs-vllm-uiper mostrare le interazioni degli utenti con il modello.

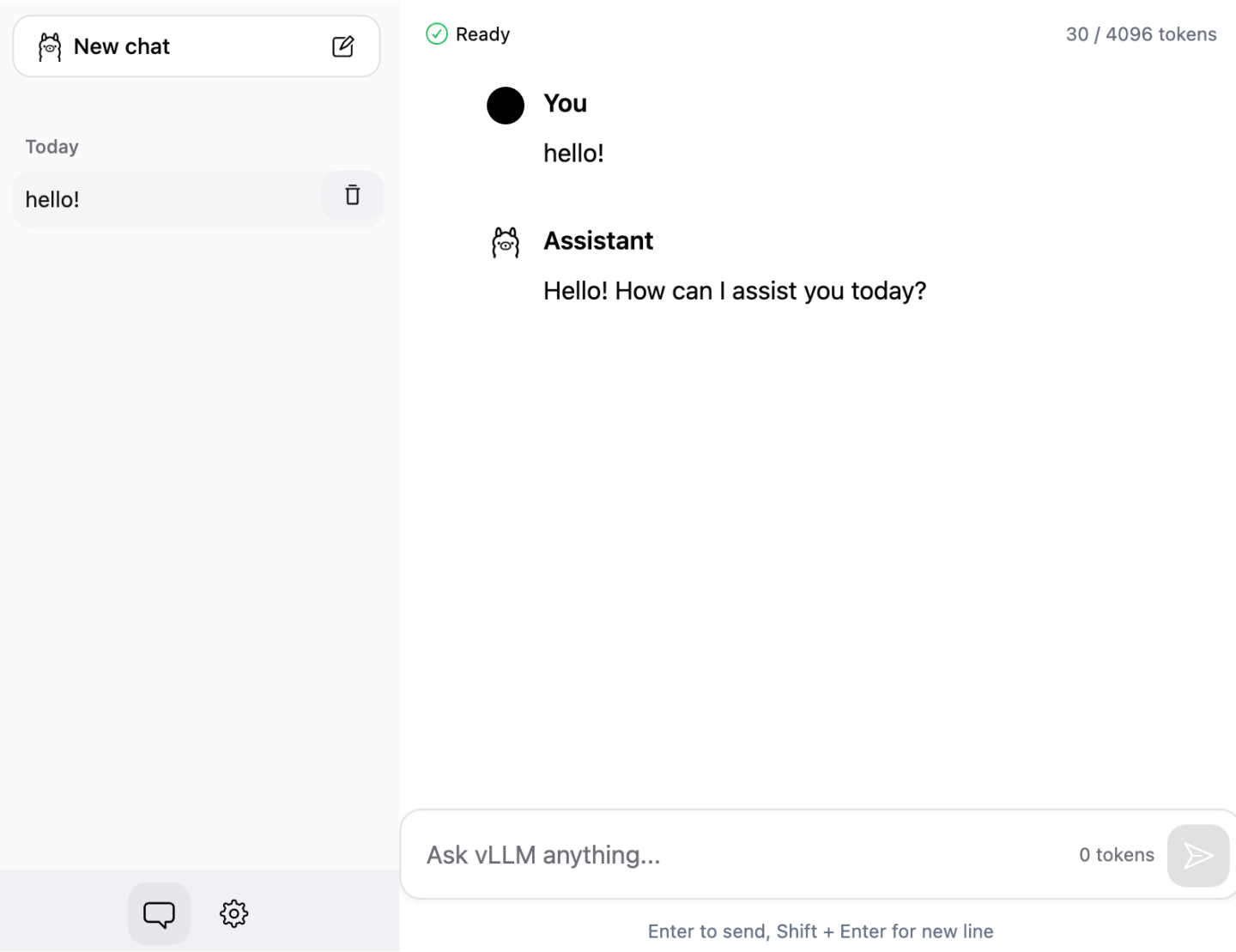
Esegui un'interfaccia utente di chatbot come contenitore Docker che mappa la porta 3000 su localhost e si connette all'endpoint VLLM NLB:

```
docker run --rm \
  -p 3000:3000 \
  -e VLLM_URL="http://${NLB}:80" \
  --name nextjs-vllm-ui-demo \
  ghcr.io/yoziru/nextjs-vllm-ui:latest
```

Apri il tuo browser web e vai a: http://localhost:3000/

Dovresti vedere l'interfaccia di chat in cui puoi interagire con il modello Llama.

Interfaccia utente di chat



Ottimizza le prestazioni di inferenza

I motori di inferenza specializzati come VLLm offrono funzionalità avanzate che aumentano in modo significativo le prestazioni di inferenza, tra cui il batching continuo, il caching KV efficiente e meccanismi di attenzione della memoria ottimizzati. È possibile ottimizzare i parametri di configurazione VLLm per migliorare le prestazioni di inferenza soddisfacendo al contempo i requisiti specifici dei casi d'uso e i modelli di carico di lavoro. Una configurazione corretta è essenziale per raggiungere la saturazione della GPU, in quanto consente di estrarre il massimo valore dalle costose risorse GPU, garantendo al contempo un throughput elevato, una bassa latenza e operazioni convenienti. Le seguenti ottimizzazioni vi aiuteranno a massimizzare le prestazioni della vostra implementazione vLLm su EKS.

Configurazioni VLLm di benchmark

Per ottimizzare i parametri di configurazione VLLM in base al tuo caso d'uso, esegui il benchmark di diverse impostazioni utilizzando uno strumento completo di benchmarking dell'inferenza come GuideLM. Questo raccoglierà metriche chiave come il throughput della richiesta al secondo (RPS), la latenza (E2E), il time to first token (TTFT) e end-to-end la latenza di coda (TPOT) per confrontare diverse configurazioni.

Configurazione vLLM di base

Questa è la configurazione di base utilizzata per eseguire VLLm:

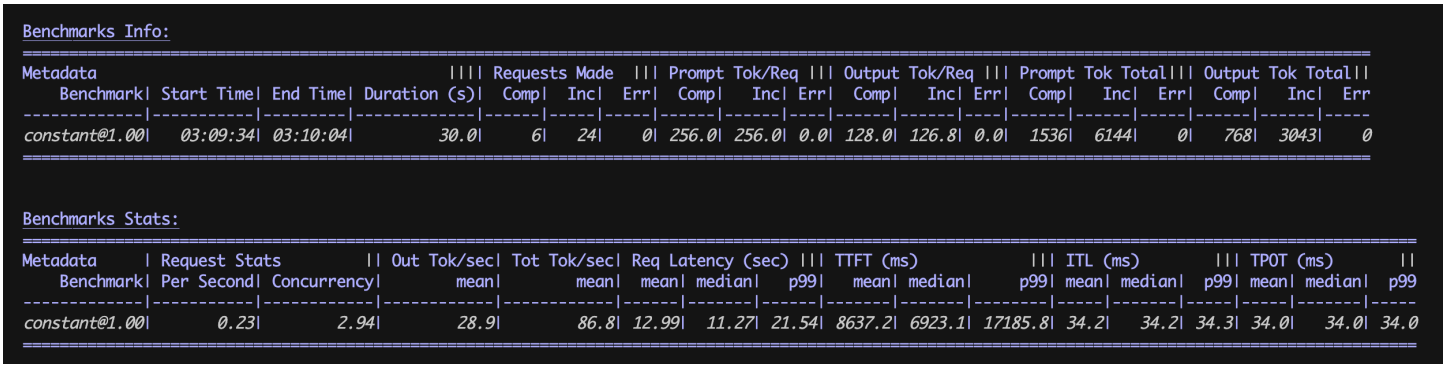
Parametro vLLm	Description
tensor_parallel_size: 1	Distribuisci il modello su 1 GPU
gpu_memory_utilization: 0,90	Riserva il 10% di memoria GPU per il sovraccarico del sistema
lunghezza massima della sequenza: 8192	Lunghezza totale massima della sequenza (ingresso+uscita)
max_num_seqs: 1	Numero massimo di richieste simultanee per GPU (Batching)

Esegui GuideLLM con questa configurazione di base per stabilire una linea di base delle prestazioni. Per questo test, GuidelIM è configurato per generare 1 richiesta al secondo, con richieste a 256 token e risposte a 128 token.

```
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"
```

Output previsto:

Risultati del benchmark di base



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

Configurazione VLLm ottimizzata

Regola i parametri VLLm per utilizzare meglio le risorse della GPU e la parallelizzazione:

Parametro VLLm	Description
tensor_parallel_size: 1	Mantieni 1 GPU. La parallelizzazione dei tensori deve corrispondere al numero di che deve essere utilizzato da vLLM GPUs
gpu_memory_utilization: 0,92	Riduci il sovraccarico di memoria della GPU, se possibile, assicurando al contempo che VLLm continui a funzionare senza errori
lunghezza massima della sequenza: 4096	Adatta la sequenza massima in base ai requisiti del caso d'uso; una sequenza massima inferiore libera risorse che possono essere utilizzate per una maggiore parallelizzazione
max_num_seqs: 8	L'aumento del numero massimo di seq aumenta il throughput ma aumenta anche la latenza. Aumentate questo valore per massimizzare il throughput assicurandovi al contempo che la latenza rimanga entro i requisiti del vostro caso d'uso

Applica queste modifiche alla distribuzione in esecuzione utilizzando il 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"}
]'
```

Verifica che il pod VLLm sia nello stato: `Ready 1/1`

```
kubectl get pod -l app=vllm-inference-app -w
```

Output previsto:

```
NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
vllm-inference-app-65df5fddc8-5kmjm   1/1     1            1           5m
```

Quindi esegui nuovamente GuideLM utilizzando gli stessi valori di benchmarking di prima:

```
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"
```

Output previsto:

Risultati di benchmark ottimizzati

```
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
==========================================================================================================================
```

Risultati del benchmarking

Calcola i risultati del benchmarking in una tabella per la configurazione vLLM di base e ottimizzata:

Valori medi	Configurazione di base	Configurazione ottimizzata
RPS	0,23 giri/sec	0,86 giri/sec
E2E	12,99 s	5,19 s
TTFT	8637,2 ms	147,9 ms
TROTTARE	34,0 ms	39,5 ms

Le configurazioni vLLM ottimizzate hanno migliorato significativamente il throughput di inferenza (RPS) e ridotto la latenza (E2E, TTFT) con solo un lieve aumento di millisecondi della latenza di coda (TPOT). Questi risultati dimostrano come vLLM migliori in modo significativo le prestazioni di inferenza, consentendo a ciascun contenitore di elaborare più richieste in meno tempo per un funzionamento conveniente.

Configurazione del cluster Amazon EKS per AI/ML carichi di lavoro

Questa sezione è progettata per aiutarti a configurare i cluster Amazon EKS ottimizzati per i AI/ML carichi di lavoro. Troverai indicazioni sull'esecuzione di container accelerati da GPU utilizzando Linux e Windows ottimizzati AMIs, sulla configurazione di cluster di formazione con Elastic Fabric Adapter (EFA) per reti ad alte prestazioni e sulla creazione di cluster di inferenza AWS con istanze Inferentia, inclusi prerequisiti, procedure e considerazioni sulla distribuzione. step-by-step

Argomenti

- Usa l'accelerazione ottimizzata per EKS per le istanze AMIs GPU
- Installa il plug-in del dispositivo Kubernetes per GPUs
- Esegui container accelerati da GPU (Windows su EC2 G-Series)
- Esecuzione dei corsi di machine learning su Amazon EKS con Elastic Fabric Adapter
- Usa le istanze AWS Inferentia con Amazon EKS for Machine Learning

Usa l'accelerazione ottimizzata per EKS per le istanze AMIs GPU

Amazon EKS supporta Amazon Linux e Bottlerocket AMIs per istanze GPU ottimizzati per EKS. L'accelerato ottimizzato per EKS AMIs semplifica l'esecuzione di carichi di lavoro AI e ML nei cluster

EKS fornendo immagini del sistema operativo predefinite e convalidate per lo stack Kubernetes accelerato. Oltre ai componenti principali di Kubernetes inclusi nello standard ottimizzato per EKS AMIs, l'accelerato ottimizzato per EKS AMIs include i moduli e i driver del kernel necessari per eseguire le GPU NVIDIA `G` e le istanze `P` EC2 e le istanze AWS GPU Inferentia e Trainium EC2 nei cluster EKS.

La tabella seguente mostra i tipi di istanza GPU supportati per ogni variante AMI accelerata ottimizzata per EKS. Consulta le AL2023 versioni ottimizzate per EKS e le versioni di Bottlerocket su GitHub per gli ultimi aggiornamenti alle varianti AMI.

Variante AMI EKS	Tipi di istanza EC2
AL2023 NVIDIA x86_64	p6-b300, p6-b200, p5, p5e, p5en, p4d, p4de, p3, p3dn, g7e, gr6, g6, g6e, g6f, gr6f, g5, g4dn
AL2023 ARM NVIDIA	p6e-gb200, 5g
AL2023 Neurone x86_64	inf1, inf2, trn1, trn2
Portabottiglie x86_64 aws-k8s-nvidia	p6-b300, p6-b200, p5, p5e, p5en, p4d, p4de, p3, p3dn, g7e, gr6, g6, g6e, g6f, gr6f, g5, g4dn
Portaborraccia aarch64/arm64 aws-k8s-nvidia	g 5 g
Portaborraccia x86_64 aws-k8s	inf1, inf2, trn1, trn2

NVIDIA ottimizzata per EKS AMIs

Utilizzando NVIDIA ottimizzata per EKS AMIs, l'utente accetta il contratto di licenza per l'utente finale del cloud (EULA) di NVIDIA.

Per trovare la versione più recente di NVIDIA ottimizzata per EKS, consulta e. AMIs the section called "Ottieni le ultime novità IDs" the section called "Ottenimento degli ID della versione più recente"

Quando si utilizza Amazon Elastic Fabric Adaptor (EFA) con NVIDIA ottimizzato per EKS AL2023 o Bottlerocket AMIs, è necessario installare il plug-in del dispositivo EFA separatamente. Per ulteriori informazioni, consulta the section called "Configurazione dei cluster di formazione con EFA".

EKS NVIDIA AL2023 AMIs

Quando si utilizza l'operatore GPU NVIDIA con AL2023 NVIDIA ottimizzato per EKS AMIs, è necessario disabilitare l'installazione da parte dell'operatore del driver e del toolkit, poiché questi sono già inclusi in EKS. AMIs I prodotti AL2023 NVIDIA ottimizzati per EKS AMIs non includono il plug-in per dispositivi NVIDIA Kubernetes o il driver NVIDIA DRA, che devono essere installati separatamente. Per ulteriori informazioni, consulta the section called "Installa il plug-in per dispositivi NVIDIA Kubernetes".

Oltre ai componenti AMI EKS standard, AL2023 NVIDIA ottimizzata per EKS AMIs include i seguenti componenti.

- Driver NVIDIA
- Driver in modalità utente NVIDIA CUDA
- Toolkit per container NVIDIA
- NVIDIA Fabric Manager
- NVIDIA ha persistito
- Driver NVIDIA IMEX
- Gestore di sottorete NVIDIA NVLink
- EFA minimal (modulo kernel e rdma-core)

Per i dettagli sul driver in modalità utente NVIDIA CUDA e sul CUDA runtime/libraries utilizzato nei contenitori delle applicazioni, consulta la documentazione NVIDIA. La versione CUDA mostrata `nvidia-smi` è la versione del driver in modalità utente NVIDIA CUDA installato sull'host, che deve essere compatibile con il CUDA utilizzato nei contenitori di applicazioni. runtime/libraries

Il kernel AL2023 NVIDIA ottimizzato per EKS AMIs supporta il kernel 6.12 per le versioni 1.33 e successive di Kubernetes e la versione del driver NVIDIA 580 per tutte le versioni di Kubernetes. Il driver NVIDIA 580 è necessario per utilizzare CUDA 13+.

Consulta le AL2023 versioni ottimizzate per EKS su GitHub per i dettagli sulle versioni dei componenti incluse in. AMIs Consulta lo script di installazione di EKS AL2023 NVIDIA AMI e lo script di caricamento del kernel per i dettagli su come le AMI EKS configurano le dipendenze NVIDIA. Puoi trovare l'elenco dei pacchetti installati e delle relative versioni su un'istanza EC2 in esecuzione con il comando. `dnf list installed`

Quando si crea una creazione personalizzata AMIs con EKS ottimizzato AMIs come base, non è consigliato o supportato l'esecuzione di un aggiornamento del sistema operativo (ad es. `dnf upgrade`) o aggiorna uno qualsiasi dei pacchetti Kubernetes o GPU inclusi in EKS-Optimized, poiché ciò rischia di compromettere la compatibilità AMIs dei componenti. Se si esegue l'aggiornamento del sistema operativo o dei pacchetti inclusi nei pacchetti ottimizzati per EKS AMIs, si consiglia di eseguire test approfonditi in un ambiente di sviluppo o di gestione temporanea prima di passare alla produzione.

Quando si creano istanze GPU personalizzate AMIs , si consiglia di creare istanze personalizzate separate AMIs per ogni tipo di istanza, generazione e famiglia di istanze che verrà eseguita. La versione accelerata ottimizzata per EKS installa AMIs in modo selettivo driver e pacchetti in fase di esecuzione in base alla generazione e alla famiglia di istanze sottostanti. Per ulteriori informazioni, consulta gli script EKS AMI per l'installazione e il runtime.

EKS Bottlerocket NVIDIA AMIs

Quando si utilizza l'operatore GPU NVIDIA con Bottlerocket NVIDIA ottimizzato per EKS, è necessario disabilitare l'installazione da parte dell'operatore del driver AMIs, del toolkit e del plug-in del dispositivo poiché questi sono già inclusi in EKS. AMIs

Oltre ai componenti AMI EKS standard, il Bottlerocket NVIDIA AMIs ottimizzato per EKS include i seguenti componenti. Le dipendenze minime per EFA (modulo kernel e rdma-core) sono installate in tutte le varianti di Bottlerocket.

- Plugin per dispositivi NVIDIA Kubernetes
- Driver NVIDIA
- Driver in modalità utente NVIDIA CUDA
- Toolkit per container NVIDIA
- NVIDIA Fabric Manager
- NVIDIA ha persistito
- Driver NVIDIA IMEX
- Gestore di sottorete NVIDIA NVLink
- Gestore NVIDIA MIG

Per i dettagli sul driver in modalità utente NVIDIA CUDA e sul CUDA runtime/libraries utilizzato nei contenitori delle applicazioni, consulta la documentazione NVIDIA. La versione CUDA mostrata

`nvidia-smi` è la versione del driver in modalità utente NVIDIA CUDA installato sull'host, che deve essere compatibile con il CUDA utilizzato nei contenitori di applicazioni. runtime/libraries

Consulta le informazioni sulla versione di Bottlerocket nella documentazione di Bottlerocket per i dettagli sui pacchetti installati e sulle relative versioni. Il Bottlerocket NVIDIA ottimizzato per EKS AMIs supporta il kernel 6.12 per le versioni 1.33 e successive di Kubernetes e la versione del driver NVIDIA 580 per le versioni 1.34 e successive di Kubernetes. Il driver NVIDIA 580 è necessario per utilizzare CUDA 13+.

Neuron ottimizzato per EKS AMIs

Per i dettagli su come eseguire carichi di lavoro di addestramento e inferenza utilizzando Neuron con Amazon EKS, consulta i seguenti riferimenti:

- Containers - Kubernetes - Guida introduttiva nella documentazione di Neuron AWS
- Esempio di formazione in AWS Neuron EKS Samples su GitHub
- Implementa carichi di lavoro di inferenza ML con Inferentia su Amazon EKS

Per trovare la versione più recente di Neuron ottimizzata per EKS, consulta e. AMIs the section called "Ottieni le ultime novità IDs" the section called "Ottenimento degli ID della versione più recente"

Quando usi Amazon Elastic Fabric Adaptor (EFA) con un dispositivo ottimizzato per EKS AL2023 o Bottlerocket Neuron AMIs, devi installare il plug-in del dispositivo EFA separatamente. Per ulteriori informazioni, consulta the section called "Configurazione dei cluster di formazione con EFA".

EKS Neuron AL2023 AMIs

I AL2023 Neuron ottimizzati per EKS AMIs non includono il plug-in del dispositivo Neuron Kubernetes o l'estensione di pianificazione Neuron Kubernetes e questi devono essere installati separatamente. Per ulteriori informazioni, consulta the section called "Installa il plug-in del dispositivo Neuron Kubernetes".

Oltre ai componenti AMI EKS standard, il AL2023 Neuron ottimizzato per EKS AMIs include i seguenti componenti.

- Driver Neuron () aws-neuronx-dkms
- Strumenti per neuroni () aws-neuronx-tools
- EFA minimal (modulo kernel e rdma-core)

Vedi lo script di installazione dell'AMI EKS AL2023 Neuron per i dettagli su come EKS AMIs configura le dipendenze Neuron. Consulta le AL2023 versioni ottimizzate per EKS su GitHub per vedere le versioni dei componenti incluse nelle AMI. Puoi trovare l'elenco dei pacchetti installati e delle relative versioni su un'istanza EC2 in esecuzione con il comando. `dnf list installed`

EKS Bottlerocket Neuron AMIs

Le varianti standard di Bottlerocket (aws-k8s) includono le dipendenze Neuron che vengono rilevate e caricate automaticamente durante l'esecuzione su istanze Inferentia o Trainium EC2. AWS

Il Bottlerocket ottimizzato per EKS non include il plug-in del dispositivo Neuron Kubernetes AMIs o l'estensione di pianificazione Neuron Kubernetes e questi devono essere installati separatamente. Per ulteriori informazioni, consulta the section called "Installa il plug-in del dispositivo Neuron Kubernetes".

Oltre ai componenti AMI EKS standard, il Bottlerocket Neuron AMIs ottimizzato per EKS include i seguenti componenti.

- aws-neuronx-dkmsDriver Neuron ()
- EFA minimal (modulo kernel e rdma-core)

Quando si utilizza Bottlerocket ottimizzato per EKS AMIs con istanze Neuron, è necessario configurare quanto segue nei dati utente di Bottlerocket. Questa impostazione consente al contenitore di assumere la proprietà del dispositivo Neuron montato in base ai valori e forniti nelle specifiche del carico di lavoro. `runAsUser` `runAsGroup` Per ulteriori informazioni sul supporto di Neuron in Bottlerocket, consultate il readme di Quickstart on EKS su. GitHub

```
[settings]
[settings.kubernetes]
device-ownership-from-security-context = true
```

Consulta il changelog del kit kernel Bottlerocket per informazioni sulla versione del driver Neuron inclusa nel Bottlerocket ottimizzato per EKS. AMIs

Installa il plug-in del dispositivo Kubernetes per GPUs

I plugin per dispositivi Kubernetes sono stati il meccanismo principale per pubblicizzare infrastrutture specializzate come interfacce di rete e adattatori di rete come GPUs risorse consumabili per i carichi di lavoro Kubernetes. Sebbene la Dynamic Resource Allocation (DRA) sia considerata il

futuro della gestione dei dispositivi in Kubernetes, la maggior parte dei fornitori di infrastrutture specializzati supporta i driver DRA nelle prime fasi. I plugin per dispositivi Kubernetes rimangono oggi un approccio ampiamente disponibile per l'utilizzo nei cluster Kubernetes. GPUs

Considerazioni

- Quando si utilizza lo AL2 023 ottimizzato per EKS con NVIDIA, è necessario installare il plug-in per dispositivi NVIDIA AMIs Kubernetes. GPUs Puoi installare e gestire il plug-in del dispositivo NVIDIA Kubernetes con Helm, gli strumenti Kubernetes che preferisci o l'operatore GPU NVIDIA.
- Quando si utilizza il Bottlerocket ottimizzato per EKS AMIs con NVIDIA GPUs, non è necessario installare il plug-in per dispositivi NVIDIA Kubernetes, poiché è già incluso nel Bottlerocket ottimizzato per EKS. AMIs Ciò include l'utilizzo di istanze GPU con EKS Auto Mode.
- Quando si utilizza lo AL2 023 o Bottlerocket ottimizzato per EKS AMIs con AWS Inferentia o Trainium GPUs, è necessario installare il plug-in del dispositivo Neuron Kubernetes e, facoltativamente, installare l'estensione di pianificazione Neuron Kubernetes. Per ulteriori informazioni, consulta la documentazione di Neuron per l'esecuzione su EKS.

Installa il plug-in per dispositivi NVIDIA Kubernetes

La procedura seguente descrive come installare il plug-in per dispositivi NVIDIA Kubernetes ed eseguire un test di esempio sulle istanze GPU NVIDIA.

Prerequisiti

- Cluster EKS creato
- Nodi GPU NVIDIA in esecuzione nel cluster utilizzando AMI NVIDIA 023 ottimizzata per EKS AL2
- Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.

Procedura

1. Aggiungi l'archivio cartografico Helm. nvdp

```
helm repo add nvdp https://nvidia.github.io/k8s-device-plugin
```

2. Aggiorna il repository locale per assicurarti di avere i grafici più recenti.

```
helm repo update
```

3. Scarica l'ultima versione del plug-in per dispositivi NVIDIA 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. Installa il plug-in per dispositivi NVIDIA Kubernetes sul tuo cluster, sostituendolo `0.17.4` con la versione più recente del comando precedente.

```
helm install nvdp nvdp/nvidia-device-plugin \
  --namespace nvidia \
  --create-namespace \
  --version 0.17.4 \
  --set gfd.enabled=true
```

5. Verifica che il plug-in del dispositivo NVIDIA Kubernetes sia in esecuzione nel cluster. L'output seguente mostra l'output con due nodi nel cluster.

```
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. Verifica che i tuoi nodi siano allocabili GPUs con il seguente comando.

```
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. Crea un file denominato `nvidia-smi.yaml` con i seguenti contenuti. Questo manifesto avvia un'immagine contenitore AL2 023 minima che viene eseguita `nvidia-smi` su un nodo.

```
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. Applicare il file manifesto con il comando seguente.

```
kubectl apply -f nvidia-smi.yaml
```

9. Quando il pod ha terminato l'esecuzione, visualizzane i registri tramite il comando seguente.

```
kubectl logs nvidia-smi
```

Di seguito viene riportato un output di esempio.

```
+-----------------------------------------------------------------------------------------
+
| 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
     |
+---------------------------------------------------------------------------------------
+
```

Installa il plug-in del dispositivo Neuron Kubernetes

La procedura seguente descrive come installare il plug-in del dispositivo Neuron Kubernetes ed eseguire un test di esempio su un'istanza Inferentia.

Prerequisiti

- Cluster EKS creato
- Nodi GPU Neuron in esecuzione nel cluster utilizzando l'AMI AL2 Neuron 023 o l'AMI Bottlerocket ottimizzati per EKS
- Helm è installato nell'ambiente a riga di comando, consulta Setup Helm instructions.

Procedura

1. Installa il plug-in del dispositivo Neuron Kubernetes sul tuo cluster.

```
helm upgrade --install neuron-helm-chart oci://public.ecr.aws/neuron/neuron-helm-
chart \
    --set "npd.enabled=false"
```

2. Verifica che il plug-in del dispositivo Neuron Kubernetes sia in esecuzione nel cluster. L'output seguente mostra l'output con un singolo nodo Neuron nel cluster.

```
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. Verifica che i tuoi nodi siano allocabili NueronCores con il seguente comando.

```
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. Verifica che i tuoi nodi siano allocabili NueronDevices con il seguente comando.

```
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. Crea un file denominato `neuron-ls.yaml` con i seguenti contenuti. Questo manifest avvia un contenitore Neuron Monitor su cui è installato lo strumento. `neuron-ls`

```
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. Applicare il file manifesto con il comando seguente.

```
kubectl apply -f neuron-ls.yaml
```

7. Quando il pod ha terminato l'esecuzione, visualizzane i registri tramite il comando seguente.

```
kubectl logs neuron-ls
```

   Di seguito viene riportato un output di esempio.

```
instance-type: inf2.xlarge
instance-id: ...
+--------+--------+--------+---------+
| NEURON | NEURON | NEURON |   PCI   |
| DEVICE | CORES  | MEMORY |   BDF   |
+--------+--------+--------+---------+
| 0      | 2      | 32 GB  | 00:1f.0 |
+--------+--------+--------+---------+
```

Esegui container accelerati da GPU (Windows su EC2 G-Series)

> **Important**
>
> Il plug-in per dispositivi Kubernetes per DirectX di TensorWorks è uno strumento di terze parti che non è approvato, supportato o gestito da AWS. AWS non si assume alcuna responsabilità per la sicurezza, l'affidabilità o le prestazioni di questo plug-in.

Scopri come eseguire carichi di lavoro di container Windows accelerati da GPU su Amazon EKS (Elastic Kubernetes Service) utilizzando GPU NVIDIA con il plug-in per dispositivi Kubernetes per DirectX di TensorWorks. Per ulteriori informazioni, consulta Kubernetes Device Plugin for DirectX.

Esistono due approcci principali per configurare l'accelerazione GPU per i container Windows:

- Opzione 1: crea un'AMI EKS ottimizzata per Windows personalizzata con i driver GPU richiesti preinstallati.
  - Utilizza questo approccio quando hai bisogno di un ambiente coerente, pre-configurato e pronto per eseguire container Windows accelerati da GPU e quando sei in grado di investire le energie aggiuntive necessarie per creare e mantenere l'AMI personalizzata.
- Opzione 2: installa i driver GPU necessari sui nodi worker EKS dopo aver avviato l'istanza.
  - Utilizza questo approccio quando desideri un processo di configurazione più semplice e puoi installare i driver GPU su ogni nuovo nodo worker. Più adatto a un ambiente di sviluppo per la valutazione o la prototipazione di carichi di lavoro accelerati da GPU.

Entrambi gli approcci possono essere sfruttati utilizzando i passaggi descritti in questa guida.

Considerazioni

Questa guida fornisce i passaggi per installare e configurare l'accelerazione GPU per i container Windows utilizzando GPU NVIDIA, driver NVIDIA GRID e il Kubernetes Device Plugin for DirectX di TensorWorks. I passaggi sono stati testati e verificati per fornire l'accelerazione GPU per i carichi di lavoro dei container Windows su Amazon EKS. Per ulteriori informazioni sui driver e plug-in dei dispositivi compatibili, consulta the section called "Limiti noti". Prima di procedere, tieni presente i seguenti punti:

- Solo i tipi di istanze della famiglia G con NVIDIA GRID drivers sono stati testati e verificati per funzionare con questa guida. Esistono sicuramente altri tipi di istanze e combinazioni di driver in

grado di eseguire container Windows con accelerazione GPU, ma potrebbero richiedere passaggi di configurazione aggiuntivi non descritti in questa guida.

- Solo i carichi di lavoro basati su DirectX sono stati testati e verificati per il funzionamento con questa guida. Esistono anche altre API GPU come OpenGL, Vulkan e OpenCL che sono potenzialmente compatibili per l'esecuzione di container Windows con accelerazione GPU, ma potrebbero richiedere passaggi di configurazione aggiuntivi non trattati in questa guida.
- Esistono alcune limitazioni note di cui tenere conto prima di eseguire container Windows con accelerazione GPU. Consulta la sezione the section called "Limiti noti" per ulteriori informazioni.

Prerequisiti

Per abilitare l'accelerazione GPU per i container Windows su Amazon EKS, devi preparare i seguenti requisiti prima di procedere:

- Avvia un cluster Amazon EKS con Kubernetes v1.27 o successive.
- Esegui il provisioning dei nodi Windows con Windows Server 2022 o versioni successive.
- Esegui il provisioning dei nodi Windows nei tipi di istanze della famiglia G, come G4 o G5.
- Esegui il provisioning dei nodi Windows con un runtime dei container con containerd `1.7.x` o `2.x.x`. (Vedi the section called "Ottenere informazioni sulla versione" per verificare la versione del containerd nella tua AMI ottimizzata di Amazon EKS).

Installa il driver GPU su ogni nodo Windows

Per installare i driver NVIDIA GRID sui nodi worker EKS, segui i passaggi descritti in NVIDIA drivers for your Amazon EC2 instance. Vai a Installation options - Option 3: GRID drivers e segui i passaggi per l'installazione.

Installa per Windows Server Core

Per Windows Server Core, che non offre un'esperienza desktop, installa i driver NVIDIA GRID in modo invisibile utilizzando i seguenti comandi:

```
$nvidiaInstallerFilePath = nvidia-driver-installer.exe # Replace with path to installer
$installerArguments = "-s -clean -noreboot -noeula"
Start-Process -FilePath $nvidiaInstallerFilePath -ArgumentList $installerArguments -
Wait -NoNewWindow -PassThru
```

Verifica dell'installazione

Esegui il seguente comando PowerShell per mostrare le informazioni di diagnostica sulle GPU sull'istanza:

```
nvidia-smi
```

Questo comando mostra la versione del driver NVIDIA e le informazioni sull'hardware della GPU. Assicurati che l'output di questo comando corrisponda alla versione del driver NVIDIA GRID che prevedi di installare.

Implementa il plug-in del dispositivo GPU su ogni nodo

Per consentire il rilevamento e l'esposizione delle risorse della GPU ai container sui nodi Windows, è necessario un plug-in per il dispositivo. Distribuisci il DirectX Device Plugin di Tensorworks su ogni nodo worker eseguendolo come DaemonSet nel tuo cluster EKS. Segui la guida per l'installazione specificata in README.md, che comporterà i seguenti passaggi. Si consiglia di:

- Implementare il plug-in del dispositivo nel namespace `kube-system`.
- Impostare limiti di risorse appropriati per DaemonSet per assicurarsi che non consumi risorse eccessive sui nodi.

> **Note**
>
> Il plug-in del dispositivo DaemonSet verrà eseguito su ogni nodo come container di processi host con privilegi elevati. Si consiglia di implementare i controlli RBAC per limitare l'accesso a questo DaemonSet in modo che solo gli utenti autorizzati possano eseguire comandi privilegiati.

Durante l'esecuzione di container accelerati da GPU, il plug-in del dispositivo supporta due modalità:

- Modalità a tenancy singola: questa modalità dedica tutte le risorse GPU a un singolo container sull'istanza. Installa i plug-in del dispositivo con supporto della tenancy singola utilizzando il comando seguente: Per ulteriori informazioni, consulta il file README.md.

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/default-daemonsets.yml"
```

- Modalità multi-tenancy: questa modalità consente di condividere le risorse GPU tra più container sull'istanza. Installa i plug-in del dispositivo con supporto della multi-tenancy utilizzando il comando seguente: Per ulteriori informazioni, consulta il file README.md.

  ```
  kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-inline.yml"
  ```

  In alternativa, usa una ConfigMap per specificare la multi-tenancy.

  ```
  kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-configmap.yml"
  ```

Verifica dell'implementazione del plug-in del dispositivo

Dopo aver distribuito il plug-in del dispositivo, sostituisci `<namespace>` ed esegui il comando seguente per verificare che il plug-in DirectX funzioni correttamente su tutti i nodi Windows.

```
kubectl get ds device-plugin-wddm -n <namespace>
```

Verifica che i container siano pronti per l'implementazione

Quando il plug-in del dispositivo DaemonSet è in esecuzione sui nodi worker Windows alimentati da GPU, utilizza il comando seguente per verificare che ogni nodo disponga di GPU allocabili. Il numero corrispondente deve corrispondere al numero di dispositivi DirectX su ciascun nodo.

```
kubectl get nodes "-o=custom-columns=NAME:.metadata.name,DirectX:.status.allocatable.directx\.microsoft\.com/display"
```

Esecuzione di container Windows con accelerazione GPU

Prima di avviare i pod, specifica il nome della risorsa `directx.microsoft.com/display` in `.spec.containers[].resources`. Questo indicherà che i container richiedono funzionalità basate su GPU e `kube-scheduler` cercherà di posizionare i pod sul nodo Windows preconfigurato con le risorse GPU disponibili.

Vedi il comando di esempio riportato di seguito che avvia `Job` per eseguire una simulazione Monte Carlo e stimare il valore di pi. Questo esempio proviene dal repository GitHub Kubernetes Device

Plugins for DirectX, che offre diversi esempi da eseguire per testare le funzionalità della GPU dei nodi 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
```

Limiti noti

Tutte le GPU sono utilizzabili

Tutte le GPU dell'istanza saranno utilizzabili da ogni container in esecuzione sull'host, anche quando si richiede un numero specifico di GPU per un determinato container. Inoltre, il comportamento predefinito prevede che tutti i container in esecuzione sull'host utilizzino la GPU con indice 0, anche se sul nodo sono disponibili più GPU. Pertanto, affinché le attività con più GPU funzionino correttamente, è necessario designare esplicitamente il dispositivo GPU specifico da utilizzare all'interno del codice dell'applicazione.

L'implementazione esatta per allocare un dispositivo da utilizzare per l'applicazione dipenderà dal linguaggio di programmazione o dal framework in uso. Ad esempio, se utilizzi la programmazione CUDA, per selezionare una GPU specifica, puoi specificare esplicitamente il dispositivo da utilizzare nel codice dell'applicazione utilizzando la funzione cudaSetDevice().

La necessità di specificare in modo esplicito il dispositivo è dovuta a un problema noto che interessa i container Windows. È possibile tenere traccia dei progressi nella risoluzione di questo problema nel

microsoft/Windows-Containers issue #333. La tabella seguente è una rappresentazione visiva e un esempio pratico di questo comportamento di allocazione delle GPU.

Consideriamo uno scenario in cui esiste un singolo nodo Windows del tipo di istanza EC2 `g4dn.12xlarge`, dotato di quattro GPU. Consideriamo uno scenario in cui vengono lanciati tre pod su questa istanza. La tabella mostra che, indipendentemente dal numero di GPU richieste da ciascun container, tutti e tre i pod hanno accesso a tutte e quattro le GPU dell'istanza e, per impostazione predefinita, utilizzeranno la GPU con indice di dispositivo 0.

Pod	GPU richieste	Accesso effettivo alla GPU	Utilizzo predefinito della GPU	Indici disponibili della GPU	Istanze totali della GPU
Pod 1	1 GPU	Tutte e 4 le GPU	GPU con indice 0	0, 1, 2, 3	4
Pod 2	2 GPU	Tutte e 4 le GPU	GPU con indice 0	0, 1, 2, 3	4
Pod 3	1 GPU	Tutte e 4 le GPU	GPU con indice 0	0, 1, 2, 3	4

Supporto per il plug-in dei dispositivi Kubernetes

L'implementazione ufficiale di NVIDIA del Kubernetes device plugin non supporta Windows. Puoi tenere traccia dei progressi nell'aggiunta del supporto ufficiale per Windows in NVIDIA/k8s-device-plugin issue #419.

Limitazioni delle istanze di calcolo GPU

A seconda della configurazione del tuo account AWS, potresti avere dei limiti di servizio sul numero e sui tipi di istanze di calcolo GPU Amazon EC2 che puoi avviare. Se hai bisogno di capacità aggiuntiva, consulta Request a quota increase.

È necessario creare un'AMI ottimizzata per GPU Windows

Amazon EKS non fornisce alcuna AMI EKS ottimizzata per GPU Windows né un componente gestito da EC2 Image Builder. È necessario seguire i passaggi di questa guida per creare un'AMI EKS

ottimizzata per Windows e personalizzata con i driver GPU richiesti preinstallati o installare i driver GPU necessari sui nodi worker EKS dopo l'avvio delle istanze.

Inferentia e Trainium non sono supportati

I carichi di lavoro basati su AWS Inferentia e AWS Trainium non sono supportati su Windows.

Esecuzione dei corsi di machine learning su Amazon EKS con Elastic Fabric Adapter

In questo argomento viene descritto come integrare Elastic Fabric Adapter (EFA) con i pod implementati nel cluster Amazon EKS. Elastic Fabric Adapter (EFA) è un'interfaccia di rete per le istanze Amazon EC2 che consente di eseguire applicazioni che richiedono livelli elevati di comunicazioni internodale su larga scala in AWS. La sua interfaccia hardware di bypass del sistema operativo personalizzata migliora le prestazioni delle comunicazioni tra istanze, che è fondamentale per dimensionare queste applicazioni. Con EFA, le applicazioni High Performance Computing (HPC) che utilizzano le applicazioni Message Passing Interface (MPI) e Machine Learning (ML) che utilizzano NVIDIA Collective Communications Library (NCCL) possono scalare fino a migliaia di o. CPUs GPUs Di conseguenza, si ottengono le prestazioni applicative dei cluster HPC locali con l'elasticità e la flessibilità on-demand del cloud. AWS L'integrazione di EFA con le applicazioni in esecuzione su cluster Amazon EKS può ridurre il tempo necessario per completare carichi di lavoro di formazione distribuiti su larga scala senza dover aggiungere ulteriori istanze al cluster. Per ulteriori informazioni su EFA, consulta Elastic Fabric Adapter.

Tipi di istanze con EFA

L' AWS EFA Kubernetes Device Plugin supporta tutti i tipi di istanze Amazon EC2 con EFA. Per visualizzare un elenco di tutti i tipi di istanze con EFA, consulta Tipi di istanze supportati nella Guida per l'utente di Amazon EC2. Tuttavia, per eseguire rapidamente le applicazioni ML, consigliamo che un'istanza disponga di chip di accelerazione hardware come GPUs NVidia AWS , Inferentia AWS o chip Trainium, oltre all'EFA. Per visualizzare un elenco di tipi di istanze con chip di accelerazione hardware ed EFA, consulta Calcolo accelerato nella Guida per l'utente di Amazon EC2.

Nel confrontare i tipi di istanza e scegliere tra di essi, considera il numero di schede di rete EFA disponibili per quel tipo di istanza, nonché il numero di schede di accelerazione, la quantità di CPU e la quantità di memoria. È possibile assegnare fino a un EFA per scheda di rete. Un EFA conta come un'interfaccia di rete. Per vedere quanti EFA sono disponibili per ciascun tipo di istanza che dispone di EFA, consulta l'elenco delle Schede di rete nella Guida per l'utente di Amazon EC2.

Interfacce EFA e solo EFA

Un Elastic Fabric Adapter (EFA) è un'interfaccia di rete che combina le funzionalità di un Elastic Network Adapter (ENA) e un'interfaccia OS-Bypass, basata sul protocollo AWS Scalable Reliable Datagram (SRD). Le funzionalità EFA consentono alle applicazioni di comunicare direttamente con l'hardware per un trasporto a bassa latenza. È possibile scegliere di accedere solo alle funzionalità EFA utilizzando interfacce solo EFA, limitando la comunicazione alle interfacce all'interno della stessa zona di disponibilità.

Per creare nodi che possono avere interfacce solo EFA, devi utilizzare un modello di lancio EC2 personalizzato e impostare `InterfaceType` su `efa-only`. Nel tuo modello di lancio personalizzato, non puoi impostare la scheda di rete `0` su un'interfaccia solo EFA, poiché si tratta della scheda di rete e dell'interfaccia di rete principali dell'istanza EC2. È necessario disporre della versione CNI di VPC `1.18.5` o successiva per le interfacce solo EFA. Se utilizzi Amazon Linux 2, la versione ami deve essere `v20240928` o successiva per le interfacce solo EFA.

La procedura seguente guida alla creazione di un cluster EKS con nodi `eksctl` dotati di interfacce NVidia GPUs ed EFA. Non è possibile utilizzare `eksctl` per creare nodi e gruppi di nodi che utilizzano interfacce solo EFA.

Prerequisiti

- Un cluster Amazon EKS esistente. Qualora non fosse disponibile un cluster esistente, potrai crearne uno utilizzando Nozioni di base. Il cluster deve essere implementato in un VPC con almeno una sottorete privata con indirizzi IP disponibili sufficienti in cui implementare i nodi. La sottorete privata deve disporre di un accesso Internet in uscita fornito da un dispositivo esterno, ad esempio un gateway NAT.

  Se prevedi di utilizzare `eksctl` per creare il gruppo di nodi, `eksctl` può anche creare un cluster per tuo conto.
- Versione `2.12.3` o successiva o versione `1.27.160` o successiva dell'interfaccia a riga di AWS comando (AWS CLI) installata e configurata sul dispositivo o. AWS CloudShell Per verificare la versione attuale, usa `aws --version | cut -d / -f2 | cut -d ' ' -f1`. I gestori di pacchetti come `yum` Homebrew per macOS sono spesso diverse versioni dell'ultima versione della CLI AWS . `apt-get` Per installare la versione più recente, consulta Installazione e configurazione rapida con aws configure nella Guida per l'utente dell'interfaccia a riga di AWS comando. La versione AWS CLI installata in AWS CloudShell potrebbe anche contenere diverse versioni

precedenti alla versione più recente. Per aggiornarlo, consulta Installazione della AWS CLI nella tua home directory nella Guida per l' AWS CloudShell utente.

- Lo strumento a riga di comando `kubectl` è installato sul dispositivo o AWS CloudShell. La versione può essere uguale oppure immediatamente precedente o successiva alla versione di Kubernetes del cluster. Ad esempio, se la versione del cluster è `1.29`, puoi usare `kubectl` versione `1.28`, `1.29` o `1.30`. Per installare o aggiornare `kubectl`, consulta the section called "Configura kubectl ed eksctl":
- Prima di avviare i nodi worker che supportano più di un Elastic Fabric Adapter, come ad esempio `p4d` o `p5`, è necessario aver installato la versione `1.7.10` o successiva del plugin CNI di Amazon VPC per Kubernetes. Per ulteriori informazioni sull'aggiornamento della versione del plugin CNI di Amazon VPC per Kubernetes, consulta the section called "CNI di Amazon VPC".
- Per le istanze p6-b200, è necessario utilizzare la versione del plugin del dispositivo EFA v0.5.6 o successiva.

> **Important**
>
> Un'importante considerazione necessaria per l'adozione di EFA con Kubernetes è la configurazione e la gestione di Huge Pages come risorsa nel cluster. Per ulteriori informazioni, consultare Gestione di Huge Pages nella documentazione Kubernetes. Le istanze Amazon EC2 con il driver EFA installato assegnano in via anticipata 5.128 Huge Pages da 2 MiB, che possono essere richieste come risorse da utilizzare nelle specifiche del processo.

Creazione di un gruppo di nodi

La procedura seguente consente di creare un gruppo di nodi con un gruppo di nodi `p4d.24xlarge` supportato con interfacce EFA e GPUDirect RDMA ed eseguire un esempio di test della NVIDIA Collective Communications Library (NCCL) per l'utilizzo di NCCL a più nodi. EFAs L'esempio può essere utilizzato come modello per la formazione distribuita di deep learning su Amazon EKS utilizzando EFAs.

1. Determina quali tipi di istanze Amazon EC2 che supportano EFA sono disponibili nella AWS regione in cui desideri distribuire i nodi. Sostituiscilo *`region-code`* con la AWS regione in cui desideri distribuire il gruppo di nodi.

```
aws ec2 describe-instance-types --region region-code \
    --filters Name=network-info.efa-supported,Values=true \
    --query "InstanceTypes[*].[InstanceType]" --output text
```

Quando distribuisci i nodi, il tipo di istanza che desideri distribuire deve essere disponibile nella AWS regione in cui si trova il cluster.

2. Determina in quali zone di disponibilità è disponibile il tipo di istanza che desideri implementare. In questo tutorial, viene utilizzato il tipo di `p5.48xlarge` istanza che deve essere restituito nell'output per la AWS regione specificata nel passaggio precedente. Quando distribuisci i nodi in un cluster di produzione, sostituiscili *`p5.48xlarge`* con qualsiasi tipo di istanza restituito nel passaggio precedente.

```
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
```

Di seguito viene riportato un output di esempio.

```
us-west-2a    us-west-2c    us-west-2b
```

Prendi nota delle zone di disponibilità restituite per l'uso nelle fasi successive. Quando implementi i nodi in un cluster, il tuo VPC deve disporre di sottoreti con indirizzi IP disponibili in una delle zone di disponibilità restituite nell'output.

3. Crea un gruppo di nodi utilizzando `eksctl`. È necessaria la versione `0.215.0` o una versione successiva dello strumento da riga di `eksctl` comando installata sul dispositivo o AWS CloudShell. Per l'installazione o l'aggiornamento di `eksctl`, consulta la sezione Installation nella documentazione di `eksctl`.

   a. Copiare i contenuti seguenti in un file denominato *`efa-cluster.yaml`*. Sostituire i valori di esempio con i propri valori. Puoi sostituire `p5.48xlarge` con un'istanza diversa, ma in questo caso assicurati che i valori per `availabilityZones` corrispondano a zone di disponibilità restituite per il tipo di istanza nel passaggio 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. Crea un gruppo di nodi gestito in un cluster esistente.

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

Se non disponi di un cluster esistente, puoi eseguire il comando seguente per creare un cluster e il gruppo di nodi.

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

> **Note**
>
> Poiché il tipo di istanza utilizzato in questo esempio installa `eksctl` automaticamente il plug-in del dispositivo NVIDIA Kubernetes su ogni istanza per te quando usi Amazon Linux 2. GPUs Ciò non è necessario per Bottlerocket, poiché il plugin del dispositivo NVIDIA è integrato nella variante NVIDIA EKS di Bottlerocket. Quando `efaEnabled` è impostato su `true` nella configurazione del gruppo di nodi, `eksctl` distribuirà automaticamente anche il plugin del dispositivo EFA sui nodi.

Utilizzo di Bottlerocket con EFA

La versione AMI di Bottlerocket 1.28.0 e successive includono il supporto ufficiale per EFA. Per utilizzare Bottlerocket per nodi abilitati a EFA, specifica `amiFamily: Bottlerocket` nella configurazione. Se è necessario utilizzare un ID AMI personalizzato, è necessario utilizzare `nodeGroups` standard anziché `managedNodeGroups`.

Ecco un esempio di configurazione:

```
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
```

L'impostazione sysctl `vm.nr_hugepages` riportata sopra configura il numero di Huge Pages di 2 Mi. In questo esempio, 3000 significa 3000 * 2 Mi = 6000 Mi di Huge Pages.

Verifica l'installazione del plugin del dispositivo EFA

Quando crei un gruppo di nodi con `efaEnabled: true`, `eksctl` distribuisce automaticamente il plugin per dispositivi EFA Kubernetes per te. Puoi verificare che il plugin del dispositivo sia installato e funzioni correttamente:

1. Controlla lo stato: DaemonSet

   ```
   kubectl get daemonsets -n kube-system
   ```

   Output di esempio:

   ```
   NAME                                  DESIRED   CURRENT   READY   UP-TO-DATE
    AVAILABLE   NODE SELECTOR   AGE
   aws-efa-k8s-device-plugin-daemonset   2         2         2       2            2
         <none>          6m16s
   ...
   ```

   Qui, il plug-in del dispositivo EFA DaemonSet è in esecuzione su due nodi. Entrambi sono PRONTO e DISPONIBILE.

2. Successivamente, verifica i pod creati da: DaemonSet

   ```
   kubectl get pods -n kube-system -l name=aws-efa-k8s-device-plugin
   ```

   Output di esempio:

   ```
   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
   ```

   I pod dei plugin del dispositivo EFA sono nello stato In esecuzione, a conferma che il plugin è stato distribuito correttamente ed è operativo.

3. Verifica la registrazione delle risorse:

   Puoi confermare che la risorsa `vpc.amazonaws.com/efa` è registrata con il kubelet descrivendo i nodi:

   ```
   kubectl describe nodes
   ```

se la risorsa EFA è registrata correttamente, la vedrai elencata nelle risorse di capacità e assegnabilità del nodo. Esempio:

```
Capacity:
  ...
  vpc.amazonaws.com/efa:  4
Allocatable:
  ...
  vpc.amazonaws.com/efa:  4
```

questo output conferma che il nodo riconosce la risorsa EFA, rendendola disponibile per i pod che la richiedono.

(Facoltativo) Test delle prestazioni dell'EFA

Consigliamo di testare la configurazione EFA. È possibile utilizzare i test NCCL presenti nel repository su. `aws-samples/awsome-distributed-training` GitHub I test NCCL valutano le prestazioni della rete utilizzando la libreria NVIDIA per comunicazioni collettive. La procedura seguente prevede l'invio dei test NCCL su Amazon EKS.

1. Implementa l'operatore MPI Kubeflow:

   Per i test NCCL è possibile applicare l'operatore Kubeflow MPI. L'operatore MPI semplifica l'esecuzione della formazione distribuita in stile Allreduce su Kubernetes. Per ulteriori informazioni, vedere MPI Operator on. GitHub

2. Esegui il test delle prestazioni NCCL multinodo per verificare RDMA/EFA: GPUDirect

   Per verificare le prestazioni NCCL con GPUDirect RDMA su EFA, esegui il test delle prestazioni NCCL standard. Per ulteriori informazioni, consulta il repository ufficiale di NCCL-Tests su. GitHub

   Completa i seguenti passaggi per eseguire un test delle prestazioni NCCL a due nodi. Nell'esempio del processo di test NCCL, ogni lavoratore richiede otto, 5210 Mi di GPUs, quattro EFAs e 8000 Mi di memoria`hugepages-2Mi`, il che significa effettivamente che ogni lavoratore consuma tutte le risorse di un'istanza. `p5.48xlarge`

   a. Crea il manifesto: MPIJob

      Copia quanto riportato di seguito in un file denominato `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
            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=8388608
            - -x
            - NCCL_P2P_NET_CHUNKSIZE=524288
            - -x
```

```
            - NCCL_TUNER_PLUGIN=/opt/amazon/ofi-nccl/lib/x86_64-linux-gnu/libnccl-
ofi-tuner.so
            - --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. Applica i test NCCL: MPIJob

   Invia `MPIJob` applicando il manifesto. Ciò creerà due istanze Amazon EC2 `p5.48xlarge`.

```
kubectl apply -f nccl-tests.yaml
```

   Di seguito viene riportato un output di esempio.

```
mpijob.kubeflow.org/nccl-tests created
```

c. Verifica che il processo abbia avviato i pod:

   Visualizza i pod in esecuzione.

```
kubectl get pods
```

   Di seguito viene riportato un output di esempio.

```
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
```

   L'operatore MPI crea un pod di avvio e 2 pod worker (uno su ciascun nodo).

d. Verifica che il processo funzioni correttamente con i log:

   Visualizza il log per il pod `nccl-tests-launcher`. Sostituisci *`nbql9`* con il valore dell'output.

```
kubectl logs -f nccl-tests-launcher-nbql9
```

Se il test è stato completato con successo, puoi implementare le applicazioni che utilizzano la libreria NVIDIA per comunicazioni collettive.

Usa le istanze AWS Inferentia con Amazon EKS for Machine Learning

Questo argomento descrive come creare un cluster Amazon EKS con nodi che eseguono istanze Amazon EC2 Inf1 e (facoltativamente) distribuire un'applicazione di esempio. Le istanze Amazon EC2 Inf1 sono alimentate da chip AWS Inferentia, creati su misura AWS per fornire inferenze ad alte prestazioni e al minor costo nel cloud. I modelli di apprendimento automatico vengono distribuiti nei contenitori utilizzando AWS Neuron, un kit di sviluppo software (SDK) specializzato che comprende un compilatore, un runtime e strumenti di profilazione che ottimizzano le prestazioni di inferenza dell'apprendimento automatico dei chip Inferentia. AWS Neuron supporta i più diffusi framework di apprendimento automatico come, e. TensorFlow PyTorch MXNet

> **Note**
>
> La logica IDs del dispositivo Neuron deve essere contiguo. Se un Pod che richiede più dispositivi Neuron è pianificato su un tipo di `inf1.24xlarge` istanza `inf1.6xlarge` o (che ha più di un dispositivo Neuron), quel Pod non si avvierà se lo scheduler Kubernetes seleziona un dispositivo non contiguo. IDs Per ulteriori informazioni, vedi Device logical must be contiguous on. IDs GitHub

Prerequisiti

- Avere `eksctl` installato sul computer. Se non è installato, consulta Installation nella documentazione di `eksctl`.
- Avere `kubectl` installato sul computer. Per ulteriori informazioni, consulta the section called "Configura kubectl ed eksctl".
- (Facoltativo) Avere `python3` installato sul computer. Se non lo hai installato, consultare Python downloads per le istruzioni di installazione.

Creazione di un cluster

1. Crea un cluster con nodi di EC2 istanze Amazon Inf1. Puoi sostituire *`inf1.2xlarge`* con qualsiasi tipo di istanza Inf1. L'`eksctl`utilità rileva che stai avviando un gruppo di nodi con un

tipo di `Inf1` istanza e avvierà i nodi utilizzando uno degli Amazon Linux accelerati ottimizzati per Amazon EKS. AMIs

> **Note**
>
> Non puoi utilizzare i ruoli IAM per gli account di servizio con Serving. TensorFlow

```
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**
>
> Notare il valore della seguente riga dell'output. Viene utilizzato in un passaggio successivo (facoltativo).

```
[9]  adding identity "arn:aws: iam::111122223333:role/eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09" to auth ConfigMap
```

Quando si avvia un gruppo di nodi con `Inf1` istanze, installa `eksctl` automaticamente il plug-in del dispositivo AWS Neuron Kubernetes. Questo plugin pubblicizza i dispositivi Neuron come risorsa di sistema allo scheduler Kubernetes, che può essere richiesto da un container. Oltre alle policy IAM dei nodi Amazon EKS di default, viene aggiunta la policy di accesso di sola lettura Amazon S3 in modo che l'applicazione di esempio, che verrà trattata in un passaggio successivo, possa caricare un modello addestrato da Amazon S3.

2. Assicurarsi che tutti i pod siano stati avviati correttamente.

```
kubectl get pods -n kube-system
```

Output abbreviato:

```
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
```

(Facoltativo) Implementa un'immagine dell'applicazione Serving TensorFlow

Un modello addestrato deve essere compilato in un target Inferentia prima di poter essere implementato nelle istanze Inferentia. Per continuare, avrai bisogno di un TensorFlow modello ottimizzato per Neuron salvato in Amazon S3. Se non ne hai già uno SavedModel, segui il tutorial per creare un modello ResNet 50 compatibile con Neuron e carica il risultato SavedModel su S3. ResNet-50 è un popolare modello di apprendimento automatico utilizzato per attività di riconoscimento delle immagini. Per ulteriori informazioni sulla compilazione dei modelli Neuron, consulta The AWS Inferentia Chip With DLAMI nella Deep Learning Developer Guide. AWS AMIs

Il manifesto di distribuzione di esempio gestisce un contenitore di servizi di inferenza precostruito TensorFlow fornito da AWS Deep Learning Containers. All'interno del contenitore si trovano AWS Neuron Runtime e l' TensorFlow applicazione Serving. Un elenco completo di Deep Learning Containers predefiniti ottimizzati per Neuron è disponibile nella GitHub sezione Immagini disponibili. All'avvio, il DLC recupererà il tuo modello da Amazon S3, avvierà Neuron TensorFlow Serving con il modello salvato e aspetterà le richieste di previsione.

Il numero di dispositivi Neuron assegnati all'applicazione di servizio può essere regolato modificando la risorsa `aws.amazon.com/neuron` nella implementazione yaml. Tieni presente che la comunicazione tra TensorFlow Serving e il runtime Neuron avviene tramite GRPC, il che richiede il trasferimento della funzionalità al contenitore. `IPC_LOCK`

1. Aggiungere la policy `AmazonS3ReadOnlyAccess` IAM al ruolo di istanza del nodo creato nel passaggio 1 di Crea un cluster. Ciò è necessario affinché l'applicazione di esempio possa caricare un modello formato da 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. Crea un file denominato `rn50_deployment.yaml` con i seguenti contenuti. Aggiornare il codice di Regione e il percorso del modello in modo che corrispondano alle impostazioni desiderate. Il nome del modello serve a scopo di identificazione quando un client effettua una richiesta al TensorFlow server. Questo esempio utilizza un nome di modello che corrisponde a uno script client di esempio di ResNet 50 client che verrà utilizzato in un passaggio successivo per l'invio di richieste di previsione.

```
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. Implementare il modello.

```
kubectl apply -f rn50_deployment.yaml
```

4. Crea un file denominato `rn50_service.yaml` con i seguenti contenuti. Le porte HTTP e gRPC vengono aperte per accettare le richieste di previsione.

```
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. Crea un servizio Kubernetes per la tua TensorFlow applicazione Model Serving.

```
kubectl apply -f rn50_service.yaml
```

(Facoltativo) Fai previsioni sul tuo servizio Serving TensorFlow

1. Per testare localmente, inoltrare la porta gRPC al servizio `eks-neuron-test`.

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. Creare uno script Python chiamato `tensorflow-model-server-infer.py` con il seguente contenuto. Questo script esegue l'inferenza tramite gRPC, che è un framework di servizi.

```
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. Eseguire lo script per inviare previsioni al servizio.

```
python3 tensorflow-model-server-infer.py
```

Di seguito viene riportato un output di esempio:

```
[[(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)]]
```

Gestisci le risorse di calcolo per i AI/ML carichi di lavoro su Amazon EKS

Questa sezione è progettata per aiutarti a gestire le risorse di calcolo per i carichi di lavoro di machine learning in Amazon Elastic Kubernetes Service (EKS). Troverai i dettagli sulla prenotazione GPUs utilizzando Capacity Blocks per gruppi di nodi gestiti e nodi autogestiti, tra cui prerequisiti, configurazione del modello di lancio, configurazioni di scalabilità, preparazione del carico di lavoro e considerazioni chiave per la gestione dei cicli di vita delle prenotazioni e la chiusura graduale dei nodi.

Argomenti

- Creare un gruppo di nodi gestito con blocchi di capacità per il ML
- Crea nodi autogestiti con i blocchi di capacità per il ML
- Usa GB2 P6e-00 con UltraServers Amazon EKS

Creare un gruppo di nodi gestito con blocchi di capacità per il ML

I blocchi di capacità per il machine learning (ML) ti consentono di riservare istanze GPU in una data futura per supportare i carichi di lavoro ML di breve durata. Per ulteriori informazioni, consulta Blocchi di capacità per il machine learning nella Guida per l'utente di Amazon EC2 per le istanze Linux.

Considerazioni

Important

- I blocchi di capacità sono disponibili solo per certi tipi di istanze e regioni AWS Amazon EC2. Per informazioni relative alla compatibilità, consulta Work with Capacity Blocks Prerequisites nella Guida per l'utente di Amazon EC2 per le istanze Linux.
- Per ulteriori informazioni, consulta Usa i blocchi di capacità per carichi di lavoro di machine learning nella Guida per l'utente di Amazon EC2 Auto Scaling.
- I gruppi di nodi gestiti con i blocchi di capacità possono essere creati solo con modelli di avvio personalizzati.
- Quando aggiorni i gruppi di nodi gestiti con i blocchi di capacità, assicurati che la dimensione desiderata del gruppo di nodi sia impostata su `0`.

Creare un gruppo di nodi gestito con blocchi di capacità di Amazon EC2

Puoi usare i blocchi di capacità con i gruppi di nodi gestiti da Amazon EKS per il provisioning e il dimensionamento dei nodi di lavoro accelerati da GPU. Gli esempi di modelli AWS CloudFormation riportati di seguito non coprono tutti gli aspetti necessari in un cluster di produzione. In genere, serve anche uno script di bootstrap per unire il nodo al cluster e specificare un'AMI accelerata Amazon EKS. Per ulteriori informazioni, consulta the section called "Crea".

1. Crea un modello di avvio adatto ai tuoi carichi di lavoro e che funzioni con i gruppi di nodi gestiti da Amazon EKS. Per ulteriori informazioni, consulta the section called "Modelli di avvio".

   Oltre ai requisiti indicati nelle procedure sopra riportate, assicurati che `LaunchTemplateData` includa quanto segue:

   - `InstanceMarketOptions` con `MarketType` impostato su `"capacity-block"`

- `CapacityReservationSpecification: CapacityReservationTarget` con `CapacityReservationId` impostato sul blocco di capacità (ad esempio: `cr-`*`02168da1478b509e0`*)
- `InstanceType` impostato su un tipo di istanza che supporta i blocchi di capacità (ad esempio: *`p5.48xlarge`*)

  Di seguito è riportato un estratto di un modello CloudFormation che crea un modello di avvio destinato a un blocco di capacità. Per creare un gruppo di nodi gestiti AMI personalizzato, è anche possibile aggiungere i parametri `ImageId` e `UserData`.

```
NodeLaunchTemplate:
  Type: "AWS::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      InstanceType: p5.48xlarge
```

2. Utilizza il modello di avvio per creare un gruppo di nodi gestito.

   Di seguito è riportato un esempio di comando per la creazione di un gruppo di nodi per i blocchi di capacità. Sostituisci i *`valori di esempio`* con quelli applicabili al cluster.

   Quando crei il gruppo di nodi gestiti del blocco di capacità, procedi come segue:

   - Imposta `capacity-type` su `"CAPACITY_BLOCK"`. Se il tipo di capacità non è impostato su `"CAPACITY_BLOCK"` se manca uno qualsiasi degli altri valori richiesti sopra indicati per il modello di avvio, la richiesta di creazione verrà rifiutata.
   - Quando specifichi `subnets` nella richiesta di creazione, assicurati di specificare solo la sottorete nella stessa zona di disponibilità della prenotazione della capacità.
   - Se specifichi un valore diverso da zero per `desiredSize` nella richiesta di creazione, Amazon EKS lo rispetterà durante la creazione del gruppo Auto Scaling (ASG). Tuttavia, se la richiesta di creazione viene effettuata prima che la prenotazione della capacità sia attiva, l'ASG non sarà in grado di avviare istanze Amazon EC2 fino a quando non diventa attiva. Di conseguenza, le attività di dimensionamento dell'ASG presenteranno errori di avvio. Quando la prenotazione

diventa attiva, l'avvio delle istanze avrà esito positivo e l'ASG aumenterà verticalmente fino alla `desiredSize` specificata al momento della creazione.

```
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. Assicurati che i nodi si uniscano dopo l'aumento verticale. I cluster Amazon EKS che usano gruppi di nodi gestiti con blocchi di capacità non eseguono alcuna verifica per accertarsi che le istanze avviate si uniscano effettivamente al cluster e vi si registrino.
4. Se imposti `desiredSize` su `0` al momento della creazione, sono disponibili diverse opzioni per aumentare verticalmente il gruppo di nodi quando la prenotazione della capacità diventa attiva:
   - Crea una policy di dimensionamento programmato per l'ASG che si allinei all'ora di inizio della prenotazione dei blocchi di capacità. Per ulteriori informazioni, consulta Dimensionamento programmato per Amazon EC2 Auto Scaling nella Guida per l'utente di Amazon EC2 Auto Scaling.
   - Utilizza la console Amazon EKS o `eks update-nodegroup-config` per aggiornare la configurazione del dimensionamento e impostare la dimensione desiderata del gruppo di nodi.
   - Utilizza l'autoscaler del cluster Kubernetes. Per ulteriori informazioni consultare Cluster Autoscaler su AWS.
5. Il gruppo di nodi è ora pronto per la pianificazione di carichi di lavoro e pod.
6. Affinché i tuoi Pod vengano svuotati senza problemi prima della fine della prenotazione, Amazon EKS usa una policy di dimensionamento pianificata per ridurre verticalmente la dimensione del gruppo di nodi a `0`. Questo dimensionamento programmato verrà impostato con il nome `Amazon EKS Node Group Capacity Scaledown Before Reservation End` . Suggeriamo di non modificare né eliminare questa azione.

   Amazon EC2 inizia a spegnere le istanze 30 minuti prima della fine della prenotazione. Di conseguenza, Amazon EKS imposterà una riduzione verticale programmata del gruppo di nodi 40 minuti prima della fine della prenotazione, al fine di espellere i pod in modo sicuro e graduale.

Crea nodi autogestiti con i blocchi di capacità per il ML

I blocchi di capacità per il machine learning (ML) ti consentono di riservare istanze GPU in una data futura per supportare i carichi di lavoro ML di breve durata. Per ulteriori informazioni, consulta Blocchi di capacità per il machine learning nella Guida per l'utente di Amazon EC2 per le istanze Linux.

Considerazioni

Important

- I blocchi di capacità sono disponibili solo per certi tipi di istanze e regioni AWS Amazon EC2. Per informazioni relative alla compatibilità, consulta Work with Capacity Blocks Prerequisites nella Guida per l'utente di Amazon EC2 per le istanze Linux.
- Se crei un gruppo di nodi autogestito prima che la prenotazione della capacità diventi attiva, imposta la capacità desiderata su `0`.
- Per avere tempo sufficiente per svuotare correttamente i nodi, suggeriamo di pianificare il dimensionamento fino a zero più di 30 minuti prima dell'orario di fine della prenotazione del blocco di capacità.
- Affinché i tuoi pod vengano svuotati correttamente, consigliamo di configurare AWS Node Termination Handler, come spiegato nei passaggi di esempio.

Utilizza i blocchi di capacità con nodi autogestiti

Puoi utilizzare i blocchi di capacità con Amazon EKS per effettuare il provisioning e il dimensionamento dei nodi autogestiti. Le seguenti fasi forniscono una panoramica generale di esempio. Gli esempi di modelli AWS CloudFormation non coprono tutti gli aspetti necessari in un carico di lavoro di produzione. In genere è anche necessario uno script di bootstrap per unire il nodo al cluster, specificare un'AMI accelerata Amazon EKS e un profilo dell'istanza adeguato per l'aggiunta al cluster. Per ulteriori informazioni, consulta the section called "Amazon Linux".

1. Crea un modello di lancio applicabile al tuo carico di lavoro. Per ulteriori informazioni, consulta Usa i blocchi di capacità per carichi di lavoro di machine learning nella Guida per l'utente di Amazon EC2 Auto Scaling.

   Assicurati che `LaunchTemplateData` includa quanto segue:

   - `InstanceMarketOptions` con `MarketType` impostato su `"capacity-block"`

- `CapacityReservationSpecification: CapacityReservationTarget` con `CapacityReservationId` impostato sul blocco di capacità (ad esempio: `cr-`*`02168da1478b509e0`*)
- `IamInstanceProfile` con l'`Arn` impostato sul valore *`iam-instance-profile-arn`* applicabile
- `ImageId` impostato sull'*`image-id`* applicabile
- `InstanceType` impostato su un tipo di istanza che supporta i blocchi di capacità (ad esempio: *`p5.48xlarge`*)
- `SecurityGroupIds` impostato sugli ID applicabili (ad esempio: *`sg-05b1d815d1EXAMPLE`*)
- `UserData` impostato *`sui dati utente`* applicabili per il gruppo di nodi autogestito

  Di seguito è riportato un estratto di un modello CloudFormation che crea un modello di avvio destinato a un blocco di capacità.

```
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
```

  Devi trasmettere la sottorete nella zona di disponibilità in cui viene effettuata la prenotazione, perché i blocchi di capacità sono zonali.

2. Utilizza il modello di avvio per creare un gruppo di nodi autogestito. Se lo fai prima che la prenotazione della capacità diventi attiva, imposta la capacità desiderata su `0`. Quando crei il gruppo di nodi, assicurati di specificare solo la rispettiva sottorete per la zona di disponibilità in cui è la capacità è riservata.

Di seguito è riportato un esempio di modello CloudFormation che puoi usare come riferimento per crearne uno applicabile al tuo carico di lavoro. Questo esempio ottiene i valori di `LaunchTemplateId` e `Version` della risorsa `AWS::Amazon EC2::LaunchTemplate` mostrata nel passaggio precedente. Ottiene anche i valori di `DesiredCapacity`, `MaxSize`, `MinSize` e `VPCZoneIdentifier` che sono dichiarati altrove nello stesso modello.

```
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. Una volta che il gruppo di nodi è stato creato, assicurati di registrare il valore `NodeInstanceRole` per il gruppo di nodi creato. Ciò è necessario per verificare che, in seguito al dimensionamento del gruppo di nodi, i nuovi nodi vengano aggiunti al cluster e Kubernetes sia in grado di riconoscerli. Per ulteriori informazioni, consulta le istruzioni Console di gestione AWS in Create self-managed Amazon Linux nodes.
4. Consigliamo di creare una policy di dimensionamento programmato per il gruppo con dimensionamento automatico che si allinei ai tempi di prenotazione dei blocchi di capacità. Per ulteriori informazioni, consulta Dimensionamento programmato per Amazon EC2 Auto Scaling nella Guida per l'utente di Amazon EC2 Auto Scaling.

   Puoi utilizzare tutte le istanze prenotate fino a 30 minuti prima dell'orario di fine del blocco di capacità. Le istanze ancora in esecuzione in quel momento inizieranno a terminare. Per avere tempo sufficiente per svuotare correttamente i nodi, suggeriamo di pianificare il dimensionamento fino a zero più di 30 minuti prima dell'orario di fine della prenotazione del blocco di capacità.

Se desideri invece aumentare le dimensioni manualmente ogni volta che la prenotazione della capacità diventa `Active`, devi aggiornare la capacità desiderata del gruppo con dimensionamento automatico all'inizio della prenotazione del blocco di capacità. Successivamente, dovrai anche ridurre le dimensioni manualmente più di 30 minuti prima dell'orario di fine della prenotazione del blocco di capacità.

5. Il gruppo di nodi è ora pronto per la pianificazione di carichi di lavoro e pod.
6. Affinché i tuoi pod vengano svuotati correttamente, consigliamo di configurare AWS Node Termination Handler. Questo strumento sarà in grado di monitorare gli eventi del ciclo di vita "Ridimensionamento di ASG" di Amazon EC2 Auto Scaling utilizzando EventBridge e consentire al piano di controllo Kubernetes di intraprendere l'azione richiesta prima che l'istanza diventi non disponibile. Altrimenti, i tuoi oggetti pod e Kubernetes rimarranno bloccati nello stato in sospeso. Per ulteriori informazioni, consulta AWS Node Termination Handler su GitHub.

   Se non configuri Node Termination Handler, consigliamo di cominciare a svuotare manualmente i pod prima di raggiungere la finestra di 30 minuti, in modo che abbiano tempo sufficiente per il corretto svuotamento.

Usa GB2 P6e-00 con UltraServers Amazon EKS

Questo argomento descrive come configurare e utilizzare Amazon EKS con GB2 P6e-00. UltraServers Il tipo di `p6e-gb200.36xlarge` istanza con 4 NVIDIA Blackwell GPUs è disponibile solo come P6e- 00. GB2 UltraServers Esistono due tipi di P6e- 00. GB2 UltraServers `u-p6e-gb200x36` UltraServer Ha 9 `p6e-gb200.36xlarge` istanze e `u-p6e-gb200x72` UltraServer ha 18 istanze. `p6e-gb200.36xlarge`

Per ulteriori informazioni, consulta la pagina Web di Amazon EC2 P6e- GB2 00 UltraServers .

Considerazioni

- Amazon EKS supporta GB2 P6e-00 UltraServers per le versioni 1.33 e successive di Kubernetes. Questa versione di Kubernetes fornisce il supporto per Dynamic Resource Allocation (DRA), abilitato di default in EKS e nella versione accelerata 023 ottimizzata per EKS. AL2 AMIs Il DRA è un requisito per utilizzare il P6e-00 con EKS. GB2 UltraServers Il DRA non è supportato in Karpenter o EKS Auto Mode e si consiglia di utilizzare i gruppi di nodi autogestiti EKS o i gruppi di nodi gestiti da EKS quando si utilizza il P6e-00 con EKS. GB2 UltraServers

- Le P6e- GB2 00 UltraServers sono rese disponibili tramite Capacity Blocks for ML. EC2 the section called "Gestione della capacità"Per informazioni su come avviare nodi EKS con Capacity Blocks, vedi.
- Quando si utilizzano gruppi di nodi gestiti EKS con Capacity Blocks, è necessario utilizzare modelli di avvio personalizzati. Quando si aggiornano i gruppi di nodi gestiti EKS con GB2 P6e-00 UltraServers, è necessario impostare la dimensione desiderata del gruppo di nodi prima dell'aggiornamento. `0`
- Si consiglia di utilizzare la variante AL2 023 ARM NVIDIA dell'accelerato ottimizzato per EKS. AMIs Questa AMI include i componenti e la configurazione del nodo necessari per funzionare con GB2 P6e-00. UltraServers Se decidi di creare la tua AMI, sei responsabile dell'installazione e della convalida della compatibilità del nodo e del software di sistema, inclusi i driver. Per ulteriori informazioni, consulta the section called "Usa la GPU EKS Linux AMIs".
- Si consiglia di utilizzare la versione AMI ottimizzata per EKS `v20251103` o successiva, che include il driver NVIDIA versione 580. Questa versione del driver NVIDIA consente alla Coherent Driver-Based Memory (CDMM) di risolvere il problema del potenziale eccesso di memoria. Quando CDMM è abilitato, le seguenti funzionalità non sono supportate: NVIDIA Multi-Instance GPU (MIG) e vGPU. Per ulteriori informazioni su CDMM, consulta NVIDIA Coherent Driver Memory Management (CDMM).
- Quando si utilizza l'operatore GPU NVIDIA con l'AMI AL2 NVIDIA 023 ottimizzata per EKS, è necessario disabilitare l'installazione da parte dell'operatore del driver e del toolkit, poiché questi sono già inclusi nell'AMI. Le versioni NVIDIA AL2 023 ottimizzate per EKS AMIs non includono il plug-in per dispositivi NVIDIA Kubernetes o il driver NVIDIA DRA, che devono essere installati separatamente.
- Ogni `p6e-gb200.36xlarge` istanza può essere configurata con un massimo di 17 schede di rete e può sfruttare EFA per la comunicazione tra. UltraServers Il traffico di rete dei carichi di lavoro può UltraServers intersecarsi, ma per ottenere prestazioni ottimali si consiglia di pianificare i carichi di lavoro nello stesso ambiente UltraServer sfruttando IMEX per la comunicazione intra-GPU. UltraServer Per ulteriori informazioni, consulta Configurazione EFA per le istanze P6e-00. GB2
- Ogni `p6e-gb200.36xlarge` istanza dispone di 3 istanze di storage da 7,5 TB. Per impostazione predefinita, l'AMI ottimizzata per EKS non formatta e monta gli instance store. Lo storage temporaneo del nodo può essere condiviso tra i pod che richiedono lo storage temporaneo e le immagini dei contenitori che vengono scaricate nel nodo. Se si utilizza l'AMI ottimizzata per EKS AL2 023, questa può essere configurata come parte del bootstrap dei nodi nei dati utente impostando la politica NodeConfigdi archiviazione locale dell'istanza su RAID0. Impostando su

RAID0 stripes, l'istanza archivia e configura il runtime del contenitore e il kubelet per utilizzare questo storage temporaneo.

Componenti

I seguenti componenti sono consigliati per eseguire carichi di lavoro su EKS con il P6e-00. GB2 UltraServers Facoltativamente, puoi utilizzare l'operatore GPU NVIDIA per installare i componenti del nodo NVIDIA. Quando si utilizza l'operatore GPU NVIDIA con l'AMI AL2 NVIDIA 023 ottimizzata per EKS, è necessario disabilitare l'installazione da parte dell'operatore del driver e del toolkit, poiché questi sono già inclusi nell'AMI.

Pila	Componente
AMI accelerata ottimizzata per EKS	Kernel 6.12
	Driver per GPU NVIDIA
	Driver in modalità utente NVIDIA CUDA
	Toolkit per container NVIDIA
	NVIDIA Fabric Manager
	Driver NVIDIA IMEX
	Gestore di sottorete NVIDIA NVLink
	Driver EFA
Componenti in esecuzione sul nodo	PENNA IN PVC
	Plugin per dispositivi EFA
	Plugin per dispositivi NVIDIA K8s
	Driver NVIDIA DRA
	NVIDIA Node Feature Discovery (NFD)
	NVIDIA GPU Feature Discovery (GFD)

I componenti del nodo nella tabella precedente svolgono le seguenti funzioni:

- VPC CNI: alloca VPC IPs come interfaccia di rete principale per i pod in esecuzione su EKS
- Plugin per dispositivi EFA: alloca i dispositivi EFA come reti secondarie per i pod in esecuzione su EKS. Responsabile del traffico di rete su P6e-00. GB2 UltraServers Per carichi di lavoro multinodo, GPU-to-GPU all'interno di un UltraServer canale che scorre su più nodi. NVLink
- Plugin per dispositivi NVIDIA Kubernetes: viene allocato come dispositivi per i GPUs pod in esecuzione su EKS. Si consiglia di utilizzare il plug-in per dispositivi NVIDIA Kubernetes fino a quando la funzionalità di allocazione GPU del driver NVIDIA DRA non sarà più sperimentale. Consulta le versioni dei driver NVIDIA DRA per informazioni aggiornate.
- Driver NVIDIA DRA: abilita risorse ComputeDomain personalizzate che facilitano la creazione di domini IMEX che seguono i carichi di lavoro in esecuzione su P6e-00. GB2 UltraServers
  - La ComputeDomain risorsa descrive un dominio Internode Memory Exchange (IMEX). Quando i carichi di lavoro con un ResourceClaim for a ComputeDomain vengono distribuiti nel cluster, il driver NVIDIA DRA crea automaticamente un IMEX DaemonSet che viene eseguito su nodi corrispondenti e stabilisce i canali IMEX tra i nodi prima dell'avvio del carico di lavoro. Per saperne di più su IMEX, consulta la panoramica di NVIDIA IMEX per sistemi multi-nodo. NVLink
  - Il driver NVIDIA DRA utilizza un'etichetta Clique ID (`nvidia.com/gpu.clique`) applicata da NVIDIA GFD che trasmette la conoscenza della topologia e del dominio di rete. NVLink
  - È consigliabile creare un lavoro per carico di lavoro. ComputeDomain
- NVIDIA Node Feature Discovery (NFD): dipendenza richiesta per GFD per applicare le etichette dei nodi in base agli attributi a livello di nodo rilevati.
- NVIDIA GPU Feature Discovery (GFD): applica un'etichetta topologica standard NVIDIA chiamata ai nodi. `nvidia.com/gpu.clique` I nodi all'interno dello stesso `nvidia.com/gpu.clique` hanno una NVLink raggiungibilità multinodo e puoi utilizzare le affinità dei pod nell'applicazione per pianificare i pod sullo stesso dominio. NVlink

Procedura

La sezione seguente presuppone che tu abbia un cluster EKS che esegue Kubernetes versione 1.33 o successiva con uno o più gruppi di nodi con GB2 P6e-00 che UltraServers esegue l'AMI accelerata ottimizzata per NVIDIA EKS AL2 023 ARM. Consulta i collegamenti per i passaggi preliminari per i nodi autogestiti e i gruppi di nodi gestiti da the section called “Gestione della capacità” EKS.

La procedura seguente utilizza i componenti riportati di seguito.

Name	Versione	Description
Operatore GPU NVIDIA	25.3.4+	Per la gestione del ciclo di vita dei plug-in richiesti come il plug-in per dispositivi NVIDIA Kubernetes e NFD/GFD.
Driver NVIDIA DRA	25.8.0+	Per la gestione ComputeDomain CRDs dei domini IMEX.
Plugin per dispositivi EFA	0.5.14 e versioni successive	Per la comunicazione incrociata. UltraServer

Installa l'operatore GPU NVIDIA

L'operatore GPU NVIDIA semplifica la gestione dei componenti necessari per l'uso nei cluster Kubernetes. GPUs Poiché il driver GPU NVIDIA e il toolkit container sono installati come parte dell'AMI accelerata ottimizzata per EKS, questi devono essere `false` impostati nella configurazione dei valori Helm.

1. Crea un file di valori Helm denominato con la seguente configurazione. `gpu-operator-values.yaml`

```
devicePlugin:
  enabled: true
nfd:
  enabled: true
gfd:
  enabled: true
driver:
  enabled: false
toolkit:
  enabled: false
migManager:
  enabled: false
```

2. Installa l'operatore GPU NVIDIA per il tuo cluster utilizzando il `gpu-operator-values.yaml` file creato nel passaggio precedente.

```
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
```

Installa il driver NVIDIA DRA

A partire dalla versione NVIDIA GPU Operator`v25.3.4`, il driver NVIDIA DRA deve essere installato separatamente. Si consiglia di tenere traccia delle note di rilascio dell'operatore della GPU NVIDIA poiché potrebbero cambiare in una versione futura.

1. Crea un file di valori Helm denominato `dra-values.yaml` con la seguente configurazione. Nota `tolerations` che configura il `nodeAffinity` driver DRA in modo che venga distribuito solo su nodi con una GPU 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. Installa il driver NVIDIA DRA per il tuo cluster utilizzando il `dra-values.yaml` file creato nel passaggio precedente.

```
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. Dopo l'installazione, il driver DRA crea `DeviceClass` risorse che consentono a Kubernetes di comprendere e allocare le `ComputeDomain` risorse, rendendo possibile la gestione IMEX per carichi di lavoro GPU distribuiti su P6e-00. GB2 UltraServers

   Verifica che le risorse DRA siano disponibili con i seguenti comandi.

```
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
```

Installa il plug-in del dispositivo EFA

Per utilizzare la comunicazione EFA tra UltraServers, devi installare il plug-in del dispositivo Kubernetes per EFA. Le istanze GB2 P6e-00 possono essere configurate con un massimo di 17 schede di rete e l'NCI primario (indice 0) deve essere di tipo `interface` e supporta fino a 100 Gbps di larghezza di banda ENA. Configura le interfacce EFA ed ENA in base alle tue esigenze durante il provisioning dei nodi. Consulta la AWS documentazione relativa alla configurazione EFA per le istanze GB2 P6e-00 per maggiori dettagli sulla configurazione EFA.

1. Crea un file di valori Helm denominato con la seguente configurazione. `efa-values.yaml`

```
tolerations:
  - key: nvidia.com/gpu
    operator: Exists
    effect: NoSchedule
```

2. Installa l'operatore NVIDIA DRA per il tuo cluster utilizzando il `dra-values.yaml` file creato nel passaggio precedente.

```
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
```

Ad esempio, se hai configurato le istanze con 1 interfaccia solo efa in ogni gruppo NCI, quando descrivi un nodo, si prevede che vengano visualizzati 4 dispositivi EFA allocabili per nodo.

```
kubectl describe node/<gb200-node-name>
```

```
Capacity:
  ...
  vpc.amazonaws.com/efa:  4
Allocatable:
  ...
  vpc.amazonaws.com/efa:  4
```

Convalida IMEX su più nodi NVLink

Per un test NVLINK NCCL multinodo e altri micro-benchmark, consulta il repository. awesome-distributed-training GitHub I passaggi seguenti mostrano come eseguire un test multinodo con nvbandwidth. NVLink

1. Per eseguire un test della larghezza di banda multinodo su due nodi del NVL72 dominio, installa prima l'operatore MPI:

   ```
   kubectl create -f https://github.com/kubeflow/mpi-operator/releases/download/v0.7.0/
   mpi-operator.yaml
   ```

2. Create un file di valori Helm denominato `nvbandwidth-test-job.yaml` che definisce il manifesto del test. Nota l'affinità del `nvidia.com/gpu.clique` pod per pianificare i lavoratori nello stesso NVLink dominio con raggiungibilità NVLink multinodo.

   A partire dalla versione NVIDIA DRA, i driver `v25.8.0` ComputeDomains sono elastici e `.spec.numNodes` possono essere impostati nella definizione. `0` ComputeDomain Consulta le ultime note di rilascio del driver NVIDIA DRA per gli aggiornamenti.

   ```
   ---
   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. Crea ComputeDomain e avvia il processo con il seguente comando.

```
kubectl apply -f nvbandwidth-test-job.yaml
```

4. ComputeDomain creazione, puoi vedere che il carico di lavoro ComputeDomain ha due nodi:

```
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. Esamina i risultati del lavoro con il seguente comando.

```
kubectl logs --tail=-1 -l job-name=nvbandwidth-test-launcher
```

6. Quando il test è completo, eliminalo con il seguente comando.

```
kubectl delete -f nvbandwidth-test-job.yaml
```

Ricette per ottimizzare il tuo cluster Amazon EKS per i AI/ML carichi di lavoro

> **Tip**
>
> Registrati ai prossimi AI/ML workshop Amazon EKS.

Questa sezione è progettata per fornire semplici ricette per l'ottimizzazione del cluster Amazon EKS, in particolare per AI/ML carichi di lavoro che coinvolgono hardware specializzato. Troverai indicazioni su come impedire che i pod vengano pianificati su nodi specifici aggiungendo problemi ai gruppi di nodi gestiti, inclusi prerequisiti, procedure e considerazioni sulla distribuzione. step-by-step

Argomenti

- Ricetta: impedire che i pod siano programmati su nodi specifici

Ricetta: impedire che i pod siano programmati su nodi specifici

Panoramica

I nodi con processori specializzati, come le GPU, possono essere più costosi da eseguire rispetto ai nodi su macchine standard. Per proteggere questi nodi da carichi di lavoro che non richiedono hardware speciale, è possibile utilizzare taint Kubernetes. I taint contrassegnano i nodi per respingere i pod che non hanno tolleranze corrispondenti, assicurando che siano pianificati solo carichi di lavoro compatibili. Per ulteriori informazioni, consulta Taint e tolleranze nella documentazione Kubernetes.

I taint dei nodi Kubernetes possono essere applicati a gruppi di nodi gestiti nuovi ed esistenti utilizzando il comando Console di gestione AWS o tramite l'API Amazon EKS. Questa ricetta mostra come applicare i taint ai gruppi di nodi gestiti di Amazon EKS utilizzando AWS CLI. Per informazioni sulla creazione di un gruppo di nodi con un taint utilizzando la Console di gestione AWS, consulta the section called "Crea".

Prerequisiti

- Un cluster Amazon EKS esistente.
- AWS CLI installata e configurata con autorizzazioni appropriate.

Fasi

Passaggio 1: creazione di un gruppo di nodi con i taint

Utilizza il comando `aws eks create-nodegroup` per creare un nuovo gruppo di nodi gestito con taint. Questo esempio applica un taint con chiave `dedicated`, valore `gpuGroup` ed effetto `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"
     }
   ]
}'
```

Per ulteriori informazioni ed esempi, consulta taint nella documentazione di riferimento di Kubernetes.

Passaggio 2: aggiornamento dei taint su un gruppo di nodi esistente

Utilizzare il comando AWS CLI aws eks update-nodegroup-config per aggiungere, rimuovere o sostituire i taint per i gruppi di nodi gestiti.

```
aws eks update-nodegroup-config
```

```
  --cluster-name my-cluster
  --nodegroup-name node-taints-example
  --taints 'removeTaints=[{key=dedicated,value=gpuGroup,effect=NO_SCHEDULE}]'
```

Note

- I taint possono essere aggiornati dopo aver creato il gruppo di nodi utilizzando l'API `UpdateNodegroupConfig`.
- La chiave del taint deve iniziare con una lettera o un numero. Può includere lettere, numeri, trattini (`-`), punti (`.`) e trattini bassi (`_`). Può contenere fino a 63 caratteri.
- Facoltativamente, la chiave del taint può iniziare con un prefisso di sottodominio DNS e un singolo `/`. Se inizia con un prefisso di sottodominio DNS, può avere una lunghezza di 253 caratteri.
- Il valore è facoltativo e deve iniziare con una lettera o un numero. Può includere lettere, numeri, trattini (`-`), punti (`.`) e trattini bassi (`_`). Può contenere fino a 63 caratteri.
- Quando si utilizza Kubernetes direttamente o Console di gestione AWS, l'effetto taint deve essere `NoSchedule`, `PreferNoSchedule` o `NoExecute`. Tuttavia, quando si utilizza AWS direttamente o l'API, l'effetto taint deve essere `NO_SCHEDULE`, `PREFER_NO_SCHEDULE` o `NO_EXECUTE`.
- Sono consentiti un massimo di 50 taint per un gruppo di nodi.
- Se i taint creati utilizzando un gruppo di nodi gestito vengono rimossi manualmente da un nodo, Amazon EKS non li aggiunge nuovamente al nodo. Questo è vero anche se i taint sono specificati nella configurazione del gruppo di nodi gestito.

Risorse per iniziare a utilizzare IA/ML su Amazon EKS

Per passare al Machine Learning su EKS, iniziare scegliendo tra questi modelli prescrittivi per preparare rapidamente un cluster EKS e un software e un hardware ML per iniziare a eseguire carichi di lavoro ML.

Workshop

Workshop sull'IA generativa su Amazon EKS

Ulteriori informazioni su come iniziare a utilizzare le applicazioni e l'inferenza di un modello linguistico di grandi dimensioni (LLM) su Amazon EKS. Ulteriori informazioni su come implementare e gestire carichi di lavoro LLM di livello di produzione. Attraverso laboratori pratici, sarà possibile scoprire come sfruttare Amazon EKS insieme a servizi e strumenti open source AWS per creare solide soluzioni

LLM. L'ambiente del workshop fornisce tutta l'infrastruttura e gli strumenti necessari, consentendo di rivolgere la propria attenzione sull'apprendimento e sull'implementazione.

IA generativa su Amazon EKS con Neuron

Ulteriori informazioni su come iniziare a utilizzare le applicazioni e l'inferenza di un modello linguistico di grandi dimensioni (LLM) su Amazon EKS. Ulteriori informazioni su come implementare e gestire carichi di lavoro LLM di livello di produzione, implementare modelli RAG avanzati con database vettoriali e creare applicazioni LLM basate su dati utilizzando framework open source. Attraverso laboratori pratici, sarà possibile scoprire come sfruttare Amazon EKS insieme a servizi e strumenti open source AWS per creare solide soluzioni LLM. L'ambiente del workshop fornisce tutta l'infrastruttura e gli strumenti necessari, consentendo di rivolgere la propria attenzione sull'apprendimento e sull'implementazione.

Best practice

Gli argomenti incentrati su IA/ML nella guida alle best practice di Amazon EKS forniscono consigli dettagliati nelle seguenti aree per ottimizzare i carichi di lavoro IA/ML su Amazon EKS.

Calcolo e scalabilità automatica di IA/ML

Questa sezione descrive le beat practice per l'ottimizzazione del calcolo di IA/ML e della scalabilità automatica in Amazon EKS, concentrandosi sulla gestione delle risorse di GPU, sulla resilienza dei nodi e sulla scalabilità delle applicazioni. Offre strategie come la pianificazione dei carichi di lavoro con etichette e affinità di nodi note, l'utilizzo di blocchi di capacità per ML o prenotazioni della capacità on-demand e l'implementazione di controlli dell'integrità dei nodi con strumenti come un agente di monitoraggio dei nodi EKS.

Rete IA/ML

Questa sezione descrive le best practice per ottimizzare le reti IA/ML in Amazon EKS per migliorare le prestazioni e la scalabilità, incluse strategie come la selezione di istanze con una maggiore larghezza di banda della rete o Elastic Fabric Adapter (EFA) per la formazione distribuita, l'installazione di strumenti come MPI e NCCL e l'abilitazione della delega dei prefissi per aumentare gli indirizzi IP e migliorare i tempi di avvio dei pod.

Sicurezza IA/ML

Questa sezione si concentra sulla protezione dell'archiviazione di dati e sulla garanzia della conformità per i carichi di lavoro IA/ML su Amazon EKS, incluse pratiche come l'utilizzo di Amazon

S3 con Servizio di gestione delle chiavi (KMS) AWS per la crittografia lato server (SSE-KMS), la configurazione di bucket con chiavi KMS regionali e chiavi bucket S3 per ridurre i costi, la concessione di autorizzazioni IAM per azioni KMS come la decrittografia dei pod EKS e l'audit con registri CloudTrail AWS.

Archiviazione IA/ML

Questa sezione fornisce le best practice per ottimizzare l'archiviazione nei carichi di lavoro IA/ML su Amazon EKS, incluse pratiche come la distribuzione di modelli che utilizzano driver CSI per montare servizi come S3, FSx for Lustre o EFS come volumi persistenti, la selezione dell'archiviazione in base alle esigenze del carico di lavoro (ad esempio, FSx per Lustre per la formazione distribuita con opzioni come Scratch-SSD o Persistent-SSD) e l'abilitazione di funzionalità come la compressione e lo striping dei dati.

Osservabilità IA/ML

Questa sezione si concentra sul monitoraggio e sull'ottimizzazione dell'utilizzo della GPU per i carichi di lavoro IA/ML su Amazon EKS per migliorare l'efficienza e ridurre i costi, incluse strategie come mirare all'utilizzo elevato della GPU con strumenti come approfondimenti sui container CloudWatch e DCGM-Exporter di NVIDIA integrato con Prometheus e Grafana, e parametri che ti consigliamo di analizzare per i propri carichi di lavoro IA/ML.

Prestazioni IA/ML

Questa sezione si concentra sul miglioramento della scalabilità e delle prestazioni delle applicazioni per i carichi di lavoro IA/ML su Amazon EKS attraverso la gestione delle immagini dei container e l'ottimizzazione dell'avvio, incluse pratiche come l'utilizzo di immagini di base piccole e leggere o container di deep learning AWS con sviluppi in più fasi, il precaricamento delle immagini tramite snapshot EBS o il preinserimento nella cache di runtime utilizzando DaemonSets o Deployment.

Architetture di riferimento

Si esplorino questi repository GitHub per architetture di riferimento, codice esemplificativo e utilità per implementare formazione e inferenza distribuite per carichi di lavoro IA/ML su Amazon EKS e altri servizi AWS.

Addestramento distribuito AWSome

Questo repository offre una raccolta di best practice, architetture di riferimento, esempi di addestramento dei modelli e utilità per addestrare modelli di grandi dimensioni su AWS. Supporta

l'addestramento distribuito con Amazon EKS, inclusi modelli CloudFormation per cluster EKS, build personalizzate di AMI e container, casi di test per framework come PyTorch (DDP/FSDP, MegatronLM, NeMo) e JAX e strumenti per la convalida, l'osservabilità e il monitoraggio delle prestazioni come EFA Prometheus exporter e Nvidia Nsight Systems.

Inferenza AWSome

Questo repository fornisce architetture di riferimento e casi di test per ottimizzare le soluzioni di inferenza su AWS, con particolare attenzione ad Amazon EKS e alle istanze EC2 accelerate. Include configurazioni di infrastruttura per cluster VPC ed EKS, progetti per framework come NVIDIA NiMS, Tensorrt-LLM, Triton Inference Server e RayService, con esempi per modelli come Llama3-8B e Llama 3.1 405B. Offre implementazioni multinodo utilizzando K8s LeaderWorkerSet, dimensionamento automatico di EKS, GPU multi-istanza (MIG) e casi d'uso reali come un bot audio per ASR, inferenza e TTS.

Tutorial

Se si desidera configurare piattaforme e framework di Machine Learning in EKS, è necessario esplorare i tutorial descritti in questa sezione. Questi tutorial coprono tutto, dai modelli per sfruttare al meglio i processori delle GPU alla scelta degli strumenti di modellazione alla creazione di framework per settori specializzati.

Creare piattaforme di IA generativa su EKS

- Implementazione dei modelli di IA generativa su Amazon EKS
- Creazione di piattaforme JupyterHub multi-tenant su Amazon EKS

Eseguire framework di IA generativa specializzati su EKS

- Accelerazione dei propri carichi di lavoro di addestramento distribuiti di IA generativa con il framework NVIDIA NeMo su Amazon EKS
- Avvio su Amazon Elastic Kubernetes Service

Massimizzare le prestazioni della GPU NVIDIA per ML su EKS

- Implementare la condivisione delle GPU per utilizzare in modo efficiente le GPU NVIDIA per i propri cluster EKS:

Condivisione della GPU su Amazon EKS con time-slicing NVIDIA e istanze EC2 accelerate

- Usare GPU multi-istanza (MiGS) e microservizi NIM per eseguire più pod per GPU sui propri cluster EKS:

  Massimizzazione dell'utilizzo della GPU con la GPU multi-istanza (MIG) di NVIDIA su Amazon EKS: esecuzione di più pod per GPU per prestazioni migliorate
- Creazione e distribuzione di un sistema di machine learning scalabile su Kubernetes con Kubeflow su AWS

Eseguire carichi di lavoro di codifica video su EKS

- Distribuzione di contenuti video con GPU frazionarie in container su Amazon EKS

Accelerare il caricamento delle immagini per i carichi di lavoro di inferenza

- In che modo H2O.ai ha ottimizzato e protetto la propria infrastruttura IA/ML con Karpenter e Bottlerocket

Monitoraggio dei carichi di lavoro ML

- Monitoraggio dei carichi di lavoro delle GPU su Amazon EKS utilizzando servizi open source gestiti AWS
- mAbilitazione dei parametri delle GPU basati su pod in Amazon CloudWatch

Strumenti Amazon EKS

Amazon EKS offre una suite di AWS strumenti che ti aiutano a gestire, risolvere i problemi e interagire con i tuoi cluster EKS in modo più efficiente. Questi strumenti sfruttano funzionalità basate sull'intelligenza artificiale e interfacce standardizzate per semplificare le operazioni dei cluster, accelerare la risoluzione dei problemi e consentire interazioni in linguaggio naturale con le risorse Kubernetes. Gli strumenti disponibili includono:

- Amazon EKS Model Context Protocol (MCP) Server: un servizio completamente gestito che consente esperienze basate sull'intelligenza artificiale per la gestione dei cluster EKS. Il server MCP EKS si integra con assistenti di codifica AI compatibili con MCP per fornire una conoscenza contestuale e in tempo reale dei cluster e delle risorse Kubernetes. Usalo per gestire i cluster, distribuire applicazioni, risolvere problemi e interrogare la documentazione EKS tramite interazioni in linguaggio naturale.
- Amazon Q sulla console Amazon EKS: un'integrazione per la risoluzione dei problemi basata sull'intelligenza artificiale disponibile direttamente nella console di AWS gestione. Amazon Q analizza automaticamente i problemi dei cluster e fornisce i risultati delle indagini, l'analisi della causa principale e le misure di mitigazione suggerite in caso di errori o avvisi nella console EKS.

Questi strumenti collaborano per fornire un supporto completo per tutto il ciclo di vita del cluster EKS, dalla configurazione e configurazione iniziali alle operazioni continue e alla risoluzione dei problemi. Che stiate sviluppando applicazioni, gestendo l'infrastruttura o risolvendo problemi di produzione, questi AWS strumenti vi aiutano a lavorare in modo più efficiente con i vostri cluster EKS.

Argomenti

- Server MCP (Model Context Protocol) Amazon EKS
- Usa Amazon Q Developer sulla console Amazon EKS

Server MCP (Model Context Protocol) Amazon EKS

Il server MCP Amazon EKS è un servizio completamente gestito che consente esperienze basate sull'intelligenza artificiale per lo sviluppo e le operazioni. Model Context Protocol (MCP) fornisce un'interfaccia standardizzata che arricchisce gli agenti e le applicazioni AI con una conoscenza contestuale e in tempo reale dei cluster EKS e delle risorse Kubernetes, consentendo risposte più accurate e sensibili al contesto e flussi di lavoro basati sull'intelligenza artificiale per tutto il

ciclo di vita dell'applicazione, dalla configurazione iniziale all'ottimizzazione della produzione e alla risoluzione dei problemi.

> **Note**
>
> Amazon EKS MCP Server è in versione di anteprima per Amazon EKS ed è soggetto a modifiche.

Panoramica di

Il server MCP EKS può essere facilmente integrato con qualsiasi assistente di codifica AI compatibile con MCP per migliorare il flusso di lavoro di sviluppo come Kiro, Amazon Q Developer CLI o strumenti di terze parti come Cursor o Cline. All'inizio, il server EKS MCP vi guida nella creazione del cluster, nel provisioning automatico dei prerequisiti e nell'applicazione delle migliori pratiche. AWS Durante lo sviluppo, semplifica le operazioni di EKS e Kubernetes fornendo flussi di lavoro di alto livello per la distribuzione delle applicazioni e la gestione dei cluster. Per il debug e la risoluzione dei problemi, il server accelera la risoluzione dei problemi attraverso strumenti integrati per la risoluzione dei problemi e l'accesso alla knowledge base disponibili sulla console EKS e sui tuoi assistenti AI preferiti. Queste funzionalità sono accessibili tramite interazioni in linguaggio naturale, che consentono di eseguire operazioni Kubernetes complesse in modo più intuitivo ed efficiente.

Il server MCP EKS completamente gestito è ospitato nel AWS cloud, eliminando la necessità di installazione e manutenzione locali. Fornisce funzionalità di livello aziendale come aggiornamenti e patch automatici, sicurezza centralizzata tramite l'integrazione AWS IAM, registrazione completa degli audit e la comprovata scalabilità AWS CloudTrail, affidabilità e supporto di. AWS Il server MCP EKS completamente gestito ospitato nel cloud offre i seguenti vantaggi chiave: AWS

- Eliminate l'installazione e la manutenzione. Con il server EKS MCP ospitato nel AWS cloud, non è più necessario gestire gli aggiornamenti delle versioni o risolvere i problemi del server locale. Basta configurare il vostro assistente AI per connettersi all'endpoint del server EKS MCP ospitato e sarete pronti per iniziare a lavorare con i vostri cluster EKS.
- Gestione centralizzata degli accessi. Il server EKS MCP si integra con IAM, fornendo un modo centralizzato e sicuro per controllare l'accesso al server. Tutte le richieste vengono firmate utilizzando AWS SigV4 tramite un proxy leggero, che consente una perfetta integrazione con le credenziali e le politiche IAM esistenti. AWS

- Monitoraggio e visibilità migliorati. AWS CloudTrail l'integrazione acquisisce le chiamate agli strumenti di inizializzazione e accesso completo effettuate tramite il servizio ospitato, consentendo audit trail dettagliati e report di conformità.
- Sempre. up-to-date Ricevi automaticamente nuovi strumenti, funzionalità e correzioni di bug senza dover aggiornare le installazioni locali. Il servizio ospitato viene continuamente migliorato in base al feedback e alle AWS best practice degli utenti.

Esempi di integrazione

Il server MCP EKS offre diversi strumenti che è possibile utilizzare per:

- Gestisci il tuo cluster Crea, configura e gestisci i cluster EKS con le migliori pratiche automatizzate.
- Gestisci le risorse Kubernetes Implementa applicazioni, gestisci le risorse Kuberentes e controlla lo stato del cluster.
- Risolvi i problemi del cluster Diagnostica i problemi utilizzando gli strumenti di risoluzione dei problemi integrati e la knowledge base dei runbook
- Interroga la documentazione Cerca e recupera la documentazione EKS pertinente contestualmente.

Esplora i tuoi cluster

```
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
```

Controlla le risorse di 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
```

Risoluzione dei problemi di

```
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
```

Crea risorse (se la modalità «scrittura» è abilitata)

```
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
```

Nozioni di base

Per iniziare, consulta the section called "Nozioni di base".

Guida introduttiva al server MCP Amazon EKS

Questa guida illustra i passaggi per configurare e utilizzare il server MCP EKS con i tuoi assistenti di codice AI. Imparerai come configurare il tuo ambiente, connetterti al server e iniziare a gestire i cluster EKS attraverso interazioni in linguaggio naturale.

> **Note**
>
> Amazon EKS MCP Server è in versione di anteprima per Amazon EKS ed è soggetto a modifiche.

Prerequisiti

Prima di iniziare, esegui le seguenti attività:

- Crea un AWS account con accesso ad Amazon EKS
- Installa e configura la AWS CLI con le credenziali
- Installa Python 3.10+
- Installa uv

Configurazione

1. Verifica dei prerequisiti di

```
# Check that your Python version is 3.10 or higher
python3 --version
```

```
# Check uv installation
uv --version

# Verify CLI configuration
aws configure list
```

2. Configura le autorizzazioni IAM

Per connettersi al server EKS MCP, il ruolo IAM deve avere le seguenti policy allegate: **`eks-mcp:InvokeMcp`**(autorizzazioni necessarie per l'inizializzazione e il recupero delle informazioni sugli strumenti disponibili), **`eks-mcp:CallReadOnlyTool`**(autorizzazioni richieste per l'utilizzo di strumenti di sola lettura) e (autorizzazioni richieste per l'utilizzo di strumenti di accesso completo **`eks-mcp:CallPrivilegedTool`**(scrittura)). Queste `eks-mcp` autorizzazioni sono incluse nelle politiche gestite di sola lettura e accesso completo fornite di seguito. AWS

- Apri la console IAM.
- Nel riquadro di navigazione a sinistra, scegli Utenti, Gruppi di utenti o Ruoli a seconda dell'identità a cui desideri allegare la politica, quindi del nome dell'utente, del gruppo o del ruolo specifico.
- Scegli la scheda Autorizzazioni.
- Scegli Allega politiche (o Aggiungi autorizzazioni se è la prima volta).
- Nell'elenco delle politiche, cerca e seleziona la politica gestita che desideri allegare:
- Operazioni di sola lettura: Amazon EKSMCPRead OnlyAccess
- Scegli Allega politiche (o Avanti e poi Aggiungi autorizzazioni per confermare).

In questo modo viene allegata la politica e le autorizzazioni hanno effetto immediato. È possibile allegare più policy alla stessa identità e ognuna può contenere diverse autorizzazioni. Per ulteriori informazioni su queste politiche, consulta le politiche AWS gestite per Amazon Elastic Kubernetes Service.

3. Scegli un assistente AI

Scegli uno dei seguenti assistenti AI compatibili con MCP o uno strumento compatibile con MCP:

- Installa Amazon Q Developer CLI
- Installa Kiro
- Installa Cursor
- Installa l'estensione Cline VS Code

Passaggio 1: configura il tuo assistente AI

Scegli una delle seguenti opzioni per configurare il tuo assistente di codice AI. Il completamento di questo passaggio configura il tuo assistente di codice AI per utilizzare il proxy MCP AWS, necessario per l'accesso sicuro e autenticato al server MCP Amazon EKS. Ciò comporta l'aggiunta o la modifica del file di configurazione MCP (ad esempio, `~/.aws/amazonq/mcp.json` per Amazon Q Developer CLI). Il proxy funge da bridge lato client, gestendo l'autenticazione AWS SigV4 utilizzando AWS le credenziali locali e abilitando l'individuazione dinamica degli strumenti per l'interazione con server MCP di backend AWS come EKS MCP Server. Per ulteriori informazioni, consulta MCP Proxy for repository. AWS

Opzione A: Amazon Q Developer CLI

La CLI Q Developer offre l'esperienza più integrata con il server MCP EKS.

1. Individua il file di configurazione MCP

- macOS/Linux: `~/.aws/q/mcp.json`
- Windows: `%USERPROFILE%\.aws\q\mcp.json`

2. Aggiungi configurazione del server MCP

Create il file di configurazione se non esiste. Assicurati di sostituire il segnaposto region (`{region}`) con la regione desiderata.

Per 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}"
      ]
    }
  }
}
```

Per 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}"
      ]
    }
  }
}
```

Nota sulla sicurezza: `--read-only` può essere utilizzato per consentire solo operazioni di sola lettura sullo strumento.

3. Verifica la configurazione

Riavvia Q Developer CLI, quindi controlla gli strumenti disponibili:

```
q /tools
```

Opzione B: Kiro IDE

Kiro è uno spazio di lavoro di codifica incentrato sull'intelligenza artificiale con supporto MCP integrato.

1. Apri le impostazioni di Kiro

- Apri Kiro
- Vai su Kiro → Impostazioni e cerca «MCP Config»
- Oppure premi `Cmd+Shift+P,` (Mac) o `Ctrl+Shift+P,` (Windows/Linux) e cerca «MCP Config»

2. Aggiungi la configurazione del server MCP

- Fai clic su «Open Workspace MCP Config» o «Open User MCP Config» per modificare direttamente il file di configurazione MCP.

Assicurati di sostituire il segnaposto region (`{region}`) con la regione desiderata.

Per 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}"
      ]
    }
  }
}
```

Per 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}"
      ]
    }
  }
}
```

Nota sulla sicurezza: `--read-only` può essere utilizzato per consentire solo operazioni di sola lettura sullo strumento.

Opzione C: Cursor IDE

Cursor fornisce il supporto MCP integrato con un'interfaccia di configurazione grafica.

1. Apri le impostazioni del cursore

- Apri il cursore
- Vai a Impostazioni → Impostazioni del cursore → Strumenti e MCP
- Oppure premi `Cmd+Shift+P` (Mac)/`Ctrl+Shift+P`(Windows) e cerca «MCP»

2. Aggiungi la configurazione del server MCP

- Fai clic su «Nuovo server MCP»

Crea il file di configurazione se non esiste. Assicurati di sostituire il segnaposto region (`{region}`) con la regione desiderata.

Per 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}"
      ]
    }
  }
}
```

Per 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}"
      ]
    }
  }
```

```
}
```

Nota sulla sicurezza: `--read-only` può essere utilizzato per consentire solo operazioni di sola lettura sullo strumento.

3. Riavvia il cursore

Chiudi e riapri il cursore per rendere effettive le modifiche.

4. Verifica nella chat di Cursor

Apri il pannello della chat e prova a:

```
What EKS MCP tools are available?
```

Dovresti vedere un elenco di strumenti di gestione EKS disponibili.

Opzione D: Cline (VS Code Extension)

Cline è una popolare estensione VS Code che porta l'assistenza AI direttamente nel tuo editor.

1. Apri le impostazioni di Cline

- Apri Cline
- Premi `Cmd+Shift+P` (Mac)/`Ctrl+Shift+P`(Windows) e cerca «MCP»

2. Aggiungi la configurazione del server MCP

- Fai clic su «Aggiungi server»
- Fai clic su «Apri configurazione utente»

Crea il file di configurazione se non esiste. Assicurati di sostituire il segnaposto region (`{region}`) con la regione desiderata.

Per 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}"
      ]
    }
  }
}
```

Per 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}"
      ]
    }
  }
}
```

Nota sulla sicurezza: `--read-only` può essere utilizzato per consentire solo operazioni di sola lettura sullo strumento.

2. Ricarica VS Code

PremiCmd+Shift+P/Ctrl+Shift+Pe seleziona «Developer: Reload Window»

3. Verifica la configurazione

Apri Cline e chiedi:

```
List the available MCP tools for EKS
```

Fase 2: (Facoltativo) Creare una politica di «scrittura»

Facoltativamente, puoi creare una policy IAM gestita dal cliente che fornisca l'accesso completo al server MCP Amazon EKS. Questa policy concede le autorizzazioni per utilizzare tutti gli strumenti nel server EKS MCP, inclusi gli strumenti privilegiati che possono comportare operazioni di scrittura e gli strumenti di sola lettura. Tieni presente che le autorizzazioni ad alto rischio (qualsiasi cosa con Delete* o una risorsa IAM senza restrizioni) sono incluse in questa policy, in quanto sono richieste per le risorse del cluster nello strumento manage_eks_stacks. setup/teardown

```
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\": \"*\"}]}"
```

Fase 3: Verifica la configurazione

Prova la connessione

Fai una semplice domanda al tuo assistente AI per verificare la connessione:

```
List all EKS clusters in my {aws} account
```

Dovresti vedere un elenco dei tuoi cluster EKS.

Fase 4: Esegui le tue prime attività

Esempio 1: esplora i tuoi cluster

```
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
```

Esempio 2: verifica le risorse di 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
```

Esempio 3: Risoluzione dei problemi

```
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
```

Esempio 4: creazione di risorse (se la modalità «scrittura» è abilitata)

```
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
```

Configurazioni comuni

Scenario 1: profili multipli AWS

Se lavori con più AWS account, crea configurazioni server MCP separate.

Per 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"
      ]
    }
  }
}
```

Per 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"
      ]
    }
  }
}
```

Scenario 2: Sola lettura per la produzione

Creare una configurazione di sola lettura per gli ambienti di produzione.

Per 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"
      ]
    }
  }
}
```

Per 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"
      ]
    }
  }
}
```

Scenario 3: sviluppo con accesso completo

Per ambienti di sviluppo con accesso completo in scrittura.

Per 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"
      ]
    }
  }
```

```
}
```

Per 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"
      ]
    }
  }
}
```

Considerazioni

Sicurezza

Non passate informazioni segrete o sensibili tramite meccanismi di input consentiti:

- Non includere segreti o credenziali nei file YAML applicati con apply_yaml.
- Non passate informazioni sensibili direttamente nel prompt al modello.
- Non includete segreti nei CloudFormation modelli o nei manifesti delle applicazioni.
- Evita di utilizzare strumenti MCP per creare Kubernetes Secrets, poiché ciò richiederebbe la fornitura dei dati segreti al modello.
- Evita di registrare informazioni sensibili nei log delle applicazioni all'interno dei pod Kubernetes.

Sicurezza dei contenuti YAML:

- Usa solo file YAML provenienti da fonti affidabili.

- Il server si basa sulla convalida dell'API Kubernetes per i contenuti YAML e non esegue la propria convalida.
- Controlla i file YAML prima di applicarli al cluster.

Invece di passare segreti tramite MCP:

- Usa AWS Secrets Manager o Parameter Store per archiviare informazioni riservate.
- Configura l'RBAC Kubernetes appropriato per gli account di servizio.
- Usa IAM roles for service accounts (IRSA) per l'accesso ai servizi dai pod. AWS

Redazione di dati sensibili:

- Il server MCP EKS oscura automaticamente gli schemi comuni relativi a token di sicurezza, certificati e altre informazioni sensibili nelle risposte degli strumenti.
- I valori redatti vengono sostituiti con `HIDDEN_FOR_SECURITY_REASONS` per evitare l'esposizione accidentale dei dati al modello.
- Questa redazione si applica a tutte le risposte degli strumenti, inclusi i registri, le descrizioni delle risorse e i dati di configurazione.

Prossimo

Per le opzioni di configurazione, vederethe section called “Configurazione”. Per un elenco completo degli strumenti, vederethe section called “Tools (Strumenti)”.

Riferimento alla configurazione del server MCP di Amazon EKS

Questa guida mostra tutte le configurazioni disponibili per lo strumento mcp-proxy-for-awslato client che ti consente di connetterti al server MCP Amazon EKS completamente gestito dal tuo IDE.

> **Note**
>
> Amazon EKS MCP Server è in versione di anteprima per Amazon EKS ed è soggetto a modifiche.

Esempio

```
{
  "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"
      ]
    }
  }
}
```

autorizzazioni IAM

Il ruolo utilizzato per la connessione al server MCP richiede le **`eks-mcp:InvokeMcp`**autorizzazioni per l'inizializzazione e il recupero delle informazioni sugli strumenti disponibili. **`eks-mcp:CallReadOnlyTool`**è necessario per l'utilizzo di strumenti di sola lettura ed **`eks-mcp:CallPrivilegedTool`**è necessario per l'utilizzo di strumenti di accesso completo (scrittura).

Variabili di ambiente

`AWS_PROFILE`(opzionale) nome del profilo delle AWS credenziali da utilizzare; può essere sovrascritto dall'argomento della `--profile` riga di comando.

- Ad esempio: `export AWS_PROFILE=production`

`AWS_REGION`(opzionale) AWS regione per la firma di SigV4; il valore predefinito è se non è impostato. `us-west-2`

- Ad esempio: `export AWS_REGION=us-east-1`

Arguments (Argomenti)

URL dell'endpoint MCP SigV4 (obbligatorio) L'URL dell'endpoint MCP a cui connettersi.

- Ad esempio: `https://eks-mcp.us-west-2.api.aws/mcp`

`--service`(opzionale) nome del AWS servizio per la firma SigV4; rilevato automaticamente dal nome host dell'endpoint se non fornito.

- Ad esempio: `--service eks-mcp`

`--profile`(opzionale) AWS profilo di credenziali da utilizzare. Il valore predefinito è la variabile di `AWS_PROFILE` ambiente se non specificata.

- Ad esempio: `--profile production`

`--region` AWS regione da usare. Utilizza la variabile di `AWS_REGION` ambiente se non è impostata, il valore predefinito è. `us-east-1`

- Ad esempio: `--region us-west-2`

`--read-only`(opzionale) Disabilita gli strumenti che potrebbero richiedere autorizzazioni di scrittura (gli strumenti che NON richiedono autorizzazioni di scrittura sono annotati con =true). readOnlyHint Per impostazione predefinita, tutti gli strumenti sono abilitati.

- Ad esempio: `--read-only`

Per ulteriori opzioni di configurazione, vedere Parametri di configurazione.

Tools (Strumenti)

Per informazioni dettagliate su tutti gli strumenti disponibili, inclusi parametri ed esempi di utilizzo, vederethe section called "Tools (Strumenti)".

Riferimento agli strumenti per server MCP di Amazon EKS

Il server espone i seguenti strumenti MCP.

> **Note**
>
> Amazon EKS MCP Server è in versione di anteprima per Amazon EKS ed è soggetto a modifiche.

Strumenti di sola lettura

Questa sezione descrive gli strumenti di sola lettura disponibili per il server MCP EKS. Tieni presente che tutte le operazioni dell'API Kubernetes in sola lettura possono accedere a entrambi:

- Cluster privati (vedi Cluster private endpoint)
- Cluster pubblici

search_eks_documentation

Cerca informazioni e linee guida nella documentazione EKS. up-to-date Questo strumento fornisce l'accesso alla documentazione EKS più recente, incluse nuove funzionalità e miglioramenti recenti di cui gli agenti potrebbero non essere a conoscenza.

Parametri:

- query (obbligatorio): la tua domanda o query di ricerca specifica relativa alla documentazione, alle funzionalità o alle migliori pratiche EKS.
- limite (opzionale): numero massimo di risultati della documentazione da restituire (1-10). Impostazione predefinita: 5.

search_eks_troubleshooting_guide

Cerca nella Guida alla risoluzione dei problemi EKS le informazioni sulla risoluzione dei problemi basate su una query. Aiuta a identificare i problemi più comuni e fornisce soluzioni. step-by-step

Parametri:

- interrogazione (obbligatoria): la tua domanda o descrizione specifica del problema relativo alla risoluzione dei problemi EKS.

describe_eks_resource

Recupera informazioni dettagliate su una specifica risorsa del cluster EKS, tra cui configurazione, stato e metadati.

Parametri:

- cluster_name (richiesto): nome del cluster EKS (richiesto per le risorse con ambito cluster).
- resource_type (richiesto): il tipo di risorsa EKS da descrivere. Valori validi:
- accessentry (richiede cluster_name e resource_name come PrincipalARN)
- addon (richiede cluster_name e resource_name come nome dell'addon)
- cluster (richiede cluster_name), nodegroup (richiede cluster_name e resource_name come nome del nodegroup).
- resource_name (opzionale): nome della risorsa specifica da descrivere (obbligatorio per la maggior parte dei tipi di risorse).

list_eks_resources

Elenca le risorse EKS di un tipo specifico, restituendo un riepilogo di tutte le risorse del tipo specificato accessibili.

Parametri:

- resource_type (richiesto): il tipo di risorsa EKS da elencare. Valori validi:
- accessentry (richiede cluster_name)
- addon (richiede cluster_name)
- cluster (non sono richiesti parametri aggiuntivi)
- nodegroup (richiede cluster_name).
- cluster_name (opzionale): nome del cluster EKS (richiesto per le risorse con ambito cluster).

get_eks_insights

Recupera le informazioni e i consigli del cluster EKS per l'ottimizzazione. Fornisce informazioni utili per la sicurezza, le prestazioni e l'ottimizzazione dei costi sulla base delle AWS migliori pratiche e dell'analisi dei cluster.

Parametri:

- cluster_name (richiesto): nome del cluster EKS.
- categoria (opzionale): categoria opzionale in base alla quale filtrare gli approfondimenti (ad esempio, «MISCONFIGURATION» o «UPGRADE_READINESS»).
- insight_id (opzionale): ID opzionale di un'analisi specifica per cui ottenere informazioni dettagliate.
- next_token (opzionale): token opzionale per l'impaginazione per ottenere il prossimo set di risultati.

get_eks_vpc_config

Recupera la configurazione VPC per un cluster EKS, incluse sottoreti, tabelle di routing e connettività di rete.

Parametri:

- cluster_name (richiesto): nome del cluster EKS per cui ottenere la configurazione VPC.
- vpc_id (opzionale): ID del VPC specifico da interrogare (opzionale, utilizzerà il VPC del cluster se non specificato).

get_k8s_events

Recupera gli eventi Kubernetes relativi a risorse specifiche per la risoluzione dei problemi e il monitoraggio.

Parametri:

- cluster_name (richiesto): nome del cluster EKS in cui si trova la risorsa.
- kind (richiesto): tipo di oggetto coinvolto (ad es. «Pod», «Deployment», «Service»). Deve corrispondere esattamente al tipo di risorsa.
- name (obbligatorio): nome dell'oggetto coinvolto per il quale ottenere gli eventi.
- namespace (opzionale): namespace dell'oggetto coinvolto. Necessario per risorse con namespace (come Pods, Deployments). Non richiesto per le risorse con ambito cluster (come Nodes,). PersistentVolumes

get_pod_logs

Recupera i log dai pod in un cluster EKS con opzioni di filtro.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS su cui è in esecuzione il pod.
- namespace (obbligatorio): lo spazio dei nomi Kubernetes in cui si trova il pod.
- pod_name (richiesto): nome del pod da cui recuperare i log.
- container_name (opzionale): nome del contenitore specifico da cui ottenere i log. Obbligatorio solo se il pod contiene più contenitori.
- limit_bytes (opzionale): numero massimo di byte da restituire. Impostazione predefinita: 10 KB (10240 byte).
- precedente (opzionale): restituisce i registri dei contenitori terminati in precedenza (il valore predefinito è false). Utile per ottenere i log dei pod che si stanno riavviando.
- since_seconds (opzionale): restituisce solo i log più recenti di così tanti secondi. Utile per ottenere i log recenti senza recuperare l'intera cronologia.
- tail_lines (opzionale): numero di righe da restituire dalla fine dei log. Valore predefinito: 100.

list_api_versions

Elenca tutte le versioni dell'API disponibili nel cluster Kubernetes specificato.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS.

list_k8s_resources

Elenca le risorse Kubernetes di un tipo specifico in un cluster EKS.

Parametri:

- cluster_name (richiesto): nome del cluster EKS in cui si trovano le risorse.
- kind (richiesto): tipo di risorse Kubernetes da elencare (ad esempio, 'Pod', 'Service', 'Deployment'). Usa lo `list_api_versions` strumento per trovare i tipi di risorse disponibili.
- api_version (richiesto): versione API delle risorse Kubernetes (ad esempio, 'v1', '1'). apps/v1', 'networking.k8s.io/v Utilizza lo strumento per trovare le versioni API `list_api_versions` disponibili.
- field_selector (opzionale): selettore di campo per filtrare le risorse (ad esempio, 'metadata.name=my-pod, status.phase=running'). Usa la stessa sintassi del flag --field-selector di kubectl.

- label_selector (opzionale): selettore di etichette per filtrare le risorse (ad esempio, 'app=nginx, tier=frontend'). Usa la stessa sintassi del flag di `--selector` kubectl.
- namespace (opzionale): namespace delle risorse Kubernetes da elencare. Se non vengono fornite, le risorse verranno elencate in tutti i namespace (per le risorse con namespace).

read_k8s_resource

Recupera informazioni dettagliate su una risorsa Kubernetes specifica in un cluster EKS.

Parametri:

- api_version (richiesto): versione API della risorsa Kubernetes (ad esempio, «v1"," 1"). apps/v1", "networking.k8s.io/v
- cluster_name (richiesto): nome del cluster EKS in cui si trova la risorsa.
- kind (richiesto): tipo di risorsa Kubernetes (ad esempio, «Pod», «Service», «Deployment»).
- name (richiesto): nome della risorsa Kubernetes da leggere.
- namespace (opzionale): namespace della risorsa Kubernetes. Obbligatorio per le risorse con namespace. Non richiesto per le risorse con ambito cluster (come Nodes,). PersistentVolumes

generate_app_manifest

Genera manifesti di distribuzione e servizio di Kubernetes standardizzati per applicazioni containerizzate.

Parametri:

- app_name (richiesto): nome dell'applicazione. Utilizzato per i nomi di distribuzione e servizio e per le etichette.
- image_uri (richiesto): URI completo dell'immagine ECR con tag (ad esempio, 123456789012.dkr.ecr.region.amazonaws.com/repo:tag). È necessario includere il percorso e il tag completi del repository.
- load_balancer_scheme (opzionale): schema di bilanciamento del carico. AWS Valori validi:
- «interno» (solo VPC privato)
- «rivolto a Internet» (accesso pubblico).
- Predefinito: «interno».

- cpu (opzionale): richiesta CPU per ogni contenitore (ad esempio, «100m» per 0,1 core CPU, «500m» per mezzo core). Impostazione predefinita: «100m».
- memoria (opzionale): richiesta di memoria per ogni contenitore (ad esempio, «128Mi» per 128 MiB, «1Gi» per 1 GiB). Predefinito: «128Mi»
- namespace (opzionale): namespace Kubernetes in cui distribuire l'applicazione. Predefinito: «default».
- port (opzionale): porta contenitore su cui l'applicazione è in ascolto. Impostazione predefinita: 80
- repliche (opzionale): numero di repliche da distribuire. Impostazione predefinita: 2

get_cloudwatch_logs

Interroga CloudWatch i log con filtraggio in base ai parametri di input e supporto per i gruppi di log standard utilizzati per l'osservabilità dei cluster EKS.

Parametri:

- cluster_name (richiesto): nome del cluster EKS in cui si trova la risorsa. Utilizzato per costruire il nome del gruppo di log CloudWatch .
- resource_type (richiesto): tipo di risorsa per la ricerca nei log. Valori validi: +"pod», «node», «container», «cluster». Ciò determina il modo in cui i log vengono filtrati.
- log_type (obbligatorio): tipo di registro da interrogare. Valori validi:
- «application»: log Container/application
- «host»: log di sistema a livello di nodo
- «performance»: registri delle metriche delle prestazioni
- «control-plane»: registri del piano di controllo EKS
- "your-log-group-name«: Fornisci direttamente un nome personalizzato per CloudWatch il gruppo di log.
- resource_name (opzionale): nome della risorsa da cercare nei messaggi di registro (ad esempio, nome del pod, nome del nodo, nome del contenitore). Utilizzato per filtrare i log per la risorsa specifica.
- minuti (opzionale): numero di minuti per cercare i log. Impostazione predefinita: 15. Ignorato se viene fornito start_time. Usa valori più piccoli per i numeri recenti, valori più grandi per l'analisi storica.

- start_time (opzionale): ora di inizio in formato ISO (ad esempio, «2023-01-01T 00:00:00 Z»). Se fornito, sostituisce il parametro minutes.
- end_time (opzionale): ora di fine in formato ISO (ad esempio, «2023-01-01T 01:00:00 Z»). Se non viene fornita, l'impostazione predefinita è l'ora corrente.
- fields (opzionale): campi personalizzati da includere nei risultati della query (il valore predefinito è "@timestamp, @message «). Usa la sintassi del CloudWatch campo Logs Insights.
- filter_pattern (opzionale): modello di filtro CloudWatch Logs aggiuntivo da applicare. Utilizza la sintassi di CloudWatch Logs Insights (ad esempio, «ERROR», «field=value»).
- limite (opzionale): numero massimo di voci di registro da restituire. Utilizza valori più bassi (10-50) per query più veloci, valori più alti (100-1000) per risultati più completi. I valori più alti possono inficiare le prestazioni.

get_cloudwatch_metrics

Recupera CloudWatch metriche e punti dati per il monitoraggio del cluster EKS e l'analisi delle prestazioni. Gestisce metriche di Container Insights, metriche personalizzate e periodi di tempo e dimensioni configurabili.

Parametri:

- cluster_name (obbligatorio): nome del cluster EKS per cui ottenere le metriche.
- dimensions (obbligatorio): dimensioni da utilizzare per la query CloudWatch metrica come stringa JSON. Deve includere dimensioni appropriate per il tipo di risorsa e la metrica (ad esempio, '{» ClusterName «: «my-cluster», "«: PodName «my-pod», «Namespace»: «default"}').
- metric_name (richiesto): nome della metrica da recuperare. Esempi comuni:
- `cpu_usage_total`: utilizzo totale della CPU
- `memory_rss`: utilizzo della memoria Resident Set Size
- `network_rx_bytes`: byte di rete ricevuti
- `network_tx_bytes`: byte di rete trasmessi
- namespace (obbligatorio): spazio dei CloudWatch nomi in cui è memorizzata la metrica. Valori comuni:
- `"ContainerInsights"`: Per le metriche dei container
- `"AWS/EC2"`: EC2 Ad esempio, le metriche
- `"AWS/EKS"`: Per le metriche del piano di controllo EKS

- minuti (opzionale): numero di minuti per esaminare le metriche. Impostazione predefinita: 15. Ignorato se viene fornito start_time.
- start_time (opzionale): ora di inizio in formato ISO (ad esempio, «2023-01-01T 00:00:00 Z»). Se fornito, sostituisce il parametro minutes.
- end_time (opzionale): ora di fine in formato ISO (ad esempio, «2023-01-01T 01:00:00 Z»). Se non viene fornita, l'impostazione predefinita è l'ora corrente.
- limite (opzionale): numero massimo di punti dati da restituire. Valori più alti (100-1000) forniscono dati più granulari ma possono influire sulle prestazioni. Valore predefinito: 50.
- periodo (opzionale): periodo in secondi per i punti dati metrici. Impostazione predefinita: 60 (1 minuto). Valori inferiori (1-60) forniscono una risoluzione più elevata, ma potrebbero essere meno disponibili.
- stat (opzionale): statistica da utilizzare per l'aggregazione delle metriche. Impostazione predefinita: «Average». Valori validi:
- `Average`: Valore medio durante il periodo
- `Sum:`Valore totale durante il periodo
- `Maximum`: Valore più alto del periodo
- `Minimum`: valore più basso del periodo
- `SampleCount`: Numero di campioni durante il periodo.

get_eks_metrics_guidance

Ottieni indicazioni sulle CloudWatch metriche per tipi di risorse specifici nei cluster EKS. Utile per l'agente per determinare le dimensioni corrette da utilizzare con lo strumento get_cloudwatch_metrics.

Parametri:

- resource_type (richiesto): tipo di risorsa per cui ottenere le metriche (cluster, node, pod, namespace, service).

get_policies_for_role

Recupera tutte le policy associate a un ruolo IAM specificato, incluse le policy di assunzione del ruolo, le policy gestite e le policy in linea.

Parametri:

- role_name (richiesto): nome del ruolo IAM per cui ottenere le policy. Il ruolo deve esistere nel tuo account. AWS

Strumenti di accesso completo (scrittura)

Questa sezione descrive gli strumenti di sola lettura disponibili per il server MCP EKS. Tieni presente che (a partire da «oggi») tutte le operazioni di scrittura dell'API Kubernetes possono accedere solo a:

- Cluster pubblici (=true) endpointPublicAccess

manage_k8s_resource

Gestisce una singola risorsa Kubernetes con operazioni di scrittura (creazione, aggiornamento, patch o eliminazione).

Parametri:

- operazione (richiesta): operazione da eseguire sulla risorsa. Valori validi:
- `create:` Crea una nuova risorsa
- `replace`: Sostituisce una risorsa esistente
- `patch`: aggiorna campi specifici di una risorsa esistente
- `delete`: Elimina una risorsa esistente
- Nota: usa read_k8s_resource per leggere le risorse e list_k8s_resources per elencare più risorse.
- cluster_name (richiesto): nome del cluster EKS in cui si trova o verrà creata la risorsa.
- kind (richiesto): tipo di risorsa Kubernetes (ad es. «Pod», «Service», «Deployment»).
- api_version (richiesto): versione API della risorsa Kubernetes (ad esempio, «v1"," 1"). apps/v1", "networking.k8s.io/v
- body (opzionale): definizione della risorsa come stringa JSON. Necessario per le operazioni di creazione, sostituzione e patch. Per creare e sostituire, questa dovrebbe essere una definizione completa della risorsa. Per quanto riguarda la patch, dovrebbe contenere solo i campi da aggiornare.
- name (opzionale): nome della risorsa Kubernetes. Obbligatorio per tutte le operazioni tranne la creazione (dove può essere specificato nel corpo).

- namespace (opzionale): namespace della risorsa Kubernetes. Obbligatorio per le risorse con namespace. Non richiesto per le risorse con ambito cluster (come Nodes,). PersistentVolumes

apply_yaml

Applica i manifesti YAML di Kubernetes a un cluster EKS.

Parametri:

- cluster_name (richiesto): nome del cluster EKS in cui verranno create o aggiornate le risorse.
- namespace (richiesto): namespace Kubernetes a cui applicare le risorse. Verrà utilizzato per le risorse con spaziatura dei nomi che non specificano uno spazio dei nomi.
- yaml_content (richiesto): contenuto YAML da applicare al cluster. Può contenere più documenti separati da '---'.
- force (opzionale): indica se aggiornare le risorse se già esistono (simile a kubectl apply). Imposta su false per creare solo nuove risorse.

manage_eks_stacks

Gestisce gli CloudFormation stack EKS con operazioni per la generazione di modelli, la distribuzione, la descrizione e l'eliminazione dei cluster EKS e della relativa infrastruttura sottostante. Il completamento della creazione del cluster richiede in genere 15-20 minuti. Per le operazioni di distribuzione ed eliminazione, lo stack deve essere stato creato da questo strumento (ad esempio contrassegnato con CreatedBy =EksMcpServer).

Parametri:

- cluster_name (richiesto): nome del cluster EKS (per le operazioni di generazione, distribuzione, descrizione ed eliminazione). Questo nome verrà utilizzato per derivare il nome dello CloudFormation stack e verrà incorporato nelle risorse del cluster.
- operazione (richiesta): operazione da eseguire. Valori validi:
- `generate`: genera un CloudFormation modello
- `deploy`: Implementa uno CloudFormation stack (richiesto) `template_content`
- `describe`: Describe/read uno CloudFormation stack (sola lettura)
- `delete`: Elimina una pila CloudFormation

- template_content (opzionale): contenuto del CloudFormation modello (per le operazioni di distribuzione). Questo dovrebbe essere il contenuto completo del modello YAML o JSON. Supporta sia risorse singole che contenuti YAML multi-documento separati da '---'.

add_inline_policy

Aggiunge una nuova policy in linea a un ruolo IAM.

Parametri:

- autorizzazioni (obbligatorie): autorizzazioni da includere nella policy come stringhe JSON che rappresentano le dichiarazioni delle policy IAM. Può essere una singola stringa JSON o un array di stringhe JSON.
- policy_name (obbligatorio): nome della politica in linea da creare. Deve essere univoco all'interno del ruolo.
- role_name (obbligatorio): nome del ruolo IAM a cui aggiungere la policy. Il ruolo deve esistere.

Usa Amazon Q Developer sulla console Amazon EKS

Amazon Elastic Kubernetes Service (EKS) si integra con Amazon Q per fornire la risoluzione dei problemi basata sull'intelligenza artificiale direttamente nella console di gestione. AWS Questa integrazione ti aiuta a indagare e risolvere rapidamente i problemi relativi a cluster, piano di controllo, nodi e carichi di lavoro con l'assistenza AI di Amazon Q.

Come funziona

La console Amazon EKS visualizza contestualmente i pulsanti Inspect with Amazon Q accanto a errori o problemi in tutta la console. Quando fai clic su questo pulsante, Amazon Q analizza automaticamente il problema e apre un pannello di chat sul lato destro della console con i risultati delle indagini, l'analisi delle cause principali e le misure di mitigazione suggerite.

L'integrazione viene visualizzata nelle seguenti posizioni all'interno della console Amazon EKS:

- Stato del cluster: esamina i problemi di integrità del cluster e i messaggi di stato nella scheda Stato del cluster nella dashboard di osservabilità
- Piano di controllo: risolvi gli errori e gli avvisi dei componenti del piano di controllo nel pannello di controllo - Monitoraggio del piano di controllo nella dashboard di osservabilità

- Informazioni sull'aggiornamento: analizza i potenziali ostacoli agli aggiornamenti e i problemi di compatibilità nella dashboard di Upgrade insights in observability
- Integrità dei nodi: analizza i problemi a livello di nodo che influiscono sulla capacità del cluster nella dashboard di osservabilità dei problemi di integrità dei nodi
- Carichi di lavoro: analizza gli eventi di Kubernetes sui pod che indicano guasti o problemi

Utilizzo di Amazon Q per la risoluzione dei problemi

Per esaminare un problema con Amazon Q

1. Aprire la Console Amazon EKS.
2. Scegli il nome del cluster da esaminare.
3. Quando visualizzi un messaggio di errore o un indicatore di problema, cerca il pulsante Ispeziona con Amazon Q. Il pulsante appare contestualmente accanto al problema o nella visualizzazione dei dettagli sullo stato dell'errore.
4. Scegli Inspect with Amazon Q.
5. Amazon Q analizza automaticamente il problema e visualizza l'analisi in un pannello di chat sul lato destro della console.
6. Esamina i risultati dell'indagine, inclusa l'analisi delle cause principali e le misure di mitigazione suggerite.
7. Puoi continuare la conversazione ponendo ad Amazon Q domande di follow-up sul problema.

Nota L'integrazione con Amazon Q viene visualizzata solo quando si verifica un errore, un avviso o un problema che richiede un'indagine. Non viene visualizzata quando le risorse sono integre.

Considerazioni

Considera quanto segue quando utilizzi Amazon Q con Amazon EKS:

- Operazioni di sola lettura: l'integrazione con Amazon Q esegue solo operazioni di lettura sulle risorse del cluster. Non apporta alcuna modifica o scrittura alla configurazione o ai carichi di lavoro del cluster.
- Elaborazione interregionale: Amazon Q può elaborare dati in diverse AWS regioni per fornire analisi basate sull'intelligenza artificiale. Per ulteriori informazioni sull'elaborazione tra regioni, consulta Elaborazione tra regioni nella Amazon Q Developer User Guide.

- AWS Solo console di gestione: questa integrazione è disponibile solo tramite la console di AWS gestione. Non è disponibile tramite la AWS CLI o l'infrastruttura come strumenti di codice. AWS APIs

Ulteriori informazioni

Per ulteriori informazioni sull'uso di Amazon Q, consulta Chatta con Amazon Q nella Amazon Q Developer User Guide.

Controllo delle versioni su Amazon EKS

Questa sezione è progettata per fornire un supporto nell'ambito del funzionamento del controllo delle versioni Kubernetes all'interno di Elastic Kubernetes Service (EKS) di Amazon. Sono riportati i dettagli sulle versioni supportate, sulle nostre policy obsolete e di supporto, sui processi di aggiornamento e sulle considerazioni chiave per la gestione delle versioni del cluster.

Argomenti

- Comprendere il ciclo di vita delle versioni di Kubernetes su EKS
- Visualizzazione delle versioni della piattaforma Amazon EKS per ciascuna versione Kubernetes

Comprendere il ciclo di vita delle versioni di Kubernetes su EKS

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

Kubernetes è in rapida evoluzione, con nuovi aggiornamenti di progettazione, funzionalità e correzioni di bug. La community rilascia nuove versioni di Kubernetes secondarie (ad esempio, `1.33`) in media una volta ogni quattro mesi. Amazon EKS segue il ciclo di rilascio e deprecazione originario per le versioni secondarie. Man mano che nuove versioni di Kubernetes diventano disponibili in Amazon EKS, si consiglia di aggiornare tempestivamente i cluster in modo da usare la versione più recente disponibile.

Una versione secondaria è supportata come standard in Amazon EKS per i primi 14 mesi dopo il rilascio. Una volta superata la data di fine del supporto standard, la versione passa al supporto esteso per i 12 mesi successivi. Il supporto esteso consente di mantenere una versione Kubernetes specifica di più a lungo a un costo aggiuntivo per ora del cluster. Se non hai aggiornato il cluster prima della fine del periodo di supporto esteso, il cluster viene aggiornato automaticamente alla versione estesa più vecchia attualmente supportata.

Il supporto esteso è abilitato per impostazione predefinita. Per disabilitarlo, consulta Disable EKS extended support.

Ti consigliamo di creare il tuo cluster con l'ultima versione di Kubernetes disponibile supportata da Amazon EKS. Se la propria applicazione richiede una versione specifica di Kubernetes, è possibile

selezionare versioni precedenti. Puoi creare nuovi cluster Amazon EKS su qualsiasi versione offerta con supporto standard o esteso.

Versioni disponibili con supporto standard

Le seguenti versioni di Kubernetes sono attualmente disponibili nel supporto standard di Amazon EKS:

- `1.35`
- `1.34`
- `1.33`
- `1.32`

Per le modifiche importanti di cui tenere conto per ogni versione del supporto standard, consulta Kubernetes versions standard support.

Versioni disponibili con supporto esteso

Le seguenti versioni di Kubernetes sono attualmente disponibili nel supporto esteso di Amazon EKS:

- `1.31`
- `1.30`
- `1.29`

Per le modifiche importanti di cui tenere conto per ogni versione del supporto esteso, consulta Kubernetes versions extended support.

Calendario di rilascio di Amazon EKS Kubernetes

La tabella seguente mostra le date importanti di rilascio e supporto da considerare per ciascuna versione di Kubernetes. La fatturazione per il supporto esteso parte all'inizio del giorno in cui la versione raggiunge la fine del supporto standard, nel fuso orario UTC+0. Le date nella tabella seguente utilizzano il fuso orario UTC+0.

> **Note**
>
> Le date con solo un mese e un anno sono approssimative e vengono aggiornate con una data esatta quando nota.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, puoi iscriverti al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/versioning/kubernetes-versions.adoc.atom
```

Versione di Kubernetes	Versione upstream	Rilascio Amazon EKS	Data di fine del supporto standard	Data di fine del supporto esteso
1.35	17 dicembre 2025	27 gennaio 2026	27 marzo 2027	27 marzo 2028
1.34	27 agosto 2025	2 ottobre 2025	2 dicembre 2026	2 dicembre 2027
1.33	23 aprile 2025	29 maggio 2025	29 luglio 2026	29 luglio 2027
1.32	11 dicembre 2024	23 gennaio 2025	23 marzo 2026	23 marzo 2027
1.31	13 agosto 2024	26 settembre 2024	26 novembre 2025	26 novembre 2026
1.30	17 aprile 2024	23 maggio 2024	23 luglio 2025	23 luglio 2026
1.29	13 dicembre 2023	23 gennaio 2024	23 marzo 2025	23 marzo 2026

Ottieni informazioni sulla versione con la AWS CLI

Puoi utilizzare la AWS CLI per ottenere informazioni sulle versioni di Kubernetes disponibili su EKS, come la data di fine di Standard Support.

Per recuperare informazioni sulle versioni di Kubernetes disponibili su EKS utilizzando la CLI AWS

1. Apri il terminale.
2. Assicurati di avere la AWS CLI installata e configurata. Per ulteriori informazioni, vedi Installing or updating to the latest version of the CLI.
3. Esegui il comando seguente:

```
aws eks describe-cluster-versions
```

4. Il comando restituirà un output JSON con dettagli sulle versioni del cluster disponibili. Ecco un esempio dell'output:

```
{
    "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"
        }
    ]
}
```

L'output fornisce le seguenti informazioni per ogni versione del cluster:

- `clusterVersion`: la versione Kubernetes del cluster EKS
- `clusterType`: il tipo di cluster (ad esempio, "eks")
- `defaultPlatformVersion`: la versione predefinita della piattaforma EKS
- `defaultVersion`: se si tratta della versione predefinita
- `releaseDate`: la data in cui è stata rilasciata questa versione
- `endOfStandardSupportDate`: la data di fine del supporto standard

- `endOfExtendedSupportDate`: la data di fine del supporto esteso
- `status`: lo stato del supporto corrente della versione, ad esempio `STANDARD_SUPPORT` o `EXTENDED_SUPPORT`
- `kubernetesPatchVersion`: la versione specifica della patch Kubernetes

Versione Amazon EKS FAQs

Quante versioni di Kubernetes sono disponibili nel supporto standard?

Amazon EKS, in linea con il supporto della community Kubernetes per le versioni di Kubernetes, si impegna a offrire il supporto per almeno tre versioni di Kubernetes in qualunque momento. L'annuncio della data di fine supporto standard di una qualsiasi versione di Kubernetes secondaria sarà comunicato almeno 60 giorni prima. A causa del processo di qualifica e rilascio di Amazon EKS per le nuove versioni di Kubernetes, la data di fine supporto standard di una versione di Kubernetes in Amazon EKS avverrà in corrispondenza o dopo la data in cui il progetto Kubernetes smetterà di supportare la versione a monte.

Per quanto tempo Kubernetes riceve il supporto standard da parte di Amazon EKS?

Una versione di Kubernetes è supportata per 14 mesi dopo la prima disponibilità su Amazon EKS. Ciò è confermato anche se Kubernetes a monte non supporterà più una versione disponibile in Amazon EKS. Viene eseguito il backporting delle patch di sicurezza applicabili alle versioni di Kubernetes che sono supportate in Amazon EKS.

Riceverò un avviso quando il supporto standard per una versione Kubernetes su Amazon EKS è terminato?

Sì. Se in alcuni cluster del tuo account è in esecuzione la versione prossima alla fine del supporto, Amazon EKS invia un avviso tramite AWS Health Dashboard circa 12 mesi dopo il rilascio della versione di Kubernetes su Amazon EKS. L'avviso include la data di fine supporto, che è successiva di almeno 60 giorni alla data dell'invio dell'avviso.

Quali funzionalità di Kubernetes sono supportate da Amazon EKS?

Amazon EKS supporta tutte le funzionalità di disponibilità generale (GA) dell'API di Kubernetes. Per impostazione predefinita, le nuove versioni beta APIs non sono abilitate nei cluster. Tuttavia, la versione beta esistente in precedenza APIs e le nuove versioni beta esistenti APIs continuano ad essere abilitate per impostazione predefinita. Le funzionalità alfa non sono supportate.

I gruppi di nodi gestiti di Amazon EKS vengono aggiornati automaticamente insieme alla versione del piano di controllo del cluster?

No, un gruppo di nodi gestiti crea istanze Amazon EC2 nel tuo account. Queste istanze non vengono aggiornate in automatico quando l'utente o Amazon EKS aggiorna il piano di controllo. Per ulteriori informazioni, consulta the section called "Aggiornamento". Si consiglia di mantenere la stessa versione di Kubernetes sul piano di controllo e sui nodi.

I gruppi di nodi autogestiti vengono aggiornati automaticamente insieme alla versione del piano di controllo del cluster?

No, un gruppo di nodi autogestito include le istanze di Amazon EC2 nel tuo account. Queste istanze non vengono aggiornate in automatico quando tu o Amazon EKS aggiornate la versione del piano di controllo per tuo conto. Nella console, un gruppo di nodi autogestito non riceve alcuna indicazione di aggiornamento. È possibile visualizzare la versione di `kubelet` installata su un nodo selezionando il nodo nella finestra di dialogo Nodi nella scheda Panoramica del cluster per determinare quali nodi devono essere aggiornati. È necessario aggiornare manualmente i nodi. Per ulteriori informazioni, consulta the section called "Metodi di aggiornamento".

Il progetto Kubernetes verifica la compatibilità tra il piano di controllo e i nodi per un massimo di due versioni secondarie. Ad esempio, i nodi `1.30` continuano a funzionare se orchestrati da un piano di controllo `1.33`. Tuttavia, non è consigliabile l'esecuzione di un cluster con nodi aggiornati in modo persistente a tre versioni secondarie precedenti rispetto al piano di controllo. Per ulteriori informazioni, consulta Policy per la versione Kubernetes e il supporto Skew della versione nella documentazione di Kubernetes. Si consiglia di mantenere la stessa versione di Kubernetes sul piano di controllo e sui nodi.

I pod in esecuzione su Fargate vengono aggiornati automaticamente con un aggiornamento automatico della versione del piano di controllo cluster?

No. Consigliamo di eseguire i pod Fargate come parte di un controller di replica, così come avviene in un'implementazione di Kubernetes. Quindi esegui un riavvio continuo di tutti i pod Fargate. La nuova versione del pod Fargate viene implementata con una versione `kubelet` che corrisponde alla versione aggiornata del piano di controllo del cluster. Per ulteriori informazioni, consulta l'argomento relativo alla funzionalità Deployments nella documentazione Kubernetes.

> **Important**
>
> Se aggiorni il piano di controllo, dovrai comunque aggiornare personalmente i nodi Fargate. Per aggiornare i nodi Fargate, elimina il pod Fargate rappresentato dal nodo e

implementalo nuovamente. Il nuovo pod viene implementato con una versione `kubelet` che corrisponde alla versione del cluster.

Quali versioni di Kubernetes sono supportate per i nodi ibridi?

Amazon EKS Hybrid Nodes supporta le stesse versioni di Kubernetes dei cluster Amazon EKS con altri tipi di calcolo dei nodi, incluso il supporto della versione Kubernetes standard ed esteso. I nodi ibridi non vengono aggiornati automaticamente quando si aggiorna la versione del piano di controllo, perché ne è responsabile l'utente. Per ulteriori informazioni, consulta the section called "Aggiornamento dei nodi ibridi".

Supporto esteso per Amazon EKS FAQs

La terminologia del supporto standard e del supporto esteso è nuova per me. Cosa significano questi termini?

Il supporto standard per una versione di Kubernetes in Amazon EKS inizia quando viene rilasciata una versione Kubernetes su Amazon EKS e termina 14 mesi dopo la data di rilascio. Il supporto esteso per una versione di Kubernetes inizierà immediatamente dopo la fine del supporto standard e terminerà trascorsi i 12 mesi successivi. Ad esempio, il supporto standard per la versione `1.23` in Amazon EKS termina l'11 ottobre 2023. Il supporto esteso per la versione `1.23` è iniziato il 12 ottobre 2023 ed è terminato l'11 ottobre 2024.

Cosa devo fare per ottenere un supporto esteso per i cluster Amazon EKS?

Dovrai abilitare il supporto esteso (vedi EKS extended support) per il tuo cluster modificando la policy di aggiornamento del cluster in EXTENDED. Per impostazione predefinita su tutti i cluster nuovi ed esistenti, la policy di aggiornamento è impostata su EXTENDED, se non diversamente specificato. Consulta Cluster upgrade policy per visualizzare la policy di aggiornamento del cluster. Il supporto standard inizia quando viene rilasciata una versione di Kubernetes su Amazon EKS e termina 14 mesi dopo la data di rilascio. Il supporto esteso per una versione di Kubernetes inizierà immediatamente dopo la fine del supporto standard e terminerà trascorsi i 12 mesi successivi.

Per quali versioni Kubernetes posso ottenere il supporto esteso?

Puoi eseguire i cluster su qualsiasi versione per un massimo di 12 mesi dopo la fine del supporto standard per quella versione. Ciò significa che ogni versione sarà supportata per 26 mesi in Amazon EKS (14 mesi di supporto standard più 12 mesi di supporto esteso).

Cosa succede se non voglio utilizzare il supporto esteso?

Se non desideri ricevere automaticamente il supporto esteso, puoi aggiornare il cluster a una versione di Kubernetes che includa il supporto standard di Amazon EKS. Per disabilitare il supporto esteso, consulta Disable EKS extended support. Nota: se disabiliti il supporto esteso, il tuo cluster sarà automaticamente aggiornato al termine del supporto standard.

Cosa succederà alla fine dei 12 mesi di supporto esteso?

I cluster in esecuzione su una versione di Kubernetes che ha completato il ciclo di vita di 26 mesi (14 mesi di supporto standard più 12 mesi di supporto esteso) verranno aggiornati automaticamente alla versione successiva. L'aggiornamento automatico include solo il piano di controllo Kubernetes. Se disponi di nodi della modalità automatica di EKS, potrebbero aggiornarsi automaticamente. I nodi autogestiti e i gruppi di nodi gestiti EKS rimarranno nella versione precedente.

Allo scadere della data di fine di supporto, non è più possibile creare nuovi cluster Amazon EKS con la versione non supportata. I piani di controllo esistenti vengono aggiornati in automatico da Amazon EKS alla prima versione supportata attraverso un processo di implementazione graduale dopo la data di fine supporto. Dopo l'aggiornamento automatico del piano di controllo, è necessario aggiornare manualmente i componenti aggiuntivi del cluster e i nodi Amazon EC2. Per ulteriori informazioni, consulta the section called "Aggiornamento della versione di Kubernetes".

Allo scadere della data di fine supporto esteso, quando avverrà esattamente l'aggiornamento automatico del piano di controllo?

Amazon EKS non è in grado di fornire periodi di tempo specifici. Gli aggiornamenti automatici possono avvenire in qualsiasi momento dopo la data di fine supporto esteso. Non riceverai alcuna notifica prima dell'aggiornamento. Consigliamo di aggiornare in modo proattivo il piano di controllo senza fare affidamento sul processo di aggiornamento automatico di Amazon EKS. Per ulteriori informazioni, consulta the section called "Aggiornamento della versione di Kubernetes".

Posso lasciare il mio piano di controllo su una versione Kubernetes a tempo indeterminato?

No. La sicurezza del cloud AWS è la massima priorità. Dopo un certo periodo (di solito 1 anno), la community di Kubernetes interrompe il rilascio di patch per vulnerabilità ed esposizioni (CVE) comuni e sconsiglia il loro invio per le versioni non supportate. Ciò significa che le vulnerabilità specifiche di una versione precedente di Kubernetes potrebbero non essere segnalate, lasciando i cluster esposti senza preavviso e senza opzioni di correzione in caso di vulnerabilità. Pertanto, Amazon EKS non consente ai piani di controllo di rimanere a una versione non più coperta dal supporto esteso.

È previsto un costo aggiuntivo per ottenere un supporto esteso?

Sì, sono previsti costi aggiuntivi per i cluster Amazon EKS in esecuzione con il supporto esteso. Per i dettagli sui prezzi, consulta i prezzi del supporto esteso di Amazon EKS per la versione Kubernetes sul AWS blog o sulla nostra pagina dei prezzi.

Cosa è incluso nel supporto esteso?

I cluster Amazon EKS in supporto esteso ricevono patch di sicurezza continue per il piano di controllo di Kubernetes. Inoltre, Amazon EKS rilascerà patch per Amazon VPC CNI, `kube-proxy` e componenti aggiuntivi per le versioni di supporto esteso su CoreDNS. Amazon EKS rilascerà anche patch per AWS Amazon EKS già pubblicato e ottimizzato per AMIs Amazon Linux, Bottlerocket e Windows, oltre ai nodi Amazon EKS Fargate per tali versioni. Tutti i cluster di Extended Support continueranno ad avere accesso al supporto tecnico da AWS.

Esistono limitazioni alle patch per i componenti non Kubernetes inclusi nel supporto esteso?

Sebbene Extended Support copra tutti i componenti specifici di Kubernetes AWS, fornirà sempre supporto solo per AWS Amazon EKS pubblicato e ottimizzato per AMIs Amazon Linux, Bottlerocket e Windows. Ciò significa che potresti avere componenti più recenti (come sistema operativo o kernel) sulla tua AMI ottimizzata per Amazon EKS durante l'utilizzo del supporto esteso. Ad esempio, una volta che Amazon Linux 2 raggiungerà la fine del suo ciclo di vita nel 2025, Amazon Linux ottimizzato per Amazon EKS AMIs verrà creato utilizzando un sistema operativo Amazon Linux più recente. Amazon EKS annuncerà e documenterà importanti discrepanze nel ciclo di vita del supporto, come questa, per ogni versione di Kubernetes.

Posso creare nuovi cluster utilizzando una versione con supporto esteso?

Sì.

Dai un'occhiata alle note di rilascio per le versioni Kubernetes sul supporto standard

> **Tip**
>
> Registrati ai prossimi workshop Amazon EKS.

Questo argomento fornisce importanti modifiche di cui tenere conto per ogni versione Kubernetes del supporto standard. Durante l'aggiornamento, esamina attentamente le modifiche apportate tra la vecchia e la nuova versione del cluster.

Kubernetes 1.35

Kubernetes `1.35` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.35`, consulta l'official release announcement.

Important

- Supporto per Cgroup v1 rimosso: Kubernetes 1.35 depreca il supporto cgroup v1, il che significa che il kubelet si rifiuterà di avviarsi per impostazione predefinita sui nodi che utilizzano cgroup v1.
  - AL2023: utilizza cgroup AL2023 v2 per impostazione predefinita e si allinea al comportamento upstream di Kubernetes.
    - Azione richiesta: i clienti che hanno configurato manualmente l'uso di cgroup v1 devono AL2023 migrare a cgroups v2 o impostarli manualmente nella configurazione kubelet. `failCgroupV1: false`
  - Bottlerocket: Bottlerocket 1.35 utilizza cgroup v2 per impostazione predefinita, tuttavia imposta la configurazione kubelet, mantenendo la compatibilità con le versioni precedenti. `failCgroupV1: false`
  - Fargate: Fargate continua a utilizzare cgroup v1.
- Containerd 1.x End of Support: Kubernetes 1.35 è l'ultima versione che supporta containerd 1.x. È necessario passare a containerd 2.0 o versione successiva prima di eseguire l'aggiornamento alla versione successiva di Kubernetes.
- A marzo 2026, il progetto upstream Kubernetes ritirerà Ingress NGINX, un componente dell'infrastruttura fondamentale per molti ambienti Kubernetes.
  - Azione richiesta: i clienti EKS devono valutare se si affidano a Ingress NGINX e iniziare a pianificare la migrazione verso alternative come l'API Gateway o i controller Ingress di terze parti, poiché non ci saranno ulteriori versioni per correzioni di bug, patch di sicurezza o aggiornamenti dopo il ritiro. Le implementazioni esistenti continueranno a funzionare, ma rimanere con Ingress NGINX dopo il pensionamento rende l'ambiente vulnerabile ai rischi per la sicurezza, poiché nessuna delle alternative disponibili è sostitutiva diretta e richiederà tempi di pianificazione e progettazione. Per ulteriori

informazioni su questo annuncio di Kubernetes, consulta la dichiarazione ufficiale del Kubernetes Steering and Security Response Committee Ingress NGINX Retirement.

- In-Place Pod Resource Updates (Stable): In-Place Pod Resource Updates consente agli utenti di regolare le risorse della CPU e della memoria senza riavviare Pod o contenitori. In precedenza, tali modifiche richiedevano la ricreazione dei Pod, che potevano interrompere i carichi di lavoro, in particolare per le applicazioni con stato o in batch. La nuova funzionalità in loco consente una scalabilità verticale più fluida e senza interruzioni, migliora l'efficienza e può anche semplificare lo sviluppo.
  - Per ulteriori informazioni, consulta Kubernetes 1.35: In-Place Pod Resize Graduates to Stable sul blog di Kubernetes.
- PreferSameNode Distribuzione del traffico (stabile): il `trafficDistribution` campo Servizi è stato aggiornato per fornire un controllo più esplicito sul routing del traffico. È stata introdotta una nuova opzione per consentire ai servizi di assegnare una priorità rigorosa agli endpoint sul nodo locale quando disponibili, altrimenti ricorrendo agli endpoint remoti. `PreferSameNode` Questa modifica rende l'API più esplicita sulla preferenza del traffico all'interno del nodo corrente.
- StatefulSet MaxUnavailable (Beta): questa funzionalità abilita gli aggiornamenti paralleli dei Pod tramite impostazioni `maxUnavailable` (ad esempio, 3 o 10%), consentendo alle applicazioni stateful come i cluster di database di aggiornarsi fino al 60% più velocemente rispetto agli one-at-a-time aggiornamenti sequenziali, riducendo significativamente le finestre di manutenzione.
- Supporto per Windows Server 2025: EKS 1.35 aggiunge il supporto per Windows Server 2025.
- Rimozione della bandiera Kubelet: la **`--pod-infra-container-image`** bandiera è stata rimossa da Kubelet. Gli utenti AMI personalizzati devono rimuovere questo flag dalla configurazione di Kubelet prima di eseguire l'aggiornamento alla 1.35.
- Avviso di deprecazione - Modalità IPVS: la modalità IPVS in kube-proxy è obsoleta e verrà rimossa in Kubernetes 1.36.

Per il `1.35` changelog completo https://github.com/kubernetes/ kubernetes/blob/master/ CHANGELOG/CHANGELOG di Kubernetes, vedi -1.35.md

Kubernetes 1.34

Kubernetes `1.34` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.34`, consulta l'official release announcement.

Important

- Containerd è stato aggiornato alla 2.1 nella versione 1.34 per il lancio.
  - In caso di problemi dopo l'aggiornamento, consulta le note di rilascio di containerd 2.1.
- AWS non rilascia un'AMI Amazon Linux 2 ottimizzata per EKS per Kubernetes 1.34.
  - AWS ti incoraggia a migrare ad Amazon Linux 2023. Informazioni su come the section called "Esegui l'aggiornamento a AL2023".
  - Per ulteriori informazioni, consulta the section called "Deprecazione dell'AMI Amazon Linux 2".
- AppArmor è obsoleto in Kubernetes 1.34.
  - Consigliamo di passare a soluzioni alternative per la sicurezza dei container come seccomp o Pod Security Standards.
- VolumeAttributesClass (VAC) si laurea in GA con Kubernetes 1.34, passando dall'API beta () all'API stabile (). `storage.k8s.io/v1beta1` `storage.k8s.io/v1`
  - Se utilizzi il driver CSI EBS con contenitori sidecar AWS gestiti (da CSI Components nella galleria ECR), la modifica del volume continuerà a funzionare senza problemi sui cluster EKS 1.31-1.33. AWS applicherà una patch ai sidecar per supportare la versione beta di VAC APIs fino alla fine del supporto standard EKS 1.33 (29 luglio 2026).
  - Se gestisci autonomamente i contenitori sidecar CSI, potrebbe essere necessario aggiungerli a versioni sidecar precedenti su cluster precedenti alla 1.34 per mantenere la funzionalità VAC.
  - Per utilizzare le VolumeAttributesClass funzionalità GA (come il rollback delle modifiche), esegui l'aggiornamento a EKS 1.34 o versione successiva.
- JWT Signer for Service Account Tokens esterno viene promosso alla versione Beta. Quando si utilizzano firmatari esterni, il flag -- service-account-extend-token -expiation non viene più rispettato completamente. Il server API impone la scadenza minima tra l'estensione desiderata (1 anno) e il limite del firmatario esterno (24 ore).
  - Ti consigliamo di utilizzare token di account di servizio vincolati, che vengono montati e ruotati automaticamente da Kubernetes.

- Dynamic Resource Allocation (DRA) Core APIs (GA): la Dynamic Resource Allocation è diventata stabile e consente una gestione efficiente dell'hardware specializzato, ad esempio GPUs attraverso

interfacce di allocazione standardizzate, semplificando la gestione delle risorse per gli acceleratori hardware e migliorando l'utilizzo di risorse specializzate.

- Projected ServiceAccount Tokens for Kubelet (Beta): questo miglioramento migliora la sicurezza utilizzando credenziali di breve durata per l'estrazione delle immagini dei container anziché segreti di lunga durata, riducendo il rischio di esposizione delle credenziali e rafforzando il livello di sicurezza generale dei cluster.
- Richieste e limiti di risorse a livello di POD (Beta): questa funzionalità semplifica la gestione delle risorse consentendo pool di risorse condivisi per pod multi-container, consentendo un'allocazione e un utilizzo più efficienti delle risorse per applicazioni complesse con più contenitori.
- Mutable CSI Node Allocatable Count (Beta): Il `MutableCSINodeAllocatableCount` feature gate è abilitato di default in EKS 1.34, rendendo mutabile l'attributo CSINode max attachable volume count e introducendo un meccanismo per aggiornarlo dinamicamente in base alla configurazione utente a livello di driver CSI. Questi aggiornamenti possono essere attivati da intervalli periodici o dal rilevamento degli errori, migliorando l'affidabilità dello stateful pod scheduling risolvendo le discrepanze tra la capacità degli allegati riportata e quella effettiva sui nodi.
  - Per ulteriori informazioni, consulta Kubernetes v1.34: Mutable CSI Node Allocatable Count sul blog di Kubernetes.
- Avviso di deprecazione - configurazione del driver cgroup: la configurazione manuale del driver cgroup è obsoleta a favore del rilevamento automatico.
  - Impatto sul cliente: se attualmente imposti il `--cgroup-driver` flag manualmente nella configurazione di kubelet, dovresti prepararti a rimuovere questa configurazione.
  - Azione richiesta: pianifica di aggiornare gli script di bootstrap dei nodi e le configurazioni AMI personalizzate per rimuovere le impostazioni manuali dei driver cgroup prima che la funzionalità venga rimossa in una futura versione di Kubernetes.
  - Per ulteriori informazioni, consulta la documentazione del driver cgroup.

Per il changelog completo di Kubernetes`1.34`, consulta -1.34.md https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.33

Kubernetes `1.33` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.33`, consulta l'official release announcement.

> **Important**
>
> - L'API Kubernetes beta dell'allocazione dinamica delle risorse è abilitata.
>   - Questa API beta migliora l'esperienza di pianificazione e monitoraggio dei carichi di lavoro che richiedono risorse come. GPUs
>   - L'API beta è definita dalla community Kubernetes e potrebbe cambiare nelle versioni future.
>   - Esamina attentamente le fasi relative alle funzionalità nella documentazione di Kubernetes per comprendere le implicazioni dell'utilizzo della versione beta. APIs
> - AWS non rilascia un'AMI Amazon Linux 2 ottimizzata per EKS per Kubernetes 1.33.
>   - AWS ti incoraggia a migrare ad Amazon Linux 2023. Informazioni su come the section called "Esegui l'aggiornamento a AL2023".
>   - Per ulteriori informazioni, consulta the section called "Deprecazione dell'AMI Amazon Linux 2".

- Ridimensionamento locale delle risorse pod (Beta): il ridimensionamento locale delle risorse è stato promosso alla versione beta e consente aggiornamenti dinamici della CPU e delle risorse di memoria per i Pod esistenti senza riavvii, permettendo il dimensionamento verticale dei carichi di lavoro stateful con zero tempi di inattività e regolazioni delle risorse senza interruzioni in base ai modelli di traffico.
- Container sidecar ora stabili: i container sidecar sono diventati stabili, implementando i sidecar come container init speciali con `restartPolicy: Always` che iniziano prima dei container applicativi, funzionano per tutto il ciclo di vita dei pod e supportano le sonde per la segnalazione dello stato operativo.
  - Per ulteriori informazioni, consulta Sidecar Containers nella documentazione Kubernetes.
- Deprecazione dell'API degli endpoint: l'API Endpoints è ora ufficialmente obsoleta e restituirà avvisi quando vi si accede: migra invece carichi di lavoro e script per utilizzare l' EndpointSlices API, che supporta funzionalità moderne come la rete dual-stack e ne gestisce più di uno per servizio. EndpointSlices
  - Per ulteriori informazioni, consulta Kubernetes v1.33: Continuing the transition from Endpoints to on the Kubernetes Blog. EndpointSlice
- Supporto di Elastic Fabric Adapter: il gruppo di sicurezza predefinito per i cluster Amazon EKS ora supporta il traffico Elastic Fabric Adapter (EFA). Il gruppo di sicurezza predefinito ha una nuova

regola in uscita che consente il traffico di EFA con la destinazione dello stesso gruppo di sicurezza. Questo consente il traffico EFA all'interno del cluster.

- Per ulteriori informazioni, consulta Elastic Fabric Adapter per AI/ML carichi di lavoro HPC su Amazon EC2 nella Amazon Elastic Compute Cloud User Guide.

Per il changelog completo di `1.33` Kubernetes, consulta -1.33.md https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.32

Kubernetes `1.32` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.32`, consulta l'official release announcement.

> **Important**
>
> - La versione `flowcontrol.apiserver.k8s.io/v1beta3` API di FlowSchema ed PriorityLevelConfiguration è stata rimossa nella versione attuale. `1.32` Se li utilizzi APIs, devi aggiornare le configurazioni per utilizzare l'ultima versione supportata prima dell'aggiornamento.
> - ServiceAccount `metadata.annotations[kubernetes.io/enforce-mountable-secrets]`è stata obsoleta nella versione `1.32` e verrà rimossa in una futura versione secondaria di Kubernetes. Si consiglia di utilizzare namespace separati per isolare l'accesso ai segreti montati.
> - La versione Kubernetes `1.32` è l'ultima versione per la quale Amazon EKS rilascerà Amazon Linux 2 (). AL2 AMIs A `1.33` partire dalla versione, Amazon EKS continuerà a rilasciare Amazon Linux 2023 (AL2023) e basato su Bottlerocket. AMIs

- La funzionalità Memory Manager è passata allo stato di disponibilità generale (GA) nella versione Kubernetes `1.32`. Questo miglioramento offre un'allocazione della memoria più efficiente e prevedibile per le applicazioni containerizzate, particolarmente utile per carichi di lavoro con requisiti di memoria specifici.
- PersistentVolumeClaims (PVCs) StatefulSets finora creati includono funzionalità di pulizia automatica. Quando non PVCs sono più necessari, verranno eliminati automaticamente mantenendo la persistenza dei dati durante StatefulSet gli aggiornamenti e le operazioni di

manutenzione dei nodi. Questa funzionalità semplifica la gestione dello storage e aiuta a evitare che rimangano orfani PVCs nel cluster.

- È stata introdotta la funzionalità Custom Resource Field Selector, che consente agli sviluppatori di aggiungere selettori di campo alle risorse personalizzate. Questa funzionalità offre le stesse funzionalità di filtraggio disponibili per gli oggetti Kubernetes integrati nelle risorse personalizzate, consentendo un filtraggio delle risorse più preciso ed efficiente e promuovendo migliori pratiche di progettazione delle API.

Per il changelog completo di `1.32` Kubernetes, vedi -1.32.md https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG

Modifiche dell'autenticazione anonima

A partire da Amazon EKS `1.32`, l'autenticazione anonima è limitata ai seguenti endpoint di controllo dell'integrità del server API:

- `/healthz`
- `/livez`
- `/readyz`

Le richieste a qualsiasi altro endpoint che utilizza l'utente `system:unauthenticated` riceveranno una risposta HTTP `401 Unauthorized`. Questo miglioramento della sicurezza aiuta a prevenire l'accesso involontario al cluster che potrebbe verificarsi a causa di policy RBAC non configurate correttamente.

> **Note**
>
> Il ruolo RBAC `public-info-viewer` continua a valere per gli endpoint di controllo dell'integrità sopra elencati.

Deprecazione dell'AMI Amazon Linux 2

La versione Kubernetes `1.32` è l'ultima versione rilasciata da Amazon EKS. AL2 AMIs Dalla versione `1.33` in poi, Amazon EKS continuerà a essere rilasciato AL2023 e sarà basato su Bottlerocket. AMIs Per ulteriori informazioni, consulta the section called "AL2 Deprecazione AMI".

Dai un'occhiata alle note di rilascio per le versioni Kubernetes sul supporto esteso

Amazon EKS supporta le versioni di Kubernetes più a lungo di quelle upstream, con supporto standard per le versioni secondarie di Kubernetes per 14 mesi dal momento in cui vengono rilasciate in Amazon EKS e supporto esteso per le versioni secondarie di Kubernetes per altri 12 mesi di supporto (26 mesi totali per versione).

Questo argomento fornisce importanti modifiche di cui tenere conto per ogni versione di Kubernetes del supporto esteso. Durante l'aggiornamento, esamina attentamente le modifiche apportate tra la vecchia e la nuova versione del cluster.

Kubernetes 1.31

Kubernetes `1.31` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.31`, consulta l'official release announcement.

> **Important**
>
> - Il flag kubelet `--keep-terminated-pod-volumes` obsoleto dal 2017 è stato rimosso come parte del rilascio della versione `1.31`. Questa modifica influisce sul modo in cui i volumi dei pod terminati vengono gestiti da kubelet. Se si utilizza questo flag nelle configurazioni dei nodi, è necessario aggiornare gli script di bootstrap e avviare i modelli per rimuoverlo prima dell'aggiornamento.

- La funzionalità di controllo `VolumeAttributesClass` beta e la risorsa API sono abilitati nella versione Amazon EKS `1.31`. Questa funzionalità consente agli operatori del cluster di modificare le proprietà mutabili di Persistent Volumes (PVs) gestiti da driver CSI compatibili, incluso il driver CSI di Amazon EBS. Per sfruttare questa funzionalità, assicurati che il driver CSI supporti la funzionalità `VolumeAttributesClass` (per il driver CSI di Amazon EBS, esegui l'upgrade alla versione `1.35.0` o successiva per abilitarla automaticamente). Sarai in grado di creare `VolumeAttributesClass` oggetti per definire gli attributi di volume desiderati, come il tipo di volume e la velocità effettiva, e associarli ai tuoi Persistent Volume Claims (). PVCs Per ulteriori informazioni, consulta official Kubernetes documentation e la documentazione del driver CSI.
  - Per ulteriori informazioni sul Driver CSI di Amazon EBS, consulta the section called "Amazon EBS".

- Il supporto di Kubernetes per AppArmorè diventato stabile ed è ora generalmente disponibile per l'uso pubblico. Questa funzionalità consente di proteggere i contenitori AppArmor impostando il `appArmorProfile.type` campo all'interno del contenitore. `securityContext` Prima della versione Kubernetes`1.30`, AppArmor era controllato da annotazioni. A partire dalla versione `1.30`, è controllato tramite campi. Per sfruttare questa funzionalità, consigliamo di abbandonare le annotazioni e utilizzare il campo `appArmorProfile.type` per garantire la compatibilità dei carichi di lavoro.
- La funzionalità del tempo di transizione dell' PersistentVolume ultima fase è diventata stabile ed è ora generalmente disponibile per l'uso pubblico nella versione Kubernetes. `1.31` Questa funzionalità introduce un nuovo campo, in `.status.lastTransitionTime` PersistentVolumeStatus, che fornisce un timestamp di quando un' PersistentVolume ultima è passata a una fase diversa. Questo miglioramento consente un migliore monitoraggio e gestione di PersistentVolumes, in particolare in scenari in cui è importante comprendere il ciclo di vita dei volumi.

Per il changelog completo di `1.31` Kubernetes, vedi -1.31.md https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.30

Kubernetes `1.30` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.30`, consulta l'official release announcement.

- A partire dalla versione Amazon EKS `1.30` o successiva, qualsiasi gruppo di nodi gestiti appena creato utilizzerà automaticamente Amazon Linux 2023 (AL2023) come sistema operativo del nodo. Per ulteriori informazioni sulla specificazione del sistema operativo per un gruppo di nodi gestito, consulta the section called "Crea".
- Con Amazon EKS `1.30`, viene aggiunta l'etichetta `topology.k8s.aws/zone-id` ai nodi worker. Puoi utilizzare Availability Zone IDs (AZ IDs) per determinare la posizione delle risorse in un account rispetto alle risorse di un altro account. Per ulteriori informazioni, consulta la sezione Zona di disponibilità IDs per le AWS risorse disponibili nella AWS RAM User Guide.
- A partire da `1.30`, Amazon EKS non include più l'annotazione `default` sulla risorsa gp2 `StorageClass` applicata ai cluster appena creati. Questo non ha alcun impatto se si utilizza il nome come riferimento a questa classe di archiviazione. Devi agire se facevi affidamento sulla presenza di un `StorageClass` predefinito nel cluster. Dovresti fare riferimento a `StorageClass` con il nome gp2. In alternativa, è possibile distribuire la classe di archiviazione predefinita

consigliata da Amazon EBS impostando il parametro `defaultStorageClass.enabled` su true durante l'installazione della versione `1.31.0` o successiva di `aws-ebs-csi-driver add-on`.

- La policy IAM minima richiesta per il ruolo IAM del cluster Amazon EKS è cambiata. L'azione `ec2:DescribeAvailabilityZones` è obbligatoria. Per ulteriori informazioni, consulta the section called "Ruolo IAM del cluster".

Per il `1.30` changelog completo di Kubernetes, vedi -1.30.md. https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.29

Kubernetes `1.29` è ora disponibile in Amazon EKS. Per ulteriori informazioni su Kubernetes `1.29`, consulta l'official release announcement.

> **Important**
>
> - La versione API `flowcontrol.apiserver.k8s.io/v1beta2` obsoleta di `FlowSchema` e `PriorityLevelConfiguration` non è più disponibile nella versione `1.29` di Kubernetes. Se disponi di manifesti o di un software client che utilizza il gruppo di API beta obsoleto, devi modificarli prima di eseguire l'aggiornamento alla versione `1.29`.

- Ora il campo `.status.kubeProxyVersion` per gli oggetti del nodo è obsoleto e il progetto Kubernetes propone di rimuovere quel campo in future versioni. Il campo obsoleto non è preciso ed è stato sempre gestito da `kubelet`, che in realtà non conosce la versione `kube-proxy` e nemmeno se `kube-proxy` è in esecuzione. Se hai utilizzato questo campo nel software client, non farlo più: le informazioni non sono affidabili e il campo è ora obsoleto.
- In Kubernetes `1.29`, per ridurre la potenziale superficie di attacco, la funzionalità `LegacyServiceAccountTokenCleanUp` etichetta i token basati sui segreti generati automaticamente come non validi se non sono stati utilizzati per un lungo periodo (1 anno per impostazione predefinita) e li rimuove automaticamente se non si tenta di utilizzarli per un lungo periodo dopo essere stati contrassegnati come non validi (1 anno aggiuntivo per impostazione predefinita). Per identificare tali token, puoi eseguire:

```
kubectl get cm kube-apiserver-legacy-service-account-token-tracking -n kube-system
```

Per il changelog completo di 1.29 Kubernetes, vedi -1.29.md# 1280. https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG changelog-since-v

Visualizzazione del periodo di supporto del cluster corrente

La sezione relativa al periodo di supporto per i cluster della console AWS indica se per il cluster è attualmente su un supporto standard o esteso. Se il periodo di supporto del cluster è Supporto esteso, ti sarà addebitato il costo del supporto esteso EKS.

Per ulteriori informazioni sul supporto standard ed esteso, consulta Comprendere il ciclo di vita delle versioni di Kubernetes su EKS.

1. Vai alla pagina Cluster nella sezione EKS della console AWS. Verifica che la console sia impostata sulla stessa regione AWS del cluster che desideri esaminare.
2. Esamina la colonna Periodo di supporto. Se il valore è Supporto standard fino a..., al momento non ti è addebitato alcun costo per il supporto esteso. Rientri nel periodo di supporto standard. Se il valore è Supporto esteso..., al momento a questo cluster è addebitato il costo del supporto esteso.

> **Note**
>
> Il periodo di supporto non può essere recuperato con AWS API o CLI.

Visualizzazione della corrente policy di aggiornamento del cluster

La policy di aggiornamento del cluster determina cosa succede al cluster quando esce dal periodo di supporto standard. Se la policy di aggiornamento è `EXTENDED`, il cluster non sarà aggiornato automaticamente ed entrerà il supporto esteso. Se la tua policy di aggiornamento è `STANDARD`, sarà aggiornata automaticamente.

La policy di controllo della versione Kubernetes di Amazon EKS ti consente di scegliere la fine del comportamento di supporto standard per i tuoi cluster EKS. Con questi controlli è possibile decidere quali cluster debbano essere sottoposti al supporto esteso e quali aggiornati automaticamente alla fine del supporto standard per una versione di Kubernetes.

Una versione secondaria è supportata come standard in Amazon EKS per i primi 14 mesi dopo il rilascio. Una volta superata la data di fine del supporto standard, la versione passa al supporto esteso per i 12 mesi successivi. Il supporto esteso consente di mantenere una versione Kubernetes specifica

di più a lungo a un costo aggiuntivo per ora del cluster. È possibile abilitare o disabilitare il supporto esteso per un cluster EKS. Se disabiliti il supporto esteso, AWS sarà automaticamente aggiornato alla versione successiva di Kubernetes al termine del supporto standard. Se abiliti il supporto esteso, è possibile mantenere la versione corrente a un costo aggiuntivo per un periodo di tempo limitato. Pianifica di aggiornare regolarmente il tuo cluster Kubernetes, anche se utilizzi il supporto esteso.

È possibile impostare la policy della versione sia per i cluster nuovi che per quelli esistenti, utilizzando la proprietà `supportType`. Esistono due opzioni che possono essere utilizzate per impostare la policy di supporto della versione:

- **`STANDARD`** : il tuo cluster EKS è idoneo all'aggiornamento automatico al termine del supporto standard. Questa impostazione non comporta costi di assistenza estesa, ma il cluster EKS eseguirà automaticamente l'aggiornamento alla successiva versione di Kubernetes supportata con supporto standard.
- **`EXTENDED`** : il tuo cluster EKS entrerà in supporto esteso una volta che la versione Kubernetes raggiungerà la fine del supporto standard. Con questa impostazione ti saranno addebitati costi di supporto estesi. È possibile aggiornare il tuo cluster a una versione Kubernetes standard supportata per evitare di incorrere in costi di supporto esteso. I cluster che utilizzano il supporto esteso saranno idonei all'aggiornamento automatico al termine del supporto esteso.

Il supporto esteso è abilitato per impostazione predefinita per i nuovi cluster e per i cluster esistenti. È possibile verificare se il supporto esteso è abilitato per un cluster in Console di gestione AWS o utilizzando AWS CLI.

> **Important**
>
> Se desideri che il cluster mantenga la versione corrente di Kubernetes per sfruttare il periodo di supporto esteso, è necessario abilitare il supporto esteso nella policy di aggiornamento prima della fine del periodo di supporto standard.

È possibile impostare la policy di supporto della versione per i cluster solo mentre è in esecuzione sulla versione Kubernetes nel supporto standard. Una volta che la versione entrerà nel supporto esteso, non potrai modificare questa impostazione finché non utilizzerai una versione con supporto standard.

Ad esempio, se hai impostato la policy di supporto della versione come `standard`, non potrai modificare questa impostazione dopo che la versione di Kubernetes in esecuzione sul cluster

avrà raggiunto la fine del supporto standard. Se hai impostato la policy di supporto della versione come `extended`, non potrai modificare questa impostazione dopo che la versione di Kubernetes in esecuzione sul cluster avrà raggiunto la fine del supporto standard. Per modificare l'impostazione della politica di supporto delle versioni, il cluster deve essere in esecuzione su una versione di Kubernetes supportata in modo standard.

Visualizzare la policy di aggiornamento del cluster (Console AWS)

1. Vai alla pagina Cluster nella sezione EKS della console AWS. Verifica che la console sia impostata sulla stessa regione AWS del cluster che desideri esaminare.
2. Esamina la colonna Policy di aggiornamento. Se il valore è Supporto standard, il cluster non accederà al supporto esteso. Se il valore è Supporto esteso, il cluster entrerà nel supporto esteso.

Visualizzare la policy di aggiornamento del cluster (AWS CLI)

1. Verifica che AWS CLI sia installata e di aver effettuato l'accesso. Informazioni su come aggiornare e installare AWS CLI.
2. Determina il nome del cluster EKS. Imposta CLI sulla stessa regione AWS del cluster EKS.
3. Esegui il comando seguente:

```
aws eks describe-cluster \
--name <cluster-name> \
--query "cluster.upgradePolicy.supportType"
```

4. Se il valore è `STANDARD`, il tuo cluster non accederà al supporto esteso. Se il valore è `EXTENDED`, il cluster accederà al supporto esteso.

Aggiungere flessibilità per pianificare gli aggiornamenti delle versioni di Kubernetes abilitando il supporto esteso EKS

Questo argomento descrive come configurare la policy di aggiornamento di un cluster EKS per abilitare il supporto esteso. La policy di aggiornamento di un cluster EKS determina cosa succede quando un cluster raggiunge la fine del periodo di supporto standard. Se una policy di aggiornamento ha il supporto esteso abilitato, alla fine del periodo di supporto standard il cluster entrerà nel periodo di supporto esteso. Alla fine del periodo di supporto standard, il cluster non verrà aggiornato automaticamente.

Vengono addebitati costi più alti per i cluster che si trovano effettivamente nel periodo di supporto esteso. Se la policy di aggiornamento per è solamente configurata per abilitare il supporto esteso, ma il cluster associato si trova nel periodo di supporto standard, vengono addebitati i costi standard.

Se si crea un cluster nella Console AWS, la policy di aggiornamento associata sarà configurata per disabilitare il supporto esteso. Se si crea un cluster con un'altra modalità, la policy di aggiornamento associata sarà configurata per abilitare il supporto esteso. Ad esempio, per i cluster creati con l'API AWS è abilitato il supporto esteso.

Per ulteriori informazioni sulle policy di aggiornamento, consultare la pagina Cluster upgrade policy.

> **Important**
>
> Se si desidera che il cluster mantenga la versione corrente di Kubernetes per sfruttare il periodo di supporto esteso, è necessario abilitare il supporto esteso nella policy di aggiornamento prima della fine del periodo di supporto standard.
> Se il supporto esteso non viene abilitato, il cluster verrà aggiornato automaticamente.

Abilitare il supporto esteso EKS (Console AWS)

1. Accedere al cluster EKS nella Console AWS. Selezionare la scheda Panoramica nella pagina Informazioni sul cluster.
2. Nella sezione Impostazioni della versione di Kubernetes, selezionare Gestisci.
3. Selezionare Supporto esteso, poi Salva modifiche.

Abilitare il supporto esteso EKS (AWS CLI)

1. Verificare che AWS CLI sia installata e di aver effettuato l'accesso. Informazioni su come aggiornare e installare AWS CLI.
2. Determinare il nome del cluster EKS.
3. Esegui il comando seguente:

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=EXTENDED
```

Prevenire l'aumento del costo dei cluster disabilitando il supporto esteso EKS

Questo argomento descrive come configurare la policy di aggiornamento di un cluster EKS per disabilitare il supporto esteso. La policy di aggiornamento di un cluster EKS determina cosa succede quando un cluster raggiunge la fine del periodo di supporto standard. Se la policy di aggiornamento di un cluster ha il supporto esteso disabilitato, verrà automaticamente aggiornato alla versione successiva di Kubernetes.

Per ulteriori informazioni sulle policy di aggiornamento, consultare la pagina Cluster upgrade policy.

> **Important**
>
> Non è possibile disabilitare il supporto esteso una volta che il cluster vi è entrato. È possibile disabilitare il supporto esteso per i cluster solo durante il supporto standard.
> AWS consiglia di aggiornare il cluster a una versione valida per il periodo di supporto standard.

Disabilitare il supporto esteso EKS (Console AWS)

1. Accedere al cluster EKS nella Console AWS. Selezionare la scheda Panoramica nella pagina Informazioni sul cluster.
2. Nella sezione Impostazione della versione di Kubernetes, selezionare Gestisci.
3. Selezionare Supporto standard, poi Salva modifiche.

Disabilitare il supporto esteso EKS (AWS CLI)

1. Verificare che AWS CLI sia installata e di aver effettuato l'accesso. Informazioni su come aggiornare e installare AWS CLI.
2. Determinare il nome del cluster EKS.
3. Esegui il comando seguente:

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=STANDARD
```

Visualizzazione delle versioni della piattaforma Amazon EKS per ciascuna versione Kubernetes

Le versioni della piattaforma Amazon EKS rappresentano le funzionalità del piano di controllo del cluster. Ad esempio quali flag del server API di Kubernetes sono abilitati e l'attuale versione della patch di Kubernetes. Ogni versione secondaria Kubernetes dispone di una o più versioni della piattaforma Amazon EKS associate. Le versioni della piattaforma per le diverse versioni secondarie di Kubernetes sono indipendenti. Puoi recuperare la versione corrente della piattaforma del cluster utilizzando la AWS Console di gestione AWS CLI o. Se disponi di un cluster locale su AWS Outposts, consulta the section called "Versioni della piattaforma EKS" invece di questo argomento.

Quando una nuova versione secondaria di Kubernetes è disponibile in Amazon EKS, ad esempio 1.33, la versione iniziale della piattaforma Amazon EKS per quella versione secondaria di Kubernetes parte da. `eks.1` Tuttavia, Amazon EKS rilascia periodicamente nuove versioni della piattaforma per abilitare nuove impostazioni del piano di controllo Kubernetes e fornire soluzioni per problemi relativi alla sicurezza.

Quando nuove versioni della piattaforma Amazon EKS diventano disponibili per una versione secondaria:

- Il numero di versione della piattaforma Amazon EKS viene incrementato (`eks.<n+1>`).
- Amazon EKS aggiorna automaticamente tutti i cluster esistenti alla versione della piattaforma Amazon EKS più recente per la versione secondaria Kubernetes corrispondente. Aggiornamenti automatici di versioni della piattaforma Amazon EKS esistenti vengono implementati in modo incrementale. Il processo di implementazione potrebbe richiedere alcuni minuti. Se le caratteristiche della versione della piattaforma Amazon EKS sono richieste immediatamente, occorre creare un nuovo cluster.

  Se il tuo cluster ha più di due versioni di piattaforma rispetto alla versione corrente della piattaforma, è possibile che Amazon EKS non sia riuscito ad aggiornare automaticamente il cluster. Per i dettagli su cosa potrebbe causare questo problema, consulta the section called "La versione della piattaforma Amazon EKS è più avanti di due versioni rispetto all'attuale versione della piattaforma".
- Amazon EKS potrebbe pubblicare un nuovo nodo AMI con una versione patch corrispondente. Tuttavia, tutte le versioni patch sono compatibili tra il piano di controllo EKS e il nodo AMIs per una determinata versione secondaria di Kubernetes.

Nuove versioni della piattaforma Amazon EKS non introducono modifiche di conflitto né causano interruzioni del servizio.

I cluster vengono sempre creati con la versione della piattaforma Amazon EKS disponibile più recente (`eks.<n>`) per la versione Kubernetes specificata. Se si aggiorna il cluster a una nuova versione secondaria Kubernetes, il cluster riceverà la versione della piattaforma Amazon EKS corrente per la versione secondaria Kubernetes a cui è stata aggiornata.

Le versioni della piattaforma Amazon EKS correnti e recenti sono descritte nelle tabelle qui di seguito.

> **Note**
>
> AWS ha recentemente disabilitato alcune versioni della piattaforma pubblicate nel giugno 2024. Le versioni della piattaforma presentavano problemi di stabilità. Non è richiesta alcuna azione.

Per ricevere notifiche di tutte le modifiche al file di origine di questa pagina di documentazione specifica, è possibile iscriversi al seguente URL con un lettore RSS:

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/versioning/platform-versions.adoc.atom
```

Versione **`1.35`** di Kubernetes

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma `1.35`: `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
`1.35.0`	`eks.3`	Rilascio iniziale della versione `1.35` di Kubernetes per	27 gennaio 2026

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
		EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.35".	

Versione **1.34** di Kubernetes

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma `1.34`: `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
`1.34.3`	`eks.13`	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 gennaio 2026
`1.34.3`	`eks.12`	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. La 1.34 `eks.11` è stata scartata internamente e non è mai stata rilasciata.	16 gennaio 2026
`1.34.2`	`eks.10`	Nuova versione della piattaforma con	19 dicembre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
		correzioni di sicurezza e miglioramenti.	
1.34.1	eks.9	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 novembre 2025
1.34.1	eks.8	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 novembre 2025
1.34.1	eks.7	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	6 novembre 2025
1.34.1	eks.6	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	30 ottobre 2025
1.34.1	eks.5	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.34.1	eks.4	Rilascio iniziale della versione 1.34 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.34".	2 ottobre 2025

Versione **1.33** di Kubernetes

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma `1.33`: `NodeRestriction`, `ExtendedResourceToleration`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `TaintNodesByCondition`, `PodSecurity`, `Priority`, `DefaultTolerationSeconds`, `DefaultStorageClass`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `RuntimeClass`, `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`, `ResourceQuota`.

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
`1.33.7`	`eks.27`	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 gennaio 2026
`1.33.7`	`eks.26`	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. La 1.33 `eks.25` è stata scartata internamente e non è mai stata rilasciata.	16 gennaio 2026
`1.33.6`	`eks.24`	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	19 dicembre 2025
`1.33.5`	`eks.23`	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 novembre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.33.5	eks.22	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 novembre 2025
1.33.5	eks.21	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	6 novembre 2025
1.33.5	eks.20	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	30 ottobre 2025
1.33.5	eks.19	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 ottobre 2025
1.33.5	eks.18	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.33.5	eks.17	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.33.5	eks.16	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 ottobre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.33.4	eks.15	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 settembre 2025
1.33.4	eks.14	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 settembre 2025
1.33.4	eks.13	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	16 settembre 2025
1.33.4	eks.12	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	5 settembre 2025
1.33.3	eks.11	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	25 agosto 2025
1.33.3	eks.10	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 agosto 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.33.2	eks.9	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.7 1.33 e eks.8 1.33 sono state scartate internamente e non sono mai state rilasciate.	30 luglio 2025
1.33.1	eks.6	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	26 giugno 2025
1.33.1	eks.5	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	11 giugno 2025
1.33.1	eks.4	Rilascio iniziale della versione 1.33 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.33".	28 maggio 2025

Versione **1.32** di Kubernetes

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.32: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection,

PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota.

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.32.11	eks.34	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 gennaio 2026
1.32.11	eks.33	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. 1.32 eks.32 è stata scartata internamente e non è mai stata rilasciata.	16 gennaio 2026
1.32.10	eks.31	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	19 dicembre 2025
1.32.9	eks.30	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 novembre 2025
1.32.9	eks.29	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 novembre 2025
1.32.9	eks.28	Nuova versione della piattaforma con	6 novembre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
		correzioni di sicurezza e miglioramenti.	
1.32.9	eks.27	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	30 ottobre 2025
1.32.9	eks.26	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 ottobre 2025
1.32.9	eks.25	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.32.9	eks.24	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.32.9	eks.23	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 ottobre 2025
1.32.8	eks.22	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 settembre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.32.8	eks.21	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 settembre 2025
1.32.8	eks.20	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	16 settembre 2025
1.32.8	eks.19	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	5 settembre 2025
1.32.7	eks.18	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	25 agosto 2025
1.32.7	eks.17	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 agosto 2025
1.32.6	eks.16	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.14 1.32 e eks.15 1.32 sono state scartate internamente e non sono mai state rilasciate.	30 luglio 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.32.5	eks.13	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	26 giugno 2025
1.32.5	eks.12	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	11 giugno 2025
1.32.5	eks.11	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	30 maggio 2025
1.32.3	eks.10	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	16 maggio 2025
1.32.3	eks.9	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 aprile 2025
1.32.3	eks.8	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 aprile 2025
1.32.3	eks.7	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 aprile 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.32.3	eks.6	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 aprile 2025
1.32.2	eks.5	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 marzo 2025
1.32.2	eks.4	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	4 marzo 2025
1.32.1	eks.3	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	24 febbraio 2025
1.32.0	eks.2	Rilascio iniziale della versione 1.32 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.32".	Gennaio 2025

Versione **1.31** di Kubernetes

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.31: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection,

PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota, ObjectCount.

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.31.14	eks.50	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 gennaio 2026
1.31.14	eks.49	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. La 1.31 eks.48 è stata scartata internamente e non è mai stata rilasciata.	16 gennaio 2026
1.31.14	eks.47	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	19 dicembre 2025
1.31.13	eks.46	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 novembre 2025
1.31.13	eks.45	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 novembre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.31.13	eks.44	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	6 novembre 2025
1.31.13	eks.43	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	30 ottobre 2025
1.31.13	eks.42	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 ottobre 2025
1.31.13	eks.41	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.31.13	eks.40	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.31.13	eks.39	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 ottobre 2025
1.31.12	eks.38	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 settembre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.31.12	eks.37	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 settembre 2025
1.31.12	eks.36	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	16 settembre 2025
1.31.12	eks.35	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	5 settembre 2025
1.31.11	eks.34	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	25 agosto 2025
1.31.11	eks.33	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 agosto 2025
1.31.10	eks.32	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.30 1.31 e eks.31 1.31 sono state scartate internamente e non sono mai state rilasciate.	30 luglio 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.31.9	eks.29	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	26 giugno 2025
1.31.9	eks.28	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	11 giugno 2025
1.31.9	eks.27	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	30 maggio 2025
1.31.7	eks.26	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	16 maggio 2025
1.31.7	eks.25	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 aprile 2025
1.31.7	eks.24	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 aprile 2025
1.31.7	eks.23	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 aprile 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.31.7	eks.22	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 aprile 2025
1.31.6	eks.21	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 marzo 2025
1.31.6	eks.20	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	4 marzo 2025
1.31.5	eks.19	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	24 febbraio 2025
1.31.5	eks.18	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	24 febbraio 2025
1.31.4	eks.17	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 gennaio 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.31.2	eks.12	Nuova versione della piattaforma con supporto per Amazon EKS Hybrid Nodes e miglioramenti per controllare l'osservabilità del piano di controllo. Scopri the section called "Nodi ibridi" e consulta Amazon EKS migliora l'osservabilità delle prestazioni, rispettivamente.	15 novembre 2024
1.31.1	eks.6	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	21 ottobre 2024
1.31.0	eks.4	Rilascio iniziale della versione 1.31 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.31".	26 settembre 2024

Versione **1.30** di Kubernetes

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.30: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority,

DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota.

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.30.14	eks.58	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 gennaio 2026
1.30.14	eks.57	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. La 1.30 eks.56 è stata scartata internamente e non è mai stata rilasciata.	16 gennaio 2026
1.30.14	eks.55	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	19 dicembre 2025
1.30.14	eks.54	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 novembre 2025
1.30.14	eks.53	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 novembre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.30.14	eks.52	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	6 novembre 2025
1.30.14	eks.51	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	30 ottobre 2025
1.30.14	eks.50	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 ottobre 2025
1.30.14	eks.49	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.30.14	eks.48	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.30.14	eks.47	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 ottobre 2025
1.30.14	eks.46	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 settembre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.30.14	eks.45	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 settembre 2025
1.30.14	eks.44	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	16 settembre 2025
1.30.14	eks.43	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	5 settembre 2025
1.30.14	eks.42	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	25 agosto 2025
1.30.14	eks.41	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 agosto 2025
1.30.14	eks.40	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.38 1.30 e eks.39 1.30 sono state scartate internamente e non sono mai state rilasciate.	30 luglio 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.30.13	eks.37	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	26 giugno 2025
1.30.13	eks.36	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	11 giugno 2025
1.30.13	eks.35	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	30 maggio 2025
1.30.11	eks.34	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	16 maggio 2025
1.30.11	eks.33	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 aprile 2025
1.30.11	eks.32	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 aprile 2025
1.30.11	eks.31	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 aprile 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.30.11	eks.30	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 aprile 2025
1.30.10	eks.29	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 marzo 2025
1.30.10	eks.28	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	4 marzo 2025
1.30.9	eks.27	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	24 febbraio 2025
1.30.8	eks.25	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 gennaio 2025
1.30.8	eks.24	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	3 gennaio 2025
1.30.7	eks.23	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	13 dicembre 2024

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.30.6	eks.22	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	13 dicembre 2024
1.30.6	eks.21	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	13 dicembre 2024
1.30.6	eks.20	Nuova versione della piattaforma con supporto per Amazon EKS Hybrid Nodes e miglioramenti per controllare l'osservabilità del piano di controllo. Scopri the section called "Nodi ibridi" e consulta Amazon EKS migliora l'osservabilità delle prestazioni, rispettivamente.	15 novembre 2024
1.30.5	eks.12	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	21 ottobre 2024
1.30.4	eks.8	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	3 settembre 2024

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.30.3	eks.7	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 agosto 2024
1.30.3	eks.6	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	9 agosto 2024
1.30.2	eks.5	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 luglio 2024
1.30.0	eks.2	Rilascio iniziale della versione 1.30 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.30".	23 maggio 2024

Versione **1.29** di Kubernetes

I seguenti controller di ammissione sono abilitati per tutte le versioni della piattaforma 1.29: NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota.

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.29.15	eks.61	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 gennaio 2026
1.29.15	eks.60	Nuova versione della piattaforma con correzioni e miglioramenti di sicurezza. La 1.29 eks.59 è stata scartata internamente e non è mai stata rilasciata.	16 gennaio 2026
1.29.15	eks.58	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	19 dicembre 2025
1.29.15	eks.57	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 novembre 2025
1.29.15	eks.56	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 novembre 2025
1.29.15	eks.55	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	6 novembre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.29.15	eks.54	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	30 ottobre 2025
1.29.15	eks.53	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 ottobre 2025
1.29.15	eks.52	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.29.15	eks.51	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	15 ottobre 2025
1.29.15	eks.50	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 ottobre 2025
1.29.15	eks.49	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 settembre 2025
1.29.15	eks.48	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 settembre 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.29.15	eks.47	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	16 settembre 2025
1.29.15	eks.46	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	5 settembre 2025
1.29.15	eks.45	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	25 agosto 2025
1.29.15	eks.44	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 agosto 2025
1.29.15	eks.43	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti. eks.41 1.29 e eks.42 1.29 sono state scartate internamente e non sono mai state rilasciate.	30 luglio 2025
1.29.15	eks.40	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	26 giugno 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.29.15	eks.39	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	11 giugno 2025
1.29.15	eks.38	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	30 maggio 2025
1.29.15	eks.37	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	16 maggio 2025
1.29.15	eks.36	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 aprile 2025
1.29.15	eks.35	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 aprile 2025
1.29.15	eks.34	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 aprile 2025
1.29.15	eks.33	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 aprile 2025

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.29.14	eks.32	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	17 marzo 2025
1.29.14	eks.31	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	4 marzo 2025
1.29.13	eks.30	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	24 febbraio 2025
1.29.13	eks.29	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	24 febbraio 2025
1.29.12	eks.28	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	20 gennaio 2025
1.29.12	eks.27	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	3 gennaio 2025
1.29.11	eks.26	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	13 dicembre 2024

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.29.10	eks.25	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	13 dicembre 2024
1.29.10	eks.24	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	13 dicembre 2024
1.29.10	eks.23	Nuova versione della piattaforma con supporto per Amazon EKS Hybrid Nodes e miglioramenti per controllare l'osservabilità del piano di controllo. Scopri the section called "Nodi ibridi" e consulta Amazon EKS migliora l'osservabilità delle prestazioni, rispettivamente.	15 novembre 2024
1.29.9	eks.17	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	21 ottobre 2024
1.29.8	eks.13	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	3 settembre 2024

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.29.7	eks.12	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	28 agosto 2024
1.29.7	eks.11	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	9 agosto 2024
1.29.6	eks.10	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	2 luglio 2024
1.29.4	eks.7	Nuova versione della piattaforma con dimensionamento automatico CoreDNS, correzioni di sicurezza e miglioramenti. Per ulteriori informazioni sul dimensionamento automatico di CoreDNS, consulta the section called "Scalare per traffico elevato".	16 maggio 2024
1.29.3	eks.6	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	18 aprile 2024

Versione di Kubernetes	Versioni della piattaforma EKS	Note di rilascio	Data di rilascio
1.29.1	eks.5	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	29 marzo 2024
1.29.1	eks.4	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	20 marzo 2024
1.29.1	eks.3	Nuova versione della piattaforma con correzioni di sicurezza e miglioramenti.	12 marzo 2024
1.29.0	eks.1	Rilascio iniziale della versione 1.29 di Kubernetes per EKS. Per ulteriori informazioni, consulta the section called "Kubernetes 1.29".	23 gennaio 2024

Ottieni versione corrente della piattaforma

1. Aprire la Console Amazon EKS.
2. Nel pannello di navigazione scegliere Cluster.
3. Nell'elenco dei cluster, scegli il Nome del cluster di cui verificare la versione della piattaforma.
4. Seleziona la scheda Panoramica.
5. La Versione della piattaforma è disponibile nella sezione Dettagli.
6. Determina il Nome del cluster di cui vuoi controllare la versione della piattaforma.
7. Esegui il comando seguente:

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

Di seguito viene riportato un output di esempio.

```
"eks.10"
```

Modificare la versione della piattaforma

Non è possibile modificare la versione della piattaforma di un cluster EKS. Quando una nuova versione della piattaforma Amazon EKS diventa disponibile per una versione Kubernetes, EKS aggiorna automaticamente tutti i cluster esistenti all'ultima versione della piattaforma di Amazon EKS per la versione Kubernetes corrispondente. Aggiornamenti automatici di versioni della piattaforma Amazon EKS esistenti vengono implementati in modo incrementale. Non è possibile utilizzare la AWS console o la CLI per modificare la versione della piattaforma.

Se aggiorni la tua versione Kubernetes, il cluster passerà alla versione della piattaforma più recente per la versione Kubernetes.

Risoluzione dei problemi con i cluster e i nodi Amazon EKS

Questo capitolo presenta alcuni errori comuni che si potrebbero verificare durante l'uso di Amazon EKS e il modo in cui gestirli. Se devi risolvere problemi in aree specifiche di Amazon EKS, consulta gli argomenti separati the section called "Risoluzione dei problemi", the section called "Risoluzione dei problemi del connettore EKS" e Risoluzione dei problemi di ADOT utilizzando i componenti aggiuntivi EKS.

Per altre informazioni sulla risoluzione dei problemi, consulta i Contenuti del Knowledge Center su Amazon Elastic Kubernetes Service su AWS re:Post.

Capacità insufficiente

Se ricevi il seguente errore mentre provi a creare un cluster Amazon EKS, significa che una delle zone di disponibilità indicate non dispone della capacità sufficiente per supportare un 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
```

Riprova a crearlo con le sottoreti del VPC del cluster ospitate nelle zone di disponibilità restituite da questo messaggio di errore.

Ci sono zone di disponibilità dove non è possibile farvi risiedere un cluster. Confronta le zone di disponibilità in cui si trovano le tue sottoreti con l'elenco delle zone di disponibilità in Requisiti e considerazioni sulle sottoreti.

Impossibile aggiungere i nodi al cluster

Ci sono diversi motivi comuni che impediscono l'aggiunta dei nodi di lavoro al cluster:

- Se i nodi sono gestiti, Amazon EKS aggiunge voci ad `aws-auth ConfigMap` quando crei il gruppo di nodi. Se la voce è stata rimossa o modificata, è necessario aggiungerla nuovamente. Per ulteriori informazioni, digita `eksctl create iamidentitymapping --help` nel terminale. Puoi verificare la versione corrente delle voci `aws-auth ConfigMap` sostituendo *`my-cluster`* nel seguente comando con il nome del tuo cluster e quindi eseguendo il comando modificato: `eksctl`

`get iamidentitymapping --cluster` *`my-cluster`* . L'ARN del ruolo specificato non può includere un percorso diverso da `/`. Ad esempio, se il nome del ruolo è `development/apps/my-role`, è necessario modificarlo in `my-role` quando si specifica l'ARN per tale ruolo. Assicurati di specificare l'ARN del ruolo IAM del nodo (non l'ARN del profilo dell'istanza).

Se i nodi sono autogestiti e non hai creato voci di accesso per l'ARN del ruolo IAM del nodo, esegui gli stessi comandi elencati per i nodi gestiti. Se hai creato una voce di accesso per l'ARN del tuo ruolo IAM del nodo, è possibile che non sia configurata correttamente nella suddetta voce di accesso. Assicurati che l'ARN del ruolo IAM del nodo (e non l'ARN del profilo dell'istanza) sia specificato come ARN principale nella tua voce `aws-auth ConfigMap` o voce di accesso. Per ulteriori informazioni sulle voci di accesso, consulta la sezione the section called "Voci di accesso".

- ClusterName nel modello del nodo AWS CloudFormation non corrisponde esattamente al nome del cluster a cui aggiungere i nodi. Se inserisci un valore non corretto in questo campo, si verificherà una configurazione errata del file `/var/lib/kubelet/kubeconfig` del nodo e i nodi non saranno aggiunti al cluster.
- Il nodo non è taggato come appartenente al cluster. Ai nodi deve essere applicato il tag seguente, dove *`my-cluster`* è sostituito con il nome del cluster.

Chiave	Valore
`kubernetes.io/cluster/` *`my-cluster`*	`owned`

- I nodi potrebbero non essere in grado di accedere al cluster utilizzando un indirizzo IP pubblico. Assicurarsi che ai nodi implementati nelle sottoreti pubbliche venga assegnato un indirizzo IP pubblico. In caso contrario, è possibile associare un indirizzo IP elastico a un nodo dopo l'avvio. Per ulteriori informazioni, consulta Associazione di un indirizzo IP elastico a un'istanza o a un'interfaccia di rete in esecuzione. Se la sottorete pubblica non è impostata per assegnare automaticamente gli indirizzi IP pubblici alle istanze implementate, è consigliabile abilitare l'impostazione. Per ulteriori informazioni, consulta Modifica dell'attributo di assegnazione degli indirizzi IPv4 pubblici della sottorete. Se il nodo viene implementato in una sottorete privata, la sottorete deve disporre di una route a un gateway NAT a cui è assegnato un indirizzo IP pubblico.
- L'endpoint AWS STS per la regione AWS in cui si implementano i nodi non è abilitato per l'account. Per abilitare la regione, consulta Attivazione e disattivazione di AWS STS in una regione AWS.
- Il nodo non dispone di una voce DNS privata, con conseguente log `kubelet` che riporta un errore `node "" not found`. Assicurati che il VPC in cui viene creato il nodo disponga di valori

impostati per `domain-name` e `domain-name-servers` come `Options` in un `DHCP options set`. I valori predefiniti sono `domain-name:<region>.compute.internal` e `domain-name-servers:AmazonProvidedDNS`. Per ulteriori informazioni, consulta Set opzioni DHCP nella Guida per l'utente di Amazon VPC.

- Se i nodi del gruppo di nodi gestiti non si connettono al cluster entro 15 minuti, sarà emesso un problema di integrità di "NoDecreationFailure" e lo stato della console sarà impostato su `Create failed`. Per le AMI Windows con tempi di avvio lenti, questo problema può essere risolto utilizzando l'avvio rapido.

Per identificare e risolvere i problemi più comuni che impediscono ai nodi worker di unirsi a un cluster, puoi utilizzare il runbook `AWSSupport-TroubleshootEKSWorkerNode`. Per ulteriori informazioni, consulta `AWSSupport-TroubleshootEKSWorkerNode` nel Riferimento dei runbook AWSAutomation Systems Manager.

Accesso negato o non autorizzato (`kubectl`)

Se ricevi uno dei seguenti errori durante l'esecuzione di comandi `kubectl`, `kubectl` non è stato configurato correttamente per Amazon EKS o le credenziali del principale IAM (ruolo o utente) in uso non sono mappate a un nome utente Kubernetes con autorizzazioni sufficienti per gli oggetti Kubernetes nel tuo cluster 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"`

Questo potrebbe essere dovuto a uno dei seguenti fattori:

- Il cluster è stato creato con le credenziali per un principale IAM e `kubectl` è configurato per utilizzare le credenziali per un altro principale IAM. Per risolvere il problema, aggiorna il file `kube config` affinché vengano utilizzate le credenziali con cui è stato creato il cluster. Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".
- Se il cluster soddisfa i requisiti minimi di piattaforma indicati nella sezione dei prerequisiti di Concedere ad utenti IAM un accesso a Kubernetes con voci di accesso EKS, non esiste una voce di accesso con il tuo principale IAM. Se esiste, non ha i nomi dei gruppi Kubernetes necessari definiti per essa, o non la policy di accesso corretta non vi è associata. Per ulteriori informazioni, consulta the section called "Voci di accesso".

- Se il tuo cluster non soddisfa i requisiti minimi di piattaforma indicati in Concedere ad utenti IAM un accesso a Kubernetes con voci di accesso EKS, non esiste una voce con il tuo principale IAM in `ConfigMap` `aws-auth`. Se esiste, non è mappata su nomi dei gruppi Kubernetes associati a `Role` o a `ClusterRole` di Kubernetes con le autorizzazioni necessarie. Per ulteriori informazioni sugli oggetti dell'autorizzazione basata sui ruoli (RBAC) di Kubernetes, consulta Utilizzare l'autorizzazione RBAC nella documentazione Kubernetes. Puoi verificare la versione corrente delle voci `aws-auth` `ConfigMap` sostituendo *`my-cluster`* nel seguente comando con il nome del tuo cluster e quindi eseguendo il comando modificato: `eksctl get iamidentitymapping --cluster` *`my-cluster`* . Se in `ConfigMap` non è presente una voce con l'ARN del tuo principale IAM, digita `eksctl create iamidentitymapping --help` nel terminale per scoprire come crearne una.

Se installi e configuri AWS CLI, è possibile configurare le credenziali IAM che utilizzi. Per ulteriori informazioni consulta Configurare AWS CLI nella Guida per l'utente dell'interfaccia a riga di comando AWS. È anche possibile configurare `kubectl` affinché utilizzi un ruolo IAM, se assumi un ruolo IAM per accedere agli oggetti Kubernetes sul cluster. Per ulteriori informazioni, consulta the section called "Accedere al cluster con kubectl".

`hostname doesn't match`

La versione Python del sistema deve essere `2.7.9` o successiva. In caso contrario, riceverai errori `hostname doesn't match` con chiamate di AWS CLI ad Amazon EKS. Per ulteriori informazioni, consulta Cosa sono gli errori "hostname non corrispondente"? nelle Domande frequenti di Python Requests.

`getsockopt: no route to host`

Docker viene eseguito nell'intervallo CIDR `172.17.0.0/16` nei cluster Amazon EKS. Consigliamo che le sottoreti VPC del cluster non si sovrappongano a questo intervallo. In caso contrario, riceverai il seguente messaggio di errore:

```
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 ricevi l'errore `Instances failed to join the Kubernetes cluster` in Console di gestione AWS, accertati che l'accesso all'endpoint privato del cluster sia abilitato o che siano stati configurati correttamente gli intervalli CIDR per l'accesso agli endpoint pubblici. Per ulteriori informazioni, consulta the section called "Accesso all'endpoint del cluster".

Codici di errore dei gruppi di nodi gestiti

Se il gruppo di nodi gestiti rileva un problema di integrità hardware, Amazon EKS restituisce un codice di errore che consente di diagnosticare il problema. Questi controlli dell'integrità non rilevano problemi software perché si basano su controlli dell'integrità di Amazon EC2. Nel seguente elenco vengono descritti i codici di errore.

AccessDenied (Accesso negato)

: Amazon EKS o uno o più nodi gestiti non sono in grado di eseguire l'autenticazione o l'autorizzazione con il server API del cluster Kubernetes. Per ulteriori informazioni sulla risoluzione di un problema comune, consulta the section called "Risoluzione di una causa comune di errori `AccessDenied` per i gruppi di nodi gestiti". Le AMI Windows private possono causare anche questo codice di errore insieme al messaggio di errore `Not authorized for images`. Per ulteriori informazioni, consulta the section called "`Not authorized for images`".

AmiIdNotFound

: Non è stato possibile trovare l'ID AMI associato al modello di avvio. Assicurati che l'AMI esista e sia condivisa con l'account.

AutoScalingGroupNotFound

: Non è stato possibile trovare il gruppo Auto Scaling associato al gruppo di nodi gestiti. Potresti ricreare un gruppo con dimensionamento automatico con le stesse impostazioni per procedere con il ripristino.

ClusterUnreachable

: Amazon EKS o uno o più nodi gestiti non sono in grado di comunicare con il server API del cluster Kubernetes. Ciò può verificarsi in caso di interruzioni di rete o se i server API stanno eseguendo il timeout delle richieste di elaborazione.

Ec2SecurityGroupNotFound

Non è stato possibile trovare il gruppo di sicurezza per il cluster. È necessario ricreare il cluster.

Ec2SecurityGroupDeletionFailure

Non è stato possibile eliminare il gruppo di sicurezza dell'accesso remoto per il gruppo di nodi gestiti. Rimuovi eventuali dipendenze dal gruppo di sicurezza.

Ec2LaunchTemplateNotFound

Non è stato possibile trovare il modello di avvio Amazon EC2 per il gruppo di nodi gestiti. È necessario ricreare il gruppo di nodi per procedere con il ripristino.

Ec2LaunchTemplateVersionMismatch

La versione del modello di avvio Amazon EC2 per il gruppo di nodi gestiti non corrisponde alla versione creata da Amazon EKS. Puoi regredire alla versione creata da Amazon EKS per il ripristino.

IamInstanceProfileNotFound

Non è stato possibile trovare il profilo dell'istanza IAM per il gruppo di nodi gestiti. Potresti ricreare un profilo dell'istanza con le stesse impostazioni per il ripristino.

IamNodeRoleNotFound

Non è stato possibile trovare il ruolo IAM per il gruppo di nodi gestiti. Potresti ricreare un ruolo IAM con le stesse impostazioni per il ripristino.

NodeCreationFailure

Il gruppo con dimensionamento automatico sta riscontrando errori nel tentativo di avviare le istanze.

NodeCreationFailure

Le istanze avviate non sono in grado di registrarsi con il cluster Amazon EKS. Le cause comuni di questo errore sono le autorizzazioni insufficienti del ruolo IAM del nodo o la mancanza di accesso a Internet in uscita per i nodi. I nodi devono soddisfare uno dei requisiti seguenti:

- Devono essere in grado di accedere a Internet utilizzando un indirizzo IP pubblico. Il gruppo di sicurezza associato alla sottorete in cui si trova il nodo deve consentire la comunicazione. Per ulteriori informazioni, consultare the section called "Considerazioni e requisiti relativi alle sottoreti" e the section called "Requisiti relativi al gruppo di sicurezza".

- I nodi e il VPC devono soddisfare i requisiti in Implementare cluster privato con accesso Internet limitato.

InstanceLimitExceeded

L'account AWS non è in grado di avviare altre istanze del tipo di istanza specificato. Potresti richiedere un aumento del limite di istanze Amazon EC2 per il ripristino.

InsufficientFreeAddresses

Una o più sottoreti associate al gruppo di nodi gestiti non dispone di indirizzi IP disponibili sufficienti per i nuovi nodi.

InternalFailure

Questi errori sono in genere causati da un problema lato server Amazon EKS.

Risoluzione di una causa comune di errori **`AccessDenied`** per i gruppi di nodi gestiti

La causa più comune degli errori `AccessDenied` durante l'esecuzione di operazioni su gruppi di nodi gestiti è la mancanza di `eks:node-manager` `ClusterRole` o `ClusterRoleBinding`. Amazon EKS imposta queste risorse nel cluster come parte dell'onboarding con i gruppi di nodi gestiti e queste sono necessarie per la gestione dei gruppi di nodi.

Il `ClusterRole` può cambiare nel tempo, ma dovrebbe essere simile all'esempio seguente:

```
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
```

Il `ClusterRoleBinding` può cambiare nel tempo, ma dovrebbe essere simile all'esempio seguente:

```
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
```

Verifica che il `eks:node-manager` `ClusterRole` esista.

```
kubectl describe clusterrole eks:node-manager
```

Se presente, confrontare l'output con il precedente esempio `ClusterRole`.

Verifica che il `eks:node-manager` `ClusterRoleBinding` esista.

```
kubectl describe clusterrolebinding eks:node-manager
```

Se presente, confrontare l'output con il precedente esempio `ClusterRoleBinding`.

Se è identificato un oggetto mancante o corrotto `ClusterRole` o `ClusterRoleBinding` come causa di un errore `AcessDenied` durante la richiesta di operazioni di gruppo di nodi gestiti, è

possibile ripristinarle. Salvai seguenti contenuti in un file denominato *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
```

Applica il file.

```
kubectl apply -f eks-node-manager-role.yaml
```

Riprova l'operazione per il gruppo di nodi per verificare se il problema è stato risolto.

`Not authorized for images`

Una potenziale causa di un messaggio di errore `Not authorized for images` è l'utilizzo di un'AMI Windows di Amazon EKS privata per avviare gruppi di nodi gestiti da Windows. Dopo il rilascio di nuove AMI Windows, AWS rende private le AMI che hanno più di 4 mesi, rendendole non più accessibili. Se il tuo gruppo di nodi gestiti utilizza un'AMI Windows privata, considera la possibilità di aggiornare il gruppo di nodi gestito da Windows. Sebbene non possiamo garantire di poter fornire l'accesso alle AMI che sono state rese private, puoi richiederne l'accesso inviando un ticket al Supporto AWS. Per maggiori informazioni, consulta Patches nella Guida per l'utente di Amazon EC2.

Il nodo è in stato **`NotReady`**

Se il nodo entra in uno stato `NotReady`, ciò probabilmente indica che il nodo non è integro e non è disponibile per pianificare nuovi pod. Ciò può verificarsi per vari motivi, ad esempio se il nodo non dispone di risorse sufficienti per CPU, memoria o spazio disponibile su disco.

Per le AMI Windows ottimizzate per Amazon EKS, non è prevista alcuna prenotazione per le risorse di elaborazione specificate per impostazione predefinita nella configurazione `kubelet`. Per evitare problemi di risorse, è possibile prenotare le risorse di calcolo per i processi di sistema fornendo a `kubelet` i valori di configurazione per kube-reserved e/o system-reserved. È possibile eseguire questa operazione utilizzando il parametro della riga di comando `-KubeletExtraArgs` nello script di bootstrap. Per ulteriori informazioni, consulta Reserve Compute Resources for System Daemons nella documentazione di Kubernetes e the section called "Parametri di configurazione dello script di bootstrap" in questa guida per l'utente.

Raccoglitore di log EKS

Per risolvere i problemi con i nodi Amazon EKS, è disponibile uno script predefinito sui nodi situati in `/etc/eks/log-collector-script/eks-log-collector.sh`. È possibile utilizzare lo script per raccogliere i registri diagnostici per i casi di supporto e per la risoluzione dei problemi generali.

Utilizza il comando seguente per eseguire lo script sul nodo:

```
sudo bash /etc/eks/log-collector-script/eks-log-collector.sh
```

> **Note**
>
> Se lo script non è presente in quella posizione. Puoi scaricare manualmente lo script ed eseguirlo con il comando seguente:
>
> ```
> curl -O https://amazon-eks.s3.amazonaws.com/support/log-collector-script/linux/
> eks-log-collector.sh
> sudo bash eks-log-collector.sh
> ```

Lo script raccoglie le seguenti informazioni di diagnostica:

```
$ 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
```

Le informazioni di diagnostica vengono raccolte e archiviate in:

```
/var/log/eks_i-EXAMPLE_2025-03-25_0000-UTC_0.7.8.tar.gz
```

Per recuperare il pacchetto di log per i nodi Bottlerocket, fai riferimento a Bottlerocket Log per maggiori dettagli.

Rete runtime container non pronta

Potresti visualizzare un errore `Container runtime network not ready` ed errori di autorizzazione analoghi al seguente:

```
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
```

Questo può verificarsi per uno dei seguenti motivi:

1. Non disponi di `aws-auth ConfigMap` sul tuo cluster oppure non include voci per il ruolo IAM con cui hai configurato i tuoi nodi.

   Per risolvere il problema, verifica le voci esistenti in `ConfigMap` sostituendo *`my-cluster`* nel seguente comando con il nome del tuo cluster e quindi eseguendo il comando modificato: `eksctl get iamidentitymapping --cluster` *`my-cluster`* . Se ricevi un messaggio di errore dal comando, è possibile che il cluster non disponga di `aws-auth ConfigMap`. Il comando seguente aggiunge una voce a `ConfigMap`. Inoltre, se `ConfigMap` non esiste, è creata dal comando. Sostituisci *`111122223333`* con l'ID dell'account AWS per il ruolo IAM e *`myAmazonEKSNodeRole`* con il nome del ruolo del tuo nodo.

   ```
   eksctl create iamidentitymapping --cluster my-cluster \
       --arn arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --group
    system:bootstrappers,system:nodes \
       --username system:node:{{EC2PrivateDNSName}}
   ```

   L'ARN del ruolo specificato non può includere un percorso diverso da `/`. Ad esempio, se il nome del tuo ruolo è `development/apps/my-role`, devi cambiarlo in `my-role` quando specifichi

l'ARN del ruolo. Assicurati di specificare l'ARN del ruolo IAM del nodo (non l'ARN del profilo dell'istanza).

2. I nodi autogestiti si trovano in un cluster con una versione della piattaforma che corrisponde a quella minima elencata nei prerequisiti dell'argomento Concedi agli utenti IAM l'accesso a Kubernetes con le voci di accesso EKS, tuttavia una voce non è elencata in `aws-auth` `ConfigMap` (consulta l'elemento precedente) per il ruolo IAM del nodo o non esiste una voce di accesso per il ruolo. Per risolvere il problema, verifica le tue voci di accesso esistenti sostituendo *`my-cluster`* nel seguente comando con il nome del tuo cluster e quindi eseguendo il comando modificato: `aws eks list-access-entries --cluster-name` *`my-cluster`* . Il comando seguente aggiunge una voce di accesso per il ruolo IAM del nodo. Sostituisci *`111122223333`* con l'ID dell'account AWS per il ruolo IAM e *`myAmazonEKSNodeRole`* con il nome del ruolo del tuo nodo. Se hai un nodo Windows, sostituisci *`EC2_LINUX`* con `EC2_Windows`. Assicurati di specificare l'ARN del ruolo IAM del nodo (non l'ARN del profilo dell'istanza).

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --type EC2_LINUX
```

Timeout dell'handshake TLS

Quando un nodo non è in grado di stabilire una connessione all'endpoint del server API pubblico, è possibile che venga visualizzato un errore analogo al seguente.

```
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\""
```

Il processo `kubelet` riprenderà continuamente e testerà l'endpoint del server API. L'errore può verificarsi anche temporaneamente durante qualsiasi procedura che esegue un aggiornamento in sequenza del cluster nel piano di controllo, ad esempio una modifica della configurazione o un aggiornamento della versione.

Per risolvere il problema, controlla la tabella di routing e i gruppi di sicurezza per assicurarti che il traffico proveniente dai nodi possa raggiungere l'endpoint pubblico.

InvalidClientTokenId

Se utilizzi i ruoli IAM per gli account di servizio per un pod o un DaemonSet implementato in un cluster in una Regione AWS Cina e non hai impostato la variabile di ambiente `AWS_DEFAULT_REGION` nelle specifiche, il pod o il DaemonSet potrebbe ricevere il seguente errore:

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation:
 The security token included in the request is invalid
```

Per risolvere il problema, è necessario aggiungere la variabile di ambiente `AWS_DEFAULT_REGION` alla specifica del pod o DaemonSet, come mostrato nella specifica del pod di esempio seguente.

```
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"
```

Prima di poter aggiornare il piano di controllo, i gruppi di nodi devono corrispondere alla versione di Kubernetes

Prima di aggiornare il piano di controllo a una nuova versione di Kubernetes, assicurati che la versione secondaria dei nodi gestiti e Fargate nel cluster sia la stessa della versione corrente del piano di controllo. L'API `update-cluster-version` di Amazon EKS rifiuta le richieste fino a quando non si aggiornano tutti i nodi gestiti Amazon EKS alla versione corrente del cluster. Amazon EKS fornisce API per aggiornare i nodi gestiti. Per informazioni sull'aggiornamento delle versioni di Kubernetes del gruppo di nodi gestiti, consulta the section called "Aggiornamento". Per aggiornare la versione di un nodo Fargate, elimina il pod rappresentato dal nodo e implementalo nuovamente dopo aver aggiornato il piano di controllo. Per ulteriori informazioni, consulta the section called "Aggiornamento della versione di Kubernetes".

All'avvio di molti nodi, si verificano errori `Too Many Requests`

Se si avviano più nodi contemporaneamente, si potrebbe visualizzare il messaggio di errore `Too Many Requests` nei registri di esecuzione dei dati utente di Amazon EC2. Ciò può verificarsi perché il piano di controllo è sovraccarico di chiamate `describeCluster`. Il sovraccarico si traduce in limitazione (della larghezza di banda della rete), nodi che non eseguono lo script bootstrap e nodo che non vengono aggiunti al cluster.

Assicurarti che gli argomenti `--apiserver-endpoint`, `--b64-cluster-ca` e `--dns-cluster-ip` siano trasferiti allo script di bootstrap del nodo. Quando includi questi argomenti, non è necessario che lo script bootstrap crei una chiamata `describeCluster`. Ciò contribuisce a impedire che il piano di controllo si sovraccarichi. Per ulteriori informazioni, consulta the section called "Fornisci i dati utente per passare argomenti al `bootstrap.sh` file incluso in un' Linux/Bottlerocket AMI ottimizzata per Amazon EKS".

Risposta agli errori HTTP 401 Unauthorized (Autorizzazione negata) per le richieste del server API Kubernetes

Questi errori sono visualizzati se il token dell'account di servizio di un pod è scaduto in un cluster.

Il server API Kubernetes del cluster Amazon EKS rifiuta le richieste con token più vecchi di 90 giorni. Nelle versioni Kubernetes precedenti, i token non avevano una scadenza. Di conseguenza, i client che si basano su questi token devono aggiornarli entro un'ora. Per impedire al server API Kubernetes di rifiutare la richiesta a causa di un token non valido, la versione dell'SDK client Kubernetes utilizzata dal carico di lavoro deve essere uguale o successiva alle versioni riportate di seguito:

- Go versione `0.15.7` e successive
- Python versione `12.0.0` e successive
- Java versione `9.0.0` e successive
- JavaScript versione `0.10.3` e successive
- Ramo `master` di Ruby
- Haskell versione `0.3.0.0`
- Versione C# `7.0.5` e successive

È possibile identificare tutti i pod esistenti nel cluster che utilizzano token obsoleti. Per ulteriori informazioni, consulta the section called "Token dell'account di servizio".

La versione della piattaforma Amazon EKS è più avanti di due versioni rispetto all'attuale versione della piattaforma

Ciò può accadere quando Amazon EKS non è in grado di aggiornare automaticamente la versione della piattaforma del cluster. Sebbene ci siano molte cause, di seguito riportiamo quelle più comuni. Se uno di questi problemi si applica al tuo cluster, potrebbe ancora funzionare, ma la sua versione della piattaforma non sarà aggiornata da Amazon EKS.

Problema

Il ruolo IAM del cluster è stato eliminato: questo ruolo è stato specificato al momento della creazione del cluster. Puoi visualizzare quale ruolo è stato specificato con il seguente comando. Sostituisci *my-cluster* con il nome del tuo cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut
 -d / -f 2
```

Di seguito viene riportato un output di esempio:

```
eksClusterRole
```

Soluzione

Crea un nuovo ruolo IAM del cluster con lo stesso nome.

Problema

Una sottorete specificata durante la creazione del cluster è stata eliminata: le sottoreti da utilizzare con il cluster sono state specificate durante la creazione del cluster. Puoi visualizzare le sottoreti specificate con il seguente comando. Sostituisci *my-cluster* con il nome del tuo cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

Di seguito viene riportato un output di esempio:

```
[
"subnet-EXAMPLE1",
"subnet-EXAMPLE2"
]
```

Soluzione

Verifica se gli ID di sottorete esistono nel tuo account.

```
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"
```

Di seguito viene riportato un output di esempio:

```
[
"subnet-EXAMPLE3",
"subnet-EXAMPLE4"
]
```

Nel caso in cui gli ID di sottorete restituiti nell'output non corrispondano agli ID di sottorete specificati al momento della creazione del cluster, se desideri che Amazon EKS aggiorni il cluster, devi modificare le sottoreti utilizzate dal cluster. Questo perché se hai specificato più di due sottoreti al momento della creazione del cluster, Amazon EKS seleziona in modo casuale le sottoreti specificate in cui creare nuove interfacce di rete elastiche. Queste interfacce di rete consentono al piano di controllo di comunicare con i nodi. Amazon EKS non aggiornerà il cluster se la sottorete selezionata non esiste. Non hai alcun controllo su quale delle sottoreti specificate al momento della creazione del cluster Amazon EKS sceglie per creare una nuova interfaccia di rete.

Quando avvii un aggiornamento della versione di Kubernetes per il cluster, l'aggiornamento può non riuscire per lo stesso motivo.

Problema

Un gruppo di sicurezza specificato durante la creazione del cluster è stato eliminato: se durante la creazione del cluster hai specificato dei gruppi di sicurezza, puoi visualizzarne gli ID con il seguente comando. Sostituisci *my-cluster* con il nome del tuo cluster.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.securityGroupIds
```

Di seguito viene riportato un output di esempio:

```
[
```

```
    "sg-EXAMPLE1"
]
```

Se `[]` è restituito, non sono stati specificati gruppi di sicurezza al momento della creazione del cluster e il problema non è dato dalla mancanza di un gruppo di sicurezza. Se vengono restituiti i gruppi di sicurezza, conferma che i gruppi di sicurezza sono presenti nel tuo account.

Soluzione

Verifica se questi gruppi di sicurezza esistono nel tuo account.

```
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"
```

Di seguito viene riportato un output di esempio:

```
[
"sg-EXAMPLE2"
]
```

Se desideri che Amazon EKS aggiorni il cluster, nel caso in cui gli ID dei gruppi di sicurezza restituiti nell'output non corrispondano agli ID dei gruppi di sicurezza specificati al momento della creazione del cluster, devi modificare i gruppi di sicurezza utilizzati dal cluster. Amazon EKS non aggiornerà un cluster se gli ID dei gruppi di sicurezza specificati al momento della creazione del cluster non esistono.

Quando avvii un aggiornamento della versione di Kubernetes per il cluster, l'aggiornamento può non riuscire per lo stesso motivo.

- Non hai almeno sei (anche se ne consigliamo 16) indirizzi IP disponibili in ciascuna delle sottoreti specificate al momento della creazione del cluster. Se non disponi di un numero sufficiente di indirizzi IP disponibili nella sottorete, dovrai liberare gli indirizzi IP nella sottorete oppure modificare le sottoreti utilizzate dal cluster in modo da utilizzare le sottoreti con un numero sufficiente di indirizzi IP disponibili.
- Hai abilitato la crittografia dei segreti durante la creazione del cluster e la chiave AWS KMS specificata è stata eliminata. Se desideri che Amazon EKS aggiorni il cluster, devi crearne uno nuovo

Domande frequenti sull’integrità del cluster e codici di errore con percorsi di risoluzione

Amazon EKS rileva problemi con i cluster EKS e l’infrastruttura del cluster e li archivia nell’oggetto integrità della risorsa del cluster EKS. Grazie alle informazioni sull'integrità del cluster è possibile rilevare, risolvere e affrontare più rapidamente eventuali problemi. Ciò consente di creare ambienti applicativi più sicuri e aggiornati. Inoltre, è possibile che tu non possa effettuare l’aggiornamento a versioni più recenti di Kubernetes o che Amazon EKS non sia in grado di installare aggiornamenti di sicurezza su un cluster degradato a causa di problemi relativi all’infrastruttura necessaria o alla configurazione del cluster. Amazon EKS può impiegare tre ore per rilevare problemi o verificare che un problema è stato risolto.

L'integrità di un cluster Amazon EKS rappresenta una responsabilità condivisa tra Amazon EKS e i suoi utenti. Sei responsabile dell'infrastruttura indispensabile per i ruoli IAM e le sottoreti Amazon VPC, nonché di altre infrastrutture necessarie, che devono essere fornite anticipatamente. Amazon EKS rileva le modifiche nella configurazione di questa infrastruttura e del cluster.

Per accedere all’integrità del cluster nella console Amazon EKS, è necessario trovare una tabella denominata Problemi di integrità nella scheda Problemi di integrità del cluster nella dashboard di osservabilità a cui si accede dalla pagina dei dettagli del cluster Amazon EKS. Questi dati saranno disponibili anche richiamando l’azione `DescribeCluster` nell’API EKS, ad esempio dall’interno della linea a riga di comando AWS.

Qual è il vantaggio di utilizzare questa funzionalità?

> Otterrai una maggiore visibilità sull'integrità del tuo cluster Amazon EKS, diagnosticando e risolvendo rapidamente eventuali problemi, senza dover dedicare tempo al debug o all'apertura di casi di supporto AWS. Ad esempio: hai eliminato accidentalmente una sottorete per il cluster Amazon EKS e Amazon EKS non è in grado di creare interfacce di rete tra account e comandi di AWS CLI di Kubernetes come `kubectl` exec o log `kubectl`. Queste operazioni non riusciranno e restituiranno l'errore: `Error from server: error dialing backend: remote error: tls: internal error.` Ora sarà visualizzato un problema di integrità di Amazon EKS con il seguente messaggio: `subnet-da60e280 was deleted: could not create network interface`.

In che modo questa funzionalità è correlata o funziona con altri servizi AWS?

I ruoli IAM e le sottoreti Amazon VPC sono due esempi di elementi dell'infrastrutture indispensabile per i quali l'integrità del cluster può rilevare problemi. Se tali risorse non sono configurate correttamente, questa funzionalità restituirà informazioni dettagliate.

Un cluster con problemi di integrità comporta costi?

Sì, ogni cluster Amazon EKS viene fatturato ai prezzi standard di Amazon EKS. La funzionalità integrità del cluster è disponibile senza costi aggiuntivi.

Questa funzionalità funziona con i cluster Amazon EKS su AWS Outposts?

Sì, vengono rilevati problemi di cluster per i cluster EKS nel cloud AWS, inclusi i cluster estesi su AWS Outposts e i cluster locali su AWS Outposts. L'integrità del cluster non rileva problemi con Amazon EKS Anywhere o Amazon EKS Distro (EKS-D).

Posso ricevere una notifica quando vengono rilevati nuovi problemi?

Sì. AWS invia un'e-mail e una notifica della dashboard di Personal Health quando sono rilevati nuovi problemi di integrità del cluster.

La console mi avvisa in caso di problemi di integrità?

Sì, qualsiasi cluster con problemi di integrità presenterà un banner di avviso nella parte superiore della console.

Le prime due colonne sono quelle necessarie per i valori di risposta dell'API. Il terzo campo dell'oggetto Problemi di integrità del cluster è ResourceIds, e il valore restituito varia in base al tipo di problema.

Codice	Messaggio	ResourceIds	Cluster recuperabile?
SUBNET_NOT_FOUND	Non è stato possibile trovare una o più sottoreti attualmente associate al tuo cluster. Effettua una chiamata all'API update-cluster-config	ID sottorete	Sì

Codice	Messaggio	ResourceIds	Cluster recuperabile?
	di Amazon EKS per aggiornare le sottoreti.		
SECURITY_GROUP_NOT_FOUND	Non è stato possibile trovare uno o più gruppi di sicurezza attualmente associati al tuo cluster. Effettua una chiamata all'API update-cluster-config di Amazon EKS per aggiornare i gruppi di sicurezza	ID gruppo di sicurezza	Sì
IP_NOT_AVAILABLE	Una o più sottoreti associate al cluster non dispone di indirizzi IP disponibili sufficienti per consentire ad Amazon EKS di eseguire operazioni di gestione dei cluster. Libera indirizzi nelle sottoreti o associa diverse sottoreti al cluster utilizzando l'API update-cluster-config di Amazon EKS.	ID sottorete	Sì
VPC_NOT_FOUND	Non è stato possibile trovare il VPC associato al cluster. È necessario eliminare e ricreare il cluster.	ID VPC	No

Codice	Messaggio	ResourceIds	Cluster recuperabile?
ASSUME_ROLE_ACCESS_DENIED	Il tuo cluster non utilizza il ruolo collegato al servizio Amazon EKS. Non è stato possibile assumere il ruolo associato al tuo cluster per eseguire le operazioni di gestione di Amazon EKS richieste. Verifica che il ruolo esista e disponga della policy di attendibilità richiesta.	Ruolo IAM del cluster	Sì
PERMISSION_ACCESS_DENIED	Il tuo cluster non utilizza il ruolo collegato al servizio Amazon EKS. Il ruolo associato al tuo cluster non concede autorizzazioni sufficienti ad Amazon EKS per eseguire le operazioni di gestione richieste. Verifica le policy associate al ruolo del cluster e controlla se vengono applicate policy di negazione separate.	Ruolo IAM del cluster	Sì

Codice	Messaggio	ResourceIds	Cluster recuperabile?
ASSUME_ROLE_ACCESS_DENIED_USING_SLR	Non è stato possibile assumere il ruolo collegato al servizio di gestione del cluster Amazon EKS. Verifica che il ruolo esista e disponga della policy di attendibilità richiesta.	Il ruolo collegato al servizio di Amazon EKS	Sì
PERMISSION_ACCESS_DENIED_USING_SLR	Il ruolo collegato al servizio di gestione del cluster Amazon EKS non concede autorizzazioni sufficienti ad Amazon EKS per eseguire le operazioni di gestione richieste. Verifica le policy associate al ruolo del cluster e controlla se vengono applicate policy di negazione separate.	Il ruolo collegato al servizio di Amazon EKS	Sì
OPT_IN_REQUIRED	Il tuo account non dispone di un abbonamento al servizio Amazon EC2. Aggiorna gli abbonamenti del tuo account nella pagina delle impostazioni dell'account.	N/D	Sì

Codice	Messaggio	ResourceIds	Cluster recuperabile?
STS_REGIONAL_ENDPOINT_DISABLED	L'endpoint regionale STS è disabilitato. Abilita l'endpoint per Amazon EKS per eseguire le operazioni di gestione dei cluster richieste.	N/D	Sì
KMS_KEY_DISABLED	La chiave KMS AWS associata al cluster è disabilitata. Riabilita la chiave per ripristinare il cluster.	L'ARN della chiave KMS	Sì
KMS_KEY_NOT_FOUND	Non è stato possibile trovare la chiave KMS AWS associata al tuo cluster. È necessario eliminare e ricreare il cluster.	L'ARN della chiave KMS	No
KMS_GRANT_REVOKED	Le concessioni per la chiave KMS AWS associata al cluster sono revocate. È necessario eliminare e ricreare il cluster.	L'ARN della chiave KMS	No

Estensione delle funzionalità di Amazon EKS con progetti open-source

Questi progetti open source estendono la funzionalità dei cluster Kubernetes in esecuzione all'interno o all'esterno AWS, inclusi i cluster gestiti da Amazon EKS.

Supporto per il software distribuito su EKS

Quando esamini i documenti di Amazon EKS, troverai riferimenti a vari strumenti e software open-source nelle nostre procedure ed esempi. Questi strumenti includono Server delle metriche Kubernetes e Sistema di gestione delle certificazioni.

Tieni presente che qualsiasi software di terze parti o open source che scegli di distribuire non rientra nell'ambito dei tuoi Contratti di AWS Supporto. Uno dei vantaggi dell'utilizzo di Kubernetes è la community open-source attiva. Consigliamo di collaborare direttamente con le comunità open-source e i responsabili del progetto pertinenti per stabilire canali di supporto appropriati per tali componenti. Per ulteriori informazioni, consulta i progetti graduati e in fase di incubazione associati alla Cloud Native Computing Foundation (CNCF).

L'ecosistema Kubernetes include numerosi progetti e componenti che prevedono diversi livelli di supporto comunitario, tempi di risposta e casi d'uso previsti. Quando implementi queste tecnologie insieme a EKS, assicurati di comprendere la matrice di supporto per ciascun componente.

AWS mantiene i componenti open source che integriamo nel piano di controllo EKS. Ciò include la nostra pipeline di sicurezza completa che copre la verifica delle build, la scansione delle vulnerabilità, i test di convalida e la gestione delle patch per tutte le immagini e i file binari dei container che distribuiamo. Ad esempio, AWS è responsabile del Kubernetes API Server. Il server API Kubernetes è coperto dall'accordo sul livello di servizio di Amazon EKS. È possibile utilizzare il tuo piano di supporto dei servizi web di Amazon per risolvere problemi con il server API Kubernetes o ottenere indicazioni generali.

È necessario esaminare attentamente il supporto offerto per i vari componenti aggiuntivi di Amazon EKS. AWS i componenti aggiuntivi sono l'unico tipo di componente aggiuntivo Amazon EKS completamente supportato da. AWS AWS I componenti aggiuntivi del Marketplace sono supportati principalmente dai AWS partner. I componenti aggiuntivi della community ricevono il supporto di base per l'intero ciclo di vita da. AWS Per ulteriori informazioni, consulta Supporto dei componenti aggiuntivi.

Ogni componente aggiuntivo EKS, indipendentemente dal tipo, riceve il supporto di base per il ciclo di vita da EKS, inclusi i componenti aggiuntivi Marketplace. Il supporto di base per il ciclo di vita include l'installazione e la disinstallazione del componente aggiuntivo. Per ulteriori informazioni sui tipi di componenti aggiuntivi Amazon EKS disponibili e sui livelli di supporto associati, consulta Ambito del supporto per i componenti aggiuntivi Amazon EKS. Per visualizzare i componenti aggiuntivi completamente supportati da AWS, consulta i componenti aggiuntivi di Amazon Web Services.

- Per ulteriori informazioni sulle nostre pratiche di sicurezza e sui limiti del supporto, consulta Sicurezza in Amazon EKS.
- Per ulteriori informazioni sui componenti aggiuntivi della community e del AWS marketplace disponibili tramite Amazon EKS Add-ons, consulta EKS Add-ons Support.

Strumenti di gestione

Strumenti di gestione correlati per cluster Amazon EKS e Kubernetes.

eksctl

`eksctl` è un semplice strumento CLI per la creazione di cluster su Amazon EKS.

- URL progetto
- Documentazione progetto
- AWS blog open source: eksctl: cluster Amazon EKS con un solo comando

AWS Controller per Kubernetes

Con AWS Controllers for Kubernetes, puoi creare e gestire risorse direttamente dal tuo cluster Kubernetes. AWS

Disponibile in EKS Capabilities.

- URL progetto
- AWS blog open source: operatore AWS di servizio per Kubernetes ora disponibile

kro (Kube Resource Orchestrator)

kro ti consente di creare APIs Kubernetes personalizzati che compongono più risorse in astrazioni di livello superiore. I team della piattaforma possono definire modelli riutilizzabili con guardrail, mentre i team applicativi utilizzano strumenti semplici e di alto livello per fornire e gestire le risorse. APIs

Disponibile in EKS Capabilities.

- URL progetto
- Documentazione progetto

Argo CD

Argo CD è uno strumento di distribuzione GitOps continua e dichiarativo per Kubernetes. Monitora continuamente i tuoi repository Git e sincronizza automaticamente le modifiche ai tuoi cluster.

Disponibile in EKS Capabilities.

- URL progetto
- Documentazione progetto

CD Flux

Flux è uno strumento utile per gestire la configurazione del cluster quando si usa Git. Utilizza un operatore nel cluster per attivare le distribuzioni all'interno di Kubernetes. Per ulteriori informazioni sugli operatori, consulta OperatorHub.io on GitHub.

- URL progetto
- Documentazione progetto

CDK per Kubernetes

Con il CDK per Kubernetes (cdk8s), è possibile definire app e componenti Kubernetes utilizzando linguaggi di programmazione familiari. Le app cdk8s sintetizzano in manifesti Kubernetes standard, che possono essere applicati a qualsiasi cluster Kubernetes.

- URL progetto

- Documentazione progetto
- AWS blog sui container: Presentazione di cdk8s+: Intent-driven for Kubernetes object APIs

Rete

Progetti di rete correlati per cluster Amazon EKS e Kubernetes.

Plug-in CNI di Amazon VPC per Kubernetes

Amazon EKS; supporta reti VPC native tramite il plug-in CNI di Amazon VPC per Kubernetes. Il plugin assegna un indirizzo IP dal VPC a ciascun pod.

- URL progetto
- Documentazione progetto

AWS Load Balancer Controller per Kubernetes

Il AWS Load Balancer Controller aiuta a gestire AWS Elastic Load Balancer per un cluster Kubernetes. Soddisfa le risorse di Kubernetes Ingress fornendo Application Load Balancers. AWS Soddisfa le risorse di servizio Kubernetes fornendo Network Load Balancer. AWS

- URL progetto
- Documentazione progetto

ExternalDNS

ExternalDNS sincronizza i servizi e gli ingressi Kubernetes esposti con provider DNS tra cui Amazon Route 53 e Service Discovery. AWS

- URL progetto
- Documentazione progetto

Machine learning

Progetti di machine learning correlati per cluster Amazon EKS e Kubernetes.

Kubeflow

Un kit di strumenti di machine learning per Kubernetes.

- URL progetto
- Documentazione progetto
- AWS blog open source: Kubeflow su Amazon EKS

Auto Scaling

Progetti di scalabilità automatica correlati per cluster Amazon EKS e Kubernetes.

Cluster Autoscaler

Cluster Autoscaler è uno strumento che consente di regolare automaticamente le dimensioni del cluster Kubernetes in base alle richieste di CPU e memoria.

- URL progetto
- Documentazione progetto
- Workshop Amazon EKS: Cluster Autoscaler

Karpenter

Karpenter è un Node Autoscaler di Kubernetes progettato per flessibilità, prestazioni e semplicità.

- URL progetto
- Documentazione progetto
- Workshop Amazon EKS: Karpenter

Escalator

Escalator è un batch o autoscaler orizzontale ottimizzato per i processi per Kubernetes.

- URL progetto
- Documentazione progetto

Monitoraggio

Progetti di monitoraggio correlati per cluster Amazon EKS e Kubernetes.

Prometheus

Prometheus è un kit di strumenti di monitoraggio e avvisi di sistema open source.

- URL progetto
- Documentazione progetto
- Workshop Amazon EKS: https://eksworkshop.com/intermediate/240_monitoring/

Integrazione continua / distribuzione continua

CI/CD Progetti imperativi correlati per i cluster Amazon EKS e Kubernetes.

Jenkins X

Soluzione IC/DC per applicazioni cloud moderne su cluster Amazon EKS e Kubernetes.

- URL progetto
- Documentazione progetto

Scopri le nuove funzionalità e le roadmap di Amazon EKS

Per maggiori informazioni sulle nuove funzionalità di Amazon EKS, scorri fino al feed Novità nella pagina Novità in AWS. È possibile anche rivedere la roadmap su GitHub, che consente di conoscere le funzionalità e le priorità in arrivo, in modo da poter pianificare come utilizzare Amazon EKS in futuro. È possibile fornirci un feedback diretto sulle priorità della roadmap.

Cronologia dei documenti

La tabella riportata di seguito illustra alcuni dei principali aggiornamenti e nuove funzionalità della Guida per l'utente di Amazon EKS. Per ricevere notifiche quando viene inserita una nuova voce in questa tabella, è possibile iscriversi al seguente URL con un lettore RSS:

```
https://docs.aws.amazon.com/eks/latest/userguide/doc-history.rss
```

Modifica	Descrizione	Data
AWS aggiornamenti delle politiche gestiti	Aggiunti `autoscaling:PutWarmPool` `autoscaling:DeleteWarmPool` e `autoscaling:DescribeWarmPool` autorizzazioni alla `AWSServiceRoleForAmazonEKSNodegroup` policy. Ciò consente ai gruppi di nodi gestiti di Amazon EKS di gestire le risorse del pool caldo ASG sottostante per tutto il ciclo di vita del gruppo di nodi.	17 febbraio 2026
AWS aggiornamenti delle politiche gestiti	È stato rimosso il requisito del prefisso «eks» nel nome del profilo dell'istanza di destinazione per l'`iam:GetInstanceProfile` autorizzazione in`AmazonEKSServiceRolePolicy` . Ciò consente ad Amazon EKS Auto Mode di convalidare e utilizzare profili di istanza personalizzati	2 febbraio 2026

	NodeClasses senza richieder e il prefisso di denominazione «eks».	
Kubernetes versione `1.35`	Sono stati aggiunti il supporto per Kubernetes versione `1.35` per nuovi cluster e aggiornam enti della versione.	27 gennaio 2026
AWS aggiornamenti delle politiche gestiti	È stata aggiunta `ec2:LockS napshot` l'autorizzazione `AmazonEBSCSIDriver Policy` per consentire al driver CSI EBS di bloccare direttamente le istantanee EBS.	15 gennaio 2026
Funzionalità di Amazon EKS	Amazon EKS ora offre funzionalità di cluster completamente gestite con EKS Capabilities, con supporto per la distribuz ione dichiarativa continua con supporto per Argo CD, gestione AWS delle risorse con supporto per AWS Controller for Kubernetes e composizione e orchestra zione delle risorse Kubernete s con supporto per Kube Resource Orchestrator.	30 novembre 2025

Nuova politica AWS gestita	Amazon EKS ha rilasciato una nuova policy gestita `AmazonEKSMCPReadOnlyAccess` per abilitare strumenti di sola lettura nel server MCP Amazon EKS per l'osservabilità e la risoluzione dei problemi. Per informazioni, consulta gli aggiornamenti di Amazon EKS alle policy AWS gestite.	21 novembre 2025
Osservabilità della rete	Amazon EKS ora offre una migliore osservabilità della rete di container con monitoraggio delle prestazioni e visibilità del traffico dei carichi di lavoro.	19 novembre 2025
AWS aggiornamenti delle politiche gestiti	È stata aggiunta `ec2:CopyVolumes` l'autorizzazione `AmazonEBSCSIDriverPolicy` per consentire al driver CSI EBS di copiare direttamente i volumi EBS.	17 novembre 2025
AWS aggiornamenti delle politiche gestiti	Aggiunti `ec2:DescribeRouteTables` e `ec2:DescribeNetworkAcls` autorizzazioni a. `AmazonEKSServiceRolePolicy` Ciò consente ad Amazon EKS di rilevare problemi di configurazione con le tabelle di routing VPC e la rete ACLs per i nodi ibridi come parte delle informazioni sui cluster.	22 ottobre 2025

Aggiornamento del ruolo collegato al servizio	È stata aggiunta `ssmmessages:OpenDataChannel` l'autorizzazione a `AmazonEKSConnectorServiceRolePolicy`	15 ottobre 2025
Kubernetes versione `1.34`	Sono stati aggiunti il supporto per Kubernetes versione `1.34` per nuovi cluster e aggiornamenti della versione.	2 ottobre 2025
Riparazione automatica dei nodi configurabile	Ora Amazon EKS offre un controllo più granulare sul comportamento di riparazione automatica del nodo.	22 settembre 2025
Aggiornamento degli approfondimenti sui cluster	Ora è possibile aggiornare manualmente gli approfondimenti sui cluster.	27 agosto 2025
AWS aggiornamenti delle politiche gestite	Autorizzazione aggiunta a `AmazonEKSServiceRolePolicy` . Questo ruolo può collegare una nuova policy di accesso `AmazonEKSEventPolicy` . Autorizzazioni limitate per `ec2:DeleteLaunchTemplate` e `ec2:TerminateInstances` .	26 agosto 2025

Prevenzione del problema "confused deputy" tra servizi	È stato aggiunto un argomento con una policy di attendibilità di esempio che è possibile applicare per la prevenzione dei problemi di "confused deputy" tra servizi. Amazon EKS accetta le condizioni `aws:SourceArn` e `aws:SourceAccount` e la policy di attendibilità di un ruolo del cluster EKS.	19 agosto 2025
Aggiornamento della versione della piattaforma Amazon EKS	Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Ciò include nuove versioni di patch per Kubernetes `1.33.2`, `1.32.6`, `1.31.10` e `1.30.14`.	30 luglio 2025

Funzionalità Multi-NIC di CNI VPC per pod multi-homed	Amazon EKS aggiunge i pod multi-homed a CNI VPC. Ora è possibile configura re un carico di lavoro e gli indirizzi IP assegnati da CNI VPC da ogni NIC sull'ista nza EC2 a ciascun pod. L'applicazione può effettuar e connessioni simultanee per utilizzare la larghezza di banda di ciascuna NIC. Ogni interfaccia di rete è configura ta nella stessa sottorete e negli stessi gruppi di sicurezza del nodo. In precedenza, era necessario utilizzare Multus CNI per eseguirne più di uno per creare pod multihomed. CNIs La documentazione e i passaggi per eseguire questa operazione sono stati spostati in. https://docs.aws.amazon.com/eks/latest/userguide/pod-multus.html	15 luglio 2025
Aggiornamento dei contenuti di risoluzione dei problemi di CNI VPC	La pagina di risoluzione dei problemi relativa alla policy di rete Kubernetes in CNI VPC è stata ampliata. Sono state aggiunte le autorizzazioni CRDs e RBAC e 12 problemi noti.	30 giugno 2025

Aggiornamento della versione della piattaforma Amazon EKS	Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Non sono presenti nuove versioni delle patch di Kubernetes in questa versione della piattaforma.	26 giugno 2025
AWS aggiornamenti delle politiche gestiti	Aggiunta l'autorizzazione `ssmmessages:OpenDataChannel` a `AmazonEKSLocalOutpostServiceRolePolicy`.	26 giugno 2025
AWS aggiornamenti delle politiche gestiti	Sono state aggiunte autorizzazioni a `AmazonEKSServiceRolePolicy` e `AmazonEKSComputePolicy` per consentire alla modalità automatica di Amazon EKS di avviare istanze utilizzando le prenotazioni della capacità on-demand EC2 nell'account. Inoltre, sono state aggiunte autorizzazioni a `AmazonEKSComputePolicy` per consentire ad Amazon EKS di creare il ruolo collegato al servizio EC2 spot per conto dell'utente.	20 giugno 2025
Aggiornamenti alle policy gestite	Aggiunta la policy `AmazonEKSDashboardConsoleReadOnly`.	19 giugno 2025

Aggiornamento della modalità automatica di Amazon EKS a `NodeClass`	Il modello `NodeClass` per i nodi della modalità automatica ha aggiunto la configurazione per sottoreti di pod separati. Vengono aggiunte le chiavi facoltative `podSubnetSelectorTerms` e `podSecurityGroupSelectorTerms` per impostare le sottoreti e i gruppi di sicurezza per i pod.	13 giugno 2025
Ruoli secondari e multi-account di destinazione con EKS Pod Identity	Amazon EKS aggiunge ruoli IAM di destinazione alle EKS Pod Identity per il concatenamento dei ruoli. # possibile utilizzarlo per assumere automaticamente un ruolo in un altro account, con EKS Pod Identity che ruota le credenziali temporanee. Ogni associazione Pod Identity deve prima disporre di un ruolo IAM da assumere nello stesso account, poi utilizza tale ruolo per assumere il ruolo di destinazione.	11 giugno 2025
Aggiornamento della versione della piattaforma Amazon EKS	Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Non sono presenti nuove versioni delle patch di Kubernetes in questa versione della piattaforma.	11 giugno 2025

Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella regione Asia Pacifico (Taipei) (`ap-east-2`) AWS .	6 giugno 2025
Controllo degli accessi `IPv6` per endpoint pubblici dual stack per nuovi cluster `IPv6`	Amazon EKS aggiunge intervalli CIDR `IPv6` per controllare l'accesso all'endpoint pubblico del cluster per nuovi cluster `IPv6`. In precedenza, era possibile aggiungere solo intervalli CIDR `IPv4` per consentire il traffico verso l'endpoint del cluster pubblico, anche per gli endpoint del cluster dual-stack.	5 giugno 2025
Aggiornamento della versione della piattaforma Amazon EKS	Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Ciò include nuove versioni di patch per Kubernetes `1.32.5`, `1.31.9` e `1.30.13`.	30 maggio 2025
Nuovi approfondimenti sui cluster per EKS Hybrid Nodes	Amazon EKS aggiunge nuovi approfondimenti sui cluster che verificano la configurazione dei nodi ibridi. Questi controlli approfonditi avviseranno dei problemi relativi ai nodi e ai pod on-premises e alla configurazione di rete remota del cluster.	29 maggio 2025

Kubernetes versione 1.33	Sono stati aggiunti il supporto per Kubernetes versione 1.33 per nuovi cluster e aggiornamenti della versione.	29 maggio 2025
Supporto aggiuntivo per il driver Amazon FSx CSI	Ora puoi utilizzare la console di AWS gestione, la AWS CLI e l'API per gestire il driver Amazon FSx CSI.	23 maggio 2025
Modifica degli scraper Prometheus nella console	Ora è possibile modificare gli scraper del Servizio gestito da Amazon per Prometheus nella console Amazon EKS.	22 maggio 2025
Aggiornamenti alle policy gestite	Aggiunta la policy AmazonEKSDashboardServiceRolePolicy .	21 maggio 2025
Aggiornamento della versione della piattaforma Amazon EKS	Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Non sono presenti nuove versioni delle patch di Kubernetes in queste versioni della piattaforma.	16 maggio 2025
Nuove pagine per le prestazioni di Amazon FSx for Lustre	Sono stati aggiunti nuovi argomenti con dettagli sull'ottimizzazione delle prestazioni di Amazon FSx for Lustre.	2 maggio 2025

Aggiornamento della modalità automatica di Amazon EKS a `NodeClass`	Il modello `NodeClass` per i nodi della modalità automatica ha aggiunto la configurazione per i proxy di inoltro di rete. Ciò aggiunge la chiave facoltativa `advancedNetworking` per configurare il proxy HTTPS.	30 aprile 2025
Bottlerocket per nodi ibridi	Bottlerocket è ora disponibile per EKS Hybrid Nodes.	29 aprile 2025
Aggiornamento della versione della piattaforma Amazon EKS	Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Non sono presenti nuove versioni delle patch di Kubernetes in queste versioni della piattaforma.	29 aprile 2025
Nuove pagine con concetti relativi alla rete ibrida	Aggiunte pagine con concetti relativi a EKS Hybrid Nodes. Vengono trattate nel dettaglio le reti on-premises e cloud con diagrammi.	18 aprile 2025
AWS aggiornamenti delle politiche gestiti	Aggiunte autorizzazioni a `AmazonEKSClusterPolicy` per consentire ad Amazon EKS di utilizzare interfacce di rete elastiche create da CNI VPC. Ciò è necessario affinché EKS possa ripulire le interfacce di rete elastiche che rimangono se CNI VPC si chiude inaspettatamente.	16 aprile 2025

AWS aggiornamenti delle politiche gestiti	Sono state aggiunte autorizzazioni per consentire `AmazonEKSServiceRolePolicy` ai AI/ML clienti EKS di aggiungere le regole Egress al gruppo di sicurezza EKS Cluster predefinito.	14 aprile 2025
Integrità dei nodi per EKS Hybrid Nodes	È possibile utilizzare `eks-node-monitoring-agent` su nodi ibridi, a partire dalla versione `1.2.0-eksbuild.1` . Eseguire `eks-node-monitoring-agent` come componente aggiuntivo di Amazon EKS per rilevare e mostrare problemi di integrità.	31 marzo 2025

EKS Hybrid Nodes per cluster esistenti	Ora è possibile aggiunger e, modificare o rimuovere la configurazione dei nodi ibridi dei cluster esistenti. In precedenza, era possibile aggiungere la configura zione dei nodi ibridi ai nuovi cluster solo al momento della creazione. Con Amazon EKS Hybrid Nodes, puoi utilizzar e la tua infrastruttura locale e perimetrale come nodi nei cluster Amazon EKS. AWS gestisce il piano AWS di controllo Kubernetes ospitato del cluster Amazon EKS e tu gestisci i nodi ibridi che vengono eseguiti nei tuoi ambienti locali o edge.	31 marzo 2025
Rollback: prevenire gli aggiornamenti accidentali con gli approfondimenti sui cluster	Amazon EKS ha temporane amente eseguito il rollback di una funzionalità che richiedev a l'utilizzo di un flag `--force` per aggiornare il cluster in caso di determinati problemi di approfondimenti sul cluster. Per ulteriori informazioni, consulta Ripristino temporane o dell'applicazione degli approfondimenti sull'aggi ornamento della versione del cluster su. GitHub	28 marzo 2025

Bottlerocket FIPS AMIs	I FIPS Bottlerocket AMIs sono ora disponibili in gruppi di nodi gestiti standard.	27 marzo 2025
AWS aggiornamenti delle politiche gestite	Aggiunte autorizzazioni a `AmazonEKSServiceRolePolicy` per consentire ad Amazon EKS di terminare le istanze EC2 create dalla modalità automatica. Aggiunte autorizzazioni a `AmazonEKSServiceRolePolicy` per consentire ad Amazon EKS di terminare le istanze EC2 create dalla modalità automatica.	28 febbraio 2025
Strategie di aggiornamento per i gruppi di nodi gestiti	Ora è possibile utilizzare strategie di aggiornamento per configurare il processo di aggiornamento delle versioni per i gruppi di nodi gestiti. Ciò introduce la strategia di aggiornamento minima per terminare i nodi prima di crearne di nuovi, utile in ambienti con capacità limitata. La strategia di aggiornamento predefinita mantiene il comportamento esistente.	27 gennaio 2025
Kubernetes versione `1.32`	Sono stati aggiunti il supporto per Kubernetes versione `1.32` per nuovi cluster e aggiornamenti della versione.	23 gennaio 2025

Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nelle Regioni AWS Asia Pacifico (Thailandia) (`ap-southeast-7`) e Messico (Centrale) (`mx-central-1`). La modalità automatica EKS e gli endpoint VPC per l'API EKS non sono disponibili in nessuna delle due Regioni.	14 gennaio 2025
AWS aggiornamenti delle politiche gestiti	Aggiunte più autorizzazioni a `AmazonEBSCSIDriverPolicy` per consentire al driver CSI per Amazon EBS di ripristinare tutti gli snapshot, abilitare il ripristino rapido degli snapshot (FSR) sui volumi EBS e modificare i tag sui volumi.	13 gennaio 2025
AWS aggiornamenti delle politiche gestiti	Aggiunte autorizzazioni a `AmazonEKSLoadBalancingPolicy` .	26 dicembre 2024
Approfondimenti sui cluster aggiornati	Ora gli approfondimenti di aggiornamento di Amazon EKS avviseranno di più problemi relativi all'integrità del cluster e alla compatibilità delle versioni. Possono rilevare problemi tra diversi componenti di Kubernetes e Amazon EKS, come `kubelet` e `kube-proxy` , nonché i componenti aggiuntivi Amazon EKS.	20 dicembre 2024

Agente di monitoraggio e riparazione automatica dei nodi	# possibile utilizzare il nuovo `eks-node-monitoring-agent` come componente aggiuntivo di Amazon EKS per rilevare e mostrare problemi di integrità. # anche possibile abilitare la riparazione automatica dei nodi per sostituire automaticamente i nodi quando vengono rilevati problemi.	16 dicembre 2024
Amazon EKS Hybrid Nodes	Ora è possibile eseguire il nodo on-premises connesso ai cluster Amazon EKS. Con Amazon EKS Hybrid Nodes, puoi utilizzare la tua infrastruttura locale e perimetrale come nodi nei cluster Amazon EKS. AWS gestisce il piano AWS di controllo Kubernetes ospitato del cluster Amazon EKS e tu gestisci i nodi ibridi che vengono eseguiti nei tuoi ambienti locali o edge.	1 dicembre 2024

Modalità automatica di Amazon EKS	La modalità automatica di Amazon EKS automatizza completamente la gestione dell'infrastruttura del cluster Kubernetes per calcolo, archiviazione e rete su AWS. Semplifica la gestione di Kubernetes effettuando automaticamente il provisioning dell'infrastruttura, selezionando le istanze di calcolo ottimali, scalando dinamicamente le risorse, ottimizzando continuamente i costi, applicando patch ai sistemi operativi e integrandosi con i servizi di sicurezza. AWS	1 dicembre 2024
Aggiornamento della versione della piattaforma Amazon EKS	Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Ciò include nuove versioni di patch per Kubernetes `1.31.2`, `1.30.6`, `1.29.10` e `1.28.15`.	22 novembre 2024
AWS aggiornamenti delle policy gestiti	Aggiornato `AWSServiceRoleForAmazonEKSNodegroup` per compatibilità con le regioni della Cina.	22 novembre 2024
La versione 1.30 di Kubernetes è ora disponibile per i cluster locali su Outposts AWS	Ora puoi creare un cluster locale Amazon EKS su AWS Outposts utilizzando la versione 1.30 di Kubernetes.	21 novembre 2024

AWS aggiornamenti delle politiche gestite	Politica AWS gestita aggiornata da EKS `AmazonEKSLocalOutpostClusterPolicy` . È stata aggiunta l'`ec2:DescribeAvailabilityZones` autorizzazione in modo che AWS Cloud Controller Manager sul piano di controllo del cluster possa identificare la zona di disponibilità in cui si trova ogni nodo.	21 novembre 2024
Bottlerocket AMIs che utilizza FIPS 140-3	AMIs Sono disponibili Bottlerocket preconfigurati per utilizzare moduli crittografici convalidati FIPS 140-3. Ciò include il modulo crittografico Amazon Linux 2023 Kernel Crypto API e il modulo crittografico AWS-LC.	20 novembre 2024
AWS aggiornamenti delle politiche gestite	Policy `AWSServiceRoleForAmazonEKSNodegroup` aggiornata per consentire `ec2:RebootInstances` per le istanze create da gruppi di nodi gestiti di Amazon EKS. Limitazione delle autorizzazioni `ec2:CreateTags` per le risorse Amazon EC2.	20 novembre 2024

Dashboard di osservabilità	La dashboard di osservabilità consente di rilevare, risolvere e correggere rapidamente i problemi. Nel namespace sono disponibili anche nuove metriche CloudWatch vendute. `AWS/EKS`	18 novembre 2024
AWS aggiornamenti delle politiche gestiti	Politica AWS gestita aggiornata da EKS`AmazonEKS ServiceRolePolicy` . Aggiunte le autorizzazioni per le politiche di accesso EKS, la gestione del bilanciatore del carico e la pulizia automatica delle risorse del cluster.	16 novembre 2024
Creazione di nuovi ruoli nella console per componenti aggiuntivi che supportano EKS Pod Identity	Sono stati aggiunti nuovi passaggi quando si utilizza la console per creare o aggiornare componenti aggiuntivi che supportano EKS Pod Identity che consentono di generare automaticamente ruoli IAM con il nome, la policy dei ruoli e la policy di attendibilità adeguati per il componente aggiuntivo.	15 novembre 2024
Gruppi di nodi gestiti in AWS Local Zones	I gruppi di nodi gestiti possono ora essere creati in AWS Local Zones.	15 novembre 2024
Disponibili nuovi parametri	Sono disponibili nuovi parametri nel gruppo API `metrics.eks.amazonaws.com` .	11 novembre 2024

AWS aggiornamenti delle politiche gestiti	Politica AWS gestita aggiornata da EKS`AmazonEKSComputePolicy`. Autorizzazioni delle risorse aggiornate per l'azione `iam:AddRoleToInstanceProfile`.	7 novembre 2024
AWS aggiornamenti delle politiche gestite	EKS ha aggiunto una nuova politica AWS gestita: `AmazonEKSComputePolicy`	1 novembre 2024
AWS aggiornamenti delle politiche gestite	Aggiunte autorizzazioni a `AmazonEKSClusterPolicy`. Aggiunta l'autorizzazione `ec2:DescribeInstanceTopology` per consentire ad Amazon EKS di collegare informazioni di topologia al nodo come etichette.	1 novembre 2024
AWS aggiornamenti delle politiche gestiti	EKS ha aggiunto una nuova politica AWS gestita: `AmazonEKSBlockStoragePolicy`	30 ottobre 2024
AWS aggiornamenti delle politiche gestite	EKS ha aggiunto una nuova politica AWS gestita: `AmazonEKSLoadBalancingPolicy`	30 ottobre 2024

AWS aggiornamenti delle politiche gestite	Sono state aggiunte `cloudwatch:PutMetricData` autorizzazioni per consentire `AmazonEKSServiceRolePolicy` ad Amazon EKS di pubblicare metriche su Amazon. CloudWatch	29 ottobre 2024
AWS aggiornamenti delle politiche gestiti	EKS ha aggiunto una nuova politica AWS gestita: `AmazonEKSNetworkingPolicy`	28 ottobre 2024
Endpoint dual stack per nuovi cluster `IPv6`	Connettersi a nuovi cluster `IPv6` con un endpoint `eks-cluster.` *`region`*`.api.aws` dual stack. Questo endpoint viene restituito quando si descrivono questi cluster. `kubectl` e altri client API Kubernetes in `IPv4`, `IPv6` o ambienti dual stack possono risolvere e connettersi a questi endpoint per cluster pubblici o privati.	21 ottobre 2024

AWS aggiornamenti delle politiche gestite	Aggiunte `autoscaling:ResumeProcesses` , `autoscaling:SuspendProcesses` e autorizzazioni associate a `AWSServiceRoleForAmazonEKSNodegroup` nelle Regioni della Cina per l'integrazione con Amazon Application Recovery Controller per EKS. Nessuna modifica in altre Regioni.	21 ottobre 2024
AL2023 accelerato AMIs	Ora puoi usare istanze accelerate `NVIDIA` e AWS Neuron per basate su. AMIs AL2023	11 ottobre 2024
Nuovo formato di origine	Abbiamo effettuato il passaggio a un nuovo formato di origine con alcune modifiche al layout. Stiamo risolvendo alcuni problemi di formattazione minori e temporanei.	10 ottobre 2024
AWS aggiornamenti delle politiche gestiti	Aggiunte autorizzazioni a `AmazonEKSServicePolicy` e `AmazonEKSServiceRolePolicy` . Aggiunte le autorizzazioni `ec2:GetSecurityGroupsForVpc` e tag associati per consentire a EKS di leggere le informazioni sui gruppi di sicurezza e aggiornare i tag correlati.	10 ottobre 2024

AWS aggiornamenti gestiti delle politiche - Nuova politica	EKS ha aggiunto una nuova politica AWS gestita.	3 ottobre 2024
Kubernetes versione `1.31`	Sono stati aggiunti il supporto per Kubernetes versione `1.31` per nuovi cluster e aggiornamenti della versione.	24 settembre 2024
AWS aggiornamenti delle politiche gestite: aggiornamento a una politica esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	21 agosto 2024
La versione 1.29 di Kubernetes è ora disponibile per i cluster locali su Outposts AWS	Ora puoi creare un cluster locale Amazon EKS su AWS Outposts utilizzando Kubernetes versione 1.29.	20 agosto 2024
EKS Pod Identity in (Stati Uniti) AWS GovCloud	Le Amazon EKS Pod Identity associano un ruolo IAM a un account di servizio Kubernetes. Grazie a questa funzionalità, non è più necessario fornire autorizzazioni estese al ruolo IAM del nodo. In questo modo, i Pod su quel nodo possono chiamare AWS APIs. A differenza dei ruoli IAM per gli account di servizio, le EKS Pod Identity sono completamente interne a EKS; non è necessario un gestore di identità OIDC.	14 agosto 2024
Aggiornamenti dei contenuti basati su scenari	Abbiamo rinominato e aggiornato gli argomenti affinché siano più basati sugli argomenti nell'intera guida.	9 agosto 2024

Endpoint di interfaccia VPC dual stack per Amazon EKS	Ora è possibile creare endpoint di interfaccia VPC dual stack per Amazon EKS con indirizzi IP e nomi DNS `IPv4` e `IPv6`.	7 agosto 2024
Nuovi endpoint dual-stack per Amazon EKS con indirizzi APIs `IPv6`	L'API EKS per la creazione e la gestione dei cluster e l'emittente URLs OIDC per i cluster dispongon o di nuovi endpoint dual-stack. Il nuovo nome DNS per l'API Amazon EKS è `eks.`*`region`*`.api.aws`, che si risolve in indirizzi `IPv4` e `IPv6`. I nuovi cluster hanno un nuovo URL dell'emittente OIDC dual stack (`oidc-eks.`*`region`*`.api.aws`).	1° agosto 2024
Blocchi di capacità per gruppi di nodi gestiti	Ora è possibile utilizzare blocchi di capacità per gruppi di nodi gestiti.	1° luglio 2024
Abilitazione della raccolta dei parametri di un gruppo Auto Scaling per impostazione predefinita	Nei gruppi di nodi gestiti Amazon EKS, i parametri dei gruppi Amazon EC2 Auto Scaling sono abilitati per impostazione predefini ta senza costi aggiuntivi. In precedenza, per abilitare questa funzionalità era necessario eseguire diversi passaggi.	28 giugno 2024

AWS aggiornamenti delle politiche gestite: aggiornamento a una politica esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	27 giugno 2024
Kubernetes versione `1.26`	La versione `1.26` di Kubernetes si trova ora nel supporto esteso.	12 giugno 2024
Miglioramenti ai riferimenti informativi sulle AMI	Sono stati apportati miglioramenti ai riferimenti informativi sulle AMI, soprattutto per Bottlerocket.	12 giugno 2024
Kubernetes versione `1.30`	Sono stati aggiunti il supporto per Kubernetes versione `1.30` per nuovi cluster e aggiornamenti della versione.	23 maggio 2024
Dimensionamento automatico di CoreDNS	L'autoscaler di CoreDNS adatterà dinamicamente il numero di repliche dell'implementazione CoreDNS in un cluster EKS in base al numero di nodi e core CPU. Questa funzionalità è disponibile per CoredNS `v1.9` e per la versione della piattaforma più recente di EKS versione `1.25` e successive.	14 maggio 2024
Aggiornamento della versione della piattaforma Amazon EKS	Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Ciò include nuove versioni di patch per Kubernetes `1.29.4`, `1.28.9` e `1.27.13`.	14 maggio 2024

CloudWatch Supporto per Container Insights per Windows	Il componente aggiuntivo Amazon CloudWatch Observability Operator ora consente anche Container Insights sui nodi di lavoro Windows del cluster.	10 aprile 2024
Concetti di Kubernetes	È stato aggiunto un nuovo argomento relativo ai concetti di Kubernetes.	5 aprile 2024
Ristrutturazione dei contenuti relativi ad accesso e IAM	Le pagine esistenti relative agli argomenti di accesso e IAM, come mappa di configurazione dell'autenticazione auth config map, voci di accesso, ID pod e IRSA, sono state spostate in una nuova sezione. Revisione del contenuto di panoramica.	2 aprile 2024
Supporto del sistema operativo Bottlerocket per il driver CSI di Amazon S3	Il driver CSI di Mountpoint per Amazon S3 è ora compatibile con Bottlerocket.	13 marzo 2024
AWS aggiornamenti gestiti delle policy: aggiornamento a una policy esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	4 marzo 2024
Amazon Linux 2023	Amazon Linux 2023 (AL2023) è un nuovo sistema operativo basato su Linux progettato per fornire un ambiente sicuro, stabile e ad alte prestazioni per le tue applicazioni cloud.	29 febbraio 2024

EKS Pod Identity e IRSA supportano i sidecar in Kubernetes 1.29	In Kubernetes 1.29, i container sidecar sono disponibili nei cluster Amazon EKS. I container sidecar sono supportati con ruoli IAM per gli account di servizio o EKS Pod Identity. Per ulteriori informazioni sui container sidecar, consultare Sidecar Containers nella documentazione di Kubernetes.	26 febbraio 2024
Kubernetes versione 1.29	Sono stati aggiunti il supporto per Kubernetes versione 1.29 per nuovi cluster e aggiornamenti della versione.	23 gennaio 2024
Versione completa: supporto esteso di Amazon EKS per le versioni Kubernetes	Il supporto esteso per le versioni Kubernetes consente di utilizzare una versione di Kubernetes specifica per più di 14 mesi.	16 gennaio 2024

Rilevamento dello stato di salute dei cluster Amazon AWS EKS nel cloud	Amazon EKS rileva i problemi nei cluster Amazon EKS e i prerequisiti necessari per l'infrastruttura del cluster in integrità del cluster. Puoi visualizzare i problemi con i tuoi cluster EKS nella Console di gestione AWS e all'interno `health` del cluster nell'API EKS. Questi problemi si aggiungono a quelli rilevati e visualizzati dalla console. In precedenza, l'integrità dei cluster era disponibile solo per i cluster locali su AWS Outposts.	28 dicembre 2023
Approfondimenti sui cluster	Ora è possibile ottenere suggerimenti relativi al cluster sulla base di controlli ricorrenti.	20 dicembre 2023
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibi le nella AWS regione Canada occidentale (Calgary) (`ca-west-1`).	20 dicembre 2023

Utilizzando le voci di accesso, ora è possibile concedere ai ruoli e agli utenti IAM l'accesso al cluster	Prima dell'introduzione delle voci di accesso, l'accesso al cluster veniva concesso ai ruoli e agli utenti IAM tramite l'aggiunta di voci a `aws-auth ConfigMap` . Adesso ogni cluster dispone di una modalità di accesso ed è possibile passare all'utilizzo di voci di accesso sulla base della propria pianificazione. Dopo aver cambiato modalità, puoi aggiungere utenti aggiungendo voci di accesso nella AWS CLI e in. AWS CloudFormation AWS SDKs	18 dicembre 2023
Aggiornamento della versione della piattaforma Amazon EKS	Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. Ciò include nuove versioni di patch per Kubernetes `1.28.4`, `1.27.8`, `1.26.11` e `1.25.16`.	12 dicembre 2023
Driver CSI di Mountpoint per Amazon S3	Ora è possibile installare il driver CSI di Mountpoint per Amazon S3 sui cluster Amazon EKS.	27 novembre 2023

Attivazione dei parametri Prometheus durante la creazione di un cluster	In Console di gestione AWS, ora puoi attivare le metriche di Prometheus durante la creazione di un cluster. È anche possibile visualizzare i dettagli dello scraper Prometheus nella scheda Osservabilità.	26 novembre 2023
Associazioni Amazon EKS Pod Identity	Le Amazon EKS Pod Identity associano un ruolo IAM a un account di servizio Kubernetes. Grazie a questa funzionalità, non è più necessario fornire autorizzazioni estese al ruolo IAM del nodo. In questo modo, i Pod su quel nodo possono effettuare chiamate. AWS APIs A differenza dei ruoli IAM per gli account di servizio, le EKS Pod Identity sono completamente interne a EKS; non è necessario un gestore di identità OIDC.	26 novembre 2023
AWS aggiornamenti gestiti delle politiche: aggiornamento a una politica esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	26 novembre 2023
Controller di snapshot CSI	Ora puoi installare il controller di snapshot CSI per utilizzarlo con i driver CSI compatibili, come il driver CSI di Amazon EBS.	17 novembre 2023

Riscrittura dell'argomento Operatore ADOT	Il supporto aggiuntivo Amazon EKS per la sezione ADOT Operator era ridondante con la Distro per la AWS documenta zione. OpenTelemetry Abbiamo migrato le informazi oni essenziali rimanenti verso quella risorsa per ridurre le informazioni obsolete e incoerenti.	14 novembre 2023
Supporto per il componente aggiuntivo CoreDNS EKS per i parametri Prometheus	Le versioni `v1.10.1-eksbuild.5` , `v1.9.3-eksbuild.9` e `v1.8.7-eksbuild.8` del componente aggiuntivo EKS per CoreDNS espongono la porta su cui CoreDNS ha pubblicato i parametri nel servizio `kube-dns`. Ciò semplifica l'inclusione dei parametri CoreDNS nei sistemi di monitoraggio.	10 novembre 2023
Componente aggiuntivo Amazon EKS CloudWatch Observability Operator	Aggiunta la pagina Amazon EKS CloudWatch Observabi lity Operator.	6 novembre 2023
Blocchi di capacità per istanze P5 autogestite nella Regione Stati Uniti orientali (Ohio)	Negli Stati Uniti orientali (Ohio), puoi ora utilizzare blocchi di capacità per istanze P5 autogestite.	31 ottobre 2023

I cluster supportano la modifica di sottoreti e gruppi di sicurezza	Puoi aggiornare il cluster per modificare le sottoreti e i gruppi di sicurezza da esso utilizzati. È possibile eseguire l' Console di gestione AWS aggiornamento dalla versione più recente della AWS CLI e dalla `eksctl` versione `v0.164.0-rc.0` o successiva. AWS CloudFormation Potrebbe essere necessario eseguire questa operazione per fornire alle sottoreti più indirizzi IP disponibili per aggiornare in modo corretto una versione del cluster.	24 ottobre 2023
Il ruolo del cluster e il ruolo del gruppo di nodi gestiti supportano le politiche di AWS Identity and Access Management gestite dal cliente	Puoi utilizzare una policy IAM personalizzata sul ruolo del cluster, anziché la EKSClusterpolicy AWS gestita da Amazon Policy. Inoltre, puoi utilizzare una policy IAM personalizzata sul ruolo del nodo in un gruppo di nodi gestiti, anziché la policy EKSWorker NodePolicy AWS gestita di Amazon. Esegui questa operazione per creare una policy con privilegio minimo per soddisfare i rigorosi requisiti di conformità.	23 ottobre 2023

Correzione del collegamento per l'installazione di eksctl	Dopo lo spostamento della pagina, correggi il collegamento di installazione per eksctl.	6 ottobre 2023
Versione di anteprima: supporto esteso di Amazon EKS per le versioni Kubernetes	Il supporto esteso per le versioni Kubernetes consente di utilizzare una versione di Kubernetes specifica per più di 14 mesi.	4 ottobre 2023
Rimuovi i riferimenti all'integrazione con AWS App Mesh	Le integrazioni di Amazon EKS con AWS App Mesh rimangono riservate solo ai clienti esistenti di App Mesh.	29 settembre 2023
Kubernetes versione `1.28`	Sono stati aggiunti il supporto per Kubernetes versione `1.28` per nuovi cluster e aggiornamenti della versione.	26 settembre 2023
I cluster esistenti supportano l'applicazione della policy di rete di Kubernetes nel plugin CNI di Amazon VPC per Kubernetes	È possibile utilizzare la policy di rete di Kubernetes nei cluster esistenti con il plugin CNI di Amazon VPC per Kubernetes anziché richiedere una soluzione di terze parti. È possibile utilizzare la policy di rete di Kubernetes nei cluster esistenti con il plugin CNI di Amazon VPC per Kubernetes anziché richiedere una soluzione di terze parti.	15 settembre 2023

Il componente aggiuntivo CoreDNS di Amazon EKS supporta la modifica di PDB	È possibile modificare il `PodDisruptionBudget` del componente aggiuntivo CoreDNS di EKS nelle versioni `v1.9.3-eksbuild.7` e successive e `v1.10.1-eksbuild.4` e successive.	15 settembre 2023
Supporto Amazon EKS per sottoreti condivise	Nuova sezione Requisiti e considerazioni condivisi per le sottoreti per creare cluster Amazon EKS in sottoreti condivise.	7 settembre 2023
Aggiornamenti a Cos'è Amazon EKS?	Sono stati aggiunti nuovi argomenti nelle sezioni Casi d'uso comuni e Architettura. Altri argomenti sono stati aggiornati.	6 settembre 2023
Applicazione della policy di rete di Kubernetes nel plugin CNI di Amazon VPC per Kubernetes	È possibile utilizzare la policy di rete di Kubernetes con il plugin CNI di Amazon VPC per Kubernetes anziché richieder e una soluzione di terze parti. È possibile utilizzare la policy di rete di Kubernetes con il plugin CNI di Amazon VPC per Kubernetes anziché richiedere una soluzione di terze parti.	29 agosto 2023
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella regione Israele (Tel Aviv) `il-central-1` (AWS).	1° agosto 2023

Archiviazione temporanea configurabile per Fargate	È possibile aumentare la quantità totale di spazio di archiviazione effimero per ciascun pod in esecuzione su Amazon EKS Fargate.	31 luglio 2023
Supporto aggiuntivo per il driver CSI per Amazon EBS	Ora puoi utilizzare la AWS CLI Console di gestione AWS e l'API per gestire il driver CSI di Amazon EFS.	26 luglio 2023
AWS aggiornamenti delle politiche gestite: nuova politica	Amazon EKS ha aggiunto una nuova policy AWS gestita.	26 luglio 2023
Gli aggiornamenti delle versioni di Kubernetes per 1.27, 1.26, 1.25 e 1.24 sono ora disponibili per i cluster locali su Outposts AWS	Gli aggiornamenti delle versioni di Kubernetes a 1.27.3, 1.26.6, 1.25.11 e 1.24.15 sono ora disponibili per i cluster locali su Outposts AWS	20 luglio 2023
Supporto dei prefissi IP per nodi Windows	L'assegnazione di prefissi IP ai nodi può consentire di ospitare un numero di gran lunga maggiore di pod sui nodi rispetto a quanti è possibile ospitarne assegnando singoli indirizzi IP secondari ai nodi.	06 luglio 2023
Driver CSI Amazon FSx per OpenZFS	Ora puoi installare il driver CSI Amazon FSx for OpenZFS sui cluster Amazon EKS.	30 giugno 2023
I pod sui nodi Linux nei cluster `IPv4` ora possono comunicare con gli endpoint `IPv6`.	Dopo aver assegnato un IPv6 indirizzo al nodo, l'indirizzo dei Pods è `IPv4` l'indirizzo di rete tradotto nell'`IPv6`indirizzo del nodo su cui è in esecuzione.	19 giugno 2023

Gruppi di nodi gestiti da Windows nelle regioni AWS GovCloud (Stati Uniti)	Nelle regioni AWS GovCloud (Stati Uniti), i gruppi di nodi gestiti di Amazon EKS possono ora eseguire contenitori Windows.	30 maggio 2023
Kubernetes versione `1.27`	Sono stati aggiunti il supporto per Kubernetes versione `1.27` per nuovi cluster e aggiornamenti della versione.	24 maggio 2023
Kubernetes versione `1.26`	Sono stati aggiunti il supporto per Kubernetes versione `1.26` per nuovi cluster e aggiornamenti della versione.	11 aprile 2023
gMSA senza dominio	Ora è possibile usare gMSA senza dominio con i pod Windows.	27 marzo 2023
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella AWS regione Asia Pacifico (Melbourne`ap-southeast-4`) ().	10 marzo 2023
Driver CSI di Amazon File Cache	Ora è possibile installare il driver CSI per Amazon File Cache sui cluster Amazon EKS.	3 marzo 2023
La versione 1.25 di Kubernetes è ora disponibile per i cluster locali su Outposts AWS	Ora è possibile creare un cluster locale Amazon EKS su un Outpost utilizzando le versioni `1.22`-`1.25` di Kubernetes.	1 marzo 2023

Kubernetes versione 1.25	Sono stati aggiunti il supporto per Kubernetes versione 1.25 per nuovi cluster e aggiornamenti della versione.	22 febbraio 2023
AWS aggiornamenti delle politiche gestite: aggiornamento a una politica esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	7 febbraio 2023
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nelle regioni Asia Pacifico (Hyderabad) (ap-south-2), Europa (Zurigo) (eu-central-2) ed Europa (Spagna) (eu-south-2). AWS	6 febbraio 2023
Versioni Kubernetes1.21: ora 1.24 sono disponibili per i cluster locali su Outposts. AWS	Ora è possibile creare un cluster locale Amazon EKS su un Outpost utilizzando le versioni 1.21-1.24 di Kubernetes. In precedenza, era disponibile solo la versione 1.21.	17 gennaio 2023
Amazon EKS ora supporta AWS PrivateLink	Puoi usare an AWS PrivateLink per creare una connessione privata tra il tuo VPC e Amazon EKS.	16 dicembre 2022
Supporto Windows per gruppi di nodi gestiti	Ora puoi utilizzare Windows per gruppi di nodi gestiti Amazon EKS.	15 dicembre 2022
I componenti aggiuntivi Amazon EKS di fornitori di software indipendenti sono ora disponibili nel Marketplace AWS	Ora puoi sfogliare e abbonarti ai componenti aggiuntivi Amazon EKS di fornitori di software indipendenti tramite Marketplace AWS .	28 novembre 2022

AWS aggiornamenti gestiti delle politiche: aggiornamento a una politica esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	17 novembre 2022
Kubernetes versione `1.24`	Sono stati aggiunti il supporto per Kubernetes versione `1.24` per nuovi cluster e aggiornamenti della versione.	15 novembre 2022
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella AWS regione Medio Oriente (Emirati Arabi Uniti) (`me-central-1`).	3 novembre 2022
AWS aggiornamenti gestiti delle politiche: aggiornamento a una politica esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	24 ottobre 2022
AWS aggiornamenti gestiti delle politiche: aggiornamento a una politica esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	20 ottobre 2022
I cluster locali su AWS Outposts sono ora disponibili	Ora puoi creare un cluster locale Amazon EKS su un Outpost.	19 settembre 2022
Quote basate su vCPU Fargate	Fargate sta passando da quote basate su pod a quote basate su vCPU.	8 settembre 2022
AWS aggiornamenti gestiti delle politiche: aggiornamento a una politica esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	31 agosto 2022

Monitoraggio dei costi	Ora Amazon EKS supporta Kubecost, che consente di monitorare i costi suddivisi per risorse Kubernetes compresi pod, nodi, namespace ed etichette.	24 agosto 2022
AWS aggiornamenti delle politiche gestite: nuova politica	Amazon EKS ha aggiunto una nuova policy AWS gestita.	24 agosto 2022
AWS aggiornamenti delle politiche gestite: nuova politica	Amazon EKS ha aggiunto una nuova policy AWS gestita.	23 agosto 2022
Aggiunta di tag alle risorse per la fatturazione	Aggiunto il supporto `aws:eks:cluster-name` generato per i tag di allocazione dei costi per tutti i cluster.	16 agosto 2022
Caratteri jolly del profilo Fargate	È stato aggiunto il supporto per i caratteri jolly del profilo Fargate nei criteri di selezione per i namespace, le chiavi e i valori delle etichette.	16 agosto 2022
Kubernetes versione `1.23`	Sono stati aggiunti il supporto per Kubernetes versione `1.23` per nuovi cluster e aggiornamenti della versione.	11 agosto 2022
Visualizza le risorse Kubernetes nel Console di gestione AWS	Ora è possibile visualizzare le informazioni relative alle risorse Kubernetes implementate nel cluster utilizzando la Console di gestione AWS.	3 maggio 2022

Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella AWS regione Asia Pacifico (Giacarta) (`ap-southeast-3`).	2 maggio 2022
Pagina Osservabilità e supporto per il componente aggiuntivo ADOT	Aggiunte la pagina Observability e AWS Distro for OpenTelemetry (ADOT).	21 aprile 2022
Kubernetes versione `1.22`	Sono stati aggiunti il supporto per Kubernetes versione `1.22` per nuovi cluster e aggiornamenti della versione.	4 aprile 2022
AWS aggiornamenti delle politiche gestite - Nuova politica	Amazon EKS ha aggiunto una nuova policy AWS gestita.	4 aprile 2022
Aggiunti i dettagli relativi all'applicazione di patch ai pod Fargate	Durante l'aggiornamento dei pod Fargate, Amazon EKS tenta innanzitutto di rimuovere i pod in base ai relativi budget di interruzione. È possibile creare regole di eventi per contrastare le espulsioni non riuscite prima che i pod vengano eliminati.	1 aprile 2022
Versione completa: supporto aggiuntivo per il driver CSI di Amazon EBS	Ora puoi utilizzare la AWS CLI Console di gestione AWS e l'API per gestire il driver CSI di Amazon EBS.	31 marzo 2022
AWS Aggiornamento dei contenuti di Outposts	Istruzioni per distribuire un cluster Amazon EKS su AWS Outposts.	22 marzo 2022

AWS aggiornamenti gestiti delle politiche: aggiornamento a una politica esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	21 marzo 2022
Supporto `containerd` Windows	Ora puoi selezionare il runtime `containerd` per i nodi Windows.	14 marzo 2022
Sono state aggiunte alcune considerazioni in merito ad Amazon EKS Connector alla documentazione di sicurezza	Descrive il modello di responsabilità condivisa in relazione ai cluster connessi.	25 febbraio 2022
Assegnazione di indirizzi `IPv6` a pod e servizi	Ora è possibile creare un cluster `1.21` o successivo che assegna gli indirizzi `IPv6` a pod e servizi.	6 gennaio 2022
AWS aggiornamenti gestiti delle politiche: aggiornamento a una politica esistente	Amazon EKS ha aggiornato una policy AWS gestita esistente.	13 dicembre 2021
Versione di anteprima: supporto del componente aggiuntivo del driver CSI per Amazon EBS	Ora puoi visualizzare l'anteprima utilizzando la Console di gestione AWS AWS CLI e l'API per gestire il driver CSI di Amazon EBS.	9 dicembre 2021
Supporto autoscaler Karpenter	Ora puoi utilizzare il progetto open source Karpenter per scalare automaticamente i nodi.	29 novembre 2021
Supporto filtro Fluent Bit Kubernetes nella registrazione di Fargate	Ora puoi usare il filtro Fluent Bit Kubernetes con la registrazione di Fargate.	10 novembre 2021

Supporto Windows disponibile nel piano di controllo	Il supporto Windows è ora disponibile nel piano di controllo. Non è più necessario attivarlo nel piano dati.	9 novembre 2021
Bottlerocket aggiunto come tipo di AMI per i gruppi di nodi gestiti	In precedenza, Bottlerocket era disponibile solo come opzione di nodo autogestito. Ora è possibile configurarlo come gruppo di nodi gestiti, riducendo lo sforzo necessari o per soddisfare i requisiti di conformità dei nodi.	28 ottobre 2021
DL1 supporto per i driver	Amazon Linux personalizzato AMIs ora supporta carichi di lavoro di deep learning per Amazon Linux 2. Questa abilitazione consente una configurazione generica on-premises o di base su cloud.	25 ottobre 2021
VT1 supporto video	Amazon Linux personalizzato AMIs ora supporta VT1 alcune distribuzioni. Questa abilitazi one pubblicizza i dispositivi Xilinx U30 sul cluster Amazon EKS.	13 settembre 2021
È ora disponibile Amazon EKS Connector	Puoi utilizzare Amazon EKS Connector per registrare e connettere qualsiasi cluster Kubernetes conforme AWS e visualizzarlo nella console Amazon EKS.	8 settembre 2021

È ora disponibile Amazon EKS Anywhere	Amazon EKS Anywhere è una nuova opzione di implement azione per Amazon EKS che consente di creare e gestire cluster Kubernetes on-premis es.	08 settembre 2021
Driver CSI Amazon FSx per NetApp ONTAP	È stato aggiunto un argomento che riassume il driver CSI Amazon FSx for NetApp ONTAP e fornisce collegame nti ad altri riferimenti.	2 settembre 2021
Ora i gruppi di nodi gestiti calcolano automaticamente il numero massimo di pod Amazon EKS per i nodi	I gruppi di nodi gestiti ora calcolano automaticamente il numero massimo di pod Amazon EKS per i nodi implementati senza un modello di avvio, o con un modello di avvio in cui non è stato specificato un ID AMI.	30 agosto 2021
Rimozione della gestione delle impostazioni dei componenti aggiuntivi da parte di Amazon EKS, senza rimuovere il software aggiuntivo Amazon EKS	Ora è possibile rimuovere un componente aggiuntivo Amazon EKS senza rimuovere il software aggiuntivo dal cluster.	20 agosto 2021
Creazione di pod multi-homed con Multus	È possibile aggiungere più interfacce di rete a un pod utilizzando Multus.	2 agosto 2021

Aggiunta di altri indirizzi IP ai nodi Linux Amazon EC2	È ora possibile aggiungere significativamente più indirizzi IP ai nodi Linux Amazon EC2. Ciò significa che è possibile eseguire una maggiore densità di pod su ciascun nodo. È ora possibile aggiungere significa tivamente più indirizzi IP ai nodi Linux Amazon EC2. Ciò significa che è possibile eseguire una maggiore densità di pod su ciascun nodo.	27 luglio 2021
`containerd` bootstrap del runtime	Amazon Linux Amazon Machine Image (AMI) con accelerazione ottimizzata per Amazon EKS ora contiene un flag bootstrap che puoi usare per abilitare il `containerd` runtime in Amazon EKS ottimizzato e Bottlerocket. AMIs Questo flag è disponibile in tutte le versioni Kubernetes supportate dell'AMI.	19 luglio 2021
Kubernetes versione `1.21`	Aggiunto il supporto per la versione `1.21` di Kubernetes.	19 luglio 2021
É stato aggiunto l'argomento relativo alle policy gestite	Un elenco di tutte le policy IAM gestite Amazon EKS e le modifiche apportate loro dal 17 giugno 2021.	17 giugno 2021
Utilizzo dei gruppi di sicurezza per i pod con Fargate	Ora è possibile usare gruppi di sicurezza per i pod con Fargate, oltre a utilizzarli con i nodi Amazon EC2.	1 giugno 2021

Sono stati aggiunti i componenti aggiuntivi CoreDNS e `kube-proxy` Amazon EKS	Amazon EKS può ora aiutarti a gestire i componenti aggiuntivi Amazon EKS `kube-proxy` e CoreDNS per il cluster.	19 maggio 2021
Kubernetes versione `1.20`	Sono stati aggiunti il supporto per Kubernetes versione `1.20` per nuovi cluster e aggiornamenti della versione.	18 maggio 2021
AWS Rilasciato il Load Balancer Controller `2.2.0`	Ora puoi utilizzare il AWS Load Balancer Controller per creare Elastic Load Balancer utilizzando istanze o destinazioni IP.	14 maggio 2021
Contaminazioni di nodi per i gruppi di nodi gestiti	Amazon EKS ora supporta l'aggiunta di note di contaminazione ai gruppi di nodi gestiti.	11 maggio 2021
Crittografia dei segreti per i cluster esistenti	Amazon EKS ora supporta l'aggiunta della crittografia dei segreti per i cluster esistenti.	26 febbraio 2021
Kubernetes versione `1.19`	Sono stati aggiunti il supporto per Kubernetes versione `1.19` per nuovi cluster e aggiornamenti della versione.	16 febbraio 2021
Amazon EKS ora supporta i provider di identità OpenID Connect (OIDC) come metodo per autenticare gli utenti nel cluster aggiornato alla versione `1.16` o successiva.	I provider di identità OIDC possono essere utilizzati con o in alternativa a AWS Identity and Access Management (IAM).	12 febbraio 2021

Visualizza le risorse dei nodi e dei carichi di lavoro in Console di gestione AWS	Ora è possibile visualizzare i dettagli sui nodi gestiti, autogestiti e Fargate e sui carichi di lavoro Kubernetes distribuiti nella Console di gestione AWS.	1 dicembre 2020
Implementazione dei tipi di istanza spot in un gruppo di nodi gestiti	È ora possibile implementare più tipi di istanza Spot o On-Demand a un gruppo di nodi gestito.	1 dicembre 2020
Ora Amazon EKS può gestire componenti aggiuntivi specifici per il tuo cluster	Puoi gestire i componenti aggiuntivi in autonomia o consentire ad Amazon EKS di controllare l'avvio e la versione di un componente aggiuntivo tramite l'API Amazon EKS.	1 dicembre 2020
Condivisione di un ALB su più ingressi	Ora puoi condividere un AWS Application Load Balancer (ALB) tra più ingressi Kubernetes. In passato, era necessario implementare un ALB separato per ogni ingresso.	23 ottobre 2020
Supporto destinazioni IP per NLB	È ora possibile implementare un Network Load Balancer con destinazioni IP. Ciò significa che è possibile utilizzare un NLB per bilanciare il carico di traffico di rete verso i pod Fargate e direttamente verso i pod in esecuzione sui nodi Amazon EC2.	23 ottobre 2020

Kubernetes versione 1.18	Sono stati aggiunti il supporto per Kubernetes versione 1.18 per nuovi cluster e aggiornamenti della versione.	13 ottobre 2020
Specificare un blocco CIDR personalizzato per l'assegnazione dell'indirizzo IP del servizio Kubernetes.	È ora possibile specificare un blocco CIDR personalizzato a cui Kubernetes assegna gli indirizzi IP di servizio.	29 settembre 2020
Assegnazione dei gruppi di sicurezza a singoli pod	Ora è possibile associare diversi gruppi di sicurezza ad alcuni dei singoli pod in esecuzione su molti tipi di istanza Amazon EC2.	09 settembre 2020
Distribuzione di Bottlerocket sui nodi	È ora possibile implementare i nodi che eseguono Bottlerocket.	31 agosto 2020
La possibilità di avviare i nodi Arm è generalmente disponibile	È possibile avviare i nodi Arm in gruppi di nodi gestiti e autogestiti.	17 agosto 2020
Modelli di avvio di gruppi di nodi gestiti e AMI personalizzate	È possibile implementare un gruppo di nodi gestito che utilizza un modello di avvio di Amazon EC2. Se lo si preferisce, il modello di avvio può specificare un'AMI personalizzata.	17 agosto 2020
Supporto EFS per AWS Fargate	Ora puoi usare Amazon EFS con AWS Fargate.	17 agosto 2020

Aggiornamento della versione della piattaforma Amazon EKS	Si tratta di una nuova versione della piattaforma con correzioni della sicurezza e miglioramenti. È incluso il supporto UDP per i servizi di tipo `LoadBalancer` quando si utilizzano Network Load Balancer con Kubernetes versione `1.15` o successiva. Per ulteriori informazioni, consulta il problema Allow UDP for AWS Network Load Balancer su. GitHub	12 agosto 2020
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nelle Regioni AWS Africa (Città del Capo) (`af-south-1`) ed Europa (Milano) (`eu-south-1`).	6 agosto 2020
Parametri di utilizzo Fargate	AWS Fargate fornisce metriche di CloudWatch utilizzo che forniscono visibilità sull'utilizzo delle risorse Fargate On-Demand da parte dell'account.	03 agosto 2020
Kubernetes versione `1.17`	Sono stati aggiunti il supporto per Kubernetes versione `1.17` per nuovi cluster e aggiornamenti della versione.	10 luglio 2020

Creazione e gestione delle risorse App Mesh da Kubernetes con il controller App Mesh per Kubernetes	Le risorse App Mesh possono essere create e gestite da Kubernetes. Inoltre, il controller inserisce automaticamente il proxy Envoy e i container init nei pod implementati.	18 giugno 2020
Amazon EKS ora supporta i nodi Inf1 Amazon EC2	É possibile aggiungere nodi Inf1 Amazon EC2 al cluster.	4 giugno 2020
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nelle regioni AWS GovCloud (Stati Uniti orientali) (`us-gov-east-1`) e AWS GovCloud (Stati Uniti occidentali) (`us-gov-west-1`) AWS .	13 maggio 2020
Kubernetes `1.12` non è più supportato in Amazon EKS	Kubernetes versione `1.12` non è più supportato in Amazon EKS. Esegui l'aggiornamento di eventuali cluster `1.12` alla versione `1.13` o successiva per evitare interruzioni del servizio.	12 maggio 2020
Kubernetes versione `1.16`	Sono stati aggiunti il supporto per Kubernetes versione `1.16` per nuovi cluster e aggiornamenti della versione.	30 aprile 2020
È stato aggiunto il ruolo collegato al servizio AWSServiceRoleForAmazonEKS	Aggiunto il ruolo collegato al servizio AWSServiceRoleForAmazonEKS.	16 aprile 2020

Kubernetes versione 1.15	Sono stati aggiunti il supporto per Kubernetes versione 1.15 per nuovi cluster e aggiornamenti della versione.	10 marzo 2020
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nelle Regioni AWS Pechino (cn-north-1) e Ningxia (cn-northwest-1).	26 febbraio 2020
FSx per il driver Lustre CSI	È stato aggiunto un argomento per l'installazione del driver CSI FSx for Lustre sui cluster Kubernetes 1.14 Amazon EKS.	23 dicembre 2019
Limitare l'accesso di rete all'endpoint di accesso pubblico di un cluster	Questo aggiornamento consente ad Amazon EKS di limitare gli intervalli CIDR che possono comunicare con l'endpoint di accesso pubblico del server API Kubernetes.	20 dicembre 2019
Risolvere l'indirizzo dell'endpoint di accesso privato per un cluster dall'esterno di un VPC	Con questo aggiornamento, è possibile utilizzare Amazon EKS per risolvere l'endpoint di accesso privato del server API Kubernetes dall'esterno di un VPC.	13 dicembre 2019
(Beta) Nodi di istanza Amazon EC2 A1 di Amazon EC2	Avvio di nodi di istanza Amazon EC2 A1 di Amazon EC2 che si registrano con il cluster Amazon EKS.	4 dicembre 2019
Creazione di un cluster su AWS Outposts	Amazon EKS ora supporta la creazione di cluster su AWS Outposts.	3 dicembre 2019

AWS Fargate su Amazon EKS	Ora i cluster Kubernetes Amazon EKS supportano i pod in esecuzione su Fargate.	3 dicembre 2019
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella AWS regione Canada (Centraleca-central-1) ().	21 novembre 2019
Gruppi di nodi gestiti	I gruppi di nodi gestiti Amazon EKS automatizzano il provisioning la gestione del ciclo di vita dei nodi (istanze Amazon EC2) per i cluster Kubernetes Amazon EKS.	18 novembre 2019
Aggiornamento della versione della piattaforma Amazon EKS	Nuove versioni della piattaforma per gestire CVE-2019-11253.	6 novembre 2019
Kubernetes 1.11 non è più supportato in Amazon EKS	Kubernetes versione 1.11 non è più supportato in Amazon EKS. Esegui l'aggiornamento di eventuali cluster 1.11 alla versione 1.12 o successiva per evitare interruzioni del servizio.	4 novembre 2019
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella AWS regione Sud America (San Paolosa-east-1) ().	16 ottobre 2019
Supporto Windows	Ora i cluster Amazon EKS che eseguono Kubernetes versione 1.14 ora supportano i carichi di lavoro di Windows.	7 ottobre 2019

Scalabilità automatica	É stato aggiunto il capitolo per descrivere alcuni dei vari tipi di scalabilità automatic a Kubernetes supportati nei cluster Amazon EKS.	30 settembre 2019
Aggiornamento della dashboard Kubernetes	Aggiornato l'argomento relativo all'installazione della dashboard Kubernetes nei cluster Amazon EKS per l'uso della versione beta `2.0`.	28 settembre 2019
Driver CSI per Amazon EFS	Aggiunto l'argomento relativo all'installazione del driver CSI per Amazon EFS nei cluster Amazon EKS con Kubernetes `1.14`.	19 settembre 2019
Parametro di Amazon EC2 Systems Manager per ID di AMI ottimizzate per Amazon EKS	É stato aggiunto l'argomen to relativo al recupero dell'ID AMI ottimizzata per Amazon EKS mediante un parametro Amazon EC2 Systems Manager. Il parametro elimina la necessità di cercare AMI IDs.	18 settembre 2019
Assegnazione di tag alle risorse di Amazon EKS	È possibile gestire l'assegna zione di tag ai propri cluster Amazon EKS.	16 settembre 2019
Driver CSI per Amazon EBS	Aggiunto l'argomento relativo all'installazione del driver CSI per Amazon EBS nei cluster Amazon EKS con Kubernetes `1.14`.	9 settembre 2019

Nuova AMI ottimizzata per Amazon EKS con patch per `CVE-2019-9512` e `CVE-2019-9514`	Amazon EKS ha aggiornato l'AMI ottimizzata per Amazon EKS per la gestione di CVE-2019-9512 e CVE-2019-9514.	6 settembre 2019
Annuncio della definizione di Kubernetes `1.11` come obsoleto in Amazon EKS	Amazon EKS ha interrotto il supporto per Kubernetes versione `1.11` il 4 novembre 2019.	4 settembre 2019
Kubernetes versione `1.14`	Sono stati aggiunti il supporto per Kubernetes versione `1.14` per nuovi cluster e aggiornamenti della versione.	3 settembre 2019
Ruoli IAM per gli account di servizio	Grazie ai ruoli IAM per gli account di servizio sui cluster Amazon EKS è possibile associare un ruolo IAM a un account di servizio Kubernetes. Grazie a questa funzionalità, non è più necessario fornire autorizzazioni estese al ruolo IAM del nodo. In questo modo, i Pod su quel nodo possono chiamare AWS APIs.	3 settembre 2019
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella regione Medio Oriente (Bahrein) (`me-south-1`). AWS	29 agosto 2019
Aggiornamento della versione della piattaforma Amazon EKS	Nuove versioni della piattaforma per gestire CVE-2019-9512 e CVE-2019-9514.	28 agosto 2019

Aggiornamento della versione della piattaforma Amazon EKS	Nuove versioni della piattaforma per gestire CVE-2019-11247 e CVE-2019-11249.	5 agosto 2019
Espansione delle Regioni di Amazon EKS	Amazon EKS è ora disponibile nella AWS regione Asia Pacifico (Hong Kongap-east-1) ().	31 luglio 2019
Kubernetes 1.10 non più supportato in Amazon EKS	Kubernetes versione 1.10 non è più supportato in Amazon EKS. Eseguire l'aggiornamento di ogni cluster 1.10 alla versione 1.11 o successiva per evitare interruzioni del servizio.	30 luglio 2019
Aggiunto argomento sul controller di ingresso ALB	L' AWS ALB Ingress Controller per Kubernetes è un controller che consente la creazione di un ALB quando vengono create risorse in ingresso.	11 luglio 2019
Nuova AMI ottimizzata per Amazon EKS	kubectlRimozione AMIs di file binari non necessari da.	3 luglio 2019
Kubernetes versione 1.13	Sono stati aggiunti il supporto per Kubernetes versione 1.13 per nuovi cluster e aggiornamenti della versione.	18 giugno 2019
Nuova AMI ottimizzata per Amazon EKS con patch per AWS-2019-005	Amazon EKS ha aggiornato le AMI ottimizzate per Amazon EKS per gestire le vulnerabilità descritte in link:security/security-bulletins/AWS-2019-005/ [AWS-2019-005,type="marketing"].	17 giugno 2019

Annuncio dell'interruzione del supporto per Kubernetes `1.10` in Amazon EKS	Amazon EKS ha interrotto il supporto per Kubernete s versione `1.10` il 22 luglio 2019.	21 maggio 2019
Aggiornamento della versione della piattaforma Amazon EKS	Nuova versione della piattafor ma per cluster Kubernetes `1.11` e `1.10` per supportare nomi DNS personalizzati nel certificato `kubelet` e migliorar e le prestazioni di `etcd`.	21 maggio 2019
Nozioni di base su `eksctl`	Questa guida alle operazion i di base ti consente di creare tutte le risorse necessarie per iniziare a utilizzare `eksctl` in Amazon EKS. Questa è una semplice utility a riga di comando per la creazione e la gestione di cluster di Kubernetes su Amazon EKS.	10 maggio 2019

AWS Comando CLI `get-token`	Il `aws eks get-token` comando è stato aggiunto alla AWS CLI. Non è più necessario installare AWS IAM Authenticator for Kubernetes per creare token di sicurezza client per la comunicazione tra server API del cluster. Aggiorna l'installazione della AWS CLI alla versione più recente per utilizzare questa nuova funzionalità. Per ulteriori informazioni, consulta Installazione dell'interfaccia a riga di AWS comando nella Guida per l'utente dell'interfaccia a riga di AWS comando.	10 maggio 2019
Aggiornamento della versione della piattaforma Amazon EKS	Nuova versione della piattaforma per cluster Kubernetes `1.12` per supportare nomi DNS personalizzati nel certificato `kubelet` e migliorare le prestazioni di `etcd`. La versione corregge un bug che costringeva i daemon `kubelet` dei nodi a richiedere un nuovo certificato a intervalli di pochi secondi.	8 maggio 2019
Tutorial su Prometheus	É stato aggiunto l'argomento relativo alla distribuzione di Prometheus nel cluster Amazon EKS.	5 aprile 2019

Registrazione del piano di controllo Amazon EKS	Con questo aggiornamento è possibile ottenere registri di verifica e di diagnostica direttamente dal piano di controllo di Amazon EKS. È possibile utilizzare questi CloudWatch registri nel proprio account come riferimento per la protezione e l'esecuzione dei cluster.	4 aprile 2019
Kubernetes versione `1.12`	Sono stati aggiunti il supporto per Kubernetes versione `1.12` per nuovi cluster e aggiornamenti della versione.	28 marzo 2019
É stata aggiunta la Guida alle operazioni di base di App Mesh	É stata aggiunta la documentazione per le nozioni di base su App Mesh e Kubernetes.	27 marzo 2019
Accesso privato all'endpoint del server API di Amazon EKS	Aggiunta la documentazione per la disabilitazione dell'accesso pubblico per l'endpoint del server API Kubernetes del cluster Amazon EKS.	19 marzo 2019
Aggiunto argomento per l'installazione del server dei parametri Kubernetes	Il server dei parametri Kubernetes è un aggregatore dei dati di utilizzo delle risorse nel cluster.	18 marzo 2019
Aggiunto elenco di progetti open source correlati	Questi progetti open source estendono la funzionalità dei cluster Kubernetes in esecuzione AWS, inclusi i cluster gestiti da Amazon EKS.	15 marzo 2019

Aggiunto argomento per l'installazione di Helm in locale	Il programma di gestione del pacchetto `helm` per Kubernetes consente di installare e gestire le applicazioni sul cluster Kubernetes. Questo argomento illustra come installare ed eseguire i binari `helm` e `tiller` localmente per consentire l'installazione e la gestione di grafici utilizzando la CLI helm nel sistema locale.	11 marzo 2019
Aggiornamento della versione della piattaforma Amazon EKS	Nuova versione della piattaforma per l'aggiornamento dei cluster Amazon EKS Kubernetes `1.11` al livello di patch `1.11.8` per risolvere CVE-2019-1002100.	8 marzo 2019
Limite cluster aumentato	Amazon EKS ha aumentato il numero di cluster che puoi creare in una AWS regione da 3 a 50.	13 febbraio 2019
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nelle AWS regioni Europa (Londra) (`eu-west-2`), Europa (Parigi) (`eu-west-3`) e Asia Pacifico (Mumbai) (`ap-south-1`).	13 febbraio 2019
Nuova AMI ottimizzata per Amazon EKS con patch per `ALAS-2019-1156`	Amazon EKS ha aggiornato le AMI ottimizzate per Amazon EKS per affrontare la vulnerabilità descritta in ALAS-2019-1156.	11 febbraio 2019

Nuova AMI ottimizzata per Amazon EKS con patch per `ALAS2-2019-1141`	Amazon EKS ha aggiornato l'AMI ottimizzata per Amazon EKS per rispondere a quelle a CVEs cui si fa riferimento in ALAS2-2019-1141.	9 gennaio 2019
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella AWS regione Asia Pacifico (Seoul) (`ap-northeast-2`).	9 gennaio 2019
Espansione delle Regioni di Amazon EKS	Amazon EKS è ora disponibile nelle seguenti AWS regioni aggiuntive: Europa (Francoforte`eu-central-1`) (), Asia Pacifico (Tokyo) (`ap-northeast-1`), Asia Pacifico (Singapore) (`ap-southeast-1`) e Asia Pacifico (Sydney) (`ap-southeast-2`).	19 dicembre 2018
Aggiornamenti del cluster Amazon EKS	É stata aggiunta la documentazione relativa all'aggiornamento della versione del cluster Kubernetes e per la sostituzione dei nodi per Amazon EKS.	12 dicembre 2018
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella AWS regione Europa (Stoccolma) (`eu-north-1`).	11 dicembre 2018

Aggiornamento della versione della piattaforma Amazon EKS	Nuova versione della piattaforma per l'aggiornamento di Kubernetes al livello di patch `1.10.11` per risolvere link:security/security-bulletins/AWS-2018-020/[CVE-2018-1002105,type="marketing"].	4 dicembre 2018
È stato aggiunto il supporto alla versione `1.0.0` per il controller in ingresso ALB	Il controller di ingresso ALB rilascia una versione `1.0.0` con supporto formale da. AWS	20 novembre 2018
Aggiunto supporto per la configurazione di rete CNI	Ora il plugin CNI di Amazon VPC per Kubernetes versione `1.2.1` supporta la configurazione di rete personalizzata per interfacce di rete di pod secondari.	16 ottobre 2018
È stato aggiunto il supporto per `MutatingAdmissionWebhook` e `ValidatingAdmissionWebhook`	La versione della piattaforma Amazon EKS `1.10-eks.2` ora supporta i controller `MutatingAdmissionWebhook` e di ammissione `ValidatingAdmissionWebhook`.	10 ottobre 2018
Sono state aggiunte le informazioni dell'AMI del partner	Canonical ha collaborato con Amazon EKS per creare nodi AMIs da utilizzare nei cluster.	3 ottobre 2018
Aggiunte istruzioni per il AWS comando CLI `update-kubeconfig`	Amazon EKS ha aggiunto il `update-kubeconfig` alla AWS CLI per semplificare il processo di creazione di un `kubeconfig` file per l'accesso al cluster.	21 settembre 2018

Nuovo Amazon EKS ottimizzato AMIs	Amazon EKS ha aggiornato Amazon EKS ottimizzato AMIs (con e senza supporto GPU) per fornire varie correzioni di sicurezza e ottimizzazioni AMI.	13 settembre 2018
Espansione AWS della regione Amazon EKS	Amazon EKS è ora disponibile nella Regione UE (Irlanda) (`eu-west-1`).	5 settembre 2018
Aggiornamento della versione della piattaforma Amazon EKS	Nuova versione della piattaforma con supporto per il livello di aggregazione Kubernetes e per Horizontal Pod Autoscaler(HPA).	31 agosto 2018
Nuovo supporto GPU ottimizzato per AMIs Amazon EKS	Amazon EKS ha aggiornato l'AMI ottimizzata per Amazon EKS per utilizzare un nuovo modello di AWS CloudFormation nodo e uno script di bootstrap. Inoltre, è disponibile una nuova AMI ottimizzata per Amazon EKS con supporto di GPU.	22 agosto 2018
Nuova AMI ottimizzata per Amazon EKS con patch per `ALAS2-2018-1058`	Amazon EKS ha aggiornato l'AMI ottimizzata per Amazon EKS per rispondere a quelle a CVEs cui si fa riferimento in ALAS2-2018-1058.	14 agosto 2018

Script della build AMI ottimizzata per Amazon EKS	Amazon EKS ha reso open-source gli script di build che vengono utilizzati per creare l'AMI ottimizzata per Amazon EKS. Questi script di build sono ora disponibili su. GitHub	10 luglio 2018
Versione iniziale di Amazon EKS	Documentazione iniziale per l'avvio del servizio	5 giugno 2018

Contribuire alla Guida per l’utente di EKS

AWS ha lanciato un’esperienza migliorata di contribuzione alla Guida per l’utente di EKS.

Ora è possibile modificare il codice sorgente della Guida per l’utente di EKS direttamente su GitHub.

I documenti ora utilizzano AsciiDoc, un potente linguaggio di authoring simile a markdown. AsciiDoc unisce una sintassi semplice a funzionalità di documentazione di livello aziendale come formattazione avanzata, riferimenti incrociati e controlli di sicurezza.

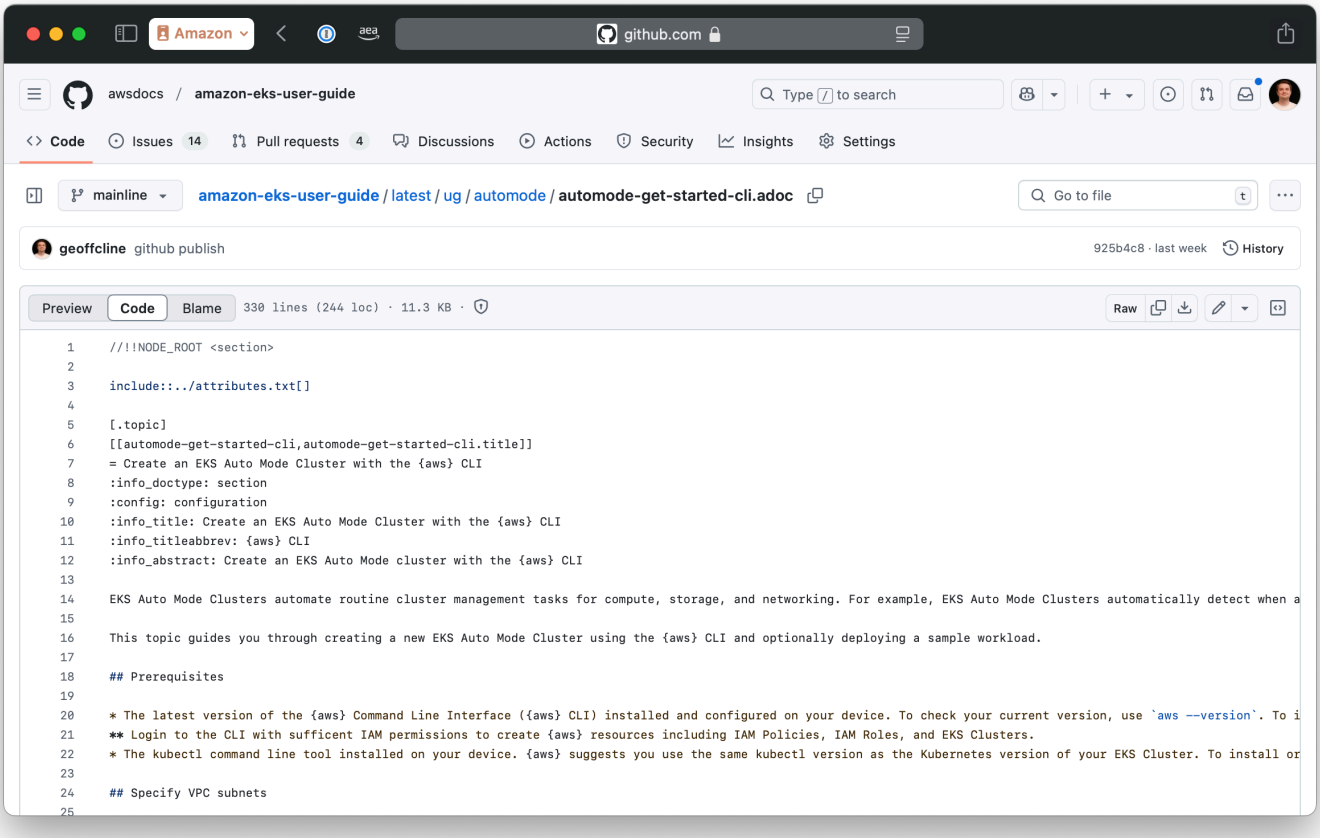
Ora è possibile modificare la documentazione EKS direttamente su GitHub. Il nostro processo ottimizzato include:

- Elaborazione più rapida delle richieste di pull.
- Fasi manuali ridotte.
- Controlli automatici della qualità dei contenuti.

Non vediamo l’ora di ricevere nuovi contributi.

Modificare una singola pagina da un browser web

È possibile modificare una singola pagina della Guida per l’utente di EKS con facilità direttamente tramite browser web.



Se si desidera modificare più pagine da browser web, consultare la pagina the section called "Modifica i file con GitHub".

Prerequisiti

- Pagina della documentazione da modificare aperta nel browser web.
- Accesso a GitHub effettuato.

Procedura

1. Accedere alla pagina da modificare nella Guida per l'utente di Amazon EKS.
2. Nel pannello a destra, scegliere il link Modifica questa pagina su GitHub.
3. Una volta su GitHub, aprire l'editor in uno dei due seguenti modi:
   - Premere il tasto e sulla tastiera.
   - Fare clic sull'icona a forma di matita e selezionare Modifica in posizione nel menu a discesa.

- Se non è disponibile l'opzione di modifica, è necessario accedere a GitHub. L'account GitHub non necessita di autorizzazioni particolari per suggerire modifiche. Tuttavia, i collaboratori interni di Amazon devono collegare il proprio profilo GitHub.

4. Apportare le modifiche necessarie al contenuto nell'editor di GitHub.
   - L'editor offre funzionalità di evidenziazione della sintassi e di anteprima.
   - È possibile utilizzare il markup AsciiDoc per formattare le modifiche.
   - È possibile utilizzare `ctrl-f` per aprire un'interfaccia di ricerca/sostituzione.
5. (Facoltativo) Visualizzare le modifiche in anteprima.
   - Utilizzare la scheda `preview` per visualizzare un'anteprima delle modifiche con formattazione avanzata.
   - Utilizzare l'opzione `show diff` per evidenziare le sezioni modificate. Le sezioni rimosse presentano un indicatore rosso nel margine sinistro. Le nuove sezioni presentano un indicatore verde nel margine sinistro.
6. Al termine della modifica, fare clic sul pulsante Commit delle modifiche… nella parte alta dell'editor.
7. Nella finestra di dialogo di commit:
   - Verificare che l'indirizzo e-mail sia corretto.
   - Aggiungere un messaggio di commit breve ma descrittivo che spieghi le modifiche.
   - Facoltativamente, aggiungere una descrizione più lunga, se necessario.
   - Selezionare per creare una nuova ramificazione e richiesta di pull.

È stata creata una richiesta di pull che include le modifiche proposte.

Panoramica della richiesta di pull

Quando si crea una richiesta pull:

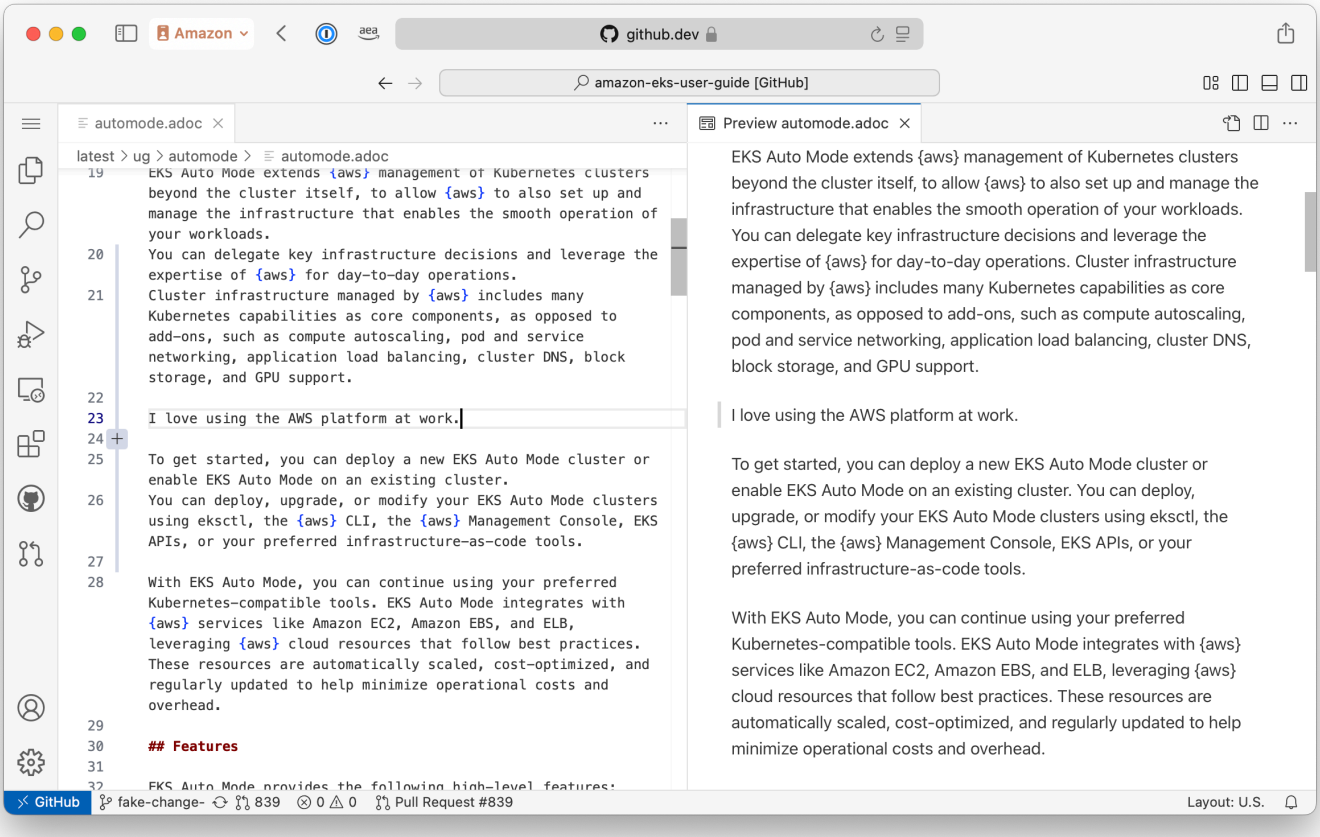
- Le modifiche vengono inviate e sottoposte a revisione da parte dei manutentori del repository.
- I revisori possono commentare i cambiamenti e richiedere modifiche.
- Per convalidare le modifiche potrebbero essere eseguiti test automatizzati.
- Una volta approvate, le modifiche possono essere unite al repository principale.

Le richieste di pull aiutano a garantire la qualità e offrono un modo per discutere le modifiche prima che vengano integrate.

Ulteriori informazioni su come vengono revisionate e approvate le richieste di pull nei GitHub Docs.

Modifica più file da un browser Web con GitHub Web Editor

Se vuoi proporre modifiche a più pagine o creare una nuova pagina di documenti, usa l'editor web GitHub .dev. basato sull'editor di testo popolare Visual Studio Code.



Prerequisiti

- Effettuato l'accesso a GitHub
- Familiarità con l'editor Visual Studio Code
- Familiarità con Git

Procedura

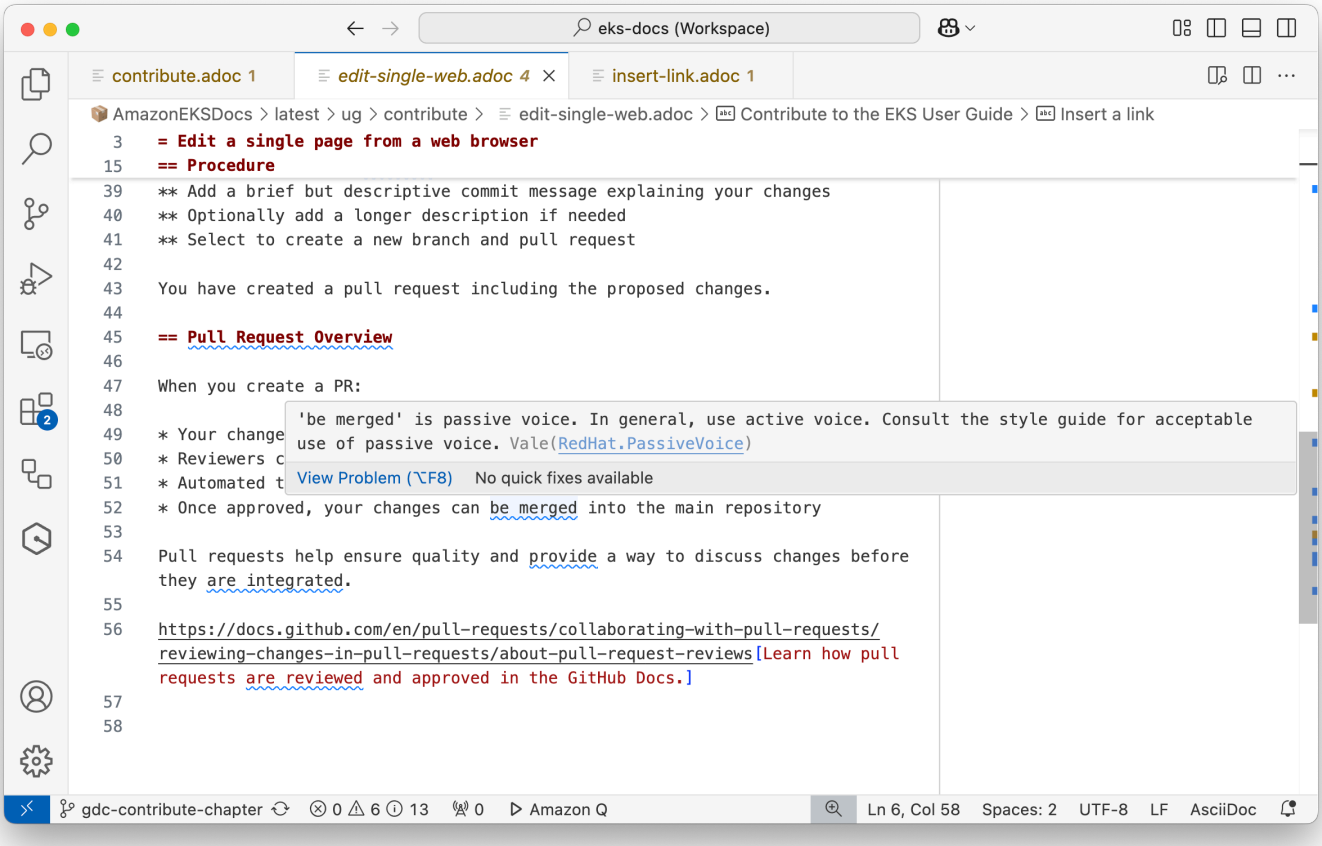
> **Note**
>
> Il team di EKS Docs ha creato un file di area di lavoro che include configurazioni suggerite per l'editor, come la disposizione del testo e l'evidenziazione della sintassi. AsciiDoc Suggeriamo di caricare questo file.

1. Apri lo spazio di lavoro su .dev. GitHub
   - È possibile aggiungere ai preferiti l'URL `https://github.dev/awsdocs/amazon-eks-user-guide/blob/mainline/eks-docs.code-workspace?workspace=true`.
2. (Solo per la prima configurazione) È possibile che ti venga richiesto di creare un fork del repository nel tuo account. GitHub Accettare questo prompt. Per ulteriori informazioni, consulta About forks nei documenti. GitHub
3. (Solo per la prima configurazione) Accetta la richiesta in basso a destra per installare l'estensione. AsciiDoc
4. Accedere ai contenuti della documentazione in `latest/ug`.
   - I file della documentazione sono organizzati per sezione principale. Ad esempio, i file di origine delle pagine del capitolo "Sicurezza" si troveranno nella directory "security/".
5. Per visualizzare un'anteprima della pagina di documentazione, utilizzare il pulsante Apri anteprima a lato in alto a destra. L'icona include una piccola lente d'ingrandimento.
6. Utilizzare la scheda Controllo del codice sorgente a sinistra per eseguire il commit delle modifiche. Per ulteriori informazioni, consultare le seguenti pagine della documentazione di Visual Studio Code:
   - Commit changes
   - Creazione di una richiesta di pull

Una volta creata una richiesta di pull, sarà revisionata dal team di documentazione.

Visualizzazione di un feedback sullo stile durante la digitazione installando Vale localmente

È possibile vedere il feedback sullo stile durante la digitazione. Ciò aiuta a identificare la scrittura scorretta e gli errori di battitura.



Panoramica:

- CLI Vale carica le guide di stile e le avvia sui file sorgente.
- Il repository EKS Docs include un file di configurazione Vale che carica guide di stile e regole locali.
- L'estensione Vale per Visual Studio (VS) Code mostra il feedback sui valori all'interno dell'editor.

Installa Java

Segui le istruzioni nei documenti della CLI Vale per installare Vale con un sistema di gestione pacchetti.

Installa l'estensione VS Code Vale

1. Apri VS Code.
2. Fai clic sull'icona delle estensioni nella barra delle attività (o premi Ctrl+Shift+X).
3. Cerca "Vale".
4. Fai clic su Installa sull'estensione "Vale VSCode" di Chris Chinchilla.
5. Ricarica VS Code quando richiesto.

Sincronizzazione di Vale

Vale utilizza il file di configurazione di `.vale.ini` nella root del progetto per determinare quali regole di stile applicare.

1. Apri VS Code.
2. Fai clic su Visualizza > Terminale (o premi Ctrl+`).
3. Se necessario, accedi alla directory principale del progetto.
4. Eseguire il comando :

```
vale sync
```

5. Attendi che Vale finisca di scaricare e sincronizzare le regole di stile

Visualizza il feedback di stile in VS Code

1. Apri un file Markdown o AsciiDoc in VS Code.
2. L'estensione Vale controllerà automaticamente il testo in base alle regole di stile.
3. I problemi di stile saranno sottolineati nell'editor.
4. Passa il mouse sul testo sottolineato per visualizzare il suggerimento di stile specifico.
5. Risolvi i problemi seguendo i suggerimenti o consultando la guida di stile.

Creazione di una nuova pagina

Scopri come creare una nuova pagina di documentazione. Questo argomento include le istruzioni per creare i metadati della pagina iniziale e aggiungere la pagina all'indice della guida.

Creazione di una pagina

1. Passa alla directory del capitolo. Ad esempio, se vuoi creare una nuova pagina nella sezione "Sicurezza", vai alla directory `latest/ug/security`.
2. Determina l'ID della pagina. Per convenzione, l'ID della pagina è tutto minuscolo e segmentato con `-`. L'ID di questa pagina è `create-page`.
3. Crea un nuovo file con l'ID della pagina e l'estensione `adoc`. Ad esempio, `create-page.adoc`.
4. Inserisci i metadati della pagina usando questo modello:

```
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.
```

Aggiungi pagina alla navigazione

1. Passa alla pagina principale. La pagina principale delle sezioni di primo livello è `book.adoc`.
2. Nella parte inferiore della pagina principale, inserisci la pagina secondaria.

   `include::${filename}[leveloffset=+1]`

   Ad esempio:

   `include::create-page.adoc[leveloffset=+1]`

Inserire un link

AsciiDoc supporta diversi tipi di link. L'utilizzo del tipo di collegamento corretto è importante affinché quest'ultimo funzioni correttamente in diversi ambienti.

Collegamento a una pagina o sezione della Guida per l’utente EKS

Utilizza i riferimenti incrociati (xref) per collegare pagine o sezioni all’interno dello stesso sito di documentazione, come la Guida per l’utente EKS. Si aggiornano automaticamente se la sezione di destinazione viene spostata o rinominata.

Usa il titolo della pagina come testo del link

Nella maggior parte dei casi, quando ti colleghi a un altro ID in questa guida per l’utente, utilizza il seguente approccio per fare in modo che il testo del link venga aggiornato automaticamente al titolo più recente, se necessario.

```
For more information, see <<page-or-section-id>>.
```

Definisci il testo del collegamento personalizzato

Nei casi in cui è necessario disporre di un testo di collegamento personalizzato, utilizza il seguente formato.

```
Here's an example of a <<page-or-section-id,link with custom text>>.
```

Collegamento a un'altra guida nei AWS Documenti

1. Trova il link alla pagina della AWS documentazione.
2. Rimuovi il prefisso `https://docs.aws.amazon.com/`, mantenendo solo il percorso. Il percorso deve iniziare con una lettera.
3. Crea un link come illustrato qui di seguito:

```
link:AmazonS3/latest/userguide/create-bucket-overview.html[Create a bucket,
 type="documentation"]
```

Collegamento a una pagina Web esterna

Questo formato crea un collegamento standard a una pagina non ospitata da Amazon. Ad esempio, usalo per GitHub i link.

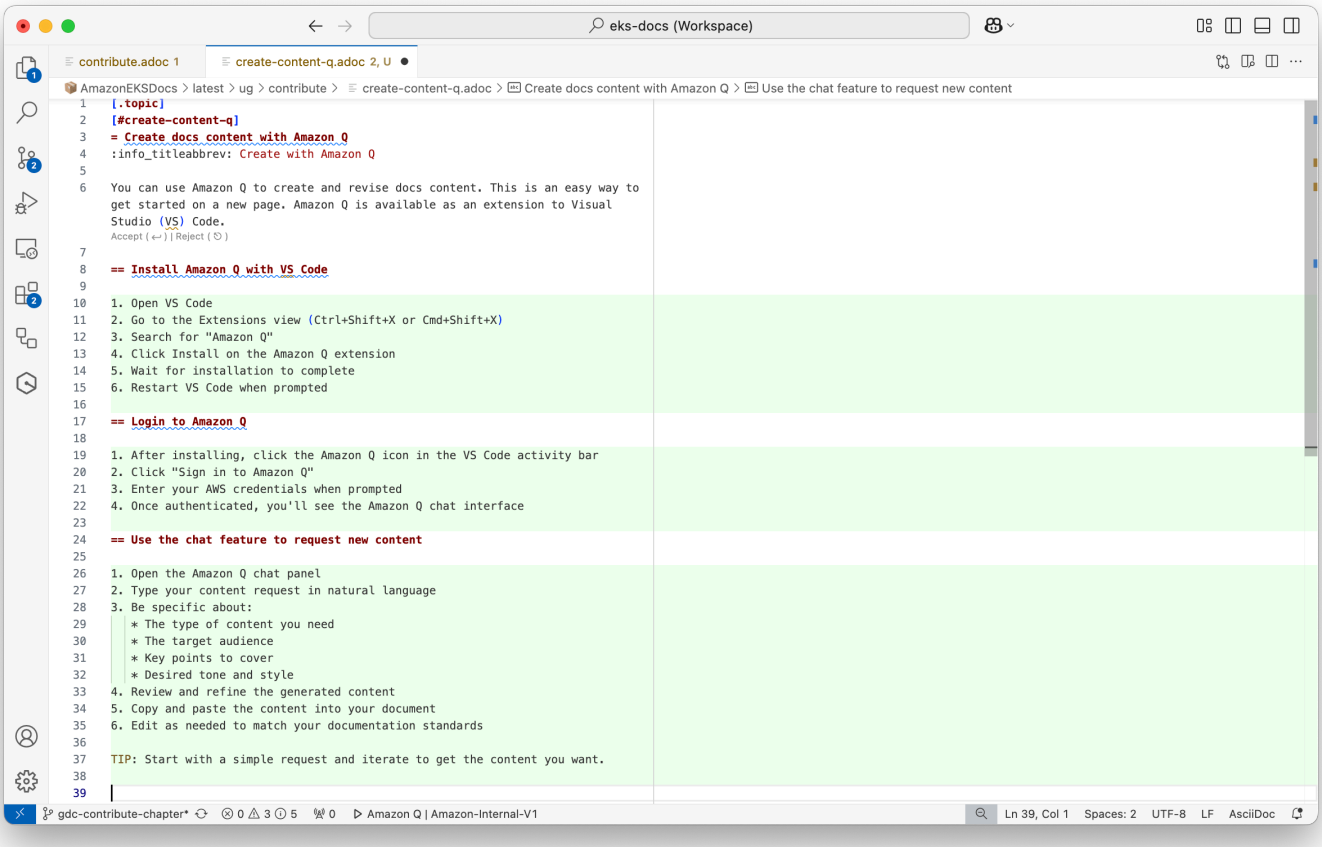
```
https://example.com[Link text]
```

> **Note**
>
> Abbiamo una lista consentita per i domini esterni. La lista consentita si trova in `vale/styles/EksDocs/ExternalDomains.yml`

Creare contenuti per documenti con Amazon Q

È possibile utilizzare Amazon Q per creare e modificare il contenuto dei documenti. Si tratta di un modo semplice per iniziare a scrivere una nuova pagina. Amazon Q è disponibile come estensione in Visual Studio (VS) Code.

Nell'immagine seguente, Amazon Q ha generato le righe evidenziate in verde.



Installare Amazon Q con VS Code

1. Aprire VS Code.

2. Andare alla vista Estensioni (Ctrl+Shift+X o Cmd+Shift+X).
3. Cercare "Amazon Q".
4. Fare clic su Installa sull'estensione Amazon Q.
5. Attendere il completamento dell'installazione.
6. Riavviare VS Code quando richiesto.

Accedere ad Amazon Q

1. Dopo l'installazione, scegliere l'icona Amazon Q nella barra di attività di VS Code.
2. Scegliere Accedi ad Amazon Q.
3. Inserire le credenziali AWS quando richiesto.
4. Dopo l'autenticazione, verrà visualizzata l'interfaccia di chat di Amazon Q.

Utilizzare Amazon Q per creare i contenuti

1. Aprire il file da modificare in VS Code.
2. Selezionare il testo da revisionare o la posizione per i nuovi contenuti.
3. Premere Ctrl+I o Cmd+I.
4. Nel prompt, specificare con precisione:
   - Il tipo di contenuto necessario.
   - I destinatari.
   - I punti principali da trattare.
   - Registro e stile desiderati.
5. Revisionare il contenuto generato nell'anteprima in linea.
6. Utilizzare invio per accettare le modifiche o esc per rifiutarle.
7. Apportare ulteriori modifiche se necessario.

Suggerimenti

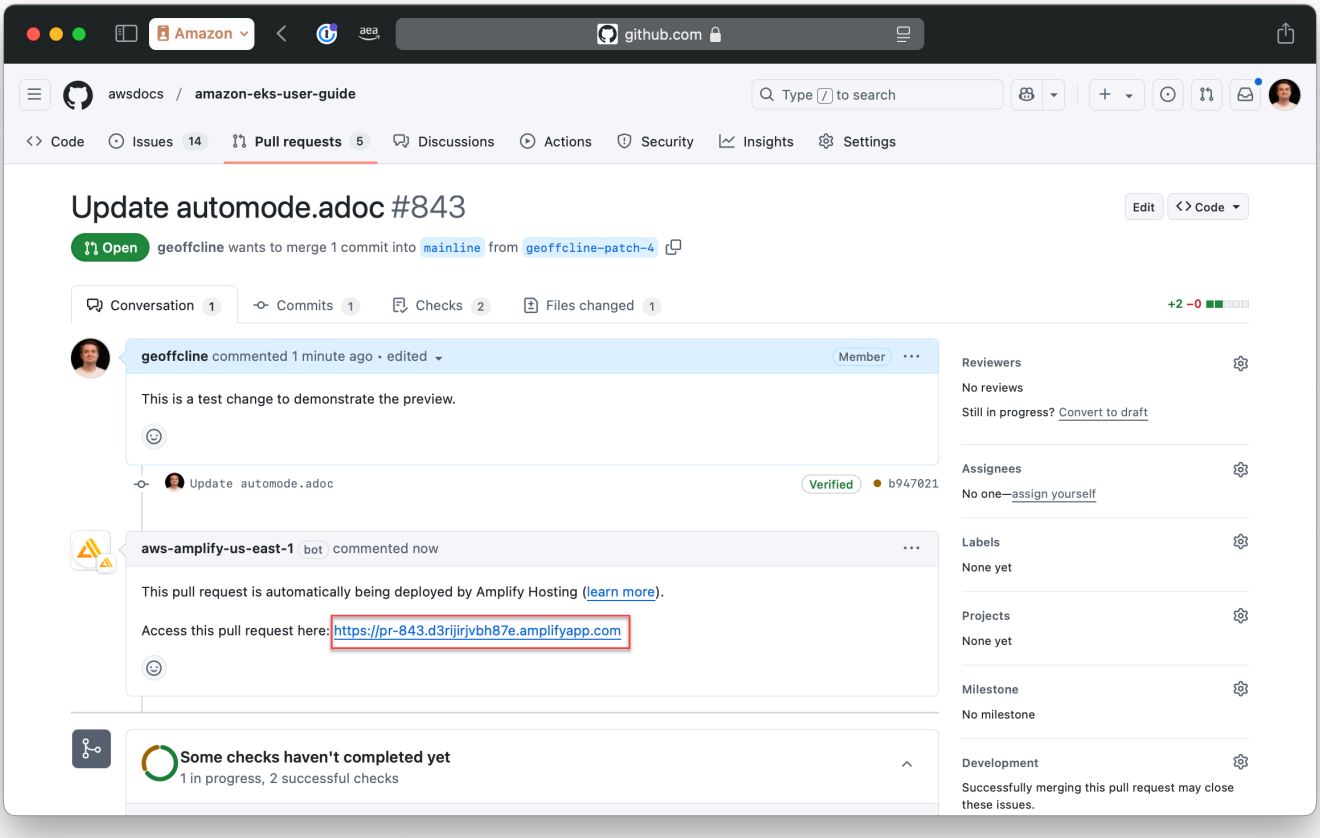
- Iniziare da una richiesta semplice e iterare per ottenere il contenuto desiderato.
- Creare una prima bozza dei titoli delle pagine, poi chiedere a Q di completarli.

- Amazon Q potrebbe generare output in Markdown. Questo linguaggio è accettabile. Gli strumenti AsciiDoc possono comprendere gran parte della sintassi markdown.

Per ulteriori informazioni su Amazon Q Developer, consultare la pagina Utilizzo di Amazon Q Developer nell'IDE.

Visualizzazione di un'anteprima del contenuto della richiesta di pull

GitHub della guida per l'utente Amazon EKS è configurata per creare e generare un'anteprima del sito dei documenti. Questa anteprima non contiene il tema AWS completo, ma verifica la corretta compilazione dei contenuti e il funzionamento dei link.



Questa anteprima è ospitata su un URL temporaneo da AWS Amplify.

Visualizzazione dell'anteprima

Quando invii una richiesta di pull, AWS Amplify tenta di creare e distribuire un'anteprima del contenuto.

Se la compilazione ha esito positivo, aws-amplify-us-east-1 aggiunge un commento alla richiesta di pull con un collegamento all'anteprima. Scegli il link a destra di "Accedi a questa richiesta di pull qui" (come indicato nello screenshot con un contorno rosso).

Se la compilazione ha esito negativo, gli amministratori del repository possono vedere i log e fornire feedback.

> **Note**
>
> Se non hai mai contribuito prima, potrebbe essere necessario che un responsabile del progetto approvi l'esecuzione della build.

Limitazioni dell'anteprima

L'anteprima è creata come un unico file HTML di grandi dimensioni. Una volta pubblicato, sarà visualizzato come più pagine.

Cosa funziona:

- Riferimenti incrociati (`xref`)
- Link a Internet
- Immagini
- Contenuti ospitati da `samples/`

Cosa non funziona:

- Link ad altri contenuti AWS, utilizzando `type="documentation"`. Ciò accade perché questo contenuto non esiste nell'ambiente di anteprima.
- L'attributo `{aws}` non sarà visualizzato correttamente. Il valore di questo cambia in base all'ambiente.

Riferimento alla sintassi AsciiDoc

AsciiDoc è il linguaggio di markup preferito per la documentazione AWS. Sebbene lo strumento supporti parzialmente la sintassi Markdown (come intestazioni ed elenchi), consigliamo di utilizzare la sintassi AsciiDoc per tutti i contenuti al fine di garantire la coerenza e il corretto rendering.

Questa guida illustra gli elementi di sintassi più comuni necessari per contribuire alla documentazione di Amazon EKS. Per una sintassi più avanzata e informazioni dettagliate, consultare AsciiDoc documentation.

Nuova pagina

Ogni nuovo documento AsciiDoc deve iniziare con la struttura definita in the section called "Creazione di una pagina".

Intestazioni

Usa le intestazioni per organizzare i contenuti in modo gerarchico:

```
= Page/topic title (level 1)
== Section title (level 2)
=== Level 3 heading
==== Level 4 heading
===== Level 5 heading
====== Level 6 heading
```

Nota: usare sempre le lettere maiuscole e minuscole per le intestazioni nella documentazione di AWS.

Formattazione del testo

Formattare il testo per enfatizzare le informazioni importanti:

```
Use *bold text* for UI labels.
Use _italic text_ for introducing terms or light emphasis.
Use `monospace text` for code, file names, and commands.
```

Elenchi

Elenchi non ordinati

Creare elenchi puntati per gli elementi senza una sequenza specifica:

```
* First item
* Second item
** Nested item
** Another nested item
* Third item
```

Elenchi ordinati

Creare elenchi numerati per fasi sequenziali o elementi con priorità:

```
. First step
. Second step
.. Substep 1
.. Substep 2
. Third step
```

Link

Consultare the section called "Inserire link" per informazioni su come formattare correttamente i link. I link in stile Markdown non sono supportati.

Esempi di codice

Codice inline

Usare gli apici retroversi per il codice inline:

```
Use the `kubectl get pods` command to list all pods.
```

Blocchi di codice

Creare blocchi di codice con evidenziazione della sintassi e supporto per gli attributi (simili alle entità):

```
 [source,python,subs="verbatim,attributes"]
```

```
----
def hello_world():
    print("Hello, World!")
----
```

Immagini

Inserire immagini con testo alternativo per l'accessibilità:

```
image::images/image-file.png[Alt text description]
```

Tabelle

Creare tabelle per organizzare le informazioni:

```
[%header,cols="2"]
|===
|Header 1
|Header 2

|Cell 1,1
|Cell 1,2

|Cell 2,1
|Cell 2,2
|===
```

Per tabelle più complesse, consultare AsciiDoc table documentation.

Callout

Usare i callout per evidenziare informazioni e ammonizioni importanti:

```
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.
```

Anteprima:

> **Note**
>
> Questo è un callout di nota.

Includere altri file

Includere contenuti da altri file:

```
include::filename.adoc[]
```

Attributi (simili alle entità)

Utilizzare attributi predefiniti per mantenere la coerenza. In particolare, DEVI utilizzare gli attributi per 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
----
```

Per un elenco degli attributi, cercare nel file `../attributes.txt`.

Procedure

Procedure dettagliate per il 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*.

...
```