Amazon EKS

Guida per l'utente di Eksctl

Guida per l'utente di Eksctl: Amazon EKS

Table of Contents

Che cos'è Eksctl?

eksctl è uno strumento di utilità a riga di comando che automatizza e semplifica il processo di creazione, gestione e funzionamento dei cluster Amazon Elastic Kubernetes Service (Amazon EKS). Scritto in Go, eksctl fornisce una sintassi dichiarativa tramite configurazioni YAML e comandi CLI per gestire operazioni complesse del cluster EKS che altrimenti richiederebbero più passaggi manuali su diversi servizi AWS.

eksctl è particolarmente utile per DevOps ingegneri, team di piattaforma e amministratori di Kubernetes che devono implementare e gestire in modo coerente i cluster EKS su larga scala. È particolarmente utile per le organizzazioni che stanno passando da Kubernetes autogestito a EKS o per quelle che implementano pratiche di infrastruttura come codice (IaC), in quanto può essere integrato nelle pipeline e nei flussi di lavoro di automazione esistenti. CI/CD Lo strumento elimina molte delle interazioni complesse tra i servizi AWS necessari per la configurazione del cluster EKS, come la configurazione VPC, la creazione di ruoli IAM e la gestione dei gruppi di sicurezza.

Le caratteristiche principali di eksctl includono la possibilità di creare cluster EKS completamente funzionali con un solo comando, il supporto per configurazioni di rete personalizzate, la gestione automatizzata dei gruppi di nodi e l'integrazione del flusso di lavoro. GitOps Lo strumento gestisce gli aggiornamenti dei cluster, ridimensiona i gruppi di nodi e gestisce la gestione dei componenti aggiuntivi tramite un approccio dichiarativo. eksctl fornisce anche funzionalità avanzate come la configurazione del profilo Fargate, la personalizzazione dei gruppi di nodi gestiti e l'integrazione di istanze spot, pur mantenendo la compatibilità con altri strumenti e servizi AWS tramite l'integrazione nativa di AWS SDK.

Funzionalità

Le funzionalità attualmente implementate sono:

- Creare, ottenere, elencare ed eliminare i cluster
- Crea, svuota ed elimina i gruppi di nodi
- Ridimensiona un gruppo di nodi
- Aggiornamento di un cluster
- Usa personalizzato AMIs
- Configurazione della rete VPC
- Configura l'accesso agli endpoint API

- Support per gruppi di nodi GPU
- Istanze spot e istanze miste
- Politiche di gestione e componenti aggiuntivi di IAM
- Elenca gli stack di formazione del cluster Cloudformation
- Installa cordens
- Scrivi il file kubeconfig per un cluster

Domande frequenti su Eksctl

Ambito generale

Posso usarlo **`eksctl`** per gestire cluster che non sono stati creati da? **`eksctl`**

Sì Dalla versione `0.40.0` è possibile eseguire su qualsiasi `eksctl` cluster, indipendentemente dal fatto che sia stato creato da `eksctl` o meno. Per ulteriori informazioni, consulta the section called “Cluster non creati da eksctl”.

Gruppi di nodi

Come posso cambiare il tipo di istanza del mio gruppo di nodi?

Dal punto di vista di`eksctl`, i gruppi di nodi sono immutabili. Ciò significa che una volta creato, l'unica cosa che `eksctl` può fare è scalare il gruppo di nodi verso l'alto o verso il basso.

Per cambiare il tipo di istanza, crea un nuovo gruppo di nodi con il tipo di istanza desiderato, quindi drenalo in modo che i carichi di lavoro si spostino su quello nuovo. Una volta completato questo passaggio, puoi eliminare il vecchio gruppo di nodi.

Come posso vedere i dati utente generati per un gruppo di nodi?

Per prima cosa avrai bisogno del nome dello stack Cloudformation che gestisce il nodegroup:

```
eksctl utils describe-stacks --region=us-west-2 --cluster NAME
```

Vedrai un nome simile a. `eksctl-CLUSTER_NAME-nodegroup-NODEGROUP_NAME`

È possibile eseguire quanto segue per ottenere i dati utente. Nota l'ultima riga che decodifica da base64 e decomprime i dati compressi con gzip.

```
NG_STACK=eksctl-scrumptious-monster-1595247364-nodegroup-ng-29b8862f # your stack here
LAUNCH_TEMPLATE_ID=$(aws cloudformation describe-stack-resources --stack-name $NG_STACK
 \
| jq -r '.StackResources | map(select(.LogicalResourceId == "NodeGroupLaunchTemplate")
 \
| .PhysicalResourceId)[0]')
aws ec2 describe-launch-template-versions --launch-template-id $LAUNCH_TEMPLATE_ID \
| jq -r '.LaunchTemplateVersions[0].LaunchTemplateData.UserData' \
| base64 -d | gunzip
```

Ingresso

Come posso configurare l'ingresso con? **`eksctl`**

Ti consigliamo di utilizzare il controller AWS Load Balancer. La documentazione su come distribuire il controller nel cluster e su come migrare dal vecchio ALB Ingress Controller è disponibile qui.

Per il Nginx Ingress Controller, la configurazione sarebbe la stessa di qualsiasi altro cluster Kubernetes.

Kubectl

Sto usando un proxy HTTPS e la convalida del certificato del cluster non riesce, come posso usare il sistema? CAs

Imposta la variabile di ambiente `KUBECONFIG_USE_SYSTEM_CA` per `kubeconfig` rispettare le autorità di certificazione del sistema.

Corsa a secco

La funzione dry-run consente di ispezionare e modificare le istanze corrispondenti al selettore di istanze prima di procedere alla creazione di un gruppo di nodi.

Quando `eksctl create cluster` viene chiamato con le opzioni del selettore di istanza e`--dry-run`, eksctl produrrà un ClusterConfig file contenente un nodegroup che rappresenta le opzioni CLI e i tipi di istanza impostati sulle istanze corrispondenti ai criteri di risorsa del selettore di istanze.

```
eksctl create cluster --name development --dry-run


apiVersion: eksctl.io/v1alpha5
```

```
cloudWatch:
  clusterLogging: {}
iam:
  vpcResourceControllerPolicy: true
  withOIDC: false
kind: ClusterConfig
managedNodeGroups:
- amiFamily: AmazonLinux2
  desiredCapacity: 2
  disableIMDSv1: true
  disablePodIMDS: false
  iam:
    withAddonPolicies:
      albIngress: false
      appMesh: false
      appMeshPreview: false
      autoScaler: false
      certManager: false
      cloudWatch: false
      ebs: false
      efs: false
      externalDNS: false
      fsx: false
      imageBuilder: false
      xRay: false
  instanceSelector: {}
  instanceType: m5.large
  labels:
    alpha.eksctl.io/cluster-name: development
    alpha.eksctl.io/nodegroup-name: ng-4aba8a47
  maxSize: 2
  minSize: 2
  name: ng-4aba8a47
  privateNetworking: false
  securityGroups:
    withLocal: null
    withShared: null
  ssh:
    allow: false
    enableSsm: false
    publicKeyPath: ""
  tags:
    alpha.eksctl.io/nodegroup-name: ng-4aba8a47
    alpha.eksctl.io/nodegroup-type: managed
```

```
  volumeIOPS: 3000
  volumeSize: 80
  volumeThroughput: 125
  volumeType: gp3
metadata:
  name: development
  region: us-west-2
  version: "1.24"
privateCluster:
  enabled: false
vpc:
  autoAllocateIPv6: false
  cidr: 192.168.0.0/16
  clusterEndpoints:
    privateAccess: false
    publicAccess: true
  manageSharedNodeSecurityGroupRules: true
  nat:
    gateway: Single
```

Il `eksctl create cluster` generato può quindi essere passato a: ClusterConfig

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

Quando un ClusterConfig file viene passato con`--dry-run`, eksctl emetterà un ClusterConfig file contenente i valori impostati nel file.

Opzioni una tantum in eksctl

Ci sono alcune opzioni una tantum che non possono essere rappresentate nel `ClusterConfig` file, ad esempio `--install-vpc-controllers`

Si prevede che:

```
eksctl create cluster --<options...> --dry-run > config.yaml
```

seguita da:

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

sarebbe equivalente a eseguire il primo comando senza`--dry-run`.

eksctl quindi non consente il passaggio di opzioni che non possono essere rappresentate nel file di configurazione quando vengono passate. `--dry-run`

> **Important**
>
> Se devi passare un profilo AWS, imposta la variabile di `AWS_PROFILE` ambiente anziché passare l'opzione `--profile` CLI.

Tutorial

Questo argomento illustra come installare e configurare eksctl, quindi utilizzarlo per creare un cluster Amazon EKS.

Passaggio 1: installa eksctl

Completa i seguenti passaggi per scaricare e installare l'ultima versione di eksctl sul tuo dispositivo Linux o macOS:

Per installare eksctl con Homebrew

1. (Prerequisito) Installa Homebrew.
2. Aggiungi il tap AWS:

```
brew tap aws/tap
```

3. Installa eksctl

```
brew install aws/tap/eksctl
```

Prima di usare eksctl, completa questi passaggi di configurazione:

1. Prerequisiti di installazione:
   - Installa AWS CLI versione 2.x o successiva.
   - Installa kubectl usando Homebrew:

```
brew install kubernetes-cli
```

2. Configura le credenziali AWS nel tuo ambiente:

```
aws configure
```

3. Verifica la configurazione dell'interfaccia a riga di comando di AWS:

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

Fase 2: Creare un file di configurazione del cluster

Crea un file di configurazione del cluster utilizzando questi passaggi:

1. Crea un nuovo file denominato`cluster.yaml`:

```
touch cluster.yaml
```

2. Aggiungere la seguente configurazione di base del cluster:

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

metadata:
   name: basic-cluster
   region: us-west-2

nodeGroups:
   - name: ng-1
     instanceType: m5.large
     desiredCapacity: 2
     minSize: 1
     maxSize: 3
     ssh:
        allow: false
```

3. Personalizza la configurazione:
   - Aggiorna il file `region` in modo che corrisponda alla regione AWS desiderata.
   - Modificalo `instanceType` in base ai requisiti del tuo carico di lavoro.
   - Regola il `desiredCapacityminSize`, e `maxSize` in base alle tue esigenze.
4. Convalida il file di configurazione:

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

Fase 3: Creare un cluster

Segui questi passaggi per creare il tuo cluster EKS:

1. Crea il cluster utilizzando il file di configurazione:

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

2. Attendi la creazione del cluster (in genere richiede 15-20 minuti).
3. Verifica la creazione del cluster:

```
eksctl get cluster
```

4. Configura kubectl per utilizzare il tuo nuovo cluster:

```
aws eks update-kubeconfig --name basic-cluster --region us-west-2
```

5. Verifica la connettività del cluster:

```
kubectl get nodes
```

Il cluster è ora pronto per l'uso.

Facoltativo: Elimina cluster

Ricordati di eliminare il cluster quando hai finito per evitare addebiti inutili:

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

> **Note**
>
> La creazione di cluster può comportare costi AWS. Assicurati di controllare i prezzi di Amazon EKS prima di creare un cluster.

Fasi successive

- Configura Kubectl per connetterti al cluster
- Implementa un'app di esempio

Opzioni di installazione per Eksctl

`eksctl`è disponibile per l'installazione dalle versioni ufficiali come descritto di seguito. Si consiglia di eseguire l'installazione solo `eksctl` dalle GitHub versioni ufficiali. Puoi scegliere di utilizzare un programma di installazione di terze parti, ma tieni presente che AWS non mantiene né supporta questi metodi di installazione. Usali a tua discrezione.

Prerequisito

Dovrai avere le credenziali API AWS configurate. Ciò che funziona per AWS CLI o qualsiasi altro strumento (kops, Terraform, ecc.) dovrebbe essere sufficiente. È possibile utilizzare variabili di `~/.aws/credentials`file o di ambiente. Per ulteriori informazioni, consulta AWS CLI Reference.

Avrai anche bisogno del comando AWS IAM Authenticator for Kubernetes (uno o più (disponibile nella versione 1.16.156 `aws-iam-authenticator` `aws eks get-token` o successiva dell'interfaccia a riga di comando di AWS) nel tuo. `PATH`

L'account IAM utilizzato per la creazione del cluster EKS deve avere questi livelli di accesso minimi.

Servizio AWS	Livello di accesso
CloudFormation	Accesso completo
EC2	Completo: Tagging Limited: Elenca, leggi, scrivi
EC2 Auto Scaling	Limitato: elenca, scrivi
EKS	Accesso completo
IAM	Limitato: elenco, lettura, scrittura, gestione delle autorizzazioni
Systems Manager	Limitato: elenco, lettura

Per Unix

Per scaricare l'ultima versione, esegui:

```
# for ARM systems, set ARCH to: `arm64`, `armv6` or `armv7`
ARCH=amd64
PLATFORM=$(uname -s)_$ARCH

curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_
$PLATFORM.tar.gz"

# (Optional) Verify checksum
curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/
eksctl_checksums.txt" | grep $PLATFORM | sha256sum --check

tar -xzf eksctl_$PLATFORM.tar.gz -C /tmp && rm eksctl_$PLATFORM.tar.gz

sudo install -m 0755 /tmp/eksctl /usr/local/bin && rm /tmp/eksctl
```

Per Windows

Download diretto (ultima versione):

- AMD64/x86_64
- ARMv6
- ARMv7
- ARM64

Assicurati di decomprimere l'archivio in una cartella nella variabile. `PATH`

Facoltativamente, verifica il checksum:

1. Scarica il file di checksum: più recente
2. Usa il prompt dei comandi per confrontare manualmente `CertUtil` l'output con il file di checksum scaricato.

   ```
   REM Replace amd64 with armv6, armv7 or arm64
   CertUtil -hashfile eksctl_Windows_amd64.zip SHA256
   ```

3. Utilizzo PowerShell per automatizzare la verifica utilizzando l'`-eq`operatore per ottenere un `True` risultato OR: `False`

   ```
   # Replace amd64 with armv6, armv7 or arm64
   ```

```
 (Get-FileHash -Algorithm SHA256 .\eksctl_Windows_amd64.zip).Hash -eq ((Get-Content .
\eksctl_checksums.txt) -match 'eksctl_Windows_amd64.zip' -split ' ')[0]
```

Usare Git Bash:

```
# for ARM systems, set ARCH to: `arm64`, `armv6` or `armv7`
ARCH=amd64
PLATFORM=windows_$ARCH

curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_
$PLATFORM.zip"

# (Optional) Verify checksum
curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/
eksctl_checksums.txt" | grep $PLATFORM | sha256sum --check

unzip eksctl_$PLATFORM.zip -d $HOME/bin

rm eksctl_$PLATFORM.zip
```

L'`eksctl`eseguibile viene inserito`$HOME/bin`, che `$PATH` proviene da Git Bash.

Homebrew

Puoi usare Homebrew per installare software su macOS e Linux.

AWS mantiene un tap Homebrew che include eksctl.

Per ulteriori informazioni sull'Homebrew tap, consulta il progetto su Github e la formula Homebrew per eksctl.

Per installare eksctl con Homebrew

1. (Prerequisito) Installa Homebrew
2. Aggiungi il tap AWS

```
brew tap aws/tap
```

3. Installa eksctl

```
brew install aws/tap/eksctl
```

Docker

Per ogni versione e RC viene inserita un'immagine del contenitore nel repository ECR. `public.ecr.aws/eksctl/eksctl` Scopri di più sull'utilizzo su ECR Public Gallery - eksctl. Ad esempio,

```
docker run --rm -it public.ecr.aws/eksctl/eksctl version
```

Completamento della shell

Bash

Per abilitare il completamento di bash, esegui quanto segue o inseriscilo in `~/.bashrc` o`~/.profile`:

```
. <(eksctl completion bash)
```

Zsh

Per completare zsh, esegui:

```
mkdir -p ~/.zsh/completion/
eksctl completion zsh > ~/.zsh/completion/_eksctl
```

e inserisci quanto segue: `~/.zshrc`

```
fpath=($fpath ~/.zsh/completion)
```

Nota che se non stai eseguendo una distribuzione come questa, oh-my-zsh potresti dover prima abilitare il completamento automatico (e inserirlo `~/.zshrc` per renderlo persistente):

```
autoload -U compinit
compinit
```

Pesce

I seguenti comandi possono essere usati per il completamento automatico di fish:

```
mkdir -p ~/.config/fish/completions
eksctl completion fish > ~/.config/fish/completions/eksctl.fish
```

PowerShell

È possibile fare riferimento al comando seguente per configurarlo. Tieni presente che il percorso potrebbe essere diverso a seconda delle impostazioni del sistema.

```
eksctl completion powershell > C:\Users\Documents\WindowsPowerShell\Scripts\eksctl.ps1
```

Aggiornamenti

> **Important**
>
> Se installi eksctl scaricandolo direttamente (senza usare un gestore di pacchetti) devi aggiornarlo manualmente.

Cluster

Questo capitolo tratta la creazione e la configurazione di cluster EKS utilizzando eksctl. Include anche componenti aggiuntivi e EKS Auto Mode.

Argomenti:

- the section called “EKS Access Entries”
  - Semplifica la gestione RBAC di Kubernetes sostituendo aws-auth con le voci di accesso EKS ConfigMap
  - Migra le mappature di identità IAM esistenti da aws-auth alle voci di accesso ConfigMap
  - Configura le modalità di autenticazione del cluster e controlla le autorizzazioni di amministrazione dei creatori del cluster
- the section called “Aggiornamenti aggiuntivi predefiniti”
  - Proteggi i cluster aggiornando i componenti aggiuntivi EKS predefiniti sui cluster più vecchi
- the section called “Componenti aggiuntivi”
  - Automatizza le attività di routine per l'installazione, l'aggiornamento e la rimozione dei componenti aggiuntivi.
  - I componenti aggiuntivi di Amazon EKS includono componenti aggiuntivi AWS, componenti aggiuntivi per community open source e componenti aggiuntivi per marketplace.
- the section called “Modalità automatica di EKS”
  - Riduci il sovraccarico operativo lasciando che AWS gestisca la tua infrastruttura EKS
  - Configura pool di nodi personalizzati anziché pool generici e di sistema predefiniti
  - Converti i cluster EKS esistenti per utilizzare la modalità automatica
- the section called “CloudWatch registrazione”
  - Risolvi i problemi relativi ai cluster abilitando i log per componenti specifici del piano di controllo EKS
  - Configura i periodi di conservazione dei log per i log del cluster EKS
  - Modifica le impostazioni di registrazione del cluster esistenti utilizzando i comandi eksctl
- the section called “Aggiornamenti del cluster”
  - Mantieni la sicurezza e la stabilità aggiornando in sicurezza le versioni del piano di controllo EKS
  - Implementa gli aggiornamenti tra i gruppi di nodi sostituendo i vecchi gruppi con quelli nuovi

- Aggiorna i componenti aggiuntivi predefiniti del cluster

- the section called “Creazione e gestione di cluster”
  - Inizia rapidamente con i cluster EKS di base utilizzando i gruppi di nodi gestiti predefiniti
  - Crea cluster personalizzati utilizzando file di configurazione con configurazioni specifiche
  - Implementa i cluster esistenti VPCs con reti private e politiche IAM personalizzate
- the section called “Configura kubelet”
  - Previeni la carenza di risorse dei nodi configurando le prenotazioni di kubelet e system daemon
  - Personalizza le soglie di sfratto per la disponibilità di memoria e file system
  - Abilita o disabilita specifiche feature gate Kubelet tra gruppi di nodi
- the section called “Connettore EKS”
  - Centralizza la gestione delle implementazioni ibride di Kubernetes tramite la console EKS
  - Configura i ruoli e le autorizzazioni IAM per l'accesso al cluster esterno
  - Rimuovi i cluster esterni e pulisci le risorse AWS associate
- the section called “Cluster completamente privato EKS”
  - Soddisfa i requisiti di sicurezza con cluster EKS completamente privati senza accesso a Internet in uscita
  - Configura l'accesso privato ai servizi AWS tramite endpoint VPC
  - Crea e gestisci gruppi di nodi privati con impostazioni di rete esplicite
- the section called “Assistenza Karpenter”
  - Automatizza il provisioning dei nodi
  - Crea configurazioni Karpenter Provisioner personalizzate
  - Configura Karpenter con la gestione delle interruzioni delle istanze spot
- the section called “Amazon EMR”
  - Crea una mappatura delle identità IAM tra EMR e cluster EKS
- the section called “Assistenza EKS Fargate”
  - Definisci profili Fargate personalizzati per la pianificazione dei pod
  - Gestisci i profili Fargate attraverso la creazione e gli aggiornamenti di configurazione
- the section called “Cluster non creati da eksctl”
  - Standardizza la gestione dei cluster creati all'esterno di eksctl

- Usa i comandi eksctl su cluster non eksctl esistenti

- the section called “Abilita Zonal Shift”
  - Migliora la disponibilità delle applicazioni abilitando funzionalità di failover rapido delle zone
  - Configura lo spostamento zonale nelle nuove implementazioni di cluster EKS
  - Abilita le funzionalità di spostamento zonale sui cluster EKS esistenti

Creazione e gestione di cluster

Questo argomento spiega come creare ed eliminare i cluster EKS utilizzando Eksctl. È possibile creare cluster con un comando CLI o creando un file YAML di configurazione del cluster.

Creazione di un cluster semplice

Crea un cluster semplice con il seguente comando:

```
eksctl create cluster
```

Ciò creerà un cluster EKS nella tua regione predefinita (come specificato dalla configurazione AWS CLI) con un nodegroup gestito contenente due nodi m5.large.

eksctl ora crea un nodegroup gestito per impostazione predefinita quando non viene utilizzato un file di configurazione. Per creare un gruppo di nodi autogestito, passa a o. `--managed=false` `eksctl create cluster` `eksctl create nodegroup`

Considerazioni

- Durante la creazione di cluster in`us-east-1`, potresti incontrare un. `UnsupportedAavailabilityZoneException` Se ciò accade, copia le zone suggerite e passa la `--zones` bandiera, per esempio:`eksctl create cluster --region=us-east-1 --zones=us-east-1a,us-east-1b,us-east-1d`. Questo problema può verificarsi in altre regioni, ma è meno comune. Nella maggior parte dei casi, non è necessario utilizzare la `--zone` bandiera.

Crea un cluster utilizzando il file di configurazione

È possibile creare un cluster utilizzando un file di configurazione anziché i flag.

Innanzitutto, crea il file`cluster.yaml`:

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

metadata:
  name: basic-cluster
  region: eu-north-1

nodeGroups:
  - name: ng-1
    instanceType: m5.large
    desiredCapacity: 10
    volumeSize: 80
    ssh:
      allow: true # will use ~/.ssh/id_rsa.pub as the default ssh key
  - name: ng-2
    instanceType: m5.xlarge
    desiredCapacity: 2
    volumeSize: 100
    ssh:
      publicKeyPath: ~/.ssh/ec2_id_rsa.pub
```

Quindi, esegui questo comando:

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

Questo creerà un cluster come descritto.

Se hai bisogno di usare un VPC esistente, puoi usare un file di configurazione come questo:

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

metadata:
  name: cluster-in-existing-vpc
  region: eu-north-1

vpc:
  subnets:
    private:
      eu-north-1a: { id: subnet-0ff156e0c4a6d300c }
      eu-north-1b: { id: subnet-0549cdab573695c03 }
      eu-north-1c: { id: subnet-0426fb4a607393184 }
```

```
nodeGroups:
  - name: ng-1-workers
    labels: { role: workers }
    instanceType: m5.xlarge
    desiredCapacity: 10
    privateNetworking: true
  - name: ng-2-builders
    labels: { role: builders }
    instanceType: m5.2xlarge
    desiredCapacity: 2
    privateNetworking: true
    iam:
      withAddonPolicies:
        imageBuilder: true
```

Il nome del cluster o del gruppo di nodi deve contenere solo caratteri alfanumerici (con distinzione tra maiuscole e minuscole) e trattini. Deve iniziare con un carattere alfabetico e non può superare i 128 caratteri, altrimenti riceverai un errore di convalida. Per ulteriori informazioni, consulta Creare uno stack dalla CloudFormation console nella guida per l'utente di AWS CLoud Formation.

Aggiorna kubeconfig per un nuovo cluster

Dopo la creazione del cluster, la configurazione kubernetes appropriata verrà aggiunta al file kubeconfig. Questo è il file che hai configurato nella variabile di ambiente o di default. `KUBECONFIG` `~/.kube/config` Il percorso del file kubeconfig può essere sovrascritto usando il flag. `--kubeconfig`

Altri flag che possono cambiare il modo in cui viene scritto il file kubeconfig:

flag	tipo	uso	Valore predefinito
--kubeconfig	stringa	percorso per scrivere kubeconfig (incompatibile con --auto-kubeconfig)	$KUBECONFIG o ~/.kube/config
--set-kubeconfig-context	bool	se vero allora current-context verrà impostato in kubeconfig; se un contesto è	true

flag	tipo	uso	Valore predefinito
		già impostato, verrà sovrascritto	
--configurazione autokubeconfig	bool	salva il file kubeconfig in base al nome del cluster	true
--write-kubeconfig	bool	attiva la scrittura di kubeconfig	true

Eliminazione di un cluster

Per eliminare questo cluster, esegui:

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

> **Warning**
>
> Usa il `--wait` flag con le operazioni di eliminazione per assicurarti che gli errori di eliminazione vengano segnalati correttamente.

Senza il `--wait` flag, eksctl emetterà solo un'operazione di cancellazione CloudFormation nello stack del cluster e non aspetterà la sua eliminazione. In alcuni casi, le risorse AWS che utilizzano il cluster o il relativo VPC possono impedire l'eliminazione del cluster. Se l'eliminazione fallisce o dimentichi il flag di attesa, potresti dover accedere alla CloudFormation GUI ed eliminare gli stack eks da lì.

> **Warning**
>
> Le politiche PDB possono bloccare la rimozione dei nodi durante l'eliminazione del cluster.

Quando si elimina un cluster con gruppi di nodi, le politiche Pod Disruption Budget (PDB) possono impedire che i nodi vengano rimossi correttamente. Ad esempio, i cluster `aws-ebs-csi-driver` installati in genere dispongono di due pod con una policy PDB che consente di non rendere

disponibile solo un pod alla volta, rendendo l'altro pod impraticabile durante l'eliminazione. Per eliminare correttamente il cluster in questi scenari, utilizzate il `disable-nodegroup-eviction` flag per aggirare i controlli delle policy PDB:

```
eksctl delete cluster -f cluster.yaml --disable-nodegroup-eviction
```

Vedi la `examples/`directory nel GitHub repository eksctl per altri file di configurazione di esempio.

Esecuzione a secco

La funzione dry-run consente di generare un ClusterConfig file che salta la creazione del cluster e genera un ClusterConfig file che rappresenta le opzioni CLI fornite e contiene i valori predefiniti impostati da eksctl.

Maggiori informazioni sono disponibili nella pagina Dry Run.

Modalità automatica di EKS

eksctl supporta EKS Auto Mode, una funzionalità che estende la gestione AWS dei cluster Kubernetes oltre il cluster stesso, per consentire ad AWS di configurare e gestire anche l'infrastruttura che consente il regolare funzionamento dei carichi di lavoro. Ciò consente di delegare le decisioni chiave sull'infrastruttura e sfruttare l'esperienza di AWS per day-to-day le operazioni. L'infrastruttura cluster gestita da AWS include molte funzionalità Kubernetes come componenti principali, al contrario dei componenti aggiuntivi, come l'autoscaling di calcolo, il networking di pod e servizi, il bilanciamento del carico delle applicazioni, il DNS del cluster, lo storage a blocchi e il supporto GPU.

Creazione di un cluster EKS con la modalità automatica abilitata

`eksctl`ha aggiunto un nuovo `autoModeConfig` campo per abilitare e configurare la modalità automatica. La forma del `autoModeConfig` campo è

```
autoModeConfig:
    # defaults to false
    enabled: boolean
    # optional, defaults to [general-purpose, system].
    # 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
```

Se `autoModeConfig.enabled` è vero, eksctl crea un cluster EKS passando e `storageConfig.blockStorage.enabled: true` all'API EKS `computeConfig.enabled: truekubernetesNetworkConfig.elasticLoadBalancing.enabled: true`, abilitando la gestione di componenti del piano dati come elaborazione, archiviazione e rete.

Per creare un cluster EKS con la modalità automatica abilitata, impostaa`autoModeConfig.enabled: true`, come in

```
# auto-mode-cluster.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
    name: auto-mode-cluster
    region: us-west-2

autoModeConfig:
    enabled: true
```

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

eksctl crea un ruolo di nodo da utilizzare per i nodi lanciati da Auto Mode. eksctl crea anche i pool di nodi and. `general-purpose` `system` Per disabilitare la creazione dei pool di nodi predefiniti, ad esempio per configurare i propri pool di nodi che utilizzano un diverso set di sottoreti, imposta, come in `nodePools: []`

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
    name: auto-mode-cluster
    region: us-west-2

autoModeConfig:
    enabled: true
    nodePools: [] # disables creation of default node pools.
```

Aggiornamento di un cluster EKS per l'utilizzo della modalità automatica

Per aggiornare un cluster EKS esistente per utilizzare la modalità Auto, esegui

```
# cluster.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
    name: cluster
    region: us-west-2

autoModeConfig:
    enabled: true
```

```
eksctl update auto-mode-config -f cluster.yaml
```

> **Note**
>
> Se il cluster è stato creato da eksctl e utilizza sottoreti pubbliche come sottoreti del cluster, Auto Mode avvierà i nodi nelle sottoreti pubbliche. Per utilizzare sottoreti private per i nodi di lavoro avviati da Auto Mode, aggiorna il cluster per utilizzare sottoreti private.

Disabilitazione della modalità automatica

Per disabilitare la modalità automatica, imposta `autoModeConfig.enabled: false` ed esegui

```
# cluster.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
    name: auto-mode-cluster
    region: us-west-2

autoModeConfig:
    enabled: false
```

```
eksctl update auto-mode-config -f cluster.yaml
```

Ulteriori informazioni

- Modalità automatica EKS

EKS Access Entries

È possibile utilizzare eksctl per gestire EKS Access Entries. Usa le voci di accesso per concedere le autorizzazioni Kubernetes alle identità AWS IAM. Ad esempio, potresti concedere a un ruolo di sviluppatore l'autorizzazione a leggere le risorse Kubernetes in un cluster.

Questo argomento spiega come usare eksctl per gestire le voci di accesso. Per informazioni generali sulle voci di accesso, consulta Concedere agli utenti IAM l'accesso a Kubernetes con le voci di accesso EKS.

Puoi allegare le policy di accesso Kubernetes definite da AWS o associare un'identità IAM a un gruppo Kubernetes.

Per ulteriori informazioni sulle politiche predefinite disponibili, consulta Associare le politiche di accesso alle voci di accesso.

Se devi definire le policy Kubernetes dei clienti, associa l'identità IAM a un gruppo Kubernetes e concedi le autorizzazioni a quel gruppo.

Modalità di autenticazione del cluster

È possibile utilizzare le voci di accesso solo se la modalità di autenticazione del cluster lo consente.

Per ulteriori informazioni, vedere Impostare la modalità di autenticazione del cluster

Imposta la modalità di autenticazione con un file YAML

`eksctl`ha aggiunto un nuovo `accessConfig.authenticationMode` campo sotto ClusterConfig, che può essere impostato su uno dei tre valori seguenti:

- `CONFIG_MAP`- impostazione predefinita nell'API EKS: `aws-auth` ConfigMap verrà utilizzato solo
- `API`- verrà utilizzata solo l'API Access Entries
- `API_AND_CONFIG_MAP`- impostazione predefinita`eksctl`: è possibile utilizzare `aws-auth` ConfigMap sia l'API di accesso che quella delle voci di accesso

Imposta la modalità di autenticazione in ClusterConfig YAML:

```
accessConfig:
```

```
  authenticationMode: <>
```

Aggiorna la modalità di autenticazione con un comando

Se si desidera utilizzare le voci di accesso su un cluster già esistente, creato senza eksctl, dove viene utilizzata `CONFIG_MAP` l'opzione, l'utente dovrà prima impostarla su. `authenticationMode` `API_AND_CONFIG_MAP` Per questo, `eksctl` ha introdotto un nuovo comando per l'aggiornamento della modalità di autenticazione del cluster, che funziona sia con i flag CLI, ad es.

```
eksctl utils update-authentication-mode --cluster my-cluster --authentication-mode
 API_AND_CONFIG_MAP
```

Accedi alle risorse di accesso

Le voci di accesso hanno un tipo, ad esempio `STANDARD` o`EC2_LINUX`. Il tipo dipende da come si utilizza la voce di accesso.

- Il `standard` tipo serve per concedere le autorizzazioni Kubernetes agli utenti IAM e ai ruoli IAM.
  - Ad esempio, puoi visualizzare le risorse Kubernetes nella console AWS allegando una policy di accesso al ruolo o all'utente che usi per accedere alla console.
- I `EC2_WINDOWS` tipi `EC2_LINUX` e servono per concedere le autorizzazioni Kubernetes alle istanze EC2. Le istanze utilizzano queste autorizzazioni per entrare a far parte del cluster.

Per ulteriori informazioni sui tipi di voci di accesso, consulta Creare voci di accesso

Entità IAM

Puoi utilizzare le voci di accesso per concedere autorizzazioni Kubernetes a identità IAM come utenti IAM e ruoli IAM.

Utilizza il `accessConfig.accessEntries` campo per associare l'ARN di una risorsa IAM a un'API Access Entries EKS. Esempio:

```
accessConfig:
  authenticationMode: API_AND_CONFIG_MAP
  accessEntries:
    - principalARN: arn:aws:iam::111122223333:user/my-user-name
      type: STANDARD
      kubernetesGroups: # optional Kubernetes groups
```

```
      - group1 # groups can used to give permissions via RBAC
      - group2

  - principalARN: arn:aws:iam::111122223333:role/role-name-1
    accessPolicies: # optional access polices
      - policyARN: arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy
        accessScope:
          type: namespace
          namespaces:
            - default
            - my-namespace
            - dev-*

  - principalARN: arn:aws:iam::111122223333:role/admin-role
    accessPolicies: # optional access polices
      - policyARN: arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy
        accessScope:
          type: cluster

  - principalARN: arn:aws:iam::111122223333:role/role-name-2
    type: EC2_LINUX
```

Oltre ad associare le policy EKS, è possibile specificare anche i gruppi Kubernetes a cui appartiene un'entità IAM, concedendo così le autorizzazioni tramite RBAC.

Gruppi di nodi gestiti e Fargate

L'integrazione con le voci di accesso per queste risorse sarà realizzata dietro le quinte, tramite l'API EKS. I gruppi di nodi gestiti e i pod Fargate appena creati creeranno voci di accesso alle API, anziché utilizzare risorse RBAC precaricate. I gruppi di nodi e i pod Fargate esistenti non verranno modificati e continueranno a fare affidamento sulle voci nella mappa di configurazione aws-auth.

Gruppi di nodi autogestiti

Ogni voce di accesso include un tipo. Per autorizzare i gruppi di nodi autogestiti, `eksctl` creerà una voce di accesso univoca per ogni gruppo di nodi con l'ARN principale impostato sul ruolo del nodo ARN e il tipo impostato su uno o in base al nodegroup AMIFamily. `EC2_LINUX` `EC2_WINDOWS`

Quando crei le tue voci di accesso, puoi anche specificare `EC2_LINUX` (per un ruolo IAM utilizzato con nodi autogestiti Linux o Bottlerocket), `EC2_WINDOWS` (per un ruolo IAM utilizzato con nodi autogestiti Windows), `FARGATE_LINUX` (per un ruolo IAM utilizzato con AWS Fargate (Fargate)) o come tipo. `STANDARD` Se non si specifica un tipo, il tipo predefinito è impostato su. `STANDARD`

> **Note**
>
> Quando si elimina un gruppo di nodi creato con un gruppo di nodi preesistente`instanceRoleARN`, è responsabilità dell'utente eliminare la voce di accesso corrispondente quando non vi sono più gruppi di nodi associati. Questo perché eksctl non tenta di scoprire se una voce di accesso è ancora utilizzata da gruppi di nodi autogestiti non creati da eksctl in quanto si tratta di un processo complicato.

Crea una voce di accesso

Questa operazione può essere eseguita in due modi diversi, durante la creazione del cluster, specificando le voci di accesso desiderate come parte del file di configurazione ed eseguendo:

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

OPPURE dopo la creazione del cluster, eseguendo:

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

Per un esempio di file di configurazione per la creazione di voci di accesso, vedi 40-access-entries.yaml nel repository eksctl. GitHub

Ottieni l'accesso

L'utente può recuperare tutte le voci di accesso associate a un determinato cluster eseguendo una delle seguenti operazioni:

```
eksctl get accessentry -f config.yaml
```

O

```
eksctl get accessentry --cluster my-cluster
```

In alternativa, per recuperare solo la voce di accesso corrispondente a una determinata entità IAM si deve usare il `--principal-arn` flag. ad es.

```
eksctl get accessentry --cluster my-cluster --principal-arn
 arn:aws:iam::111122223333:user/admin
```

Eliminare la voce di accesso

Per eliminare una singola voce di accesso alla volta, usa:

```
eksctl delete accessentry --cluster my-cluster --principal-arn
 arn:aws:iam::111122223333:user/admin
```

Per eliminare più voci di accesso, utilizzate il `--config-file` flag e specificate tutte le voci di accesso `principalARN's` corrispondenti, `accessEntry` nel campo di primo livello, ad es.

```
...
accessEntry:
  - principalARN: arn:aws:iam::111122223333:user/my-user-name
  - principalARN: arn:aws:iam::111122223333:role/role-name-1
  - principalARN: arn:aws:iam::111122223333:role/admin-role
```

```
eksctl delete accessentry -f config.yaml
```

Esegui la migrazione da aws-auth ConfigMap

L'utente può migrare le proprie identità IAM esistenti da `aws-auth` configmap alle voci di accesso eseguendo quanto segue:

```
eksctl utils migrate-to-access-entry --cluster my-cluster --target-authentication-mode
 <API or API_AND_CONFIG_MAP>
```

Quando `--target-authentication-mode` flag è impostato su`API`, la modalità di autenticazione passa alla `API` modalità (ignorata se è già attiva), le mappature delle identità IAM verranno migrate alle voci di accesso e configmap viene eliminata dal cluster. `API` `aws-auth`

Quando `--target-authentication-mode` flag è impostato su`API_AND_CONFIG_MAP`, la modalità di autenticazione passa alla modalità (ignorata se è già in `API_AND_CONFIG_MAP` `API_AND_CONFIG_MAP` modalità), le mappature delle identità IAM verranno migrate alle voci di accesso, ma configmap viene preservata. `aws-auth`

> **Note**
>
> Quando `--target-authentication-mode` flag è impostato su`API`, questo comando non aggiornerà la modalità di autenticazione alla `API` modalità se `aws-auth` configmap presenta uno dei seguenti vincoli.

- Esiste una mappatura delle identità a livello di account.
- Uno o più Roles/Users sono mappati sui gruppi Kubernetes che iniziano con il prefisso `system:` (ad eccezione dei gruppi specifici EKS, ad esempio, ecc.). `system:masters` `system:bootstrappers` `system:nodes`
- Una o più mappature di identità IAM sono per un [Service Linked Role] (link: - .html). IAM/latest/UserGuide/using service-linked-roles

Disattiva le autorizzazioni di amministratore del cluster Creator

`eksctl`ha aggiunto un nuovo campo `accessConfig.bootstrapClusterCreatorAdminPermissions: boolean` che, se impostato su false, disabilita la concessione delle autorizzazioni di amministratore del cluster all'identità IAM che crea il cluster. ad es.

aggiungi l'opzione al file di configurazione:

```
accessConfig:
  bootstrapClusterCreatorAdminPermissions: false
```

ed esegui:

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

Cluster non creati da eksctl

È possibile eseguire `eksctl` comandi su cluster che non sono stati creati da. `eksctl`

> **Note**
>
> Eksctl può supportare solo cluster sconosciuti con nomi compatibili con AWS. CloudFormation Qualsiasi nome di cluster che non corrisponde a questo non CloudFormation supererà il controllo di convalida dell'API.

Comandi supportati

I seguenti comandi possono essere utilizzati contro i cluster creati con qualsiasi mezzo diverso da. `eksctl` I comandi, i flag e le opzioni del file di configurazione possono essere utilizzati esattamente nello stesso modo.

Se abbiamo perso alcune funzionalità, fatecelo sapere.

- ✓ Crea:
  - ✓ `eksctl create nodegroup` (vedi nota sotto)
  - ✓ `eksctl create fargateprofile`
  - ✓ `eksctl create iamserviceaccount`
  - ✓ `eksctl create iamidentitymapping`
- ✓ Ottieni:
  - ✓ `eksctl get clusters/cluster`
  - ✓ `eksctl get fargateprofile`
  - ✓ `eksctl get nodegroup`
  - ✓ `eksctl get labels`
- ✓ Elimina:
  - ✓ `eksctl delete cluster`
  - ✓ `eksctl delete nodegroup`
  - ✓ `eksctl delete fargateprofile`
  - ✓ `eksctl delete iamserviceaccount`
  - ✓ `eksctl delete iamidentitymapping`
- ✓ Aggiornamento:
  - ✓ `eksctl upgrade cluster`
  - ✓ `eksctl upgrade nodegroup`

- ✓ Impostazione/disattivazione:
  - ✓ `eksctl set labels`
  - ✓ `eksctl unset labels`
- ✓ Scala:
  - ✓ `eksctl scale nodegroup`
- ✓ Scarico:
  - ✓ `eksctl drain nodegroup`
- ✓ Abilita:
  - ✓ `eksctl enable profile`
  - ✓ `eksctl enable repo`
- ✓ Utilità:
  - ✓ `eksctl utils associate-iam-oidc-provider`
  - ✓ `eksctl utils describe-stacks`
  - ✓ `eksctl utils install-vpc-controllers`
  - ✓ `eksctl utils nodegroup-health`
  - ✓ `eksctl utils set-public-access-cidrs`
  - ✓ `eksctl utils update-cluster-endpoints`
  - ✓ `eksctl utils update-cluster-logging`
  - ✓ `eksctl utils write-kubeconfig`
  - ✓ `eksctl utils update-coredns`
  - ✓ `eksctl utils update-aws-node`
  - ✓ `eksctl utils update-kube-proxy`

Creazione di gruppi di nodi

`eksctl create nodegroup`è l'unico comando che richiede un input specifico da parte dell'utente.

Poiché gli utenti possono creare i propri cluster con qualsiasi configurazione di rete preferiscano, per il momento non `eksctl` cercheranno di recuperare o indovinare questi valori. Questo potrebbe cambiare in futuro man mano che impareremo di più su come le persone utilizzano questo comando su cluster non creati da eksctl.

Ciò significa che per creare gruppi di nodi o gruppi di nodi gestiti su un cluster che non è stato creato da`eksctl`, è necessario fornire un file di configurazione contenente i dettagli VPC. Come minimo:

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

metadata:
  name: non-eksctl-created-cluster
  region: us-west-2

vpc:
  id: "vpc-12345"
  securityGroup: "sg-12345"    # this is the ControlPlaneSecurityGroup
  subnets:
    private:
      private1:
          id: "subnet-12345"
      private2:
          id: "subnet-67890"
    public:
      public1:
          id: "subnet-12345"
      public2:
          id: "subnet-67890"

...
```

Per ulteriori informazioni sulle opzioni di configurazione del VPC, consulta Rete.

Registrazione di cluster non EKS con EKS Connector

Puoi utilizzare EKS Connector per visualizzare i cluster al di fuori di AWS nella console EKS. Questo processo richiede la registrazione del cluster con EKS e l'esecuzione dell'agente EKS Connector sul cluster Kubernetes esterno.

`eksctl`semplifica la registrazione di cluster non EKS creando le risorse AWS richieste e generando manifesti Kubernetes per EKS Connector da applicare al cluster esterno.

Registra il cluster

Per registrare o connettere un cluster Kubernetes non EKS, esegui

```
eksctl register cluster --name <name> --provider <provider>
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
```

Questo comando registrerà il cluster e scriverà tre file che contengono i manifesti di Kubernetes per EKS Connector che devono essere applicati al cluster esterno prima della scadenza della registrazione.

> **Note**
>
> `eks-connector-clusterrole.yamlgete eks-connector-console-dashboard-full-access-clusterrole.yaml` assegna `list` le autorizzazioni per le risorse Kubernetes in tutti i namespace all'identità IAM chiamante e devono essere modificate di conseguenza, se necessario, prima di applicarle al cluster. Per configurare un accesso più limitato, vedi Concessione dell'accesso a un utente per visualizzare un cluster.

Per fornire un ruolo IAM esistente da utilizzare per EKS Connector, passalo tramite `--role-arn` as in:

```
eksctl register cluster --name <name> --provider <provider> --role-arn=<role-arn>
```

Se il cluster esiste già, eksctl restituirà un errore.

Annulla la registrazione del cluster

Per annullare la registrazione o disconnettere un cluster registrato, esegui

```
eksctl deregister cluster --name <name>
2021-08-19 16:04:09 [#]  unregistered cluster "<name>" successfully
2021-08-19 16:04:09 [#]  run `kubectl delete namespace eks-connector` and `kubectl
 delete -f eks-connector-binding.yaml` on your cluster to remove EKS Connector
 resources
```

Questo comando annullerà la registrazione del cluster esterno e rimuoverà le risorse AWS associate, ma è necessario rimuovere le risorse Kubernetes del connettore EKS dal cluster.

Ulteriori informazioni

- Connettore EKS

Personalizzazione della configurazione di Kubelet

Le risorse di sistema possono essere riservate tramite la configurazione del kubelet. Questo è consigliato, perché in caso di carenza di risorse il kubelet potrebbe non essere in grado di eliminare i pod e alla fine far sì che il nodo diventi. `NotReady` Per fare ciò, i file di configurazione possono includere il `kubeletExtraConfig` campo che accetta un yaml in formato libero che verrà incorporato in. `kubelet.yaml`

Alcuni campi in `kubelet.yaml` sono impostati da eksctl e quindi non sono sovrascrivibili, come,, o. `address` `clusterDomain` `authentication` `authorization` `serverTLSBootstrap`

Il seguente file di configurazione di esempio crea un gruppo di nodi che riserva `300m` vCPU, `300Mi` memoria e storage temporaneo per il kubelet; `300m` vCPU`300Mi`, `1Gi` di memoria e di storage temporaneo per i demoni di sistema del sistema operativo; `1Gi` e avvia l'eliminazione dei pod quando c'è meno della memoria disponibile o meno del 10% del filesystem root. `200Mi`

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

metadata:
  name: dev-cluster-1
  region: eu-north-1

nodeGroups:
  - name: ng-1
    instanceType: m5a.xlarge
```

```
    desiredCapacity: 1
    kubeletExtraConfig:
        kubeReserved:
            cpu: "300m"
            memory: "300Mi"
            ephemeral-storage: "1Gi"
        kubeReservedCgroup: "/kube-reserved"
        systemReserved:
            cpu: "300m"
            memory: "300Mi"
            ephemeral-storage: "1Gi"
        evictionHard:
            memory.available:  "200Mi"
            nodefs.available: "10%"
        featureGates:
            RotateKubeletServerCertificate: true # has to be enabled, otherwise it will
be disabled
```

In questo esempio, date le istanze di tipo `m5a.xlarge` che hanno 4 v CPUs e 16 GiB di memoria, la `Allocatable` quantità di CPUs sarebbe 3,4 e 15,4 GiB di memoria. È importante sapere che i valori specificati nel file di configurazione per i campi in `kubeletExtraconfig` sovrascriveranno completamente i valori predefiniti specificati da eksctl. Tuttavia, l'omissione di uno o più `kubeReserved` parametri farà sì che i parametri mancanti vengano impostati come valori predefiniti in base al tipo di istanza aws utilizzato.

`kubeReserved`calcolo

Sebbene in genere sia consigliabile configurare un'istanza mista NodeGroup per utilizzare istanze con la stessa configurazione di CPU e RAM, non si tratta di un requisito rigoroso. Pertanto, il `kubeReserved` calcolo utilizza l'istanza più piccola sul `InstanceDistribution.InstanceTypes` campo. In questo modo, NodeGroups con tipi di istanze diversi, non si riservano troppe risorse sull'istanza più piccola. Tuttavia, ciò potrebbe portare a una prenotazione troppo piccola per il tipo di istanza più grande.

> **Warning**
>
> Le impostazioni sono `eksctl` predefinite`featureGates.RotateKubeletServerCertificate=true`, ma quando `featureGates` vengono fornite impostazioni personalizzate, non verranno impostate.

Dovresti sempre includerlo`featureGates.RotateKubeletServerCertificate=true`, a meno che tu non debba disabilitarlo.

CloudWatch registrazione

Questo argomento spiega come configurare il CloudWatch logging di Amazon per i componenti del piano di controllo del cluster EKS. CloudWatch la registrazione offre visibilità sulle operazioni del piano di controllo del cluster, essenziale per la risoluzione dei problemi, il controllo delle attività del cluster e il monitoraggio dello stato dei componenti Kubernetes.

Abilitazione della registrazione CloudWatch

CloudWatch la registrazione per il piano di controllo EKS non è abilitata per impostazione predefinita a causa dei costi di inserimento e archiviazione dei dati.

Per abilitare la registrazione del piano di controllo quando viene creato il cluster, è necessario definire l'**`cloudWatch.clusterLogging.enableTypes`**impostazione nel proprio `ClusterConfig` (vedi sotto per alcuni esempi).

Quindi, se hai un file di configurazione con le **`cloudWatch.clusterLogging.enableTypes`**impostazioni corrette, puoi creare un cluster con. `eksctl create cluster --config-file=<path>`

Se hai già creato un cluster, puoi usare`eksctl utils update-cluster-logging`.

> **Note**
>
> questo comando viene eseguito in modalità piano per impostazione predefinita, sarà necessario specificare `--approve` flag per applicare le modifiche al cluster.

Se stai usando un file di configurazione, esegui:

```
eksctl utils update-cluster-logging --config-file=<path>
```

In alternativa, puoi usare i flag CLI.

Per abilitare tutti i tipi di log, esegui:

```
eksctl utils update-cluster-logging --enable-types all
```

Per abilitare `audit` i log, esegui:

```
eksctl utils update-cluster-logging --enable-types audit
```

Per abilitare tutti tranne i `controllerManager` log, esegui:

```
eksctl utils update-cluster-logging --enable-types=all --disable-
types=controllerManager
```

Se i tipi `api` e `scheduler` log erano già abilitati, per disabilitarli `scheduler` e `controllerManager` attivarli contemporaneamente, esegui:

```
eksctl utils update-cluster-logging --enable-types=controllerManager --disable-
types=scheduler
```

Questo lascerà `api` e `controllerManager` come unici tipi di log saranno abilitati.

Per disabilitare tutti i tipi di log, esegui:

```
eksctl utils update-cluster-logging --disable-types all
```

Esempi di **`ClusterConfig`**

In un cluster EKS, il `enableTypes` campo sottostante `clusterLogging` può contenere un elenco di valori possibili per abilitare i diversi tipi di log per i componenti del piano di controllo.

Di seguito sono riportati i valori possibili:

- `api`: abilita i log del server dell'API Kubernetes.
- `audit`: abilita i log di controllo di Kubernetes.
- `authenticator`: abilita i log dell'autenticatore.
- `controllerManager`: abilita i log del gestore dei controller Kubernetes.
- `scheduler`: abilita i log dello scheduler Kubernetes.

Per ulteriori informazioni, consulta la documentazione EKS.

Disabilita tutti i registri

Per disabilitare tutti i tipi, usa `[]` o rimuovi completamente la `cloudWatch` sezione.

Abilita tutti i registri

Puoi abilitare tutti i tipi con `"*"` o`"all"`. Esempio:

```
cloudWatch:
  clusterLogging:
    enableTypes: ["*"]
```

Abilita uno o più registri

Puoi abilitare un sottoinsieme di tipi elencando i tipi che desideri abilitare. Esempio:

```
cloudWatch:
  clusterLogging:
    enableTypes:
      - "audit"
      - "authenticator"
```

Periodo di conservazione dei registri

Per impostazione predefinita, i registri vengono archiviati in CloudWatch Registri a tempo indeterminato. È possibile specificare il numero di giorni per i quali i log del piano di controllo devono essere conservati in Logs. CloudWatch L'esempio seguente conserva i log per 7 giorni:

```
cloudWatch:
  clusterLogging:
    logRetentionInDays: 7
```

Esempio completo

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

metadata:
  name: cluster-11
  region: eu-west-2
```

```
nodeGroups:
  - name: ng-1
    instanceType: m5.large
    desiredCapacity: 1

cloudWatch:
  clusterLogging:
    enableTypes: ["audit", "authenticator"]
    logRetentionInDays: 7
```

Cluster completamente privato EKS

eksctl supporta la creazione di cluster completamente privati che non hanno accesso a Internet in uscita e dispongono solo di sottoreti private. Gli endpoint VPC vengono utilizzati per consentire l'accesso privato ai servizi AWS.

Questa guida descrive come creare un cluster privato senza accesso a Internet in uscita.

Creazione di un cluster completamente privato

L'unico campo obbligatorio per creare un cluster completamente privato è: `privateCluster.enabled`

```
privateCluster:
  enabled: true
```

Dopo la creazione del cluster, i comandi eksctl che richiedono l'accesso al server API Kubernetes dovranno essere eseguiti dall'interno del VPC del cluster, da un VPC peered o utilizzando altri mezzi come AWS Direct Connect. I comandi eksctl che richiedono l'accesso a EKS non APIs funzioneranno se vengono eseguiti dall'interno del VPC del cluster. Per risolvere questo problema, crea un endpoint di interfaccia per Amazon EKS per accedere in modo privato alla gestione di Amazon Elastic Kubernetes Service (Amazon APIs EKS) dal tuo Amazon Virtual Private Cloud (VPC). In una versione futura, eksctl aggiungerà il supporto per creare questo endpoint in modo che non debba essere creato manualmente. I comandi che richiedono l'accesso all'URL del provider OpenID Connect dovranno essere eseguiti dall'esterno del VPC del cluster dopo aver abilitato AWS per PrivateLink Amazon EKS.

La creazione di gruppi di nodi gestiti continuerà a funzionare e la creazione di gruppi di nodi autogestiti funzionerà in quanto richiede l'accesso al server API tramite l'endpoint dell'interfaccia

EKS se il comando viene eseguito dall'interno del VPC del cluster, da un VPC peer o utilizzando altri mezzi come AWS Direct Connect.

> **Note**
>
> Gli endpoint VPC vengono addebitati su base oraria e in base all'utilizzo. Ulteriori dettagli sui prezzi sono disponibili nella pagina dei PrivateLink prezzi di AWS

> **Warning**
>
> I cluster completamente privati non sono supportati in. `eu-south-1`

Configurazione dell'accesso privato a servizi AWS aggiuntivi

Per consentire ai nodi di lavoro di accedere ai servizi AWS in modo privato, eksctl crea endpoint VPC per i seguenti servizi:

- Endpoint di interfaccia per ECR (entrambi `ecr.api` e`ecr.dkr`) per estrarre le immagini dei contenitori (plug-in AWS CNI ecc.)
- Un endpoint gateway per S3 per estrarre i livelli effettivi dell'immagine
- Un endpoint di interfaccia per EC2 richiesto dall'integrazione `aws-cloud-provider`
- Un endpoint di interfaccia per STS per supportare Fargate e IAM Roles for Services Accounts (IRSA)
- Un endpoint di interfaccia per CloudWatch logging (`logs`) se la registrazione è abilitata CloudWatch

Questi endpoint VPC sono essenziali per un cluster privato funzionale e, come tali, eksctl non supporta la configurazione o la disabilitazione. Tuttavia, un cluster potrebbe aver bisogno dell'accesso privato ad altri servizi AWS (ad esempio, l'Autoscaling richiesto dal Cluster Autoscaler). Questi servizi possono essere specificati in`privateCluster.additionalEndpointServices`, che indica a eksctl di creare un endpoint VPC per ciascuno di essi.

Ad esempio, per consentire l'accesso privato alla scalabilità automatica e alla registrazione: CloudWatch

```
privateCluster:
  enabled: true
  additionalEndpointServices:
  # For Cluster Autoscaler
  - "autoscaling"
  # CloudWatch logging
  - "logs"
```

Gli endpoint supportati in sono, e. `additionalEndpointServices` `autoscaling` `cloudformation` `logs`

Ignorare le creazioni degli endpoint

Se è già stato creato un VPC con gli endpoint AWS necessari configurati e collegati alle sottoreti descritte nella documentazione EKS, `eksctl` puoi saltare la creazione fornendo l'opzione in questo modo: `skipEndpointCreation`

```
privateCluster:
  enabled: true
  skipEndpointCreation: true
```

Questa impostazione non può essere utilizzata insieme a. `additionalEndpointServices` Salterà tutta la creazione degli endpoint. Inoltre, questa impostazione è consigliata solo se la topologia della <# sottorete dell'endpoint è configurata correttamente. Se gli ID di sottorete sono corretti, `vpce` il routing viene impostato con indirizzi prefissi, tutti gli endpoint EKS necessari vengono creati e collegati al VPC fornito. `eksctl`non altererà nessuna di queste risorse.

Gruppi di nodi

Solo i gruppi di nodi privati (gestiti e autogestiti) sono supportati in un cluster completamente privato perché il VPC del cluster viene creato senza sottoreti pubbliche. Il `privateNetworking` campo (e) deve essere impostato in modo esplicito. `nodeGroup[].privateNetworking` `managedNodeGroup[` È un errore lasciare `privateNetworking` unset in un cluster completamente privato.

```
nodeGroups:
- name: ng1
  instanceType: m5.large
  desiredCapacity: 2
```

```
  # privateNetworking must be explicitly set for a fully-private cluster
  # Rather than defaulting this field to `true`,
  # we require users to explicitly set it to make the behaviour
  # explicit and avoid confusion.
  privateNetworking: true

managedNodeGroups:
- name: m1
  instanceType: m5.large
  desiredCapacity: 2
  privateNetworking: true
```

Accesso all'endpoint del cluster

Un cluster completamente privato non supporta la modifica durante la creazione del cluster. `clusterEndpointAccess` È un errore impostarlo `clusterEndpoints.publicAccess` oppure`clusterEndpoints.privateAccess`, poiché un cluster completamente privato può avere solo accesso privato, e consentire la modifica di questi campi può compromettere il funzionamento del cluster.

VPC e sottoreti forniti dall'utente

eksctl supporta la creazione di cluster completamente privati utilizzando un VPC e sottoreti preesistenti. È possibile specificare solo sottoreti private ed è un errore specificare le sottoreti. `vpc.subnets.public`

eksctl crea endpoint VPC nel VPC fornito e modifica le tabelle di routing per le sottoreti fornite. Ogni sottorete dovrebbe avere una tabella di routing esplicita associata perché eksctl non modifica la tabella di route principale.

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

metadata:
  name: private-cluster
  region: us-west-2

privateCluster:
  enabled: true
  additionalEndpointServices:
  - "autoscaling"
```

```
vpc:
  subnets:
    private:
      us-west-2b:
        id: subnet-0818beec303f8419b
      us-west-2c:
        id: subnet-0d42ef09490805e2a
      us-west-2d:
        id: subnet-0da7418077077c5f9


nodeGroups:
- name: ng1
  instanceType: m5.large
  desiredCapacity: 2
  # privateNetworking must be explicitly set for a fully-private cluster
  # Rather than defaulting this field to true for a fully-private cluster, we require
 users to explicitly set it
  # to make the behaviour explicit and avoid confusion.
  privateNetworking: true

managedNodeGroups:
- name: m1
  instanceType: m5.large
  desiredCapacity: 2
  privateNetworking: true
```

Gestire un cluster completamente privato

Affinché tutti i comandi funzionino dopo la creazione del cluster, eksctl avrà bisogno dell'accesso privato all'endpoint del server dell'API EKS e dell'accesso a Internet in uscita (for). `EKS:DescribeCluster` I comandi che non richiedono l'accesso al server API saranno supportati se eksctl dispone di un accesso a Internet in uscita.

Eliminazione forzata di un cluster completamente privato

È probabile che si verifichino degli errori quando si elimina un cluster completamente privato tramite eksctl poiché eksctl non ha automaticamente accesso a tutte le risorse del cluster. `--force`esiste per risolvere questo problema: forzerà l'eliminazione del cluster e continuerà quando si verificano errori.

Limitazioni

Una limitazione dell'attuale implementazione è che eksctl inizialmente crea il cluster con l'accesso agli endpoint pubblico e privato abilitato e disabilita l'accesso pubblico agli endpoint dopo che tutte le operazioni sono state completate. Ciò è necessario perché eksctl deve accedere al server API Kubernetes per consentire ai nodi autogestiti di unirsi al cluster e supportare Fargate. GitOps Una volta completate queste operazioni, eksctl imposta l'accesso agli endpoint del cluster in modalità «solo privato». Questo aggiornamento aggiuntivo significa che la creazione di un cluster completamente privato richiederà più tempo rispetto a un cluster standard. In futuro, eksctl potrebbe passare a una funzione Lambda abilitata per VPC per eseguire queste operazioni API.

Accesso in uscita tramite server proxy HTTP

eksctl è in grado di comunicare con AWS APIs tramite un server proxy HTTP (S) configurato, tuttavia dovrai assicurarti di impostare correttamente l'elenco di esclusione dei proxy.

In genere, è necessario assicurarsi che le richieste per l'endpoint VPC per il cluster non vengano instradate tramite i proxy impostando una variabile di ambiente appropriata `no_proxy` che includa il valore. `.eks.amazonaws.com`

Se il tuo server proxy esegue l' «intercettazione SSL» e utilizzi IAM Roles for Service Accounts (IRSA), dovrai assicurarti di bypassare esplicitamente SSL per il dominio. Man-in-the-Middle `oidc.<region>.amazonaws.com` In caso contrario, eksctl otterrà l'impronta del certificato root errata per il provider OIDC e il plugin AWS VPC CNI non si avvierà perché non è in grado di ottenere le credenziali IAM, rendendo il cluster non operativo.

Ulteriori informazioni

- Cluster privati EKS

Componenti aggiuntivi

Questo argomento descrive come gestire i componenti aggiuntivi Amazon EKS per i cluster Amazon EKS utilizzando eksctl. EKS Add-Ons è una funzionalità che consente di abilitare e gestire il software operativo Kubernetes tramite l'API EKS, semplificando il processo di installazione, configurazione e aggiornamento dei componenti aggiuntivi del cluster.

> **Warning**
>
> eksctl ora installa componenti aggiuntivi predefiniti (vpc-cni, coredns, kube-proxy) come componenti aggiuntivi EKS anziché componenti aggiuntivi autogestiti. Ciò significa che è necessario utilizzare al posto dei comandi per i `eksctl update addon` cluster creati con eksctl `eksctl utils update-*` v0.184.0 e versioni successive.

È possibile creare cluster senza componenti aggiuntivi di rete predefiniti quando si desidera utilizzare plugin CNI alternativi come Cilium e Calico.

I componenti aggiuntivi EKS ora supportano la ricezione di autorizzazioni IAM tramite EKS Pod Identity Associations, permettendo loro di connettersi ai servizi AWS esterni al cluster

Creazione di componenti aggiuntivi

Eksctl offre maggiore flessibilità per la gestione dei componenti aggiuntivi del cluster:

Nel tuo file di configurazione, puoi specificare i componenti aggiuntivi che desideri e (se necessario) il ruolo o le politiche da allegare ad essi:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: example-cluster
  region: us-west-2

iam:
  withOIDC: true

addons:
- name: vpc-cni
  # all below properties are optional
  version: 1.7.5
  tags:
    team: eks
  # you can specify at most one of:
  attachPolicyARNs:
  - arn:aws:iam::account:policy/AmazonEKS_CNI_Policy
  # or
  serviceAccountRoleARN: arn:aws:iam::account:role/AmazonEKSCNIAccess
  # or
```

```
  attachPolicy:
    Statement:
    - Effect: Allow
      Action:
      - ec2:AssignPrivateIpAddresses
      - ec2:AttachNetworkInterface
      - ec2:CreateNetworkInterface
      - ec2:DeleteNetworkInterface
      - ec2:DescribeInstances
      - ec2:DescribeTags
      - ec2:DescribeNetworkInterfaces
      - ec2:DescribeInstanceTypes
      - ec2:DetachNetworkInterface
      - ec2:ModifyNetworkInterfaceAttribute
      - ec2:UnassignPrivateIpAddresses
      Resource: '*'
```

Puoi specificare al massimo uno di`attachPolicy`, e. `attachPolicyARNs` `serviceAccountRoleARN`

Se non viene specificato nessuno di questi, l'addon verrà creato con un ruolo a cui sono allegate tutte le politiche consigliate.

> **Note**
>
> Per allegare le politiche ai componenti aggiuntivi, il cluster deve essere abilitato. `OIDC` Se non è abilitato, ignoriamo tutte le politiche allegate.

Puoi quindi creare questi componenti aggiuntivi durante il processo di creazione del cluster:

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

Oppure crea i componenti aggiuntivi in modo esplicito dopo la creazione del cluster utilizzando il file di configurazione o i flag CLI:

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

```
eksctl create addon --name vpc-cni --version 1.7.5 --service-account-role-arn <role-
arn>
```

```
eksctl create addon --name aws-ebs-csi-driver --namespace-config 'namespace=custom-
namespace'
```

> **Tip**
>
> Usa il `--namespace-config` flag per distribuire i componenti aggiuntivi in uno spazio dei nomi personalizzato anziché nello spazio dei nomi predefinito.

Durante la creazione del componente aggiuntivo, se nel cluster esiste già una versione autogestita dell'addon, è possibile scegliere come risolvere i potenziali `configMap` conflitti impostando l'opzione tramite il file di configurazione, ad es. `resolveConflicts`

```
addons:
- name: vpc-cni
  attachPolicyARNs:
    - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
  resolveConflicts: overwrite
```

Per la creazione di addon, il `resolveConflicts` campo supporta tre valori distinti:

- `none`- EKS non modifica il valore. La creazione potrebbe fallire.
- `overwrite`- EKS sovrascrive qualsiasi modifica alla configurazione riportandola ai valori predefiniti di EKS.
- `preserve`- EKS non modifica il valore. La creazione potrebbe fallire. (Analogamente`none`, ma diverso dall'`preserve`aggiornamento dei componenti aggiuntivi).

Elencare componenti aggiuntivi abilitati

Puoi vedere quali componenti aggiuntivi sono abilitati nel tuo cluster eseguendo:

```
eksctl get addons --cluster <cluster-name>
```

or

```
eksctl get addons -f config.yaml
```

Impostazione della versione dell'addon

L'impostazione della versione del componente aggiuntivo è facoltativa. Se il `version` campo viene lasciato vuoto, `eksctl` verrà risolta la versione predefinita dell'addon. Ulteriori informazioni su quale versione è la versione predefinita per componenti aggiuntivi specifici sono disponibili nella documentazione AWS su EKS. Tieni presente che la versione predefinita potrebbe non essere necessariamente l'ultima versione disponibile.

La versione del componente aggiuntivo può essere impostata su. `latest` In alternativa, la versione può essere impostata con il tag di build EKS specificato, ad esempio `v1.7.5-eksbuild.1` o`v1.7.5-eksbuild.2`. Può anche essere impostato sulla versione di rilascio dell'addon, ad esempio `v1.7.5` o`1.7.5`, e il tag del `eksbuild` suffisso verrà scoperto e impostato automaticamente.

Consulta la sezione seguente su come scoprire i componenti aggiuntivi disponibili e le relative versioni.

Alla scoperta degli addon

Puoi scoprire quali componenti aggiuntivi sono disponibili per l'installazione sul tuo cluster eseguendo:

```
eksctl utils describe-addon-versions --cluster <cluster-name>
```

Questo scoprirà la versione Kubernetes del tuo cluster e filtrerà in base a quella. In alternativa, se vuoi vedere quali componenti aggiuntivi sono disponibili per una particolare versione di Kubernetes, puoi eseguire:

```
eksctl utils describe-addon-versions --kubernetes-version <version>
```

Puoi anche scoprire i componenti aggiuntivi filtrando in base ai loro,. `type` `owner` and/or `publisher` Ad esempio, per visualizzare i componenti aggiuntivi di un determinato proprietario e tipo, puoi eseguire:

```
eksctl utils describe-addon-versions --kubernetes-version 1.22 --types "infra-
management, policy-management" --owners "aws-marketplace"
```

I `publishers` flag`types`, `owners` e sono opzionali e possono essere specificati insieme o singolarmente per filtrare i risultati.

Alla scoperta dello schema di configurazione per i componenti aggiuntivi

Dopo aver scoperto l'addon e la versione, puoi visualizzare le opzioni di personalizzazione recuperando lo schema di configurazione JSON corrispondente.

```
eksctl utils describe-addon-configuration --name vpc-cni --version v1.12.0-eksbuild.1
```

Ciò restituisce uno schema JSON delle varie opzioni disponibili per questo componente aggiuntivo.

Lavorare con i valori di configurazione

`ConfigurationValues`può essere fornito nel file di configurazione durante la creazione o l'aggiornamento dei componenti aggiuntivi. Sono supportati solo i formati JSON e YAML.

Ad es. ,

```
addons:
- name: coredns
  configurationValues: |-
    replicaCount: 2
```

```
addons:
- name: coredns
  version: latest
  configurationValues: "{\"replicaCount\":3}"
  resolveConflicts: overwrite
```

> **Note**
>
> Tenete presente che quando i valori di configurazione degli addon vengono modificati, si verificano conflitti di configurazione.

```
Thus, we need to specify how to deal with those by setting the `resolveConflicts` field
 accordingly.
As in this scenario we want to modify these values, we'd set `resolveConflicts:
 overwrite`.
```

Inoltre, il comando get ora recupererà anche l'addon. `ConfigurationValues` Ad es.

```
eksctl get addon --cluster my-cluster --output yaml
```

```
- ConfigurationValues: '{"replicaCount":3}'
  IAMRole: ""
  Issues: null
  Name: coredns
  NewerVersion: ""
  Status: ACTIVE
  Version: v1.8.7-eksbuild.3
```

Utilizzo di uno spazio dei nomi personalizzato

È possibile fornire uno spazio dei nomi personalizzato nel file di configurazione durante la creazione di componenti aggiuntivi. Un namespace non può essere aggiornato una volta creato un componente aggiuntivo.

Utilizzo del file di configurazione

```
addons:
  - name: aws-ebs-csi-driver
    version: latest
    namespaceConfig:
      namespace: custom-namespace
```

Utilizzo del flag CLI

In alternativa, puoi specificare uno spazio dei nomi personalizzato usando il flag: `--namespace-config`

```
eksctl create addon --cluster my-cluster --name aws-ebs-csi-driver --namespace-config
 'namespace=custom-namespace'
```

Il comando get recupererà anche il valore dello spazio dei nomi per l'addon

```
- ConfigurationValues: ""
  IAMRole: ""
  Issues: null
  Name: aws-ebs-csi-driver
  NamespaceConfig:
```

```
    namespace: custom-namespace
  NewerVersion: ""
  PodIdentityAssociations: null
  Status: ACTIVE
  Version: v1.47.0-eksbuild.1
```

Aggiornamento dei componenti aggiuntivi

Puoi aggiornare i componenti aggiuntivi alle versioni più recenti e modificare le politiche allegate eseguendo:

```
eksctl update addon -f config.yaml
```

```
eksctl update addon --name vpc-cni --version 1.8.0 --service-account-role-arn <new-role>
```

> **Note**
>
> La configurazione dello spazio dei nomi non può essere aggiornata una volta creato un componente aggiuntivo. Il `--namespace-config` flag è disponibile solo durante la creazione dell'addon.

Analogamente alla creazione di un componente aggiuntivo, quando aggiorni un componente aggiuntivo, hai il pieno controllo sulle modifiche alla configurazione che potresti aver applicato in precedenza a quel componente aggiuntivo. `configMap` In particolare, puoi conservarli o sovrascriverli. Questa funzionalità opzionale è disponibile tramite lo stesso campo `resolveConflicts` del file di configurazione. ad esempio,

```
addons:
- name: vpc-cni
  attachPolicyARNs:
    - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
  resolveConflicts: preserve
```

Per l'aggiornamento dei componenti aggiuntivi, il `resolveConflicts` campo accetta tre valori distinti:

- none- EKS non modifica il valore. L'aggiornamento potrebbe fallire.

- `overwrite`- EKS sovrascrive qualsiasi modifica alla configurazione riportandola ai valori predefiniti di EKS.
- `preserve`- EKS conserva il valore. Se scegli questa opzione, ti consigliamo di testare qualsiasi modifica di campo e valore su un cluster non di produzione prima di aggiornare il componente aggiuntivo nel tuo cluster di produzione.

Eliminazione di componenti aggiuntivi

Puoi eliminare un componente aggiuntivo eseguendo:

```
eksctl delete addon --cluster <cluster-name> --name <addon-name>
```

Questo eliminerà l'addon e tutti i ruoli IAM ad esso associati.

Quando elimini il cluster, vengono eliminati anche tutti i ruoli IAM associati ai componenti aggiuntivi.

Flessibilità di creazione di cluster per componenti aggiuntivi di rete predefiniti

Quando viene creato un cluster, EKS installa automaticamente VPC CNI, CoredNS e kube-proxy come componenti aggiuntivi autogestiti. Per disabilitare questo comportamento al fine di utilizzare altri plugin CNI come Cilium e Calico, eksctl ora supporta la creazione di un cluster senza componenti aggiuntivi di rete predefiniti. Per creare un cluster di questo tipo, imposta, come in: `addonsConfig.disableDefaultAddons`

```
addonsConfig:
  disableDefaultAddons: true
```

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

Per creare un cluster con solo CoredNS e kube-proxy e non VPC CNI, specifica gli addon in modo esplicito e imposta, come in: `addons` `addonsConfig.disableDefaultAddons`

```
addonsConfig:
  disableDefaultAddons: true
addons:
  - name: kube-proxy
```

```
  - name: coredns
```

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

Come parte di questa modifica, eksctl ora installa i componenti aggiuntivi predefiniti come componenti aggiuntivi EKS anziché componenti aggiuntivi autogestiti durante la creazione del cluster se non è impostato esplicitamente su true. `addonsConfig.disableDefaultAddons` Pertanto, `eksctl utils update-*` i comandi non possono più essere utilizzati per aggiornare i componenti aggiuntivi per i cluster creati con eksctl v0.184.0 e versioni successive:

- `eksctl utils update-aws-node`
- `eksctl utils update-coredns`
- `eksctl utils update-kube-proxy`

Invece`eksctl update addon`, dovrebbe essere usato ora.

Per ulteriori informazioni, consulta Amazon EKS introduce la flessibilità di creazione di cluster per componenti aggiuntivi di rete.

Abilitazione dell'accesso per Amazon EMR

Per consentire a EMR di eseguire operazioni sull'API Kubernetes, alla sua SLR devono essere concesse le autorizzazioni RBAC richieste. eksctl fornisce un comando che crea le risorse RBAC richieste per EMR e aggiorna il ruolo per associare il ruolo alla SLR per EMR. `aws-auth` ConfigMap

```
eksctl create iamidentitymapping --cluster dev --service-name emr-containers --
namespace default
```

Assistenza EKS Fargate

AWS Fargate è un motore di elaborazione gestito per Amazon ECS in grado di eseguire container. In Fargate non è necessario gestire server o cluster.

Amazon EKS ora può lanciare pod su AWS Fargate. Ciò elimina la necessità di preoccuparsi di come fornire o gestire l'infrastruttura per i pod e semplifica la creazione e l'esecuzione di applicazioni Kubernetes performanti e ad alta disponibilità su AWS.

Creazione di un cluster con il supporto di Fargate

Puoi aggiungere un cluster con supporto Fargate con:

```
eksctl create cluster --fargate
[#]  eksctl version 0.11.0
[#]  using region ap-northeast-1
[#]  setting availability zones to [ap-northeast-1a ap-northeast-1d ap-northeast-1c]
[#]  subnets for ap-northeast-1a - public:192.168.0.0/19 private:192.168.96.0/19
[#]  subnets for ap-northeast-1d - public:192.168.32.0/19 private:192.168.128.0/19
[#]  subnets for ap-northeast-1c - public:192.168.64.0/19 private:192.168.160.0/19
[#]  nodegroup "ng-dba9d731" will use "ami-02e124a380df41614" [AmazonLinux2/1.14]
[#]  using Kubernetes version 1.14
[#]  creating EKS cluster "ridiculous-painting-1574859263" in "ap-northeast-1" region
[#]  will create 2 separate CloudFormation stacks for cluster itself and the initial
 nodegroup
[#]  if you encounter any issues, check CloudFormation console or try 'eksctl utils
 describe-stacks --region=ap-northeast-1 --cluster=ridiculous-painting-1574859263'
[#]  CloudWatch logging will not be enabled for cluster "ridiculous-
painting-1574859263" in "ap-northeast-1"
[#]  you can enable it with 'eksctl utils update-cluster-logging --enable-
types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-northeast-1 --
cluster=ridiculous-painting-1574859263'
[#]  Kubernetes API endpoint access will use default of {publicAccess=true,
 privateAccess=false} for cluster "ridiculous-painting-1574859263" in "ap-northeast-1"
[#]  2 sequential tasks: { create cluster control plane "ridiculous-
painting-1574859263", create nodegroup "ng-dba9d731" }
[#]  building cluster stack "eksctl-ridiculous-painting-1574859263-cluster"
[#]  deploying stack "eksctl-ridiculous-painting-1574859263-cluster"
[#]  building nodegroup stack "eksctl-ridiculous-painting-1574859263-nodegroup-ng-
dba9d731"
[#]  --nodes-min=2 was set automatically for nodegroup ng-dba9d731
[#]  --nodes-max=2 was set automatically for nodegroup ng-dba9d731
[#]  deploying stack "eksctl-ridiculous-painting-1574859263-nodegroup-ng-dba9d731"
[#]  all EKS cluster resources for "ridiculous-painting-1574859263" have been created
[#]  saved kubeconfig as "/Users/marc/.kube/config"
[#]  adding identity "arn:aws:iam::123456789012:role/eksctl-ridiculous-painting-157485-
NodeInstanceRole-104DXUJOFDPO5" to auth ConfigMap
[#]  nodegroup "ng-dba9d731" has 0 node(s)
[#]  waiting for at least 2 node(s) to become ready in "ng-dba9d731"
[#]  nodegroup "ng-dba9d731" has 2 node(s)
[#]  node "ip-192-168-27-156.ap-northeast-1.compute.internal" is ready
[#]  node "ip-192-168-95-177.ap-northeast-1.compute.internal" is ready
```

```
[#]  creating Fargate profile "default" on EKS cluster "ridiculous-painting-1574859263"
[#]  created Fargate profile "default" on EKS cluster "ridiculous-painting-1574859263"
[#]  kubectl command should work with "/Users/marc/.kube/config", try 'kubectl get
 nodes'
[#]  EKS cluster "ridiculous-painting-1574859263" in "ap-northeast-1" region is ready
```

Questo comando avrà creato un cluster e un profilo Fargate. Questo profilo contiene alcune informazioni necessarie ad AWS per creare istanze di pod in Fargate. Questi sono:

- ruolo di esecuzione del pod per definire le autorizzazioni necessarie per eseguire il pod e la posizione di rete (sottorete) per eseguire il pod. Ciò consente di applicare le stesse autorizzazioni di rete e sicurezza a più pod Fargate e semplifica la migrazione dei pod esistenti su un cluster a Fargate.
- Selettore per definire quali pod devono essere eseguiti su Fargate. Questo è composto da una e. `namespace` `labels`

Quando il profilo non è specificato ma il supporto per Fargate è abilitato, viene creato `--fargate` un profilo Fargate predefinito. Questo profilo ha come target gli `default` e i `kube-system` namespace in modo che i pod in quei namespace vengano eseguiti su Fargate.

Il profilo Fargate che è stato creato può essere controllato con il seguente comando:

```
eksctl get fargateprofile --cluster ridiculous-painting-1574859263 -o yaml
- name: fp-default
  podExecutionRoleARN: arn:aws:iam::123456789012:role/eksctl-ridiculous-
painting-1574859263-ServiceRole-EIFQOH0S1GE7
  selectors:
  - namespace: default
  - namespace: kube-system
  subnets:
  - subnet-0b3a5522f3b48a742
  - subnet-0c35f1497067363f3
  - subnet-0a29aa00b25082021
```

Per ulteriori informazioni sui selettori, consulta Progettazione dei profili Fargate.

Creazione di un cluster con supporto Fargate utilizzando un file di configurazione

Il seguente file di configurazione dichiara un cluster EKS con un nodegroup composto da un'istanza EC2 `m5.large` e due profili Fargate. Tutti i pod definiti nei `kube-system` namespace `default` and verranno eseguiti su Fargate. Tutti i pod nel `dev` namespace che hanno anche l'etichetta `dev=passed` funzioneranno anche su Fargate. Tutti gli altri pod verranno programmati sul nodo in. `ng-1`

```
# An example of ClusterConfig with a normal nodegroup and a Fargate profile.
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: fargate-cluster
  region: ap-northeast-1

nodeGroups:
  - name: ng-1
    instanceType: m5.large
    desiredCapacity: 1

fargateProfiles:
  - name: fp-default
    selectors:
      # All workloads in the "default" Kubernetes namespace will be
      # scheduled onto Fargate:
      - namespace: default
      # All workloads in the "kube-system" Kubernetes namespace will be
      # scheduled onto Fargate:
      - namespace: kube-system
  - name: fp-dev
    selectors:
      # All workloads in the "dev" Kubernetes namespace matching the following
      # label selectors will be scheduled onto Fargate:
      - namespace: dev
        labels:
          env: dev
          checks: passed
```

```
eksctl create cluster -f cluster-fargate.yaml
[#]  eksctl version 0.11.0
[#]  using region ap-northeast-1
[#]  setting availability zones to [ap-northeast-1c ap-northeast-1a ap-northeast-1d]
[#]  subnets for ap-northeast-1c - public:192.168.0.0/19 private:192.168.96.0/19
[#]  subnets for ap-northeast-1a - public:192.168.32.0/19 private:192.168.128.0/19
[#]  subnets for ap-northeast-1d - public:192.168.64.0/19 private:192.168.160.0/19
[#]  nodegroup "ng-1" will use "ami-02e124a380df41614" [AmazonLinux2/1.14]
[#]  using Kubernetes version 1.14
[#]  creating EKS cluster "fargate-cluster" in "ap-northeast-1" region with Fargate
 profile and un-managed nodes
[#]  1 nodegroup (ng-1) was included (based on the include/exclude rules)
[#]  will create a CloudFormation stack for cluster itself and 1 nodegroup stack(s)
[#]  will create a CloudFormation stack for cluster itself and 0 managed nodegroup
 stack(s)
[#]  if you encounter any issues, check CloudFormation console or try 'eksctl utils
 describe-stacks --region=ap-northeast-1 --cluster=fargate-cluster'
[#]  CloudWatch logging will not be enabled for cluster "fargate-cluster" in "ap-
northeast-1"
[#]  you can enable it with 'eksctl utils update-cluster-logging --enable-
types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-northeast-1 --
cluster=fargate-cluster'
[#]  Kubernetes API endpoint access will use default of {publicAccess=true,
 privateAccess=false} for cluster "fargate-cluster" in "ap-northeast-1"
[#]  2 sequential tasks: { create cluster control plane "fargate-cluster", create
 nodegroup "ng-1" }
[#]  building cluster stack "eksctl-fargate-cluster-cluster"
[#]  deploying stack "eksctl-fargate-cluster-cluster"
[#]  building nodegroup stack "eksctl-fargate-cluster-nodegroup-ng-1"
[#]  --nodes-min=1 was set automatically for nodegroup ng-1
[#]  --nodes-max=1 was set automatically for nodegroup ng-1
[#]  deploying stack "eksctl-fargate-cluster-nodegroup-ng-1"
[#]  all EKS cluster resources for "fargate-cluster" have been created
[#]  saved kubeconfig as "/home/user1/.kube/config"
[#]  adding identity "arn:aws:iam::123456789012:role/eksctl-fargate-cluster-nod-
NodeInstanceRole-42Q80B2Z147I" to auth ConfigMap
[#]  nodegroup "ng-1" has 0 node(s)
[#]  waiting for at least 1 node(s) to become ready in "ng-1"
[#]  nodegroup "ng-1" has 1 node(s)
[#]  node "ip-192-168-71-83.ap-northeast-1.compute.internal" is ready
[#]  creating Fargate profile "fp-default" on EKS cluster "fargate-cluster"
[#]  created Fargate profile "fp-default" on EKS cluster "fargate-cluster"
[#]  creating Fargate profile "fp-dev" on EKS cluster "fargate-cluster"
```

```
[#]  created Fargate profile "fp-dev" on EKS cluster "fargate-cluster"
[#]  "coredns" is now schedulable onto Fargate
[#]  "coredns" is now scheduled onto Fargate
[#]  "coredns" is now scheduled onto Fargate
[#]  "coredns" pods are now scheduled onto Fargate
[#]  kubectl command should work with "/home/user1/.kube/config", try 'kubectl get
 nodes'
[#]  EKS cluster "fargate-cluster" in "ap-northeast-1" region is ready
```

Progettazione dei profili Fargate

Ogni voce del selettore ha fino a due componenti, namespace e un elenco di coppie chiave-valore. Per creare una voce di selezione è necessario solo il componente namespace. Tutte le regole (namespace, coppie chiave-valore) devono essere applicate a un pod per corrispondere a una voce del selettore. Un pod deve corrispondere solo a una voce del selettore per essere eseguito sul profilo. Qualsiasi pod che soddisfi tutte le condizioni di un campo di selezione verrà programmato per essere eseguito su Fargate. Tutti i pod che non corrispondono ai namespace inseriti nella whitelist ma in cui l'utente imposta manualmente lo scheduler: fargate-scheduler filed sarebbero bloccati in uno stato Pending, poiché non erano autorizzati a funzionare su Fargate.

I profili devono soddisfare i seguenti requisiti:

- Un selettore è obbligatorio per profilo
- Ogni selettore deve includere uno spazio dei nomi; le etichette sono opzionali

Esempio: pianificazione del carico di lavoro in Fargate

Per pianificare i pod su Fargate per l'esempio sopra menzionato, si potrebbe, ad esempio, creare un namespace `dev` chiamato e distribuirvi il carico di lavoro:

```
kubectl create namespace dev
namespace/dev created

kubectl run nginx --image=nginx --restart=Never --namespace dev
pod/nginx created

kubectl get pods --all-namespaces --output wide
NAMESPACE     NAME                        READY   STATUS    AGE   IP                NODE
dev           nginx                       1/1     Running   75s   192.168.183.140
 fargate-ip-192-168-183-140.ap-northeast-1.compute.internal
```

```
kube-system   aws-node-44qst             1/1     Running   21m   192.168.70.246
 ip-192-168-70-246.ap-northeast-1.compute.internal
kube-system   aws-node-4vr66             1/1     Running   21m   192.168.23.122
 ip-192-168-23-122.ap-northeast-1.compute.internal
kube-system   coredns-699bb99bf8-84x74   1/1     Running   26m   192.168.2.95
 ip-192-168-23-122.ap-northeast-1.compute.internal
kube-system   coredns-699bb99bf8-f6x6n   1/1     Running   26m   192.168.90.73
 ip-192-168-70-246.ap-northeast-1.compute.internal
kube-system   kube-proxy-brxhg           1/1     Running   21m   192.168.23.122
 ip-192-168-23-122.ap-northeast-1.compute.internal
kube-system   kube-proxy-zd7s8           1/1     Running   21m   192.168.70.246
 ip-192-168-70-246.ap-northeast-1.compute.internal
```

Dall'output dell'ultimo `kubectl get pods` comando possiamo vedere che il `nginx` pod è distribuito in un nodo chiamato. `fargate-ip-192-168-183-140.ap-northeast-1.compute.internal`

Gestione dei profili Fargate

Per distribuire i carichi di lavoro Kubernetes su Fargate, EKS necessita di un profilo Fargate. Quando crea un cluster come negli esempi precedenti, si `eksctl` occupa di questo problema creando un profilo predefinito. Dato un cluster già esistente, è anche possibile creare un profilo Fargate con il `eksctl create fargateprofile` comando:

> **Note**
>
> Questa operazione è supportata solo su cluster che funzionano sulla versione della piattaforma EKS `eks.5` o superiore.

> **Note**
>
> Se l'esistente è stato creato con una versione `eksctl` precedente alla 0.11.0, sarà necessario eseguirlo `eksctl upgrade cluster` prima di creare il profilo Fargate.

```
eksctl create fargateprofile --namespace dev --cluster fargate-example-cluster
[#]  creating Fargate profile "fp-9bfc77ad" on EKS cluster "fargate-example-cluster"
[#]  created Fargate profile "fp-9bfc77ad" on EKS cluster "fargate-example-cluster"
```

È inoltre possibile specificare il nome del profilo Fargate da creare. Questo nome non deve iniziare con il prefisso`eks-`.

```
eksctl create fargateprofile --namespace dev --cluster fargate-example-cluster --name
 fp-development
[#]  created Fargate profile "fp-development" on EKS cluster "fargate-example-cluster"
```

Utilizzando questo comando con i flag CLI eksctl può creare un solo profilo Fargate con un semplice selettore. Per selettori più complessi, ad esempio con più namespace, eksctl supporta l'uso di un file di configurazione:

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

metadata:
  name: fargate-example-cluster
  region: ap-northeast-1

fargateProfiles:
  - name: fp-default
    selectors:
      # All workloads in the "default" Kubernetes namespace will be
      # scheduled onto Fargate:
      - namespace: default
      # All workloads in the "kube-system" Kubernetes namespace will be
      # scheduled onto Fargate:
      - namespace: kube-system
  - name: fp-dev
    selectors:
      # All workloads in the "dev" Kubernetes namespace matching the following
      # label selectors will be scheduled onto Fargate:
      - namespace: dev
        labels:
          env: dev
          checks: passed
```

```
eksctl create fargateprofile -f fargate-example-cluster.yaml
[#]  creating Fargate profile "fp-default" on EKS cluster "fargate-example-cluster"
[#]  created Fargate profile "fp-default" on EKS cluster "fargate-example-cluster"
[#]  creating Fargate profile "fp-dev" on EKS cluster "fargate-example-cluster"
[#]  created Fargate profile "fp-dev" on EKS cluster "fargate-example-cluster"
[#]  "coredns" is now scheduled onto Fargate
```

```
[#]  "coredns" pods are now scheduled onto Fargate
```

Per visualizzare i profili Fargate esistenti in un cluster:

```
eksctl get fargateprofile --cluster fargate-example-cluster
NAME         SELECTOR_NAMESPACE  SELECTOR_LABELS  POD_EXECUTION_ROLE_ARN
                                                    SUBNETS
fp-9bfc77ad  dev                 <none>           arn:aws:iam::123456789012:role/
eksctl-fargate-example-cluster-ServiceRole-1T5F78E5FSH79
 subnet-00adf1d8c99f83381,subnet-04affb163ffab17d4,subnet-035b34379d5ef5473
```

E per vederli in `yaml` formato:

```
eksctl get fargateprofile --cluster fargate-example-cluster -o yaml
- name: fp-9bfc77ad
  podExecutionRoleARN: arn:aws:iam::123456789012:role/eksctl-fargate-example-cluster-
ServiceRole-1T5F78E5FSH79
  selectors:
  - namespace: dev
  subnets:
  - subnet-00adf1d8c99f83381
  - subnet-04affb163ffab17d4
  - subnet-035b34379d5ef5473
```

O in `json` formato:

```
eksctl get fargateprofile --cluster fargate-example-cluster -o json
[
    {
        "name": "fp-9bfc77ad",
        "podExecutionRoleARN": "arn:aws:iam::123456789012:role/eksctl-fargate-example-
cluster-ServiceRole-1T5F78E5FSH79",
        "selectors": [
            {
                "namespace": "dev"
            }
        ],
        "subnets": [
            "subnet-00adf1d8c99f83381",
            "subnet-04affb163ffab17d4",
            "subnet-035b34379d5ef5473"
        ]
```

```
    }
]
```

I profili Fargate sono immutabili per progettazione. Per cambiare qualcosa, create un nuovo profilo Fargate con le modifiche desiderate ed eliminate quello vecchio con il `eksctl delete fargateprofile` comando come nell'esempio seguente:

```
eksctl delete fargateprofile --cluster fargate-example-cluster --name fp-9bfc77ad --
wait
2019-11-27T19:04:26+09:00 [#]  deleting Fargate profile "fp-9bfc77ad"
  ClusterName: "fargate-example-cluster",
  FargateProfileName: "fp-9bfc77ad"
}
```

Tieni presente che l'eliminazione del profilo è un processo che può richiedere fino a qualche minuto. Quando il `--wait` flag non è specificato, si aspetta `eksctl` ottimisticamente che il profilo venga eliminato e ritorna non appena la richiesta API AWS è stata inviata. Per `eksctl` attendere che il profilo sia stato eliminato con successo, usa `--wait` come nell'esempio precedente.

Approfondimenti

- AWS Fargate
- Amazon EKS ora può lanciare pod su AWS Fargate

Aggiornamenti del cluster

Un cluster gestito da `eksctl`può essere aggiornato in 3 semplici passaggi:

1. aggiorna la versione del piano di controllo con `eksctl upgrade cluster`
2. aggiornare i gruppi di nodi
3. aggiorna i componenti aggiuntivi di rete predefiniti (per ulteriori informazioni, consulta): the section called “Aggiornamenti aggiuntivi predefiniti”

Esamina attentamente le risorse relative all'aggiornamento del cluster:

- Aggiorna il cluster esistente alla nuova versione di Kubernetes nella Guida per l'utente di Amazon EKS

- Le migliori pratiche per gli aggiornamenti dei cluster nella Guida alle migliori pratiche di EKS

> **Note**
>
> Il vecchio `eksctl update cluster` sarà obsoleto. Usare invece `eksctl upgrade cluster`.

Aggiornamento della versione del piano di controllo

Gli aggiornamenti della versione del piano di controllo devono essere eseguiti per una versione secondaria alla volta.

Per aggiornare Control Plane alla prossima versione disponibile, esegui:

```
eksctl upgrade cluster --name=<clusterName>
```

Questo comando non applicherà immediatamente alcuna modifica, sarà necessario eseguirlo nuovamente `--approve` per applicare le modifiche.

La versione di destinazione per l'aggiornamento del cluster può essere specificata sia con il flag CLI:

```
eksctl upgrade cluster --name=<clusterName> --version=1.16
```

o con il file di configurazione

```
cat cluster1.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: cluster-1
  region: eu-north-1
  version: "1.16"

eksctl upgrade cluster --config-file cluster1.yaml
```

> **Warning**
>
> Gli unici valori consentiti per gli `metadata.version` argomenti `--version` and sono la versione corrente del cluster o una versione successiva. Gli aggiornamenti di più di una versione di Kubernetes non sono supportati.

Aggiornamenti aggiuntivi predefiniti

Questo argomento spiega come aggiornare i componenti aggiuntivi preinstallati preinstallati inclusi nei cluster EKS.

> **Warning**
>
> eksctl ora installa i componenti aggiuntivi predefiniti come componenti aggiuntivi EKS anziché componenti aggiuntivi autogestiti. Scopri di più sulle sue implicazioni in Flessibilità di creazione di cluster per componenti aggiuntivi di rete predefiniti.
> Per l'aggiornamento dei componenti aggiuntivi, `eksctl utils update-<addon>` non possono essere utilizzati per i cluster creati con eksctl v0.184.0 e versioni successive. Questa guida è valida solo per i cluster creati prima di questa modifica.

Esistono 3 componenti aggiuntivi predefiniti che vengono inclusi in ogni cluster EKS:

- `kube-proxy`
- `aws-node`
- `coredns`

Aggiorna il componente aggiuntivo preinstallato

Per i componenti aggiuntivi EKS ufficiali che vengono creati manualmente tramite `eksctl create addons` o al momento della creazione del cluster, il modo per gestirli è tramite. `eksctl create/get/update/delete addon` In questi casi, consulta la documentazione relativa ai componenti aggiuntivi EKS.

Il processo di aggiornamento di ciascuno di essi è diverso, quindi è necessario eseguire 3 comandi distinti. Tutti i seguenti comandi accettano`--config-file`. Per impostazione predefinita, ognuno di

questi comandi viene eseguito in modalità piano, se sei soddisfatto delle modifiche proposte, esegui nuovamente con`--approve`.

Per aggiornare`kube-proxy`, esegui:

```
eksctl utils update-kube-proxy --cluster=<clusterName>
```

Per aggiornare`aws-node`, esegui:

```
eksctl utils update-aws-node --cluster=<clusterName>
```

Per aggiornare`coredns`, esegui:

```
eksctl utils update-coredns --cluster=<clusterName>
```

Una volta aggiornato, assicuratevi di avviare `kubectl get pods -n kube-system` e controllate se tutti gli addon pod sono pronti, dovreste vedere qualcosa del genere:

```
NAME                       READY   STATUS    RESTARTS   AGE
aws-node-g5ghn             1/1     Running   0          2m
aws-node-zfc9s             1/1     Running   0          2m
coredns-7bcbfc4774-g6gg8   1/1     Running   0          1m
coredns-7bcbfc4774-hftng   1/1     Running   0          1m
kube-proxy-djkp7           1/1     Running   0          3m
kube-proxy-mpdsp           1/1     Running   0          3m
```

Support per Zonal Shift nei cluster EKS

EKS ora supporta lo spostamento di zona di Amazon Application Recovery Controller (ARC) e lo spostamento automatico di zona che migliorano la resilienza degli ambienti cluster Multi-AZ. Con AWS Zonal Shift, i clienti possono spostare il traffico all'interno del cluster da una zona di disponibilità ridotta, assicurando che i nuovi pod e nodi Kubernetes vengano lanciati solo in zone di disponibilità integre.

Creazione di un cluster con lo spostamento di zona abilitato

```
# zonal-shift-cluster.yaml
---
```

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

metadata:
  name: highly-available-cluster
  region: us-west-2


zonalShiftConfig:
  enabled: true
```

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

Abilitazione dello spostamento zonale su un cluster esistente

Per abilitare o disabilitare lo spostamento zonale su un cluster esistente, esegui

```
eksctl utils update-zonal-shift-config -f zonal-shift-cluster.yaml
```

o senza un file di configurazione:

```
eksctl utils update-zonal-shift-config --cluster=zonal-shift-cluster --enabled
```

Ulteriori informazioni

- Spostamento zonale EKS

Assistenza Karpenter

`eksctl`fornisce supporto per aggiungere Karpenter a un cluster appena creato. Creerà tutti i prerequisiti necessari descritti nella sezione Guida introduttiva di Karpenter, inclusa l'installazione di Karpenter stesso tramite Helm. Al `0.28.0`+ momento supportiamo l'installazione delle versioni. Consulta la sezione sulla compatibilità di Karpenter per ulteriori dettagli.

La seguente configurazione del cluster delinea una tipica installazione di Karpenter:

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

```
metadata:
  name: cluster-with-karpenter
  region: us-west-2
  version: '1.32' # requires a version of Kubernetes compatible with Karpenter
  tags:
    karpenter.sh/discovery: cluster-with-karpenter # here, it is set to the cluster
 name
iam:
  withOIDC: true # required

karpenter:
  version: '1.2.1' # Exact version should be specified according to the Karpenter
 compatibility matrix

managedNodeGroups:
  - name: managed-ng-1
    minSize: 1
    maxSize: 2
    desiredCapacity: 1
```

La versione è la versione di Karpenter in quanto può essere trovata nel loro Helm Repository. È inoltre possibile impostare le seguenti opzioni:

```
karpenter:
  version: '1.2.1'
  createServiceAccount: true # default is false
  defaultInstanceProfile: 'KarpenterNodeInstanceProfile' # default is to use the IAM
 instance profile created by eksctl
  withSpotInterruptionQueue: true # adds all required policies and rules for supporting
 Spot Interruption Queue, default is false
```

È necessario definire OIDC per installare Karpenter.

Una volta installato correttamente Karpenter, aggiungi NodePool(s) e NodeClass(es) per consentire a Karpenter di iniziare ad aggiungere nodi al cluster.

La `nodeClassRef` sezione NodePool 's' deve corrispondere al nome di un. `EC2NodeClass` Esempio:

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

```
  name: example
  annotations:
    kubernetes.io/description: "Example NodePool"
spec:
  template:
    spec:
      requirements:
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64"]
        - key: kubernetes.io/os
          operator: In
          values: ["linux"]
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["on-demand"]
        - key: karpenter.k8s.aws/instance-category
          operator: In
          values: ["c", "m", "r"]
        - key: karpenter.k8s.aws/instance-generation
          operator: Gt
          values: ["2"]
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: example # must match the name of an EC2NodeClass
```

```
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: example
  annotations:
    kubernetes.io/description: "Example EC2NodeClass"
spec:
  role: "eksctl-KarpenterNodeRole-${CLUSTER_NAME}" # replace with your cluster name
  subnetSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${CLUSTER_NAME}" # replace with your cluster name
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${CLUSTER_NAME}" # replace with your cluster name
  amiSelectorTerms:
    - alias: al2023@latest # Amazon Linux 2023
```

Nota che devi specificare uno dei `role` o `instanceProfile` per i nodi di avvio. Se scegli di utilizzare `instanceProfile` il nome del profilo creato da `eksctl` segue lo schema:`eksctl-KarpenterNodeInstanceProfile-<cluster-name>`.

Etichettatura automatica dei gruppi di sicurezza

`eksctl`etichetta automaticamente il gruppo di sicurezza del nodo condiviso del cluster con `karpenter.sh/discovery` quando sia Karpenter è abilitato (`karpenter.version`specificato) sia il `karpenter.sh/discovery` tag è presente in. `metadata.tags` Ciò consente la compatibilità con AWS Load Balancer Controller.

Nota: con karpenter 0.32.0+, i Provisioner sono stati resi obsoleti e sostituiti da. NodePool

Schema di configurazione del cluster

> **Note**
>
> La posizione dello schema è attualmente in fase di migrazione.

È possibile utilizzare un file yaml per creare un cluster. Visualizza il riferimento allo schema.

Esempio:

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

Il riferimento allo schema per questo file è disponibile su GitHub.

Per ulteriori informazioni sull'utilizzo del file, vederethe section called “Creazione e gestione di cluster”.

Gruppi di nodi

Questo capitolo include informazioni su come creare e configurare i gruppi di nodi con Eksctl. I gruppi di nodi sono gruppi di istanze EC2 collegate a un cluster EKS.

Argomenti:

- the section called “Istanze Spot”
  - Crea e gestisci cluster EKS con istanze Spot utilizzando gruppi di nodi gestiti
  - Configura le istanze Spot per gruppi di nodi non gestiti utilizzando il MixedInstancesPolicy
  - Distingui le istanze Spot e On-Demand utilizzando l'etichetta Kubernetes `node-lifecycle`
- the section called “Auto Scaling”
  - Abilita la scalabilità automatica dei nodi del cluster Kubernetes creando un cluster o un gruppo di nodi con ruolo IAM che consente l'uso dell'autoscaler del cluster
  - Configura le definizioni dei gruppi di nodi per includere i tag e le annotazioni necessari affinché l'autoscaler del cluster possa scalare il gruppo di nodi
  - Crea gruppi di nodi separati per ogni zona di disponibilità se i carichi di lavoro hanno requisiti specifici della zona, come lo storage specifico della zona o le regole di affinità
- the section called “Gruppi di nodi gestiti da EKS”
  - Esegui il provisioning e gestisci istanze EC2 (nodi) per i cluster Amazon EKS Kubernetes
  - Applica facilmente correzioni di bug, patch di sicurezza e aggiorna i nodi alle ultime versioni di Kubernetes
- the section called “Nodi ibridi EKS”
  - Abilita l'esecuzione di applicazioni locali e periferiche su un'infrastruttura gestita dal cliente con gli stessi cluster, funzionalità e strumenti AWS EKS utilizzati nel cloud AWS
  - Configura la rete per connettere le reti locali a un VPC AWS, utilizzando opzioni come AWS VPN o Site-to-Site AWS Direct Connect
  - Configura le credenziali per i nodi remoti per l'autenticazione con il cluster EKS, utilizzando AWS Systems Manager (SSM) o AWS IAM Roles Anywhere
- the section called “Config di riparazione dei nodi”
  - Attivazione di Node Repair for EKS Managed Nodegroups per monitorare e sostituire o riavviare automaticamente nodi di lavoro non integri

- the section called “Supporto ARM”
  - Crea un cluster EKS con istanze Graviton basate su ARM per migliorare le prestazioni e l'efficienza dei costi
- the section called “Tinte”
  - Applica i taint a gruppi di nodi specifici in un cluster Kubernetes
  - Controlla la pianificazione e l'eliminazione dei pod in base a chiavi, valori ed effetti di taint
- the section called “Supporto modello di avvio”
  - Avvio di gruppi di nodi gestiti utilizzando un modello di avvio EC2 fornito
  - Aggiornamento di un gruppo di nodi gestiti per utilizzare una versione diversa di un Launch Template
  - Comprensione delle limitazioni e delle considerazioni relative all'utilizzo di modelli personalizzati AMIs e di Launch con gruppi di nodi gestiti
- the section called “Lavora con i gruppi di nodi”
  - Abilita l'accesso SSH alle istanze EC2 nel gruppo di nodi
  - Aumenta o riduci il numero di nodi in un gruppo di nodi
- the section called “Sottoreti personalizzate”
  - Estendi un VPC esistente con una nuova sottorete e aggiungi un gruppo di nodi a quella sottorete
- the section called “Bootstrap dei nodi”
  - Comprendi il nuovo processo di inizializzazione dei nodi (nodeadm) introdotto nel 2023 AmazonLinux
  - Scopri NodeConfig le impostazioni predefinite applicate da eksctl per i nodi autogestiti e gestiti da EKS
  - Personalizza il processo di avvio dei nodi fornendone uno personalizzato overrideBootstrapCommand NodeConfig
- the section called “Gruppi di nodi non gestiti”
  - Crea o aggiorna gruppi di nodi non gestiti in un cluster EKS
  - Aggiorna i componenti aggiuntivi predefiniti di Kubernetes come kube-proxy, aws-node e CoredNS
- the section called “Supporto GPU”
  - Eksctl supporta la selezione dei tipi di istanze GPU per i gruppi di nodi, abilitando l'uso di carichi di lavoro accelerati da GPU sui cluster EKS.

  - Eksctl installa automaticamente il plug-in per dispositivi NVIDIA Kubernetes quando viene selezionato un tipo di istanza abilitato alla GPU, facilitando l'uso delle risorse GPU nel cluster.
  - Gli utenti possono disabilitare l'installazione automatica del plug-in e installare manualmente una versione specifica del plug-in del dispositivo NVIDIA Kubernetes utilizzando i comandi forniti.
- the section called “Selettore di istanze”
  - Genera automaticamente un elenco di tipi di istanze EC2 adatti in base a criteri di risorse come vCPUs, memoria e architettura CPU GPUs
  - Crea cluster e gruppi di nodi con i tipi di istanze corrispondenti ai criteri di selezione delle istanze specificati
  - Eseguite un dry run per ispezionare e modificare i tipi di istanze corrispondenti al selettore di istanze prima di creare un gruppo di nodi
- the section called “Mappature di volume aggiuntive”
  - Configura mappature di volume aggiuntive per un gruppo di nodi gestito in un cluster EKS
  - Personalizza le proprietà del volume come dimensione, tipo, crittografia, IOPS e velocità effettiva per i volumi aggiuntivi
  - Allega le istantanee EBS esistenti come volumi aggiuntivi al gruppo di nodi
- the section called “Nodi Windows Worker”
  - Aggiungi gruppi di nodi Windows a un cluster Linux Kubernetes esistente per consentire l'esecuzione di carichi di lavoro Windows
  - Pianifica i carichi di lavoro sul sistema operativo appropriato (Windows o Linux) utilizzando selettori di nodi basati sulle etichette e `kubernetes.io/os` `kubernetes.io/arch`
- the section called “Supporto AMI personalizzato”
  - Usa il `--node-ami` flag per specificare un'AMI personalizzata per i gruppi di nodi, interroga AWS per l'ultima AMI ottimizzata per EKS o usa AWS Systems Manager Parameter Store per trovare l'AMI.
  - Imposta il `--node-ami-family` flag per specificare la famiglia di sistemi operativi per l'AMI del gruppo di nodi, ad esempio AmazonLinux 2, Ubuntu2204 o 2022. WindowsServer CoreContainer
  - Per i gruppi di nodi Windows, specifica un AMI personalizzato e fornisci uno script di PowerShell bootstrap tramite. `overrideBootstrapCommand`
- the section called “DNS personalizzato”
  - Sovrascrivi l'indirizzo IP del server DNS utilizzato per le ricerche DNS interne ed esterne

Lavora con i gruppi di nodi

Creazione di gruppi di nodi

È possibile aggiungere uno o più gruppi di nodi oltre al gruppo di nodi iniziale creato insieme al cluster.

Per creare un gruppo di nodi aggiuntivo, usa:

```
eksctl create nodegroup --cluster=<clusterName> [--name=<nodegroupName>]
```

> **Note**
>
> `--version`il flag non è supportato per i gruppi di nodi gestiti. Eredita sempre la versione dal piano di controllo.

Per impostazione predefinita, i nuovi gruppi di nodi non gestiti ereditano la versione dal piano di controllo (`--version=auto`), ma è possibile specificare una versione diversa, da utilizzare anche `--version=latest` per forzare l'uso della versione più recente.

Inoltre, è possibile utilizzare lo stesso file di configurazione utilizzato per: `eksctl create cluster`

```
eksctl create nodegroup --config-file=<path>
```

Creare un gruppo di nodi da un file di configurazione

I gruppi di nodi possono essere creati anche tramite una definizione del cluster o un file di configurazione. Dati il seguente file di configurazione di esempio e un cluster esistente chiamato: `dev-cluster`

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

metadata:
  name: dev-cluster
  region: eu-north-1
```

```
managedNodeGroups:
  - name: ng-1-workers
    labels: { role: workers }
    instanceType: m5.xlarge
    desiredCapacity: 10
    volumeSize: 80
    privateNetworking: true
  - name: ng-2-builders
    labels: { role: builders }
    instanceType: m5.2xlarge
    desiredCapacity: 2
    volumeSize: 100
    privateNetworking: true
```

Il nodo si raggruppa `ng-1-workers` e `ng-2-builders` può essere creato con questo comando:

```
eksctl create nodegroup --config-file=dev-cluster.yaml
```

Bilanciamento del carico

Se vi siete già preparati a collegare i gruppi or/and target di Classic Load Balancer esistenti ai gruppi di nodi, potete specificarli nel file di configurazione. I gruppi or/and target dei sistemi di bilanciamento del carico classici vengono associati automaticamente all'ASG durante la creazione dei gruppi di nodi. Questo è supportato solo per i gruppi di nodi autogestiti definiti tramite il campo. `nodeGroups`

```
# dev-cluster-with-lb.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: dev-cluster
  region: eu-north-1

nodeGroups:
  - name: ng-1-web
    labels: { role: web }
    instanceType: m5.xlarge
    desiredCapacity: 10
    privateNetworking: true
    classicLoadBalancerNames:
      - dev-clb-1
      - dev-clb-2
```

```
    asgMetricsCollection:
      - granularity: 1Minute
        metrics:
          - GroupMinSize
          - GroupMaxSize
          - GroupDesiredCapacity
          - GroupInServiceInstances
          - GroupPendingInstances
          - GroupStandbyInstances
          - GroupTerminatingInstances
          - GroupTotalInstances
  - name: ng-2-api
    labels: { role: api }
    instanceType: m5.2xlarge
    desiredCapacity: 2
    privateNetworking: true
    targetGroupARNs:
      - arn:aws:elasticloadbalancing:eu-north-1:01234567890:targetgroup/dev-target-
group-1/abcdef0123456789
```

Selezione del gruppo di nodi nei file di configurazione

Per eseguire un'`delete`operazione `create` or solo su un sottoinsieme dei gruppi di nodi specificati in un file di configurazione, ci sono due flag CLI che accettano un elenco di globs e, ad esempio: `0` 1

```
eksctl create nodegroup --config-file=<path> --include='ng-prod-*-??' --exclude='ng-
test-1-ml-a,ng-test-2-?'
```

Utilizzando il file di configurazione di esempio sopra riportato, è possibile creare tutti i nodegroup di lavoratori tranne quelli di lavoro con il seguente comando:

```
eksctl create nodegroup --config-file=dev-cluster.yaml --exclude=ng-1-workers
```

Oppure si potrebbe eliminare il nodegroup dei costruttori con:

```
eksctl delete nodegroup --config-file=dev-cluster.yaml --include=ng-2-builders --
approve
```

In questo caso, dobbiamo anche fornire il `--approve` comando per eliminare effettivamente il nodegroup.

Regole di inclusione ed esclusione

- se no `--include` o `--exclude` è specificato tutto è incluso
- se `--include` viene specificato only, verranno inclusi solo i gruppi di nodi che corrispondono a quei globi
- se `--exclude` viene specificato only, vengono inclusi tutti i gruppi di nodi che non corrispondono a quei globi
- se vengono specificati entrambi, `--exclude` le regole hanno la precedenza su `--include` (ad esempio, i gruppi di nodi che corrispondono alle regole di entrambi i gruppi verranno esclusi)

Elenco dei gruppi di nodi

Per elencare i dettagli su un gruppo di nodi o su tutti i gruppi di nodi, usa:

```
eksctl get nodegroup --cluster=<clusterName> [--name=<nodegroupName>]
```

Per elencare uno o più gruppi di nodi in formato YAML o JSON, che restituisce più informazioni rispetto alla tabella di log predefinita, usa:

```
# YAML format
eksctl get nodegroup --cluster=<clusterName> [--name=<nodegroupName>] --output=yaml

# JSON format
eksctl get nodegroup --cluster=<clusterName> [--name=<nodegroupName>] --output=json
```

Immutabilità dei gruppi di nodi

In base alla progettazione, i gruppi di nodi sono immutabili. Ciò significa che se è necessario modificare qualcosa (diverso dal ridimensionamento) come l'AMI o il tipo di istanza di un gruppo di nodi, è necessario creare un nuovo gruppo di nodi con le modifiche desiderate, spostare il carico ed eliminare quello vecchio. Vedi la sezione Eliminazione e drenaggio dei gruppi di nodi.

Ridimensionamento dei gruppi di nodi

Il ridimensionamento dei gruppi di nodi è un processo che può richiedere fino a qualche minuto. Quando il `--wait` flag non è specificato, si aspetta `eksctl` ottimisticamente che il nodegroup venga

ridimensionato e ritorna non appena la richiesta API AWS è stata inviata. Per `eksctl` attendere che i nodi siano disponibili, aggiungi un `--wait` flag come nell'esempio seguente.

> **Note**
>
> Il ridimensionamento di un gruppo di nodi down/in (ovvero la riduzione del numero di nodi) può causare errori poiché ci basiamo esclusivamente sulle modifiche all'ASG. Ciò significa che i nodi in questione removed/terminated non vengono svuotati in modo esplicito. Questa potrebbe essere un'area di miglioramento in futuro.

La scalabilità di un gruppo di nodi gestito si ottiene chiamando direttamente l'API EKS che aggiorna la configurazione di un gruppo di nodi gestito.

Ridimensionamento di un singolo gruppo di nodi

Un gruppo di nodi può essere ridimensionato utilizzando il comando: `eksctl scale nodegroup`

```
eksctl scale nodegroup --cluster=<clusterName> --nodes=<desiredCount> --
name=<nodegroupName> [ --nodes-min=<minSize> ] [ --nodes-max=<maxSize> ] --wait
```

Ad esempio, per ridimensionare il nodegroup fino a 5 nodi, `ng-a345f4e1` `cluster-1` esegui:

```
eksctl scale nodegroup --cluster=cluster-1 --nodes=5 ng-a345f4e1
```

Un gruppo di nodi può anche essere ridimensionato utilizzando un file di configurazione passato a `--config-file` e specificando il nome del gruppo di nodi con cui deve essere ridimensionato. `--name` Eksctl cercherà nel file di configurazione e scoprirà quel nodegroup e i suoi valori di configurazione di ridimensionamento.

Se il numero di nodi desiderato `NOT` rientra nell'intervallo del numero minimo corrente e del numero massimo attuale, verrà visualizzato un errore specifico. Questi valori possono essere passati anche con flag `--nodes-min` e `--nodes-max` rispettivamente.

Ridimensionamento di più gruppi di nodi

Eksctl può scoprire e scalare tutti i gruppi di nodi presenti in un file di configurazione trasmesso con. `--config-file`

Analogamente al ridimensionamento di un singolo gruppo di nodi, lo stesso set di convalide si applica a ciascun gruppo di nodi. Ad esempio, il numero di nodi desiderato deve rientrare nell'intervallo del numero minimo e massimo di nodi.

Eliminazione e svuotamento dei gruppi di nodi

Per eliminare un gruppo di nodi, esegui:

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

Le regole di inclusione ed esclusione possono essere utilizzate anche con questo comando.

> **Note**
>
> Questo prosciugherà tutti i pod da quel gruppo di nodi prima che le istanze vengano eliminate.

Per saltare le regole di sfratto durante il processo di drenaggio, esegui:

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

Tutti i nodi sono isolati e tutti i pod vengono rimossi da un gruppo di nodi al momento dell'eliminazione, ma se hai bisogno di drenare un gruppo di nodi senza eliminarlo, esegui:

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

Per unire un gruppo di nodi, esegui:

```
eksctl drain nodegroup --cluster=<clusterName> --name=<nodegroupName> --undo
```

Per ignorare le regole di sfratto, come le impostazioni, esegui: PodDisruptionBudget

```
eksctl drain nodegroup --cluster=<clusterName> --name=<nodegroupName> --disable-
eviction
```

Per velocizzare il processo di drenaggio è possibile specificare `--parallel <value>` il numero di nodi da drenare in parallelo.

Altre funzionalità

Puoi anche abilitare SSH, accesso ASG e altre funzionalità per un gruppo di nodi, ad esempio:

```
eksctl create nodegroup --cluster=cluster-1 --node-
labels="autoscaling=enabled,purpose=ci-worker" --asg-access --full-ecr-access --ssh-
access
```

Aggiorna le etichette

Non ci sono comandi specifici `eksctl` per aggiornare le etichette di un gruppo di nodi, ma può essere facilmente ottenuto utilizzando`kubectl`, ad esempio:

```
kubectl label nodes -l alpha.eksctl.io/nodegroup-name=ng-1 new-label=foo
```

Accesso SSH

È possibile abilitare l'accesso SSH per i gruppi di nodi configurando uno dei gruppi di nodi `publicKeyName` e `publicKeyPath` nella configurazione del `publicKey` gruppo di nodi. In alternativa puoi usare AWS Systems Manager (SSM) per SSH sui nodi, configurando il nodegroup con: `enableSsm`

```
managedNodeGroups:
  - name: ng-1
    instanceType: m5.large
    desiredCapacity: 1
    ssh: # import public key from file
      publicKeyPath: ~/.ssh/id_rsa_tests.pub
  - name: ng-2
    instanceType: m5.large
    desiredCapacity: 1
    ssh: # use existing EC2 key
      publicKeyName: ec2_dev_key
  - name: ng-3
    instanceType: m5.large
    desiredCapacity: 1
    ssh: # import inline public key
      publicKey: "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDqZEdzvHnK/GVP8nLngRHu/
GDi/3PeES7+Bx6l3koXn/Oi/UmM9/jcW5XGziZ/
oe1cPJ777eZV7muEvXg5ZMQBrYxUtYCdvd8Rt6DIoSqDLsIPqbuuNlQoBHq/PU2IjpWnp/
wrJQXMk94IIrGjY8QHfCnpuMENCucVaifgAhwyeyuO5KiqUmD8E0RmcsotHKBV9X8H5eqLXd8zMQaPl
```

```
+Ub7j5PG+9KftQu0F/QhdFvpSLsHaxvBzA5nhIltjkaFcwGQnD1rpCM3+UnQE7Izoa5Yt1xoUWRwnF
+L2TKovW7+bYQ1kxsuuiX149jXTCJDVjkYCqi7HkrXYqcC1sbsror someuser@hostname"
  - name: ng-4
    instanceType: m5.large
    desiredCapacity: 1
    ssh: # enable SSH using SSM
      enableSsm: true
```

Gruppi di nodi non gestiti

In`eksctl`, l'impostazione `--managed=false` o l'utilizzo del `nodeGroups` campo crea un gruppo di nodi non gestito. Tieni presente che i gruppi di nodi non gestiti non vengono visualizzati nella console EKS, che come regola generale conosce solo i gruppi di nodi gestiti da EKS.

Dovresti aggiornare i gruppi di nodi solo dopo l'esecuzione. `eksctl upgrade cluster` (Vedi Aggiornamento dei cluster).

Se hai un cluster semplice con solo un gruppo di nodi iniziale (cioè creato con`eksctl create cluster`), il processo è molto semplice:

1. Ottieni il nome del vecchio nodegroup:

```
eksctl get nodegroups --cluster=<clusterName> --region=<region>
```

> **Note**
>
> ```
> You should see only one nodegroup here, if you see more - read the next
>  section.
> ```

2. Crea un nuovo gruppo di nodi:

```
 eksctl create nodegroup --cluster=<clusterName> --region=<region> --
name=<newNodeGroupName> --managed=false
```

3. Elimina il vecchio nodegroup:

```
 eksctl delete nodegroup --cluster=<clusterName> --region=<region> --
name=<oldNodeGroupName>
```

> **Note**
>
> ```
> This will drain all pods from that nodegroup before the instances are deleted.
>  In some scenarios, Pod Disruption Budget (PDB) policies can prevent pods to
>  be evicted. To delete the nodegroup regardless of PDB, one should use the `--
> disable-eviction` flag, will bypass checking PDB policies.
> ```

Aggiornamento di più gruppi di nodi

Se disponi di più gruppi di nodi, è tua responsabilità tenere traccia della configurazione di ciascuno di essi. Puoi farlo utilizzando i file di configurazione, ma se non li hai già utilizzati, dovrai ispezionare il cluster per scoprire come è stato configurato ogni gruppo di nodi.

In termini generali, stai cercando di:

- rivedi quali gruppi di nodi hai e quali possono essere eliminati o devono essere sostituiti per la nuova versione
- annota la configurazione di ogni gruppo di nodi, valuta la possibilità di utilizzare il file di configurazione per facilitare gli aggiornamenti la prossima volta

Aggiornamento con file di configurazione

Se si utilizza il file di configurazione, è necessario effettuare le seguenti operazioni.

Modifica il file di configurazione per aggiungere nuovi gruppi di nodi e rimuovere i vecchi gruppi di nodi. Se vuoi solo aggiornare i gruppi di nodi e mantenere la stessa configurazione, puoi semplicemente cambiare i nomi dei gruppi di nodi, ad esempio aggiungendoli al nome. `-v2`

Per creare tutti i nuovi gruppi di nodi definiti nel file di configurazione, esegui:

```
eksctl create nodegroup --config-file=<path>
```

Una volta che hai creato nuovi gruppi di nodi, puoi eliminare quelli vecchi:

```
eksctl delete nodegroup --config-file=<path> --only-missing
```

> **Note**
>
> La prima esecuzione è in modalità piano, se sei soddisfatto delle modifiche proposte, esegui nuovamente con. `--approve`

Aggiornamento dei componenti aggiuntivi predefiniti

Potrebbe essere necessario aggiornare i componenti aggiuntivi di rete installati nel cluster. Per ulteriori informazioni, consulta the section called “Aggiornamenti aggiuntivi predefiniti”.

Gruppi di nodi gestiti da EKS

I gruppi di nodi gestiti di Amazon EKS sono una funzionalità che automatizza il provisioning e la gestione del ciclo di vita dei nodi (istanze EC2) per i cluster Amazon EKS Kubernetes. I clienti possono fornire gruppi di nodi ottimizzati per i propri cluster ed EKS manterrà i nodi aggiornati con le ultime versioni di Kubernetes e del sistema operativo host.

Un gruppo di nodi gestiti EKS è un gruppo con scalabilità automatica e istanze EC2 associate gestite da AWS per un cluster Amazon EKS. Ogni gruppo di nodi utilizza l'AMI Amazon Linux 2 ottimizzata per Amazon EKS. Amazon EKS semplifica l'applicazione di correzioni di bug e patch di sicurezza ai nodi, nonché l'aggiornamento alle versioni più recenti di Kubernetes. Ogni gruppo di nodi avvia un gruppo di scalabilità automatica per il cluster, che può estendersi su più zone di disponibilità e sottoreti AWS VPC per un'elevata disponibilità.

NOVITÀ Supporto di Launch Template per gruppi di nodi gestiti

> **Note**
>
> Il termine «gruppi di nodi non gestiti» è stato usato per riferirsi ai gruppi di nodi che eksctl ha supportato sin dall'inizio (rappresentati tramite il campo). `nodeGroups` Il `ClusterConfig` file continua a utilizzare il `nodeGroups` campo per definire i gruppi di nodi non gestiti e i gruppi di nodi gestiti vengono definiti con il campo. `managedNodeGroups`

Creazione di gruppi di nodi gestiti

```
$ eksctl create nodegroup
```

Nuovi cluster

Per creare un nuovo cluster con un gruppo di nodi gestito, esegui

```
eksctl create cluster
```

Per creare più gruppi di nodi gestiti e avere un maggiore controllo sulla configurazione, è possibile utilizzare un file di configurazione.

> **Note**
>
> I gruppi di nodi gestiti non hanno una parità di funzionalità completa con i gruppi di nodi non gestiti.

```
# cluster.yaml
# A cluster with two managed nodegroups
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: managed-cluster
  region: us-west-2

managedNodeGroups:
  - name: managed-ng-1
    minSize: 2
    maxSize: 4
    desiredCapacity: 3
    volumeSize: 20
    ssh:
      allow: true
      publicKeyPath: ~/.ssh/ec2_id_rsa.pub
      # new feature for restricting SSH access to certain AWS security group IDs
      sourceSecurityGroupIds: ["sg-00241fbb12c607007"]
    labels: {role: worker}
    tags:
      nodegroup-role: worker
    iam:
      withAddonPolicies:
        externalDNS: true
```

```
        certManager: true

  - name: managed-ng-2
    instanceType: t2.large
    minSize: 2
    maxSize: 3
```

Un altro esempio di file di configurazione per la creazione di un gruppo di nodi gestito è disponibile qui.

È possibile avere un cluster con gruppi di nodi gestiti e non gestiti. I gruppi di nodi non gestiti non vengono visualizzati nella console AWS EKS ma `eksctl get nodegroup` elencheranno entrambi i tipi di gruppi di nodi.

```
# cluster.yaml
# A cluster with an unmanaged nodegroup and two managed nodegroups.
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: managed-cluster
  region: us-west-2

nodeGroups:
  - name: ng-1
    minSize: 2

managedNodeGroups:
  - name: managed-ng-1
    minSize: 2
    maxSize: 4
    desiredCapacity: 3
    volumeSize: 20
    ssh:
      allow: true
      publicKeyPath: ~/.ssh/ec2_id_rsa.pub
      # new feature for restricting SSH access to certain AWS security group IDs
      sourceSecurityGroupIds: ["sg-00241fbb12c607007"]
    labels: {role: worker}
    tags:
      nodegroup-role: worker
    iam:
```

```
      withAddonPolicies:
        externalDNS: true
        certManager: true

  - name: managed-ng-2
    instanceType: t2.large
    privateNetworking: true
    minSize: 2
    maxSize: 3
```

NUOVO Supporto per AMI personalizzate, gruppi di sicurezza `instancePrefix`,`instanceName`,`ebsOptimized`,`volumeType`,`volumeName`,`volumeEncrypted`,`volum`
e `disableIMDSv1`

```
# cluster.yaml
# A cluster with a managed nodegroup with customization.
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: managed-cluster
  region: us-west-2

managedNodeGroups:
  - name: custom-ng
    ami: ami-0e124de4755b2734d
    securityGroups:
      attachIDs: ["sg-1234"]
    maxPodsPerNode: 80
    ssh:
      allow: true
    volumeSize: 100
    volumeName: /dev/xvda
    volumeEncrypted: true
    # defaults to true, which enforces the use of IMDSv2 tokens
    disableIMDSv1: false
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh managed-cluster --kubelet-extra-args '--node-
labels=eks.amazonaws.com/nodegroup=custom-ng,eks.amazonaws.com/nodegroup-
image=ami-0e124de4755b2734d'
```

Se stai richiedendo un tipo di istanza disponibile solo in una zona (e la configurazione eksctl richiede la specificazione di due) assicurati di aggiungere la zona di disponibilità alla richiesta del tuo gruppo di nodi:

```
# cluster.yaml
# A cluster with a managed nodegroup with "availabilityZones"
---

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: flux-cluster
  region: us-east-2
  version: "1.23"

availabilityZones: ["us-east-2b", "us-east-2c"]
managedNodeGroups:
  - name: workers
    instanceType: hpc6a.48xlarge
    minSize: 64
    maxSize: 64
    labels: { "fluxoperator": "true" }
    availabilityZones: ["us-east-2b"]
    efaEnabled: true
    placement:
      groupName: eks-efa-testing
```

Questo può essere vero, ad esempio, per tipi come la famiglia Hpc6 che sono disponibili solo in una zona.

Cluster esistenti

```
eksctl create nodegroup --managed
```

Suggerimento: se utilizzi un `ClusterConfig` file per descrivere l'intero cluster, descrivi il nuovo gruppo di nodi gestiti nel `managedNodeGroups` campo ed esegui:

```
eksctl create nodegroup --config-file=YOUR_CLUSTER.yaml
```

Aggiornamento dei gruppi di nodi gestiti

Puoi aggiornare un nodegroup all'ultima versione di release AMI ottimizzata per EKS per il tipo di AMI che stai utilizzando in qualsiasi momento.

Se il tuo nodegroup è la stessa versione di Kubernetes del cluster, puoi eseguire l'aggiornamento all'ultima versione dell'AMI per quella versione di Kubernetes del tipo di AMI che stai utilizzando. Se il tuo nodegroup è la versione di Kubernetes precedente alla versione Kubernetes del cluster, puoi aggiornare il nodegroup all'ultima versione dell'AMI che corrisponde alla versione Kubernetes del nodegroup o eseguire l'aggiornamento all'ultima versione dell'AMI che corrisponde alla versione Kubernetes del cluster. Non puoi ripristinare un nodegroup a una versione precedente di Kubernetes.

Per aggiornare un nodegroup gestito all'ultima versione dell'AMI release:

```
eksctl upgrade nodegroup --name=managed-ng-1 --cluster=managed-cluster
```

Il nodegroup può essere aggiornato all'ultima versione AMI per una versione di Kubernetes specificata utilizzando:

```
eksctl upgrade nodegroup --name=managed-ng-1 --cluster=managed-cluster --kubernetes-
version=<kubernetes-version>
```

Per eseguire l'aggiornamento a una versione di rilascio AMI specifica anziché alla versione più recente, passa`--release-version`:

```
eksctl upgrade nodegroup --name=managed-ng-1 --cluster=managed-cluster --release-
version=1.19.6-20210310
```

> **Note**
>
> Se i nodi gestiti vengono distribuiti utilizzando l'AMI personalizzata AMIs, è necessario seguire il seguente flusso di lavoro per distribuire una nuova versione dell'AMI personalizzata.

- la distribuzione iniziale del nodegroup deve essere eseguita utilizzando un modello di avvio. ad es.

```
managedNodeGroups:
  - name: launch-template-ng
```

```
    launchTemplate:
      id: lt-1234
      version: "2" #optional (uses the default version of the launch template if
 unspecified)
```

- crea una nuova versione dell'AMI personalizzata (utilizzando la console AWS EKS).
- crea una nuova versione del modello di lancio con il nuovo ID AMI (utilizzando la console AWS EKS).
- aggiorna i nodi alla nuova versione del modello di lancio. ad es.

```
eksctl upgrade nodegroup --name nodegroup-name --cluster cluster-name --launch-
template-version new-template-version
```

Gestione degli aggiornamenti paralleli per i nodi

È possibile aggiornare più nodi gestiti contemporaneamente. Per configurare gli aggiornamenti paralleli, definisci il nodegroup durante la creazione `updateConfig` del nodegroup. Un esempio `updateConfig` può essere trovato qui.

Per evitare interruzioni dei carichi di lavoro dovute all'aggiornamento di più nodi contemporaneamente, puoi limitare il numero di nodi che possono diventare non disponibili durante un aggiornamento specificandolo nel campo di un. `maxUnavailable` `updateConfig` In alternativa, usa`maxUnavailablePercentage`, che definisce il numero massimo di nodi non disponibili come percentuale del numero totale di nodi.

Nota che `maxUnavailable` non può essere superiore a`maxSize`. Inoltre, `maxUnavailable` `maxUnavailablePercentage` non può essere utilizzato contemporaneamente.

Questa funzionalità è disponibile solo per i nodi gestiti.

Aggiornamento dei gruppi di nodi gestiti

`eksctl`consente di aggiornare la UpdateConfigsezione di un gruppo di nodi gestito. Questa sezione definisce due campi. `MaxUnavailable`e`MaxUnavailablePercentage`. I tuoi gruppi di nodi non vengono modificati durante l'aggiornamento, quindi non dovrebbero essere previsti tempi di inattività.

Il comando `update nodegroup` deve essere usato con un file di configurazione usando il flag. `--config-file` Il nodegroup deve contenere una sezione. `nodeGroup.updateConfig` Ulteriori informazioni possono essere trovate qui.

Problemi di Nodegroup Health

EKS Managed Nodegroups controlla automaticamente la configurazione del gruppo di nodi e dei nodi per rilevare eventuali problemi di salute e li segnala tramite l'API e la console EKS. Per visualizzare i problemi di salute di un nodegroup:

```
eksctl utils nodegroup-health --name=managed-ng-1 --cluster=managed-cluster
```

Gestione delle etichette

EKS Managed Nodegroups supporta l'aggiunta di etichette applicate ai nodi Kubernetes nel nodegroup. Questo viene specificato tramite il `labels` campo in eksctl durante la creazione di cluster o gruppi di nodi.

Per impostare nuove etichette o aggiornare le etichette esistenti su un gruppo di nodi:

```
eksctl set labels --cluster managed-cluster --nodegroup managed-ng-1 --labels
 kubernetes.io/managed-by=eks,kubernetes.io/role=worker
```

Per annullare l'impostazione o rimuovere le etichette da un gruppo di nodi:

```
eksctl unset labels --cluster managed-cluster --nodegroup managed-ng-1 --labels
 kubernetes.io/managed-by,kubernetes.io/role
```

Per visualizzare tutte le etichette impostate su un gruppo di nodi:

```
eksctl get labels --cluster managed-cluster --nodegroup managed-ng-1
```

Ridimensionamento dei gruppi di nodi gestiti

`eksctl scale nodegroup`supporta anche i gruppi di nodi gestiti. La sintassi per scalare un gruppo di nodi gestito o non gestito è la stessa.

```
eksctl scale nodegroup --name=managed-ng-1 --cluster=managed-cluster --nodes=4 --nodes-
min=3 --nodes-max=5
```

Ulteriori informazioni

- Gruppi di nodi gestiti da EKS

Bootstrap dei nodi

AmazonLinux2023

AL2023 ha introdotto un nuovo processo di inizializzazione dei nodi nodeadm che utilizza uno schema di configurazione YAML, eliminando l'uso dello script. `/etc/eks/bootstrap.sh`

> **Note**
>
> Con le versioni 1.30 e successive di Kubernetes, Amazon Linux 2023 è il sistema operativo predefinito.

Impostazioni predefinite per AL2

Per i nodi autogestiti e i nodi gestiti da EKS in base alla personalizzazione AMIs, `eksctl` crea un valore predefinito, minimo `NodeConfig` e lo inserisce automaticamente nei dati utente del modello di avvio dei nodegroup. ad es.

```
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: https://XXXX.us-west-2.eks.amazonaws.com
    certificateAuthority: XXXX
    cidr: 10.100.0.0/16
    name: my-cluster
  kubelet:
    config:
      clusterDNS:
      - 10.100.0.10
    flags:
    - --node-labels=alpha.eksctl.io/cluster-name=my-cluster,alpha.eksctl.io/nodegroup-
name=my-nodegroup
    - --register-with-taints=special=true:NoSchedule
```

```
--//--
```

Per i nodi gestiti da EKS basati su nativi AMIs, l'impostazione predefinita `NodeConfig` viene aggiunta da EKS MNG sotto il cofano, aggiunta direttamente ai dati utente di EC2. Pertanto, in questo scenario, `eksctl` non è necessario includerlo nel modello di lancio.

Configurazione del processo di bootstrap

Per impostare proprietà avanzate o semplicemente sovrascrivere i valori predefiniti, eksctl consente di specificare un valore personalizzato tramite o ad es. `NodeConfig` `NodeConfig` `nodeGroup.overrideBootstrapCommand` `managedNodeGroup.overrideBootstrapCommand`

```
managedNodeGroups:
  - name: mng-1
    amiFamily: AmazonLinux2023
    ami: ami-0253856dd7ab7dbc8
    overrideBootstrapCommand: |
      apiVersion: node.eks.aws/v1alpha1
      kind: NodeConfig
      spec:
        instance:
          localStorage:
            strategy: RAID0
```

Questa configurazione personalizzata verrà aggiunta agli userdata da eksctl e unita alla configurazione predefinita. `nodeadm` Scopri di più sulla capacità di unire più oggetti `nodeadm` di configurazione qui.

Avvia il supporto dei modelli per i gruppi di nodi gestiti

eksctl supporta l'avvio di gruppi di nodi gestiti utilizzando un modello di avvio EC2 fornito. Ciò consente diverse opzioni di personalizzazione per i gruppi di nodi, tra cui la fornitura di gruppi personalizzati AMIs e di sicurezza e il trasferimento dei dati utente per il bootstrap dei nodi.

Creazione di gruppi di nodi gestiti utilizzando un modello di avvio fornito

```
# managed-cluster.yaml
```

```
# A cluster with two managed nodegroups
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: managed-cluster
  region: us-west-2

managedNodeGroups:
  - name: managed-ng-1
    launchTemplate:
      id: lt-12345
      version: "2" # optional (uses the default launch template version if unspecified)

  - name: managed-ng-2
    minSize: 2
    desiredCapacity: 2
    maxSize: 4
    labels:
      role: worker
    tags:
      nodegroup-name: managed-ng-2
    privateNetworking: true
    launchTemplate:
      id: lt-12345
```

Aggiornamento di un gruppo di nodi gestito per utilizzare una versione diversa del modello di lancio

```
eksctl upgrade nodegroup --name=managed-ng-1 --cluster=managed-cluster --launch-
template-version=3
```

> **Note**
>
> Se un modello di avvio utilizza un'AMI personalizzata, anche la nuova versione deve utilizzare un'AMI personalizzata, altrimenti l'operazione di aggiornamento avrà esito negativo

Se un modello di avvio non utilizza un'AMI personalizzata, è possibile specificare anche la versione di Kubernetes a cui eseguire l'aggiornamento:

```
eksctl upgrade nodegroup --name=managed-ng-1 --cluster=managed-cluster --launch-
template-version=3 --kubernetes-version=1.17
```

Note sull'AMI personalizzata e sul supporto dei modelli di avvio

- Quando viene fornito un modello di lancio, i seguenti campi non sono supportati: `instanceType` `amissh.allow`,`ssh.sourceSecurityGroupIds`,`securityGroups`,`instancePrefix`,`instanceNam` `overrideBootstrapCommand` `edisableIMDSv1`.
- Quando si utilizza un AMI (`ami`) personalizzato, `overrideBootstrapCommand` deve essere impostato anche per eseguire il bootstrap.
- `overrideBootstrapCommand`può essere impostato solo quando si utilizza un'AMI personalizzata.
- Quando viene fornito un modello di avvio, i tag specificati nella configurazione del nodegroup si applicano solo alla risorsa EKS Nodegroup e non vengono propagati alle istanze EC2.

Sottoreti personalizzate

È possibile estendere un VPC esistente con una nuova sottorete e aggiungere un gruppo di nodi a quella sottorete.

Perché

Se il cluster dovesse esaurire i dati preconfigurati IPs, è possibile ridimensionare il VPC esistente con un nuovo CIDR per aggiungervi una nuova sottorete. Per scoprire come farlo, leggi questa guida su AWS Extending VPCs.

A; DR

Vai alla configurazione del VPC e aggiungi, fai clic su Azioni-> Modifica CIDRs e aggiungi un nuovo intervallo. Esempio:

```
192.168.0.0/19 -> existing CIDR
+ 192.169.0.0/19 -> new CIDR
```

Ora devi aggiungere una nuova sottorete. A seconda che si tratti di una nuova sottorete privata o pubblica, sarà necessario copiare le informazioni di routing rispettivamente da una sottorete privata o pubblica.

Una volta creata la sottorete, aggiungete il routing e copiate l'ID del gateway NAT o l'Internet Gateway da un'altra sottorete nel VPC. Fai attenzione che, se si tratta di una sottorete pubblica, abilita l'assegnazione automatica degli IP. Azioni->Modifica le impostazioni IP di assegnazione automatica->Abilita l'assegnazione automatica dell'indirizzo pubblico. IPv4

Non dimenticare di copiare anche i TAG delle sottoreti esistenti a seconda della configurazione della sottorete pubblica o privata. Questo è importante, altrimenti la sottorete non farà parte del cluster e le istanze nella sottorete non potranno unirsi.

Al termine, copia l'ID della nuova sottorete. Ripetere l'operazione tutte le volte che è necessario.

In che modo

Per creare un gruppo di nodi nelle sottoreti create, esegui il seguente comando:

```
eksctl create nodegroup --cluster <cluster-name> --name my-new-subnet --subnet-ids
 subnet-0edeb3a04bec27141,subnet-0edeb3a04bec27142,subnet-0edeb3a04bec27143
# or for a single subnet id
eksctl create nodegroup --cluster <cluster-name> --name my-new-subnet --subnet-ids
 subnet-0edeb3a04bec27141
```

Oppure, usa la configurazione come tale:

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

Con una configurazione come questa:

```
# A simple example of ClusterConfig object with two nodegroups:
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: cluster-3
  region: eu-north-1

nodeGroups:
  - name: new-subnet-nodegroup
    instanceType: m5.large
    desiredCapacity: 1
    subnets:
```

```
      - subnet-id1
      - subnet-id2
```

Attendi la creazione del gruppo di nodi e le nuove istanze dovrebbero avere i nuovi intervalli IP delle sottoreti.

Eliminazione del cluster

Poiché la nuova aggiunta ha modificato il VPC esistente aggiungendo una dipendenza all'esterno dello CloudFormation stack, non è più CloudFormation possibile rimuovere il cluster.

Prima di eliminare il cluster, rimuovi manualmente tutte le sottoreti aggiuntive create, quindi procedi chiamando: `eksctl`

```
eksctl delete cluster -n <cluster-name> --wait
```

DNS personalizzato

Esistono due modi per sovrascrivere l'indirizzo IP del server DNS utilizzato per tutte le ricerche DNS interne ed esterne. Questo è l'equivalente del flag per. `--cluster-dns` `kubelet`

Il primo, è attraverso il `clusterDNS` campo. I file di configurazione accettano un `string` campo chiamato `clusterDNS` con l'indirizzo IP del server DNS da utilizzare. Questo verrà passato a chi a sua volta lo passerà ai pod tramite il file. `kubelet` `/etc/resolv.conf` Per ulteriori informazioni, consulta lo schema del file di configurazione.

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

metadata:
  name: cluster-1
  region: eu-north-1

nodeGroups:
- name: ng-1
  clusterDNS: 169.254.20.10
```

Nota che questa configurazione accetta solo un indirizzo IP. Per specificare più di un indirizzo, utilizzate il `kubeletExtraConfig`parametro:

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

metadata:
  name: cluster-1
  region: eu-north-1

nodeGroups:
  - name: ng-1
    kubeletExtraConfig:
      clusterDNS: ["169.254.20.10","172.20.0.10"]
```

Tinte

Per applicare le tinte a uno specifico gruppo di nodi, usa la sezione di `taints` configurazione in questo modo:

```
    taints:
      - key: your.domain.com/db
        value: "true"
        effect: NoSchedule
      - key: your.domain.com/production
        value: "true"
        effect: NoExecute
```

Un esempio completo può essere trovato qui.

Selettore di istanze

eksctl supporta la specificazione di più tipi di istanze per gruppi di nodi gestiti e autogestiti, ma con oltre 270 tipi di istanze EC2, gli utenti devono dedicare del tempo a capire quali tipi di istanza sarebbero più adatti al loro gruppo di nodi. È ancora più difficile quando si utilizzano istanze Spot, perché è necessario scegliere un set di istanze che funzioni bene insieme a Cluster Autoscaler.

eksctl ora si integra con il selettore di istanze EC2, che risolve questo problema generando un elenco di tipi di istanze basato su criteri di risorse: v, memory, # of e architettura CPU. CPUs GPUs Quando vengono passati i criteri del selettore di istanza, eksctl crea un nodegroup con i tipi di istanza impostati sui tipi di istanza che corrispondono ai criteri forniti.

Crea cluster e gruppi di nodi

Per creare un cluster con un singolo gruppo di nodi che utilizzi tipi di istanza corrispondenti ai criteri di risorsa del selettore di istanze passati a eksctl, esegui

```
eksctl create cluster --instance-selector-vcpus=2 --instance-selector-memory=4
```

Questo creerà un cluster e un gruppo di nodi gestito con il `instanceTypes` campo impostato su `[c5.large, c5a.large, c5ad.large, c5d.large, t2.medium, t3.medium, t3a.medium]` (l'insieme di tipi di istanza restituiti potrebbe cambiare).

Per i gruppi di nodi non gestiti, il `instancesDistribution.instanceTypes` campo verrà impostato:

```
eksctl create cluster --managed=false --instance-selector-vcpus=2 --instance-selector-
memory=4
```

I criteri del selettore di istanza possono essere specificati anche in: ClusterConfig

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

metadata:
  name: cluster
  region: us-west-2

nodeGroups:
- name: ng
  instanceSelector:
    vCPUs: 2
    memory: "4" # 4 GiB, unit defaults to GiB

managedNodeGroups:
- name: mng
  instanceSelector:
    vCPUs: 2
    memory: 2GiB #
    cpuArchitecture: x86_64 # default value
```

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

Le seguenti opzioni CLI del selettore di istanze sono supportate da `eksctl create cluster` e: `eksctl create nodegroup`

`--instance-selector-vcpus`, e `--instance-selector-memory` `--instance-selector-gpus` `instance-selector-cpu-architecture`

Un file di esempio può essere trovato qui.

Dry Run

La funzione dry-run consente di ispezionare e modificare le istanze corrispondenti al selettore di istanze prima di procedere alla creazione di un gruppo di nodi.

```
eksctl create cluster --name development --instance-selector-vcpus=2 --instance-
selector-memory=4 --dry-run

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
# ...
managedNodeGroups:
- amiFamily: AmazonLinux2
  instanceSelector:
    memory: "4"
    vCPUs: 2
  instanceTypes:
  - c5.large
  - c5a.large
  - c5ad.large
  - c5d.large
  - t2.medium
  - t3.medium
  - t3a.medium
...
# other config
```

ClusterConfig Il generato può `eksctl create cluster` quindi essere passato a:

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

Il `instanceSelector` campo che rappresenta le opzioni CLI verrà inoltre aggiunto al ClusterConfig file per scopi di visibilità e documentazione. Quando `--dry-run` viene omesso, questo campo verrà ignorato e il `instanceTypes` campo verrà utilizzato, altrimenti qualsiasi modifica verrà sovrascritta da `instanceTypes` eksctl.

Quando viene passato un ClusterConfig file`--dry-run`, eksctl produrrà un ClusterConfig file contenente lo stesso set di gruppi di nodi dopo aver ampliato i criteri di risorsa del selettore di istanze di ciascun gruppo di nodi.

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

metadata:
  name: cluster
  region: us-west-2

nodeGroups:
- name: ng
  instanceSelector:
    vCPUs: 2
    memory: 4 # 4 GiB, unit defaults to GiB

managedNodeGroups:
- name: mng
  instanceSelector:
    vCPUs: 2
    memory: 2GiB #
    cpuArchitecture: x86_64 # default value
```

```
eksctl create cluster -f instance-selector-cluster.yaml --dry-run

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
# ...
managedNodeGroups:
- amiFamily: AmazonLinux2
  # ...
  instanceSelector:
    cpuArchitecture: x86_64
    memory: 2GiB
```

```
    vCPUs: 2
  instanceTypes:
  - t3.small
  - t3a.small
nodeGroups:
- amiFamily: AmazonLinux2
  # ...
  instanceSelector:
    memory: "4"
    vCPUs: 2
  instanceType: mixed
  instancesDistribution:
    capacityRebalance: false
    instanceTypes:
    - c5.large
    - c5a.large
    - c5ad.large
    - c5d.large
    - t2.medium
    - t3.medium
    - t3a.medium
# ...
```

Istanze Spot

Gruppi di nodi gestiti

`eksctl`supporta i nodi di lavoro Spot utilizzando EKS Managed Nodegroups, una funzionalità che consente ai clienti EKS con applicazioni con tolleranza ai guasti di fornire e gestire facilmente le istanze Spot EC2 per i propri cluster EKS. EKS Managed Nodegroup configurerà e lancerà un gruppo EC2 Autoscaling di istanze Spot seguendo le best practice Spot e svuotando automaticamente i nodi di lavoro Spot prima che le istanze vengano interrotte da AWS. Non è previsto alcun costo incrementale per l'utilizzo di questa funzionalità e i clienti pagano solo per l'utilizzo delle risorse AWS, come le istanze Spot EC2 e i volumi EBS.

Per creare un cluster con un gruppo di nodi gestito utilizzando istanze Spot, passa il `--spot` flag e un elenco opzionale di tipi di istanze:

```
eksctl create cluster --spot --instance-types=c3.large,c4.large,c5.large
```

Per creare un gruppo di nodi gestito utilizzando istanze Spot su un cluster esistente:

```
eksctl create nodegroup --cluster=<clusterName> --spot --instance-
types=c3.large,c4.large,c5.large
```

Per creare istanze Spot utilizzando gruppi di nodi gestiti tramite un file di configurazione:

```
# spot-cluster.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: spot-cluster
  region: us-west-2

managedNodeGroups:
- name: spot
  instanceTypes: ["c3.large","c4.large","c5.large","c5d.large","c5n.large","c5a.large"]
  spot: true


# `instanceTypes` defaults to [`m5.large`]
- name: spot-2
  spot: true

# On-Demand instances
- name: on-demand
  instanceTypes: ["c3.large", "c4.large", "c5.large"]
```

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

> **Note**
>
> I gruppi di nodi non gestiti non supportano i `instanceTypes` campi `spot` and, ma il `instancesDistribution` campo viene utilizzato per configurare le istanze Spot. Vedi sotto

Ulteriori informazioni

- Gruppi di nodi Spot EKS

- Tipi di capacità dei gruppi di nodi gestiti da EKS

Gruppi di nodi non gestiti

`eksctl`supporta le istanze spot tramite i gruppi MixedInstancesPolicy di Auto Scaling.

Ecco un esempio di nodegroup che utilizza il 50% di istanze spot e il 50% di istanze on demand:

```
nodeGroups:
  - name: ng-1
    minSize: 2
    maxSize: 5
    instancesDistribution:
      maxPrice: 0.017
      instanceTypes: ["t3.small", "t3.medium"] # At least one instance type should be
 specified
      onDemandBaseCapacity: 0
      onDemandPercentageAboveBaseCapacity: 50
      spotInstancePools: 2
```

Nota che il `nodeGroups.X.instanceType` campo non deve essere impostato quando si utilizza il campo. `instancesDistribution`

Questo esempio utilizza istanze GPU:

```
nodeGroups:
  - name: ng-gpu
    instanceType: mixed
    desiredCapacity: 1
    instancesDistribution:
      instanceTypes:
        - p2.xlarge
        - p2.8xlarge
        - p2.16xlarge
      maxPrice: 0.50
```

Questo esempio utilizza la strategia di allocazione spot ottimizzata per la capacità:

```
nodeGroups:
  - name: ng-capacity-optimized
```

```
    minSize: 2
    maxSize: 5
    instancesDistribution:
      maxPrice: 0.017
      instanceTypes: ["t3.small", "t3.medium"] # At least one instance type should be
 specified
      onDemandBaseCapacity: 0
      onDemandPercentageAboveBaseCapacity: 50
      spotAllocationStrategy: "capacity-optimized"
```

Questo esempio utilizza la strategia di allocazione capacity-optimized-prioritized spot:

```
nodeGroups:
  - name: ng-capacity-optimized-prioritized
    minSize: 2
    maxSize: 5
    instancesDistribution:
      maxPrice: 0.017
      instanceTypes: ["t3a.small", "t3.small"] # At least two instance types should be
 specified
      onDemandBaseCapacity: 0
      onDemandPercentageAboveBaseCapacity: 0
      spotAllocationStrategy: "capacity-optimized-prioritized"
```

Utilizza la strategia di `capacity-optimized-prioritized` allocazione e quindi imposta l'ordine dei tipi di istanza nell'elenco delle sostituzioni dei modelli di lancio dalla priorità più alta a quella più bassa (dalla prima all'ultima nell'elenco). Amazon EC2 Auto Scaling rispetta le priorità dei tipi di istanza con il massimo impegno, ma ottimizza innanzitutto la capacità. Questa è una buona opzione per i carichi di lavoro in cui è necessario ridurre al minimo la possibilità di interruzioni, ma è importante anche la preferenza per determinati tipi di istanze. Per ulteriori informazioni, consulta ASG Purchase Options.

Nota che il `spotInstancePools` campo non deve essere impostato quando si utilizza il campo. `spotAllocationStrategy` Se non `spotAllocationStrategy` è specificato, EC2 utilizzerà per impostazione predefinita la `lowest-price` strategia.

Ecco un esempio minimo:

```
nodeGroups:
  - name: ng-1
    instancesDistribution:
```

```
      instanceTypes: ["t3.small", "t3.medium"] # At least one instance type should be
 specified
```

Per distinguere i nodi tra istanze spot o on-demand, puoi usare l'etichetta Kubernetes `node-lifecycle` che avrà il valore `spot` o `on-demand` dipenderà dal tipo.

Parametri in InstancesDistribution

Consulta lo schema di configurazione del cluster per i dettagli.

Supporto GPU

Eksctl supporta la selezione dei tipi di istanze GPU per i gruppi di nodi. Basta fornire un tipo di istanza compatibile al comando create o tramite il file di configurazione.

```
eksctl create cluster --node-type=p2.xlarge
```

> **Note**
>
> Non è più necessario abbonarsi all'AMI del marketplace per il supporto delle GPU su EKS.

I resolver AMI (`auto`and`auto-ssm`) vedranno che desideri utilizzare un tipo di istanza GPU e selezioneranno l'AMI accelerata ottimizzata EKS corretta.

Eksctl rileverà che è stata selezionata un'AMI con un tipo di istanza abilitato alla GPU e installerà automaticamente il plug-in del dispositivo NVIDIA Kubernetes.

> **Note**
>
> Windows e Ubuntu AMIs non vengono forniti con i driver GPU installati, quindi l'esecuzione di carichi di lavoro accelerati da GPU non funzionerà immediatamente.

Per disabilitare l'installazione automatica del plugin e installare manualmente una versione specifica, usala con il comando create. `--install-nvidia-plugin=false` Esempio:

```
eksctl create cluster --node-type=p2.xlarge --install-nvidia-plugin=false
```

e, per le versioni 0.15.0 e successive,

```
kubectl create -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/<VERSION>/
deployments/static/nvidia-device-plugin.yml
```

oppure, per le versioni precedenti,

```
kubectl create -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/<VERSION>/
nvidia-device-plugin.yml
```

L'installazione del plug-in per dispositivi NVIDIA Kubernetes verrà ignorata se il cluster include solo gruppi di nodi Bottlerocket, poiché Bottlerocket gestisce già l'esecuzione del plug-in del dispositivo. Se utilizzi diverse famiglie di AMI nelle configurazioni del tuo cluster, potresti dover utilizzare contaminazioni e tolleranze per impedire che il plug-in del dispositivo venga eseguito sui nodi Bottlerocket.

Supporto ARM

Questo argomento spiega come creare un cluster con un gruppo di nodi ARM e come aggiungere un gruppo di nodi ARM a un cluster esistente.

EKS supporta l'architettura ARM a 64 bit con i suoi processori Graviton. Per creare un cluster, seleziona uno dei tipi di istanza basati su Graviton (`a1`,,`t4g`,`m6g`,`m7g`,,`m6gd`,`c6g`,`c7g`,`c6gd`, `r6g` `r7g` `r6gd` `m8gr8g`,`c8g`) ed esegui:

```
eksctl create cluster --node-type=a1.large
```

o usa un file di configurazione:

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

metadata:
  name: cluster-arm-1
  region: us-west-2


nodeGroups:
  - name: ng-arm-1
```

```
    instanceType: m6g.medium
    desiredCapacity: 1
```

```
eksctl create cluster -f cluster-arm-1.yaml
```

ARM è supportato anche nei gruppi di nodi gestiti:

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

metadata:
  name: cluster-arm-2
  region: us-west-2

managedNodeGroups:
  - name: mng-arm-1
    instanceType: m6g.medium
    desiredCapacity: 1
```

```
eksctl create cluster -f cluster-arm-2.yaml
```

I resolver AMI `auto` e `auto-ssm` dedurranno l'AMI corretto in base al tipo di istanza ARM. Solo le famiglie AmazonLinux 2023, AmazonLinux 2 e Bottlerocket dispongono di EKS ottimizzato per ARM. AMIs

> **Note**
>
> ARM è supportato per i cluster con versione 1.15 e successive.

Auto Scaling

Abilita Auto Scaling

Puoi creare un cluster (o un gruppo di nodi in un cluster esistente) con un ruolo IAM che consentirà l'uso del cluster autoscaler:

```
eksctl create cluster --asg-access
```

Questo flag imposta anche `k8s.io/cluster-autoscaler/<clusterName>` tag, quindi la scoperta `k8s.io/cluster-autoscaler/enabled` dei gruppi di nodi dovrebbe funzionare.

Una volta che il cluster è in esecuzione, sarà necessario installare Cluster Autoscaler stesso.

È inoltre necessario aggiungere quanto segue alle definizioni dei gruppi di nodi gestiti o non gestiti per aggiungere i tag necessari affinché Cluster Autoscaler possa scalare il gruppo di nodi:

```
nodeGroups:
  - name: ng1-public
    iam:
      withAddonPolicies:
        autoScaler: true
```

Scalabilità verso l'alto da 0

Se desideri essere in grado di scalare il tuo gruppo di nodi da 0 e hai delle etichette and/or definite sui tuoi gruppi di nodi, dovrai propagarle come tag sui tuoi Auto Scaling Groups (). ASGs

Un modo per farlo è impostare i tag ASG nel campo delle definizioni dei gruppi di nodi. `tags` Ad esempio, dato un gruppo di nodi con le seguenti etichette e sfumature:

```
nodeGroups:
  - name: ng1-public
    ...
    labels:
      my-cool-label: pizza
    taints:
      key: feaster
      value: "true"
      effect: NoSchedule
```

Dovresti aggiungere i seguenti tag ASG:

```
nodeGroups:
  - name: ng1-public
    ...
    labels:
      my-cool-label: pizza
    taints:
      feaster: "true:NoSchedule"
```

```
    tags:
      k8s.io/cluster-autoscaler/node-template/label/my-cool-label: pizza
      k8s.io/cluster-autoscaler/node-template/taint/feaster: "true:NoSchedule"
```

Sia per i gruppi di nodi gestiti che per quelli non gestiti, questa operazione può essere eseguita automaticamente impostando `propagateASGTags` su`true`, che aggiungerà le etichette e le tonalità come tag al gruppo Auto Scaling:

```
nodeGroups:
  - name: ng1-public
    ...
    labels:
      my-cool-label: pizza
    taints:
      feaster: "true:NoSchedule"
    propagateASGTags: true
```

Auto Scaling con riconoscimento della zona

Se i tuoi carichi di lavoro sono specifici per zona, dovrai creare gruppi di nodi separati per ogni zona. Questo perché `cluster-autoscaler` presuppone che tutti i nodi di un gruppo siano esattamente equivalenti. Quindi, ad esempio, se un evento di scale-up viene attivato da un pod che richiede un PVC specifico per una zona (ad esempio un volume EBS), il nuovo nodo potrebbe essere programmato nella AZ sbagliata e il pod non si avvierà.

Non avrai bisogno di un gruppo di nodi separato per ogni AZ se il tuo ambiente soddisfa i seguenti criteri:

- Nessun requisito di archiviazione specifico per zona.
- Nessun PodAffinity richiesto con topologia diversa dall'host.
- Nessun NodeAffinity richiesto sull'etichetta della zona.
- Nessun NodeSelector su un'etichetta di zona.

(Leggi di più qui e qui.)

Se soddisfi tutti i requisiti di cui sopra (e forse anche altri), dovresti essere al sicuro con un singolo gruppo di nodi che si estende su più nodi. AZs Altrimenti ti consigliamo di creare gruppi di nodi Single-AZ separati:

PRIMA:

```
nodeGroups:
  - name: ng1-public
    instanceType: m5.xlarge
    # availabilityZones: ["eu-west-2a", "eu-west-2b"]
```

DOPO:

```
nodeGroups:
  - name: ng1-public-2a
    instanceType: m5.xlarge
    availabilityZones: ["eu-west-2a"]
  - name: ng1-public-2b
    instanceType: m5.xlarge
    availabilityZones: ["eu-west-2b"]
```

Supporto AMI personalizzato

Impostazione dell'ID AMI del nodo

Il `--node-ami` flag consente una serie di casi d'uso avanzati, come l'utilizzo di un'AMI personalizzata o l'esecuzione di query su AWS in tempo reale per determinare quale AMI utilizzare. Il flag può essere utilizzato sia per immagini non GPU che per immagini GPU.

Il flag può utilizzare l'ID dell'immagine AMI per un'immagine da utilizzare in modo esplicito. Può anche accettare le seguenti parole chiave «speciali»:

Parola chiave	Description
auto	Indica che l'AMI da utilizzare per i nodi deve essere trovata interrogando AWS EC2. Questo si riferisce al resolver automatico.
ssm automatico	Indica che l'AMI da utilizzare per i nodi deve essere trovata interrogando AWS SSM Parameter Store.

> **Note**
>
> Al momento, i gruppi di nodi gestiti da EKS supportano solo le seguenti famiglie AMI quando si lavora con applicazioni personalizzate AMIs:`AmazonLinux2023`,`AmazonLinux2`,, `BottlerocketUbuntu2004`, e `UbuntuPro2004` `Ubuntu2204` `Ubuntu2404`

Quando `--node-ami` si imposta su una stringa ID, `eksctl` si presume che sia stata richiesta un'AMI personalizzata. Per i nodi AmazonLinux 2 e Ubuntu, sia gestiti da EKS che autogestiti, ciò significa che `overrideBootstrapCommand` è necessario. Per il AmazonLinux 2023, poiché smette di usare lo `/etc/eks/bootstrap.sh` script per il bootstrap dei nodi, a favore di un processo di inizializzazione di nodeadm (per maggiori informazioni, consulta i documenti di bootstrapping dei nodi), non è supportato. `overrideBootstrapCommand`

Esempi di flag CLI:

```
eksctl create cluster --node-ami=auto

# with a custom ami id
eksctl create cluster --node-ami=ami-custom1234
```

Esempio di file Config:

```
nodeGroups:
  - name: ng1
    instanceType: p2.xlarge
    amiFamily: AmazonLinux2
    ami: auto
  - name: ng2
    instanceType: m5.large
    amiFamily: AmazonLinux2
    ami: ami-custom1234
managedNodeGroups:
  - name: m-ng-2
    amiFamily: AmazonLinux2
    ami: ami-custom1234
    instanceType: m5.large
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh <cluster-name>
```

La `--node-ami` bandiera può essere utilizzata anche con`eksctl create nodegroup`.

Impostazione del nodo AMI Family

`--node-ami-family`Possono assumere le seguenti parole chiave:

Parola chiave	Description
AmazonLinux2	Indica che deve essere utilizzata l'immagine AMI EKS basata su Amazon Linux 2 (impostazione predefinita).
AmazonLinux2023	Indica che deve essere utilizzata l'immagine AMI EKS basata su Amazon Linux 2023.
Ubuntu 2004	Indica che deve essere utilizzata l'immagine AMI EKS basata su Ubuntu 20.04 LTS (Focal) (supportata per EKS 1.29).
UbuntuPro2004	Indica che deve essere utilizzata l'immagine AMI EKS basata su Ubuntu Pro 20.04 LTS (Focal) (disponibile per EKS >= 1.27, <1.29).
Ubuntu 2204	Indica che deve essere utilizzata l'immagine EKS AMI basata su Ubuntu 22.04 LTS (Jammy) (disponibile per EKS >= 1.29).
UbuntuPro2204	Indica che deve essere utilizzata l'immagine EKS AMI basata su Ubuntu Pro 22.04 LTS (Jammy) (disponibile per EKS >= 1.29).
Ubuntu 2404	Indica che deve essere utilizzata l'immagine AMI EKS basata su Ubuntu 24.04 LTS (Noble) (disponibile per EKS >= 1.31).
UbuntuPro2404	Indica che deve essere utilizzata l'immagine AMI EKS basata su Ubuntu Pro 24.04 LTS (Noble) (disponibile per EKS >= 1.31).

Parola chiave	Description
Portabottiglie	Indica che deve essere utilizzata l'immagin e AMI EKS basata su Bottlerocket.
WindowsServer2019 FullContainer	Indica che deve essere utilizzat a l'immagine AMI EKS basata su Windows Server 2019 Full Container.
WindowsServer2019 CoreContainer	Indica che deve essere utilizzat a l'immagine AMI EKS basata su Windows Server 2019 Core Container.
WindowsServer2022 FullContainer	Indica che deve essere utilizzat a l'immagine AMI EKS basata su Windows Server 2022 Full Container.
WindowsServer2022 CoreContainer	Indica che deve essere utilizzat a l'immagine AMI EKS basata su Windows Server 2022 Core Container.

Esempio di flag CLI:

```
eksctl create cluster --node-ami-family=AmazonLinux2
```

Esempio di file Config:

```
nodeGroups:
  - name: ng1
    instanceType: m5.large
    amiFamily: AmazonLinux2
managedNodeGroups:
  - name: m-ng-2
    instanceType: m5.large
    amiFamily: Ubuntu2204
```

La `--node-ami-family` bandiera può essere utilizzata anche con`eksctl create nodegroup`. `eksctl`richiede che la famiglia AMI sia impostata esplicitamente tramite il file di configurazione o tramite il flag `--node-ami-family` CLI, ogni volta che si lavora con un'AMI personalizzata.

> **Note**
>
> Al momento, i gruppi di nodi gestiti da EKS supportano solo le seguenti famiglie AMI quando si lavora con applicazioni personalizzate AMIs:`AmazonLinux2023`,`AmazonLinux2`,, `BottlerocketUbuntu2004`, e `UbuntuPro2004` `Ubuntu2204` `Ubuntu2404`

Supporto AMI personalizzato per Windows

Solo i gruppi di nodi Windows autogestiti possono specificare un'AMI personalizzata. `amiFamily`deve essere impostato su una famiglia di AMI Windows valida.

Le seguenti PowerShell variabili saranno disponibili per lo script bootstrap:

```
$EKSBootstrapScriptFile
$EKSClusterName
$APIServerEndpoint
$Base64ClusterCA
$ServiceCIDR
$KubeletExtraArgs
$KubeletExtraArgsMap: A hashtable containing arguments for the kubelet, e.g., @{ 'node-
labels' = ''; 'register-with-taints' = ''; 'max-pods' = '10'}
$DNSClusterIP
$ContainerRuntime
```

Esempio di file Config:

```
nodeGroups:
  - name: custom-windows
    amiFamily: WindowsServer2022FullContainer
    ami: ami-01579b74557facaf7
    overrideBootstrapCommand: |
      & $EKSBootstrapScriptFile -EKSClusterName "$EKSClusterName" -APIServerEndpoint
 "$APIServerEndpoint" -Base64ClusterCA "$Base64ClusterCA" -ContainerRuntime
 "containerd" -KubeletExtraArgs "$KubeletExtraArgs" 3>&1 4>&1 5>&1 6>&1
```

Supporto AMI personalizzato Bottlerocket

Per i nodi Bottlerocket, non è supportato. `overrideBootstrapCommand` Invece, per designare il proprio contenitore bootstrap, è necessario utilizzare il `bottlerocket` campo come parte del file di configurazione. (ad esempio

```
  nodeGroups:
  - name: bottlerocket-ng
    ami: ami-custom1234
    amiFamily: Bottlerocket
    bottlerocket:
      enableAdminContainer: true
      settings:
        bootstrap-containers:
          bootstrap:
            source: <MY-CONTAINER-URI>
```

Nodi Windows Worker

A partire dalla versione 1.14, Amazon EKS supporta Windows Nodes che consentono l'esecuzione di contenitori Windows. Oltre a disporre di nodi Windows, è necessario un nodo Linux nel cluster per eseguire CoredNS, poiché Microsoft non supporta ancora la modalità di rete host. Pertanto, un cluster Windows EKS sarà una combinazione di nodi Windows e almeno un nodo Linux. I nodi Linux sono fondamentali per il funzionamento del cluster e pertanto, per un cluster di livello di produzione, si consiglia di disporre di almeno due nodi `t2.large` Linux per HA.

> **Note**
>
> Non è più necessario installare il controller di risorse VPC sui nodi di lavoro Linux per eseguire carichi di lavoro Windows nei cluster EKS creati dopo il 22 ottobre 2021. È possibile abilitare la gestione degli indirizzi IP di Windows sul piano di controllo EKS tramite un'impostazione ConfigMap (vedi link: eks/latest/userguide/windows -support.html per i dettagli). eksctl applicherà automaticamente una patch per abilitare la gestione degli indirizzi IP di Windows quando viene ConfigMap creato un gruppo di nodi Windows.

Creazione di un nuovo cluster con supporto per Windows

La sintassi del file di configurazione consente di creare un cluster completamente funzionante con supporto per Windows in un unico comando:

```
# cluster.yaml
# An example of ClusterConfig containing Windows and Linux node groups to support
 Windows workloads
---
```

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

metadata:
  name: windows-cluster
  region: us-west-2

nodeGroups:
  - name: windows-ng
    amiFamily: WindowsServer2019FullContainer
    minSize: 2
    maxSize: 3

managedNodeGroups:
  - name: linux-ng
    instanceType: t2.large
    minSize: 2
    maxSize: 3

  - name: windows-managed-ng
    amiFamily: WindowsServer2019FullContainer
    minSize: 2
    maxSize: 3
```

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

Per creare un nuovo cluster con un gruppo di nodi non gestito da Windows senza utilizzare un file di configurazione, esegui i seguenti comandi:

```
eksctl create cluster --managed=false --name=windows-cluster --node-ami-
family=WindowsServer2019CoreContainer
```

Aggiungere il supporto per Windows a un cluster Linux esistente

Per consentire l'esecuzione di carichi di lavoro Windows su un cluster esistente con nodi Linux (famiglia `AmazonLinux2` AMI), è necessario aggiungere un gruppo di nodi Windows.

È stato aggiunto il NUOVO supporto per il gruppo di nodi gestito da Windows (--managed=true o omit the flag).

```
eksctl create nodegroup --managed=false --cluster=existing-cluster --node-ami-
family=WindowsServer2019CoreContainer
```

```
eksctl create nodegroup --cluster=existing-cluster --node-ami-
family=WindowsServer2019CoreContainer
```

Per garantire che i carichi di lavoro siano pianificati sul sistema operativo giusto, devono avere come destinazione il sistema operativo su cui devono essere eseguiti: `nodeSelector`

```
# Targeting Windows
  nodeSelector:
    kubernetes.io/os: windows
    kubernetes.io/arch: amd64
```

```
# Targeting Linux
  nodeSelector:
    kubernetes.io/os: linux
    kubernetes.io/arch: amd64
```

Se si utilizza un cluster più vecchio di `kubernetes.io/os` e `1.19` le `kubernetes.io/arch` etichette devono essere sostituite rispettivamente con `beta.kubernetes.io/os` e`beta.kubernetes.io/arch`.

Ulteriori informazioni

- Supporto EKS per Windows

Mappature di volume aggiuntive

Come opzione di configurazione aggiuntiva, quando si tratta di mappature di volume, è possibile configurare mappature aggiuntive quando viene creato il gruppo di nodi.

Per fare ciò, imposta il campo come segue: `additionalVolumes`

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

metadata:
  name: dev-cluster
  region: eu-north-1

managedNodeGroups:
  - name: ng-1-workers
```

```
    labels: { role: workers }
    instanceType: m5.xlarge
    desiredCapacity: 10
    volumeSize: 80
    additionalVolumes:
      - volumeName: '/tmp/mount-1' # required
        volumeSize: 80
        volumeType: 'gp3'
        volumeEncrypted: true
        volumeKmsKeyID: 'id'
        volumeIOPS: 3000
        volumeThroughput: 125
      - volumeName: '/tmp/mount-2'  # required
        volumeSize: 80
        volumeType: 'gp2'
        snapshotID: 'snapshot-id'
```

Per ulteriori dettagli sulla selezione dei nomi dei volumi, consultate la documentazione sulla denominazione dei dispositivi. Per saperne di più sui volumi EBS, sui limiti di volume delle istanze o sulle mappature dei dispositivi a blocchi, visita questa pagina.

Nodi ibridi EKS

Introduzione

AWS EKS introduce Hybrid Nodes, una nuova funzionalità che consente di eseguire applicazioni locali e periferiche su un'infrastruttura gestita dal cliente con gli stessi cluster, caratteristiche e strumenti AWS EKS che usi nel cloud AWS. AWS EKS Hybird Nodes offre un'esperienza Kubernetes gestita da AWS negli ambienti locali per consentire ai clienti di semplificare e standardizzare il modo in cui esegui le applicazioni in ambienti locali, edge e cloud. Scopri di più su EKS Hybrid Nodes.

Per facilitare il supporto di questa funzionalità, eksctl introduce un nuovo campo di primo livello chiamato. `remoteNetworkConfig` Qualsiasi configurazione relativa ai nodi ibridi deve essere impostata tramite questo campo, come parte del file di configurazione; non ci sono flag CLI equivalenti. Inoltre, all'avvio, qualsiasi configurazione di rete remota può essere configurata solo durante la creazione del cluster e non può essere aggiornata in seguito. Ciò significa che non sarà possibile aggiornare i cluster esistenti per utilizzare i nodi ibridi.

La **`remoteNetworkConfig`** sezione del file di configurazione consente di configurare le due aree principali quando si tratta di unire nodi remoti ai cluster EKS: rete e credenziali.

Rete

EKS Hybrid Nodes è flessibile e si adatta al metodo preferito per connettere le reti locali a un VPC AWS. Sono disponibili diverse opzioni documentate, tra cui AWS Site-to-Site VPN e AWS Direct Connect, e puoi scegliere il metodo più adatto al tuo caso d'uso. Nella maggior parte dei metodi che puoi scegliere, il tuo VPC sarà collegato a un gateway privato virtuale (VGW) o a un gateway di transito (TGW). Se ti affidi a eksctl per creare un VPC per te, eksctl configurerà anche, nell'ambito del tuo VPC, tutti i prerequisiti relativi alla rete per facilitare la comunicazione tra il tuo piano di controllo EKS e i nodi remoti, ad es.

- regole SGS di ingresso/uscita
- percorsi nelle tabelle delle rotte delle sottoreti private
- l'attacco del gateway VPC al dato TGW o VGW

File di configurazione di esempio:

```
remoteNetworkConfig:
  vpcGatewayID: tgw-xxxx # either VGW or TGW to be attached to your VPC
  remoteNodeNetworks:
    # eksctl will create, behind the scenes, SG rules, routes, and a VPC gateway
 attachment,
    # to facilitate communication between remote network(s) and EKS control plane, via
 the attached gateway
    - cidrs: ["10.80.146.0/24"]
  remotePodNetworks:
    - cidrs: ["10.86.30.0/23"]
```

Se il tuo metodo di connettività preferito non prevede l'utilizzo di un TGW o VGW, non devi fare affidamento su eksctl per creare il VPC per te, ma fornirne invece uno preesistente. In una nota correlata, se utilizzi un VPC preesistente, eksctl non apporterà alcuna modifica ad esso e la garanzia che tutti i requisiti di rete siano soddisfatti è sotto la tua responsabilità.

> **Note**
>
> eksctl non configura alcuna infrastruttura di rete al di fuori del VPC AWS (ovvero alcuna infrastruttura che collega VGW/TGW le reti remote)

Credenziali

I nodi ibridi EKS utilizzano AWS IAM Authenticator e credenziali IAM temporanee fornite da AWS SSM o AWS IAM Roles Anywhere per l'autenticazione con il cluster EKS. Analogamente ai gruppi di nodi autogestiti, se non diversamente specificato, eksctl creerà per te un ruolo IAM di Hybrid Nodes che verrà assunto dai nodi remoti. Inoltre, quando si utilizza IAM Roles Anywhere come fornitore di credenziali, eksctl configurerà un profilo e tratterà un ancoraggio basato su un determinato pacchetto di autorità di certificazione (), ad es. `iam.caBundleCert`

```
remoteNetworkConfig:
  iam:
    # the provider for temporary IAM credentials. Default is SSM.
    provider: IRA
    # the certificate authority bundle that serves as the root of trust,
    # used to validate the X.509 certificates provided by your nodes.
    # can only be set when provider is IAMRolesAnywhere.
    caBundleCert: xxxx
```

L'ARN del ruolo Hybrid Nodes creato da eksctl è necessario più avanti nel processo di unione dei nodi remoti al clusternodeadm, `NodeConfig` per la configurazione e la creazione di attivazioni (se si utilizza SSM). Per recuperarlo, usa:

```
aws cloudformation describe-stacks \
  --stack-name eksctl-<CLUSTER_NAME>-cluster \
  --query 'Stacks[].Outputs[?OutputKey==`RemoteNodesRoleARN`].[OutputValue]' \
  --output text
```

Allo stesso modo, se usi IAM Roles Anywhere, puoi recuperare l'ARN del trust anchor e del profilo anywhere creato da eksctl, modificando il comando precedente sostituendolo rispettivamente con o. `RemoteNodesRoleARN` `RemoteNodesTrustAnchorARN` `RemoteNodesAnywhereProfileARN`

Se disponi di una configurazione IAM Roles Anywhere preesistente o stai utilizzando SSM, puoi fornire un ruolo IAM per i nodi ibridi tramite. `remoteNetworkConfig.iam.roleARN` Tieni presente che in questo scenario, eksctl non creerà per te il profilo trust anchor e anywhere. Ad es.

```
remoteNetworkConfig:
  iam:
    roleARN: arn:aws:iam::000011112222:role/HybridNodesRole
```

Per mappare il ruolo a un'identità Kubernetes e autorizzare i nodi remoti a unirsi al cluster EKS, eksctl crea una voce di accesso con Hybrid Nodes IAM Role come ARN principale e di tipo. ad es. `HYBRID_LINUX`

```
eksctl get accessentry --cluster my-cluster --principal-arn
 arn:aws:iam::000011112222:role/eksctl-my-cluster-clust-HybridNodesSSMRole-XiIAg0d29PkO
 --output json
[
    {
        "principalARN": "arn:aws:iam::000011112222:role/eksctl-my-cluster-clust-
HybridNodesSSMRole-XiIAg0d29PkO",
        "kubernetesGroups": [
            "system:nodes"
        ]
    }
]
```

Supporto per componenti aggiuntivi

Container Networking Interface (CNI): la CNI di AWS VPC non può essere utilizzata con nodi ibridi. Le funzionalità principali di Cilium e Calico sono supportate per l'uso con nodi ibridi. Puoi gestire il tuo CNI con strumenti di tua scelta come Helm. Per ulteriori informazioni, consulta Configurare un CNI per nodi ibridi.

> **Note**
>
> Se installi VPC CNI nel tuo cluster per i tuoi gruppi di nodi autogestiti o gestiti da EKS, devi utilizzare `v1.19.0-eksbuild.1` o versioni successive, poiché include un aggiornamento del daemonset del componente aggiuntivo per escluderne l'installazione sui nodi ibridi.

Ulteriori riferimenti

- Nodi ibridi EKS UserDocs
- Annuncio di lancio

Configurazione di riparazione dei nodi di supporto per i gruppi di nodi gestiti da EKS

EKS Managed Nodegroups supporta Node Repair, in cui lo stato dei nodi gestiti viene monitorato e i nodi di lavoro non sani vengono sostituiti o riavviati in risposta. eksctl ora fornisce opzioni di configurazione complete per un controllo granulare sul comportamento di riparazione dei nodi.

Configurazione base di riparazione dei nodi

Utilizzo dei flag CLI

Per creare un cluster con un gruppo di nodi gestito utilizzando la riparazione di base dei nodi, passa il flag: `--enable-node-repair`

```
eksctl create cluster --enable-node-repair
```

Per creare un gruppo di nodi gestito con riparazione dei nodi su un cluster esistente:

```
eksctl create nodegroup --cluster=<clusterName> --enable-node-repair
```

Utilizzo dei file di configurazione

```
# basic-node-repair.yaml
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: basic-node-repair-cluster
  region: us-west-2

managedNodeGroups:
- name: ng-1
  nodeRepairConfig:
    enabled: true
```

```
eksctl create cluster -f basic-node-repair.yaml
```

Configurazione avanzata di riparazione dei nodi

Configurazione della soglia

È possibile configurare quando le azioni di riparazione dei nodi smetteranno di utilizzare soglie basate sulla percentuale o sul conteggio. Nota: non è possibile utilizzare contemporaneamente la soglia percentuale e quella di conteggio.

Flag CLI per le soglie

```
# Percentage-based threshold - repair stops when 20% of nodes are unhealthy
eksctl create cluster --enable-node-repair \
  --node-repair-max-unhealthy-percentage=20

# Count-based threshold - repair stops when 5 nodes are unhealthy
eksctl create cluster --enable-node-repair \
  --node-repair-max-unhealthy-count=5
```

File di configurazione per le soglie

```
managedNodeGroups:
- name: threshold-ng
  nodeRepairConfig:
    enabled: true
    # Stop repair actions when 20% of nodes are unhealthy
    maxUnhealthyNodeThresholdPercentage: 20
    # Alternative: stop repair actions when 3 nodes are unhealthy
    # maxUnhealthyNodeThresholdCount: 3
    # Note: Cannot use both percentage and count thresholds simultaneously
```

Limiti di riparazione parallela

Controlla il numero massimo di nodi che possono essere riparati contemporaneamente o in parallelo. In questo modo è possibile controllare in modo più preciso il ritmo delle sostituzioni dei nodi. Nota: non è possibile utilizzare contemporaneamente i limiti percentuali e di conteggio.

Flag CLI per limiti paralleli

```
# Percentage-based parallel limits - repair at most 15% of unhealthy nodes in parallel
eksctl create cluster --enable-node-repair \
  --node-repair-max-parallel-percentage=15
```

```
# Count-based parallel limits - repair at most 2 unhealthy nodes in parallel
eksctl create cluster --enable-node-repair \
  --node-repair-max-parallel-count=2
```

File di configurazione per limiti paralleli

```
managedNodeGroups:
- name: parallel-ng
  nodeRepairConfig:
    enabled: true
    # Repair at most 15% of unhealthy nodes in parallel
    maxParallelNodesRepairedPercentage: 15
    # Alternative: repair at most 2 unhealthy nodes in parallel
    # maxParallelNodesRepairedCount: 2
    # Note: Cannot use both percentage and count limits simultaneously
```

Sostituzioni di riparazione personalizzate

Specificate le 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. Se si utilizza questa opzione, è necessario specificare tutti i valori per ogni sostituzione.

```
managedNodeGroups:
- name: custom-repair-ng
  instanceType: g4dn.xlarge  # GPU instances
  nodeRepairConfig:
    enabled: true
    maxUnhealthyNodeThresholdPercentage: 25
    maxParallelNodesRepairedCount: 1
    nodeRepairConfigOverrides:
      # Handle GPU-related failures with immediate termination
      - nodeMonitoringCondition: "AcceleratedInstanceNotReady"
        nodeUnhealthyReason: "NvidiaXID13Error"
        minRepairWaitTimeMins: 10
        repairAction: "Terminate"
      # Handle network issues with restart after waiting
      - nodeMonitoringCondition: "NetworkNotReady"
        nodeUnhealthyReason: "InterfaceNotUp"
        minRepairWaitTimeMins: 20
        repairAction: "Restart"
```

Esempi di configurazione completi

Per un esempio completo con tutte le opzioni di configurazione, vedi examples/44-node-repair.yaml.

Esempio 1: riparazione di base con soglie percentuali

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

metadata:
  name: basic-repair-cluster
  region: us-west-2

managedNodeGroups:
- name: basic-ng
  instanceType: m5.large
  desiredCapacity: 3
  nodeRepairConfig:
    enabled: true
    maxUnhealthyNodeThresholdPercentage: 20
    maxParallelNodesRepairedPercentage: 15
```

Esempio 2: riparazione conservativa per carichi di lavoro critici

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

metadata:
  name: critical-workload-cluster
  region: us-west-2

managedNodeGroups:
- name: critical-ng
  instanceType: c5.2xlarge
  desiredCapacity: 6
  nodeRepairConfig:
    enabled: true
    # Very conservative settings
    maxUnhealthyNodeThresholdPercentage: 10
    maxParallelNodesRepairedCount: 1
    nodeRepairConfigOverrides:
      # Wait longer before taking action on critical workloads
      - nodeMonitoringCondition: "NetworkNotReady"
```

```
        nodeUnhealthyReason: "InterfaceNotUp"
        minRepairWaitTimeMins: 45
        repairAction: "Restart"
```

Esempio 3: carico di lavoro su GPU con riparazione specializzata

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

metadata:
  name: gpu-workload-cluster
  region: us-west-2

managedNodeGroups:
- name: gpu-ng
  instanceType: g4dn.xlarge
  desiredCapacity: 4
  nodeRepairConfig:
    enabled: true
    maxUnhealthyNodeThresholdPercentage: 25
    maxParallelNodesRepairedCount: 1
    nodeRepairConfigOverrides:
      # GPU failures require immediate termination
      - nodeMonitoringCondition: "AcceleratedInstanceNotReady"
        nodeUnhealthyReason: "NvidiaXID13Error"
        minRepairWaitTimeMins: 5
        repairAction: "Terminate"
```

Documentazione di riferimento dell'interfaccia a riga di comando

Bandiere di riparazione dei nodi

Flag	Description	Esempio
`--enable-node-repair`	Abilita la riparazione automatica dei nodi	`--enable-node-repair`
`--node-repair-max-unhealthy-percentage`	Percentuale massima di nodi non integri prima della riparazione	`--node-repair-max-unhealthy-percentage=20`

Flag	Description	Esempio
`--node-repair-max-unhealthy-count`	Numero massimo di nodi non integri prima della riparazione	`--node-repair-max-unhealthy-count=5`
`--node-repair-max-parallel-percentage`	Percentuale massima di nodi da riparare in parallelo	`--node-repair-max-parallel-percentage=15`
`--node-repair-max-parallel-count`	Numero massimo di nodi da riparare in parallelo	`--node-repair-max-parallel-count=2`

Nota: le modifiche alla configurazione per la riparazione dei nodi sono supportate solo tramite i file di configurazione YAML a causa della loro complessità.

Informazioni di riferimento sulla configurazione

nodeRepairConfig

Campo	Tipo	Description	Vincoli	Esempio
`enabled`	booleano	Abilita/disabilita la riparazione dei nodi	-	`true`
`maxUnhealthyNodeThresholdPercentage`	intero	Soglia percentuale di nodi non integri, al di sopra della quale le azioni di riparazione automatica del nodo verranno interrotte	Non può essere utilizzato con `maxUnhealthyNodeThresholdCount`	`20`
`maxUnhealthyNodeTh`	intero	Conta la soglia dei nodi non integri, oltre la quale le azioni	Non può essere utilizzato con	`5`

Campo	Tipo	Description	Vincoli	Esempio
`resholdCount`		di riparazione automatica del nodo si interromperanno	`maxUnhealthyNodeThresholdPercentage`	
`maxParallelNodesRepairedPercentage`	intero	Percentuale massima di nodi non integri che possono essere riparati contemporaneamente o in parallelo	Non può essere utilizzato con `maxParallelNodesRepairedCount`	15
`maxParallelNodesRepairedCount`	intero	Numero massimo di nodi non integri che possono essere riparati contemporaneamente o in parallelo	Non può essere utilizzato con `maxParallelNodesRepairedPercentage`	2
`nodeRepairConfigOverrides`	array	Sostituzioni granulari per azioni di riparazione specifiche che controllano l'azione di riparazione e il tempo di ritardo	Tutti i valori devono essere specificati per ogni sostituzione	Vedi gli esempi precedenti

nodeRepairConfigSostituzioni

Campo	Tipo	Description	Valori validi
`nodeMonitoringCondition`	stringa	Condizione non integra segnalata dall'agente di monitoraggio del nodo a cui si applica questa sostituzione	`"AcceleratedInstanceNotRead`

Campo	Tipo	Description	Valori validi
			`y" , "NetworkNotReady"`
`nodeUnhealthyReason`	stringa	Motivo segnalato dall'agente di monitoraggio dei nodi a cui si applica questa eccezione	`"NvidiaXID13Error" ,"InterfaceNotUp"`
`minRepairWaitTimeMins`	intero	Tempo minimo di attesa in minuti prima di tentare di riparare un nodo con la condizione e il motivo specificati	Qualsiasi numero intero positivo
`repairAction`	stringa	Azione di riparazione da eseguire per i nodi quando tutte le condizioni specificate sono soddisfatte	`"Terminate" , "Restart" , "NoAction"`

Ulteriori informazioni

- EKS Managed Nodegroup Node Health

Rete

Questo capitolo include informazioni su come Eksctl crea reti Virtual Private Cloud (VPC) per i cluster EKS.

Argomenti:

- the section called “Configurazione del VPC”
  - Modifica l'intervallo VPC CIDR e configura l'indirizzamento IPv6
  - Usa un VPC esistente
  - Personalizza il VPC, le sottoreti, i gruppi di sicurezza e i gateway NAT per il nuovo cluster EKS
- the section called “Impostazioni della sottorete”
  - Utilizza sottoreti private per il gruppo di nodi iniziale per isolarlo dalla rete Internet pubblica
  - Personalizza la topologia delle sottoreti elencando più sottoreti per zona di disponibilità e specificando le sottoreti nelle configurazioni dei gruppi di nodi
  - Limita i gruppi di nodi a sottoreti denominate specifiche nella configurazione VPC
  - Quando si utilizzano sottoreti private per gruppi di nodi, imposta su `privateNetworking` `true`
  - Fornisci una specifica di sottorete completa con entrambe `public` le `private` configurazioni nella specifica VPC
  - Solo una delle `subnets` o `availabilityZones` può essere fornita nella configurazione del gruppo di nodi
- the section called “Accesso al cluster”
  - Gestisci l'accesso pubblico e privato agli endpoint del server dell'API Kubernetes in un cluster EKS
  - Limita l'accesso all'endpoint dell'API pubblica EKS Kubernetes specificando gli intervalli CIDR consentiti
  - Aggiorna la configurazione di accesso agli endpoint del server API e le restrizioni CIDR di accesso pubblico per un cluster esistente
- the section called “Rete del piano di controllo”
  - Aggiorna le sottoreti utilizzate dal piano di controllo EKS per un cluster
- the section called “IPv6 Support”

- Specificare la versione (IPv4 o IPv6) IP da utilizzare durante la creazione di un VPC con cluster EKS

Configurazione del VPC

VPC dedicato per cluster

Per impostazione predefinita, `eksctl create cluster` creerà un VPC dedicato per il cluster. Questo viene fatto per evitare interferenze con le risorse esistenti per una serie di motivi, tra cui la sicurezza, ma anche perché è difficile rilevare tutte le impostazioni in un VPC esistente.

- Il VPC CIDR predefinito utilizzato da è. `eksctl` `192.168.0.0/16`
  - È diviso in 8 (`/19`) sottoreti (3 private, 3 pubbliche e 2 riservate).
- Il nodegroup iniziale viene creato in sottoreti pubbliche.
- L'accesso SSH è disabilitato a meno che non sia specificato. `--allow-ssh`
- Per impostazione predefinita, il nodegroup consente il traffico in entrata dal gruppo di sicurezza del piano di controllo sulle porte 1025 - 65535.

> **Note**
>
> In `us-east-1` eksctl crea solo 2 sottoreti pubbliche e 2 private per impostazione predefinita.

Cambia VPC CIDR

Se devi configurare il peering con un altro VPC o semplicemente hai bisogno di un intervallo più o meno ampio IPs di, puoi `--vpc-cidr` usare flag per cambiarlo. Consulta i documenti AWS per le guide sulla scelta dei blocchi CIDR consentiti per l'uso in un VPC AWS.

Se stai creando un IPv6 cluster, puoi configurarlo `VPC.IPv6Cidr` nel file di configurazione del cluster. Questa impostazione è solo nel file di configurazione, non in un flag CLI.

Se possiedi un blocco di indirizzi IPv6 IP, puoi anche creare un pool tutto tuo. IPv6 Consulta Bring your own IP address (BYOIP) su Amazon EC2 per informazioni su come importare il tuo pool. Quindi usa il `VPC.IPv6Cidr` file di configurazione del cluster per configurare Eksctl.

Usa un VPC esistente: condiviso con kops

Puoi utilizzare il VPC di un cluster Kubernetes esistente gestito da kops. Questa funzionalità viene fornita per facilitare la migrazione e il peering del cluster. and/or

Se in precedenza hai creato un cluster con kops, ad esempio utilizzando comandi simili a questo:

```
export KOPS_STATE_STORE=s3://kops
kops create cluster cluster-1.k8s.local --zones=us-west-2c,us-west-2b,us-west-2a --
networking=weave --yes
```

È possibile creare un cluster EKS nello stesso AZs utilizzando le stesse sottoreti VPC (NOTA: ne sono necessarie almeno 2 AZs/subnets):

```
eksctl create cluster --name=cluster-2 --region=us-west-2 --vpc-from-kops-
cluster=cluster-1.k8s.local
```

Usa VPC esistente: altra configurazione personalizzata

`eksctl`offre una certa flessibilità, ma non completa, per topologie VPC e sottoreti personalizzate.

È possibile utilizzare un VPC esistente fornendo sottoreti and/or pubbliche private utilizzando i flag and. `--vpc-private-subnets` `--vpc-public-subnets` Spetta all'utente assicurarsi che le sottoreti utilizzate siano classificate correttamente, poiché non esiste un modo semplice per verificare se una sottorete sia effettivamente privata o pubblica, poiché le configurazioni variano.

Dati questi flag, `eksctl create cluster` determinerà automaticamente l'ID VPC, ma non creerà tabelle di routing o altre risorse, come i gateway. internet/NAT Tuttavia, creerà gruppi di sicurezza dedicati per il nodegroup iniziale e il piano di controllo.

È necessario assicurarsi di fornire almeno 2 sottoreti diverse AZs e questa condizione viene verificata da EKS. Se utilizzi un VPC esistente, i seguenti requisiti non vengono applicati o controllati da EKS o Eksctl e EKS crea il cluster. Alcune funzioni di base del cluster funzionano senza questi requisiti. (Ad esempio, il tagging non è strettamente necessario, i test hanno dimostrato che è possibile creare un cluster funzionale senza alcun tag impostato nelle sottoreti, tuttavia non vi è alcuna garanzia che ciò sia sempre valido e l'etichettatura è consigliata.)

Requisiti standard:

- tutte le sottoreti specificate devono trovarsi nello stesso VPC, all'interno dello stesso blocco di IPs

- è disponibile un numero sufficiente di indirizzi IP, in base alle esigenze
- un numero sufficiente di sottoreti (minimo 2), in base alle esigenze
- le sottoreti sono etichettate con almeno quanto segue:
  - `kubernetes.io/cluster/<name>`tag impostato su uno o `shared` `owned`
  - `kubernetes.io/role/internal-elb`tag impostato su `1` per le sottoreti private
  - `kubernetes.io/role/elb`tag impostato su *`1`* per le sottoreti pubbliche
- gateway NAT Internet and/or configurati correttamente
- le tabelle di routing hanno le voci corrette e la rete è funzionante
- NOVITÀ: tutte le sottoreti pubbliche devono avere la proprietà `MapPublicIpOnLaunch` abilitata (ad esempio `Auto-assign public IPv4 address` nella console AWS). I gruppi di nodi gestiti e Fargate non assegnano IPv4 indirizzi pubblici, la proprietà deve essere impostata nella sottorete.

Potrebbero esserci altri requisiti imposti da EKS o Kubernetes e sta interamente a te attenerti a qualsiasi requisito. up-to-date and/or recommendations, and implement those as needed/possible

Le impostazioni predefinite dei gruppi di sicurezza applicate da `eksctl` possono o meno essere sufficienti per condividere l'accesso con le risorse di altri gruppi di sicurezza. Se desideri modificare ingress/egress le regole dei gruppi di sicurezza, potresti dover utilizzare un altro strumento per automatizzare le modifiche o farlo tramite la console EC2.

In caso di dubbio, non utilizzare un VPC personalizzato. L'utilizzo `eksctl create cluster` senza `--vpc-*` flag configurerà sempre il cluster con un VPC dedicato completamente funzionale.

Esempi

Crea un cluster utilizzando un VPC personalizzato con 2 sottoreti private e 2 pubbliche:

```
eksctl create cluster \
  --vpc-private-subnets=subnet-0ff156e0c4a6d300c,subnet-0426fb4a607393184 \
  --vpc-public-subnets=subnet-0153e560b3129a696,subnet-009fa0199ec203c37
```

oppure usa il seguente file di configurazione equivalente:

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

```
metadata:
  name: my-test
  region: us-west-2

vpc:
  id: "vpc-11111"
  subnets:
    private:
      us-west-2a:
          id: "subnet-0ff156e0c4a6d300c"
      us-west-2c:
          id: "subnet-0426fb4a607393184"
    public:
      us-west-2a:
          id: "subnet-0153e560b3129a696"
      us-west-2c:
          id: "subnet-009fa0199ec203c37"

nodeGroups:
  - name: ng-1
```

Crea un cluster utilizzando un VPC personalizzato con 3 sottoreti private e fai in modo che il nodegroup iniziale utilizzi quelle sottoreti:

```
eksctl create cluster \
  --vpc-private-
subnets=subnet-0ff156e0c4a6d300c,subnet-0549cdab573695c03,subnet-0426fb4a607393184 \
  --node-private-networking
```

oppure usa il seguente file di configurazione equivalente:

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

metadata:
  name: my-test
  region: us-west-2

vpc:
  id: "vpc-11111"
  subnets:
    private:
      us-west-2d:
```

```
          id: "subnet-0ff156e0c4a6d300c"
      us-west-2c:
          id: "subnet-0549cdab573695c03"
      us-west-2a:
          id: "subnet-0426fb4a607393184"

nodeGroups:
  - name: ng-1
    privateNetworking: true
```

Crea un cluster utilizzando una sottorete pubblica VPC 4x personalizzata:

```
eksctl create cluster \
  --vpc-public-
subnets=subnet-0153e560b3129a696,subnet-0cc9c5aebe75083fd,subnet-009fa0199ec203c37,subnet-018fa
```

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

metadata:
  name: my-test
  region: us-west-2

vpc:
  id: "vpc-11111"
  subnets:
    public:
      us-west-2d:
          id: "subnet-0153e560b3129a696"
      us-west-2c:
          id: "subnet-0cc9c5aebe75083fd"
      us-west-2a:
          id: "subnet-009fa0199ec203c37"
      us-west-2b:
          id: "subnet-018fa0176ba320e45"

nodeGroups:
  - name: ng-1
```

Altri esempi possono essere trovati nella cartella del repository: `examples`

- utilizzo di un VPC esistente

- utilizzando un VPC CIDR personalizzato

Gruppo di sicurezza dei nodi condivisi personalizzato

`eksctl`creerà e gestirà un gruppo di sicurezza dei nodi condivisi che consente la comunicazione tra i nodi non gestiti e il piano di controllo del cluster e i nodi gestiti.

Se invece desideri fornire il tuo gruppo di sicurezza personalizzato, puoi sovrascrivere il `sharedNodeSecurityGroup` campo nel file di configurazione:

```
vpc:
  sharedNodeSecurityGroup: sg-0123456789
```

Per impostazione predefinita, durante la creazione del cluster, `eksctl` aggiungerà regole a questo gruppo di sicurezza per consentire la comunicazione da e verso il gruppo di sicurezza del cluster predefinito creato da EKS. Il gruppo di sicurezza del cluster predefinito viene utilizzato sia dal piano di controllo EKS che dai gruppi di nodi gestiti.

Se desideri gestire tu stesso le regole del gruppo di sicurezza, puoi evitare `eksctl` di creare le regole `manageSharedNodeSecurityGroupRules` impostando `false` nel file di configurazione su:

```
vpc:
  sharedNodeSecurityGroup: sg-0123456789
  manageSharedNodeSecurityGroupRules: false
```

Gateway NAT

Il gateway NAT per un cluster può essere configurato come`Disable`, `Single` (impostazione predefinita) o. `HighlyAvailable` L'`HighlyAvailable`opzione implementerà un gateway NAT in ogni zona di disponibilità della regione, in modo che, se una zona di disponibilità non funziona, i nodi dell'altra AZs siano ancora in grado di comunicare con Internet.

Può essere specificato tramite il flag `--vpc-nat-mode` CLI o nel file di configurazione del cluster come nell'esempio seguente:

```
vpc:
  nat:
    gateway: HighlyAvailable # other options: Disable, Single (default)
```

Guarda l'esempio completo qui.

> **Note**
>
> La specificazione del gateway NAT è supportata solo durante la creazione del cluster. Non viene toccato durante un aggiornamento del cluster.

Impostazioni della sottorete

Usa sottoreti private per il gruppo di nodi iniziale

Se preferisci isolare il gruppo di nodi iniziale dalla rete Internet pubblica, puoi usare il flag. `--node-private-networking` Se utilizzata insieme al `--ssh-access` flag, è possibile accedere alla porta SSH solo dall'interno del VPC.

> **Note**
>
> L'utilizzo del `--node-private-networking` flag farà sì che il traffico in uscita attraversi il gateway NAT utilizzando il relativo IP elastico. D'altra parte, se i nodi si trovano in una sottorete pubblica, il traffico in uscita non passerà attraverso il gateway NAT e quindi il traffico in uscita ha l'IP di ogni singolo nodo.

Topologia di sottorete personalizzata

`eksctl`la versione `0.32.0` ha introdotto un'ulteriore personalizzazione della topologia di sottorete con la possibilità di:

- Elenca più sottoreti per AZ nella configurazione VPC
- Specificare le sottoreti nella configurazione del gruppo di nodi

Nelle versioni precedenti le sottoreti personalizzate dovevano essere fornite per zona di disponibilità, il che significa che era possibile elencare solo una sottorete per AZ. `0.32.0`Le chiavi di identificazione possono essere arbitrarie.

```
vpc:
```

```
  id: "vpc-11111"
  subnets:
    public:
      public-one:                           # arbitrary key
          id: "subnet-0153e560b3129a696"
      public-two:
          id: "subnet-0cc9c5aebe75083fd"
      us-west-2b:                           # or list by AZ
          id: "subnet-018fa0176ba320e45"
    private:
      private-one:
          id: "subnet-0153e560b3129a696"
      private-two:
          id: "subnet-0cc9c5aebe75083fd"
```

> **Important**
>
> Se si utilizza l'AZ come chiave di identificazione, il `az` valore può essere omesso.

Se si utilizza una stringa arbitraria come chiave di identificazione, come sopra, è possibile:

- `id`deve essere impostato (`az` e `cidr` facoltativo)
- o `az` deve essere impostato (`cidr`opzionale)

Se un utente specifica una sottorete tramite AZ senza specificare CIDR e ID, una sottorete in quella AZ verrà scelta dal VPC, arbitrariamente se esistono più sottoreti di questo tipo.

> **Note**
>
> È necessario fornire una specifica di sottorete completa, ovvero entrambe le `public` `private` configurazioni dichiarate nelle specifiche VPC.

I gruppi di nodi possono essere limitati a sottoreti denominate tramite la configurazione. Quando si specificano le sottoreti nella configurazione del gruppo di nodi, utilizzare la chiave di identificazione fornita nelle specifiche VPC, non l'id della sottorete. Esempio:

```
vpc:
```

```
  id: "vpc-11111"
  subnets:
    public:
      public-one:
          id: "subnet-0153e560b3129a696"
    ... # subnet spec continued

nodeGroups:
  - name: ng-1
    instanceType: m5.xlarge
    desiredCapacity: 2
    subnets:
      - public-one
```

> **Note**
>
> È possibile fornire solo una delle o può essere fornita nella configurazione del gruppo di nodi. `subnets` `availabilityZones`

Quando si posizionano i gruppi di nodi all'interno di una sottorete privata, `privateNetworking` devono essere impostati su on the nodegroup: `true`

```
vpc:
  id: "vpc-11111"
  subnets:
    public:
      private-one:
          id: "subnet-0153e560b3129a696"
    ... # subnet spec continued

nodeGroups:
  - name: ng-1
    instanceType: m5.xlarge
    desiredCapacity: 2
    privateNetworking: true
    subnets:
      - private-one
```

Vedi 24-nodegroup-subnets.yaml nel repository eksctl per un esempio di configurazione completo. GitHub

Accesso al cluster

Gestione dell'accesso agli endpoint del server API Kubernetes

Per impostazione predefinita, un cluster EKS espone il server API Kubernetes pubblicamente ma non direttamente dall'interno delle sottoreti VPC (public=true, private=false). Il traffico destinato al server API dall'interno del VPC deve prima uscire dalle reti VPC (ma non dalla rete di Amazon) e quindi rientrare per raggiungere il server API.

L'accesso agli endpoint del server dell'API Kubernetes per un cluster può essere configurato per l'accesso pubblico e privato durante la creazione del cluster utilizzando il file di configurazione del cluster. Di seguito è riportato un esempio:

```
vpc:
  clusterEndpoints:
    publicAccess:  <true|false>
    privateAccess: <true|false>
```

Esistono alcune avvertenze aggiuntive durante la configurazione dell'accesso agli endpoint dell'API Kubernetes:

1. EKS non consente i cluster senza l'accesso privato o pubblico abilitato.
2. EKS consente di creare una configurazione che consente solo l'accesso privato, ma eksctl non la supporta durante la creazione del cluster in quanto impedisce a eksctl di unire i nodi di lavoro al cluster.
3. L'aggiornamento di un cluster per avere un accesso privato agli endpoint dell'API Kubernetes significa che i comandi Kubernetes, per impostazione predefinita (ad esempio`kubectl`) e, possibilmente`eksctl delete cluster`, `eksctl utils write-kubeconfig` il comando `eksctl utils update-kube-proxy` devono essere eseguiti all'interno del VPC del cluster.
   - Ciò richiede alcune modifiche a varie risorse AWS. Per ulteriori informazioni, consulta Cluster API Server Endpoint.
   - Puoi fornire `vpc.extraCIDRs` che aggiungerà intervalli CIDR aggiuntivi a ControlPlaneSecurityGroup, consentendo alle sottoreti esterne al VPC di raggiungere l'endpoint dell'API Kubernetes. Allo stesso modo puoi provvedere ad aggiungere anche intervalli CIDR. `vpc.extraIPv6CIDRs` IPv6

Di seguito è riportato un esempio di come è possibile configurare l'accesso agli endpoint dell'API Kubernetes utilizzando il sottocomando: `utils`

```
eksctl utils update-cluster-vpc-config --cluster=<clustername> --private-access=true --
public-access=false
```

Per aggiornare l'impostazione utilizzando un file, usa: **`ClusterConfig`**

```
eksctl utils update-cluster-vpc-config -f config.yaml --approve
```

Nota che se non passi un flag, manterrà il valore corrente. Quando sei soddisfatto delle modifiche proposte, aggiungi il `approve` flag per apportare la modifica al cluster in esecuzione.

Limitazione dell'accesso all'endpoint dell'API pubblica EKS Kubernetes

La creazione predefinita di un cluster EKS espone pubblicamente il server API Kubernetes.

Questa funzionalità si applica solo all'endpoint pubblico. Le opzioni di configurazione dell'accesso agli endpoint del server API non cambieranno e avrai comunque la possibilità di disabilitare l'endpoint pubblico in modo che il cluster non sia accessibile da Internet. (Fonte: https://github.com/aws/containers-roadmap/issues/108 #issuecomment -552766489)

Per limitare l'accesso all'endpoint dell'API pubblica a un set di durante la creazione di un cluster, imposta il campo: CIDRs **`publicAccessCIDRs`**

```
vpc:
  publicAccessCIDRs: ["1.1.1.1/32", "2.2.2.0/24"]
```

Per aggiornare le restrizioni su un cluster esistente, usa:

```
eksctl utils update-cluster-vpc-config --cluster=<cluster> 1.1.1.1/32,2.2.2.0/24
```

Per aggiornare le restrizioni utilizzando un **`ClusterConfig`** file, imposta il nuovo CIDRs in **`vpc.publicAccessCIDRs`** ed esegui:

```
eksctl utils update-cluster-vpc-config -f config.yaml
```

> **Important**
>
> Se si impostano `publicAccessCIDRs` e si creano gruppi di nodi, è `privateAccess` necessario impostare su `true` o aggiungere i nodi IPs all'elenco. `publicAccessCIDRs`

Se i nodi non possono accedere all'endpoint dell'API del cluster a causa dell'accesso limitato, la creazione del cluster avrà esito negativo `context deadline exceeded` perché i nodi non saranno in grado di accedere all'endpoint pubblico e non riusciranno a unirsi al cluster.

Per aggiornare sia l'accesso agli endpoint del server API che l'accesso pubblico CIDRs per un cluster con un unico comando, esegui:

```
eksctl utils update-cluster-vpc-config --cluster=<cluster> --public-access=true --
private-access=true --public-access-cidrs=1.1.1.1/32,2.2.2.0/24
```

Per aggiornare l'impostazione utilizzando un file di configurazione:

```
vpc:
  clusterEndpoints:
    publicAccess:  <true|false>
    privateAccess: <true|false>
  publicAccessCIDRs: ["1.1.1.1/32"]
```

```
eksctl utils update-cluster-vpc-config --cluster=<cluster> -f config.yaml
```

Aggiornamento delle sottoreti e dei gruppi di sicurezza del piano di controllo

Questa documentazione spiega come modificare la configurazione di rete del piano di controllo del cluster EKS dopo la creazione iniziale. Ciò include l'aggiornamento delle sottoreti e dei gruppi di sicurezza del piano di controllo.

Aggiornamento delle sottoreti del piano di controllo

Quando un cluster viene creato con eksctl, viene creata una serie di sottoreti pubbliche e private che vengono passate all'API EKS. EKS crea da 2 a 4 interfacce di rete elastiche multiaccount (ENIs) in

tali sottoreti per consentire la comunicazione tra il piano di controllo Kubernetes gestito da EKS e il VPC.

Per aggiornare le sottoreti utilizzate dal piano di controllo EKS, esegui:

```
eksctl utils update-cluster-vpc-config --cluster=<cluster> --control-plane-subnet-
ids=subnet-1234,subnet-5678
```

Per aggiornare l'impostazione utilizzando un file di configurazione:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: cluster
  region: us-west-2

vpc:
  controlPlaneSubnetIDs: [subnet-1234, subnet-5678]
```

```
eksctl utils update-cluster-vpc-config -f config.yaml
```

Senza il `--approve` flag, eksctl registra solo le modifiche proposte. Quando sei soddisfatto delle modifiche proposte, esegui nuovamente il comando con il flag. `--approve`

Aggiornamento dei gruppi di sicurezza del piano di controllo

Per gestire il traffico tra il piano di controllo e i nodi di lavoro, EKS supporta l'invio di gruppi di sicurezza aggiuntivi che vengono applicati alle interfacce di rete tra account fornite da EKS. Per aggiornare i gruppi di sicurezza per il piano di controllo EKS, esegui:

```
eksctl utils update-cluster-vpc-config --cluster=<cluster> --control-plane-security-
group-ids=sg-1234,sg-5678
```

Per aggiornare l'impostazione utilizzando un file di configurazione:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: cluster
  region: us-west-2
```

```
vpc:
  controlPlaneSecurityGroupIDs: [sg-1234, sg-5678]
```

```
eksctl utils update-cluster-vpc-config -f config.yaml
```

Per aggiornare sia le sottoreti del piano di controllo che i gruppi di sicurezza per un cluster, esegui:

```
eksctl utils update-cluster-vpc-config --cluster=<cluster> --control-plane-subnet-
ids=<> --control-plane-security-group-ids=<>
```

Per aggiornare entrambi i campi utilizzando un file di configurazione:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: cluster
  region: us-west-2

vpc:
  controlPlaneSubnetIDs: [subnet-1234, subnet-5678]
  controlPlaneSecurityGroupIDs: [sg-1234, sg-5678]
```

```
eksctl utils update-cluster-vpc-config -f config.yaml
```

Per un esempio completo, consulta cluster-subnets-sgs.yaml.

Senza il `--approve` flag, eksctl registra solo le modifiche proposte. Quando sei soddisfatto delle modifiche proposte, esegui nuovamente il comando con il flag. `--approve`

IPv6 Support

Definisci la famiglia IP

Quando si `eksctl` crea un vpc, è possibile definire la versione IP che verrà utilizzata. Le seguenti opzioni sono disponibili per essere configurate:

- IPv4
- IPv6

Il valore predefinito è `IPv4`.

Per definirlo, usa il seguente esempio:

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

metadata:
  name: my-test
  region: us-west-2
  version: "1.21"

kubernetesNetworkConfig:
  ipFamily: IPv6 # or IPv4

addons:
  - name: vpc-cni
  - name: coredns
  - name: kube-proxy

iam:
  withOIDC: true
```

> **Note**
>
> Questa impostazione è solo nel file di configurazione, non in un flag CLI.

Se si utilizza IPv6, è necessario configurare i seguenti requisiti:

- OIDC è abilitato
- i componenti aggiuntivi gestiti sono definiti come illustrato sopra
- la versione del cluster deve essere => 1.21
- la versione del componente aggiuntivo vpc-cni deve essere => 1.10.0
- i gruppi di nodi autogestiti non sono supportati IPv6 con i cluster
- i gruppi di nodi gestiti non sono supportati con cluster non proprietari IPv6
- `vpc.nate serviceIPv4CIDR` i campi sono creati da eksctl per i cluster ipv6 e non sono opzioni di configurazione supportate
- AutoAllocateIPv6 non è supportato insieme a IPv6

- Per il IPv6 cluster, il ruolo IAM per vpc-cni deve avere politiche IAM richieste per la modalità associata IPv6

Le reti private possono essere eseguite anche con la famiglia IPv6 IP. Segui le istruzioni riportate in EKS Private Cluster.

IAM

Questo capitolo include informazioni sull'utilizzo di AWS IAM.

Argomenti:

- the section called “Gestione degli utenti e dei ruoli IAM”
  - Gestisci le mappature di utenti e ruoli IAM per controllare l'accesso a un cluster EKS
  - Configura le mappature delle identità IAM tramite il file di configurazione del cluster o i comandi CLI
- the section called “Ruoli IAM per gli account di servizio”
  - Gestisci le autorizzazioni granulari per le applicazioni in esecuzione su Amazon EKS che utilizzano altri servizi AWS
  - Crea e configura coppie di ruoli IAM e account di servizio Kubernetes utilizzando eksctl
  - Abilita IAM OpenID Connect Provider per un cluster EKS per abilitare IAM Roles for Service Accounts
- the section called “Limite delle autorizzazioni IAM”
  - Controlla le autorizzazioni massime concesse alle entità IAM (utenti o ruoli) impostando un limite di autorizzazioni
- the section called “Associazioni EKS Pod Identity”
  - Configura le autorizzazioni IAM per i componenti aggiuntivi EKS utilizzando le associazioni di identità dei pod consigliate
  - Consenti alle applicazioni Kubernetes di ricevere le autorizzazioni IAM necessarie per connettersi ai servizi AWS all'esterno del cluster
  - Semplifica il processo di automazione dei ruoli e degli account di servizio IAM su più cluster EKS
- the section called “Policy IAM”
  - Gestisci le policy IAM per i gruppi di nodi EKS, incluso il supporto per varie policy aggiuntive come image builder, auto scaler, DNS esterno, gestore di certificati e altro ancora.
  - Allega ruoli di istanza personalizzati o policy in linea ai gruppi di nodi per ottenere autorizzazioni aggiuntive.
  - Collega specifiche policy gestite da AWS tramite ARN ai gruppi di nodi, assicurandoti che siano incluse le policy richieste come Amazon e EKSWorker NodePolicy Amazoneks_CNI_Policy.

- the section called “Policy IAM minime”
  - Gestisci le risorse AWS EC2, inclusi sistemi di bilanciamento del carico, gruppi di auto-scaling e monitoraggio CloudWatch
  - Crea e gestisci CloudFormation stack AWS
  - Gestisci cluster Amazon Elastic Kubernetes Service (EKS), gruppi di nodi e risorse correlate come ruoli e policy IAM

Policy IAM minime

Questo documento descrive le politiche IAM minime necessarie per eseguire i principali casi d'uso di eksctl. Queste sono quelle utilizzate per eseguire i test di integrazione.

> **Note**
>
> Ricordati di sostituirli `<account_id>` con i tuoi.

> **Note**
>
> Una AWS Managed Policy viene creata e amministrata da AWS. Non è possibile modificare le autorizzazioni definite nelle policy gestite da AWS.

Amazon EC2 FullAccess (politica gestita da AWS)

Visualizza la definizione EC2 FullAccess della politica di Amazon.

AWSCloudFormationFullAccess (Politica gestita da AWS)

Visualizza AWSCloud FormationFullAccess la definizione della policy.

EksAllAccess

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

```
            "Action": "eks:*",
            "Resource": "*"
        },
        {
            "Action": [
                "ssm:GetParameter",
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:*:123456789012:parameter/aws/*",
                "arn:aws:ssm:*::parameter/aws/*"
            ],
            "Effect": "Allow"
        },
        {
             "Action": [
               "kms:CreateGrant",
               "kms:DescribeKey"
             ],
             "Resource": "*",
             "Effect": "Allow"
        },
        {
             "Action": [
               "logs:PutRetentionPolicy"
             ],
             "Resource": "*",
             "Effect": "Allow"
        }
    ]
}
```

IamLimitedAccess

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:GetInstanceProfile",
```

```
                "iam:RemoveRoleFromInstanceProfile",
                "iam:GetRole",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "iam:UpdateAssumeRolePolicy",
                "iam:AddRoleToInstanceProfile",
                "iam:ListInstanceProfilesForRole",
                "iam:PassRole",
                "iam:DetachRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:GetRolePolicy",
                "iam:GetOpenIDConnectProvider",
                "iam:CreateOpenIDConnectProvider",
                "iam:DeleteOpenIDConnectProvider",
                "iam:TagOpenIDConnectProvider",
                "iam:ListAttachedRolePolicies",
                "iam:TagRole",
                "iam:UntagRole",
                "iam:GetPolicy",
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:ListPolicyVersions"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:instance-profile/eksctl-*",
                "arn:aws:iam::123456789012:role/eksctl-*",
                "arn:aws:iam::123456789012:policy/eksctl-*",
                "arn:aws:iam::123456789012:oidc-provider/*",
                "arn:aws:iam::123456789012:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "arn:aws:iam::123456789012:role/eksctl-managed-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetUser"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/*",
                "arn:aws:iam::123456789012:user/*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "eks.amazonaws.com",
                        "eks-nodegroup.amazonaws.com",
                        "eks-fargate.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Limite delle autorizzazioni IAM

Un limite di autorizzazioni è una funzionalità avanzata di AWS IAM in cui sono state impostate le autorizzazioni massime che una policy basata sull'identità può concedere a un'entità IAM, laddove tali entità siano utenti o ruoli. Quando viene impostato un limite di autorizzazioni per un'entità, tale entità può eseguire solo le azioni consentite sia dalle sue politiche basate sull'identità che dai suoi limiti di autorizzazioni.

Puoi fornire il limite delle tue autorizzazioni in modo che tutte le entità basate sull'identità create da eksctl vengano create all'interno di quel limite. Questo esempio dimostra come è possibile fornire un limite di autorizzazioni alle varie entità basate sull'identità create da eksctl:

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

metadata:
  name: cluster-17
  region: us-west-2

iam:
  withOIDC: true
```

```
  serviceRolePermissionsBoundary: "arn:aws:iam::11111:policy/entity/boundary"
  fargatePodExecutionRolePermissionsBoundary: "arn:aws:iam::11111:policy/entity/
boundary"
  serviceAccounts:
    - metadata:
        name: s3-reader
      attachPolicyARNs:
      - "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
      permissionsBoundary: "arn:aws:iam::11111:policy/entity/boundary"

nodeGroups:
  - name: "ng-1"
    desiredCapacity: 1
    iam:
      instanceRolePermissionsBoundary: "arn:aws:iam::11111:policy/entity/boundary"
```

> **Warning**
>
> Non è possibile fornire sia un ARN di ruolo che un limite di autorizzazioni.

Impostazione del limite di autorizzazione VPC CNI

Tieni presente che quando crei un cluster con OIDC abilitato eksctl creerà automaticamente un `iamserviceaccount` per VPC-CNI per motivi di sicurezza. Se desideri aggiungere un limite di autorizzazione, devi specificarlo manualmente nel tuo file di configurazione: `iamserviceaccount`

```
iam:
  serviceAccounts:
    - metadata:
        name: aws-node
        namespace: kube-system
      attachPolicyARNs:
      - "arn:aws:iam::<arn>:policy/AmazonEKS_CNI_Policy"
      permissionsBoundary: "arn:aws:iam::11111:policy/entity/boundary"
```

Policy IAM

Puoi collegare Instance Roles ai gruppi di nodi. I carichi di lavoro in esecuzione sul nodo riceveranno le autorizzazioni IAM dal nodo. Per ulteriori informazioni, consulta Ruoli IAM per Amazon EC2.

Questa pagina elenca i modelli di policy IAM predefiniti disponibili in eksctl. Questi modelli semplificano il processo di concessione ai nodi EKS delle autorizzazioni di servizio AWS appropriate senza dover creare manualmente policy IAM personalizzate.

Policy aggiuntive IAM supportate

Esempio di tutte le politiche aggiuntive supportate:

```
nodeGroups:
  - name: ng-1
    instanceType: m5.xlarge
    desiredCapacity: 1
    iam:
      withAddonPolicies:
        imageBuilder: true
        autoScaler: true
        externalDNS: true
        certManager: true
        appMesh: true
        appMeshPreview: true
        ebs: true
        fsx: true
        efs: true
        awsLoadBalancerController: true
        xRay: true
        cloudWatch: true
```

Politica di Image Builder

La `imageBuilder` policy consente l'accesso completo all'ECR (Elastic Container Registry). Ciò è utile per creare, ad esempio, un server CI che deve inviare immagini a ECR.

Politica EBS

La `ebs` policy abilita il nuovo driver EBS CSI (Elastic Block Store Container Storage Interface).

Politica di Cert Manager

La `certManager` policy consente di aggiungere record a Route 53 per risolvere il DNS01 problema. Ulteriori informazioni possono essere trovate qui.

Aggiungere un ruolo di istanza personalizzato

Questo esempio crea un gruppo di nodi che riutilizza un ruolo di istanza IAM esistente da un altro cluster:

```
apiVersion: eksctl.io/v1alpha4
kind: ClusterConfig
metadata:
  name: test-cluster-c-1
  region: eu-north-1

nodeGroups:
  - name: ng2-private
    instanceType: m5.large
    desiredCapacity: 1
    iam:
      instanceProfileARN: "arn:aws:iam::123:instance-profile/eksctl-test-cluster-a-3-
nodegroup-ng2-private-NodeInstanceProfile-Y4YKHLNINMXC"
      instanceRoleARN: "arn:aws:iam::123:role/eksctl-test-cluster-a-3-nodegroup-
NodeInstanceRole-DNGMQTQHQHBJ"
```

Allegare politiche in linea

```
nodeGroups:
  - name: my-special-nodegroup
    iam:
      attachPolicy:
        Version: "2012-10-17"
        Statement:
        - Effect: Allow
          Action:
          - 's3:GetObject'
          Resource: 'arn:aws:s3:::example-bucket/*'
```

Allegare politiche tramite ARN

```
nodeGroups:
  - name: my-special-nodegroup
    iam:
      attachPolicyARNs:
        - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
```

```
        - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
        - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
        - arn:aws:iam::1111111111:policy/kube2iam
      withAddonPolicies:
        autoScaler: true
        imageBuilder: true
```

> **Warning**
>
> Se un gruppo di nodi include il, `attachPolicyARNs` deve includere anche le politiche dei nodi predefinite`AmazonEKSWorkerNodePolicy`, `AmazonEKS_CNI_Policy` come in questo esempio. `AmazonEC2ContainerRegistryPullOnly`

Gestione degli utenti e dei ruoli IAM

> **Note**
>
> AWS suggerisce di migrare a da. the section called “Associazioni EKS Pod Identity” `aws-auth` ConfigMap

I cluster EKS utilizzano utenti e ruoli IAM per controllare l'accesso al cluster. Le regole sono implementate in una mappa di configurazione

Modifica ConfigMap con un comando CLI

chiamato`aws-auth`. `eksctl`fornisce comandi per leggere e modificare questa mappa di configurazione.

Ottieni tutte le mappature delle identità:

```
eksctl get iamidentitymapping --cluster <clusterName> --region=<region>
```

Ottieni tutte le mappature di identità corrispondenti a un arn:

```
eksctl get iamidentitymapping --cluster <clusterName> --region=<region> --arn
 arn:aws:iam::123456:role/testing-role
```

Crea una mappatura dell'identità:

```
eksctl create iamidentitymapping --cluster  <clusterName> --region=<region> --arn
arn:aws:iam::123456:role/testing --group system:masters --username admin
```

Eliminare una mappatura dell'identità:

```
eksctl delete iamidentitymapping --cluster  <clusterName> --region=<region> --arn
 arn:aws:iam::123456:role/testing
```

> **Note**
>
> Il comando precedente elimina una singola mappatura FIFO a meno che non `--all` venga fornito, nel qual caso rimuove tutte le corrispondenze. Avviserà se vengono trovate altre mappature corrispondenti a questo ruolo.

Crea una mappatura dell'account:

```
eksctl create iamidentitymapping --cluster  <clusterName> --region=<region> --account
user-account
```

Eliminare una mappatura dell'account:

```
eksctl delete iamidentitymapping --cluster  <clusterName> --region=<region> --account
user-account
```

Modifica ConfigMap utilizzando un ClusterConfig file

Le mappature delle identità possono essere specificate anche in: ClusterConfig

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

metadata:
  name: cluster-with-iamidentitymappings
  region: us-east-1
```

```
iamIdentityMappings:
  - arn: arn:aws:iam::000000000000:role/myAdminRole
    groups:
      - system:masters
    username: admin
    noDuplicateARNs: true # prevents shadowing of ARNs

  - arn: arn:aws:iam::000000000000:user/myUser
    username: myUser
    noDuplicateARNs: true # prevents shadowing of ARNs

  - serviceName: emr-containers
    namespace: emr # serviceName requires namespace

  - account: "000000000000" # account must be configured with no other options

nodeGroups:
  - name: ng-1
    instanceType: m5.large
    desiredCapacity: 1
```

```
eksctl create iamidentitymapping -f cluster-with-iamidentitymappings.yaml
```

Ruoli IAM per gli account di servizio

> **Tip**
>
> `eksctl`supporta la configurazione di autorizzazioni granulari per le app che eseguono EKS tramite EKS Pod Identity Associations

Amazon EKS supporta here Roles for Service Accounts (IRSA)] che consente agli operatori del cluster di mappare i ruoli AWS IAM agli account di servizio Kubernetes.

Ciò fornisce una gestione granulare delle autorizzazioni per le app eseguite su EKS e che utilizzano altri servizi AWS. Potrebbero essere app che utilizzano S3, qualsiasi altro servizio dati (RDS, MQ, STS, DynamoDB) o componenti Kubernetes come il controller AWS Load Balancer o ExternalDNS.

Puoi creare facilmente coppie di ruoli e account di servizio IAM con. `eksctl`

> **Note**
>
> Se hai utilizzato ruoli di istanza e stai pensando di utilizzare invece IRSA, non dovresti mischiare i due.

Come funziona

Funziona tramite IAM OpenID Connect Provider (OIDC) esposto da EKS e i ruoli IAM devono essere costruiti con riferimento al provider IAM OIDC (specifico per un determinato cluster EKS) e un riferimento all'account di servizio Kubernetes a cui sarà associato. Una volta creato un ruolo IAM, un account di servizio dovrebbe includere l'ARN di quel ruolo come annotazione (). `eks.amazonaws.com/role-arn` Per impostazione predefinita, l'account di servizio verrà creato o aggiornato per includere l'annotazione del ruolo, che può essere disabilitata utilizzando il flag. `--role-only`

All'interno di EKS, c'è un controller di ammissione che inietta le credenziali di sessione AWS nei pod rispettivamente dei ruoli in base all'annotazione sull'account di servizio utilizzato dal pod. Le credenziali verranno esposte dalle variabili di ambiente &. `AWS_ROLE_ARN` `AWS_WEB_IDENTITY_TOKEN_FILE` Dato che viene utilizzata una versione recente di AWS SDK (vedi qui per i dettagli della versione esatta), l'applicazione utilizzerà queste credenziali.

Nel nome `eksctl` della risorsa c'è iamserviceaccount, che rappresenta una coppia IAM Role e Service Account.

Utilizzo da CLI

> **Note**
>
> IAM Roles for Service Accounts richiede la versione 1.13 o successiva di Kubernetes.

Il provider IAM OIDC non è abilitato per impostazione predefinita, puoi utilizzare il seguente comando per abilitarlo o utilizzare il file di configurazione (vedi sotto):

```
eksctl utils associate-iam-oidc-provider --cluster=<clusterName>
```

Una volta associato il provider IAM OIDC al cluster, per creare un ruolo IAM associato a un account di servizio, esegui:

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=<serviceAccountName> --
namespace=<serviceAccountNamespace> --attach-policy-arn=<policyARN>
```

> **Note**
>
> Puoi specificare `--attach-policy-arn` più volte di utilizzare più di una policy.

Più specificamente, puoi creare un account di servizio con accesso in sola lettura a S3 eseguendo:

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=s3-read-only --attach-
policy-arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
```

Per impostazione predefinita, verrà creato nello spazio dei `default` nomi, ma puoi specificare qualsiasi altro spazio dei nomi, ad esempio:

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=s3-read-only --
namespace=s3-app --attach-policy-arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
```

> **Note**
>
> Se il namespace non esiste già, verrà creato.

Se hai già un account di servizio creato nel cluster (senza un ruolo IAM), dovrai usare `--override-existing-serviceaccounts` flag.

I tag personalizzati possono essere applicati anche al ruolo IAM `--tags` specificando:

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=<serviceAccountName> --
tags "Owner=John Doe,Team=Some Team"
```

CloudFormation genererà un nome di ruolo che include una stringa casuale. Se preferisci un nome di ruolo predeterminato puoi specificare`--role-name`:

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=<serviceAccountName> --
role-name "custom-role-name"
```

Quando l'account di servizio viene creato e gestito da un altro strumento, ad esempio helm, utilizzalo `--role-only` per prevenire i conflitti. L'altro strumento è quindi responsabile del mantenimento dell'annotazione ARN del ruolo. Nota che non `--override-existing-serviceaccounts` ha alcun effetto sugli account `roleOnly` `--role-only` /service, il ruolo verrà sempre creato.

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=<serviceAccountName> --
role-only --role-name=<customRoleName>
```

Se disponi di un ruolo esistente che desideri utilizzare con un account di servizio, puoi fornire il `--attach-role-arn` flag invece di fornire le politiche. Per garantire che il ruolo possa essere assunto solo dall'account di servizio specificato, è necessario impostare qui un documento sulla politica di relazione].

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=<serviceAccountName> --
attach-role-arn=<customRoleARN>
```

Per aggiornare i ruoli di un account di servizio, le autorizzazioni che puoi eseguire`eksctl update iamserviceaccount`.

> **Note**
>
> `eksctl delete iamserviceaccount`elimina Kubernetes `ServiceAccounts` anche se non sono stati creati da. `eksctl`

Utilizzo con file di configurazione

Per gestire `iamserviceaccounts` l'utilizzo del file di configurazione, dovrai impostare `iam.withOIDC: true` ed elencare l'account che desideri utilizzare. `iam.serviceAccount`

Tutti i comandi supportati*`--config-file`*, puoi gestire iamserviceaccounts allo stesso modo dei nodegroups. I supporti `--include` e i `--exclude` flag dei `eksctl create iamserviceaccount` comandi (consulta questa sezione per maggiori dettagli su come funzionano). Inoltre, il `eksctl delete iamserviceaccount` comando `--only-missing` lo supporta, quindi puoi eseguire le eliminazioni allo stesso modo dei gruppi di nodi.

> **Note**
>
> Gli account di servizio IAM rientrano nell'ambito di un namespace, ovvero possono esistere due account di servizio con lo stesso nome in namespace diversi. Pertanto, per definire in modo univoco un account di servizio come parte di`--include`, `--exclude` flags, è necessario passare la stringa del nome nel formato. `namespace/name` (ad esempio

```
eksctl create iamserviceaccount --config-file=<path> --include backend-apps/s3-reader
```

L'opzione di attivazione `wellKnownPolicies` è inclusa per utilizzare IRSA con casi d'uso noti come `cluster-autoscaler` `ecert-manager`, come abbreviazione per elenchi di policy.

Le politiche note supportate e altre proprietà di `serviceAccounts` sono documentate nello schema di configurazione.

Si utilizza il seguente esempio di configurazione con: `eksctl create cluster`

```
# An example of ClusterConfig with IAMServiceAccounts:
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: cluster-13
  region: us-west-2

iam:
  withOIDC: true
  serviceAccounts:
  - metadata:
      name: s3-reader
      # if no namespace is set, "default" will be used;
      # the namespace will be created if it doesn't exist already
      namespace: backend-apps
      labels: {aws-usage: "application"}
    attachPolicyARNs:
    - "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    tags:
      Owner: "John Doe"
      Team: "Some Team"
```

```
  - metadata:
      name: cache-access
      namespace: backend-apps
      labels: {aws-usage: "application"}
    attachPolicyARNs:
    - "arn:aws:iam::aws:policy/AmazonDynamoDBReadOnlyAccess"
    - "arn:aws:iam::aws:policy/AmazonElastiCacheFullAccess"
  - metadata:
      name: cluster-autoscaler
      namespace: kube-system
      labels: {aws-usage: "cluster-ops"}
    wellKnownPolicies:
      autoScaler: true
    roleName: eksctl-cluster-autoscaler-role
    roleOnly: true
  - metadata:
      name: some-app
      namespace: default
    attachRoleARN: arn:aws:iam::123:role/already-created-role-for-app
nodeGroups:
  - name: "ng-1"
    tags:
      # EC2 tags required for cluster-autoscaler auto-discovery
      k8s.io/cluster-autoscaler/enabled: "true"
      k8s.io/cluster-autoscaler/cluster-13: "owned"
    desiredCapacity: 1
```

Se crei un cluster senza questi campi impostati, puoi utilizzare i seguenti comandi per abilitare tutto ciò di cui hai bisogno:

```
eksctl utils associate-iam-oidc-provider --config-file=<path>
eksctl create iamserviceaccount --config-file=<path>
```

Ulteriori informazioni

- Presentazione dei ruoli IAM dettagliati per gli account di servizio
- Guida per l'utente EKS - IAM Roles For Service Accounts
- Mappatura degli utenti e del ruolo IAM ai ruoli RBAC di Kubernetes

Associazioni EKS Pod Identity

AWS EKS ha introdotto un nuovo meccanismo avanzato chiamato Pod Identity Association per consentire agli amministratori di cluster di configurare le applicazioni Kubernetes per ricevere le autorizzazioni IAM necessarie per connettersi ai servizi AWS esterni al cluster. Pod Identity Association sfrutta IRSA, tuttavia la rende configurabile direttamente tramite l'API EKS, eliminando del tutto la necessità di utilizzare l'API IAM.

Di conseguenza, i ruoli IAM non devono più fare riferimento a un provider OIDC e quindi non saranno più legati a un singolo cluster. Ciò significa che i ruoli IAM possono ora essere utilizzati su più cluster EKS senza la necessità di aggiornare la policy di fiducia dei ruoli ogni volta che viene creato un nuovo cluster. Questo, a sua volta, elimina la necessità di duplicare i ruoli e semplifica del tutto il processo di automazione dell'IRSA.

Prerequisiti

Dietro le quinte, l'implementazione delle associazioni di identità dei pod consiste nell'eseguire un agente come demone sui nodi di lavoro. Per eseguire l'agente prerequisito sul cluster, EKS fornisce un nuovo componente aggiuntivo chiamato EKS Pod Identity Agent. Pertanto, la creazione di associazioni di identità dei pod (in generale e con`eksctl`) richiede l'`eks-pod-identity-agent`addon preinstallato nel cluster. Questo componente aggiuntivo può essere creato utilizzando nello stesso modo `eksctl` in cui viene utilizzato qualsiasi altro componente aggiuntivo supportato.

```
eksctl create addon --cluster my-cluster --name eks-pod-identity-agent
```

Inoltre, se si utilizza un ruolo IAM preesistente durante la creazione di un'associazione di identità del pod, è necessario configurare il ruolo in modo che consideri attendibile il service principal EKS appena introdotto (). `pods.eks.amazonaws.com` Di seguito è riportato un esempio di politica di fiducia IAM:

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

```
                "sts:TagSession"
            ]
        }
    ]
}
```

Se invece non fornisci l'ARN di un ruolo esistente al comando create, ne creerà uno dietro le quinte e `eksctl` configurerà la politica di fiducia di cui sopra.

Creazione di associazioni di identità Pod

Per manipolare le associazioni di identità dei pod, `eksctl` ha aggiunto un nuovo campo sotto`iam.podIdentityAssociations`, ad es.

```
iam:
  podIdentityAssociations:
  - namespace: <string> #required
    serviceAccountName: <string> #required
    createServiceAccount: true #optional, default is false
    roleARN: <string> #required if none of permissionPolicyARNs, permissionPolicy and
 wellKnownPolicies is specified. Also, cannot be used together with any of the three
 other referenced fields.
    roleName: <string> #optional, generated automatically if not provided, ignored if
 roleARN is provided
    permissionPolicy: {} #optional
    permissionPolicyARNs: [] #optional
    wellKnownPolicies: {} #optional
    permissionsBoundaryARN: <string> #optional
    tags: {} #optional
```

Per un esempio completo, fai riferimento a pod-identity-associations .yaml.

> **Note**
>
> Oltre `permissionPolicy` a essere utilizzato come documento di policy in linea, tutti gli altri campi hanno una controparte con flag CLI.

La creazione di associazioni di identità pod può essere eseguita nei seguenti modi. Durante la creazione del cluster, specificando le associazioni di identità dei pod desiderate come parte del file di configurazione ed eseguendo:

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

Dopo la creazione del cluster, utilizzando un file di configurazione, ad es.

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

O utilizzando i flag CLI, ad es.

```
eksctl create podidentityassociation \
    --cluster my-cluster \
    --namespace default \
    --service-account-name s3-reader \
    --permission-policy-arns="arn:aws:iam::111122223333:policy/permission-policy-1,
 arn:aws:iam::111122223333:policy/permission-policy-2" \
    --well-known-policies="autoScaler,externalDNS" \
    --permissions-boundary-arn arn:aws:iam::111122223333:policy/permissions-boundary
```

> **Note**
>
> A un account di servizio può essere associato un solo ruolo IAM alla volta. Pertanto, il tentativo di creare una seconda associazione di identità del pod per lo stesso account di servizio genererà un errore.

Recupero delle associazioni di identità dei pod

Per recuperare tutte le associazioni di identità dei pod per un determinato cluster, esegui uno dei seguenti comandi:

```
eksctl get podidentityassociation -f config.yaml
```

O

```
eksctl get podidentityassociation --cluster my-cluster
```

Inoltre, per recuperare solo le associazioni di identità dei pod all'interno di un determinato namespace, usa il flag, ad es. `--namespace`

```
eksctl get podidentityassociation --cluster my-cluster --namespace default
```

Infine, per recuperare una singola associazione, corrispondente a un determinato account di servizio K8s, includi anche il `--service-account-name` comando precedente, ad es.

```
eksctl get podidentityassociation --cluster my-cluster --namespace default --service-
account-name s3-reader
```

Aggiornamento delle associazioni di identità Pod

Per aggiornare il ruolo IAM di una o più associazioni di identità dei pod, passa il nuovo `roleARN(s)` al file di configurazione, ad es.

```
iam:
  podIdentityAssociations:
    - namespace: default
      serviceAccountName: s3-reader
      roleARN: new-role-arn-1
    - namespace: dev
      serviceAccountName: app-cache-access
      roleARN: new-role-arn-2
```

ed esegui:

```
eksctl update podidentityassociation -f config.yaml
```

OPPURE (per aggiornare una singola associazione) passa la nuova `--role-arn` tramite i flag CLI:

```
eksctl update podidentityassociation --cluster my-cluster --namespace default --
service-account-name s3-reader --role-arn new-role-arn
```

Eliminazione delle associazioni di identità Pod

Per eliminare una o più associazioni di identità dei pod, passale `namespace(s)` e `serviceAccountName(s)` al file di configurazione, ad es.

```
iam:
  podIdentityAssociations:
    - namespace: default
      serviceAccountName: s3-reader
```

```
    - namespace: dev
      serviceAccountName: app-cache-access
```

ed esegui:

```
eksctl delete podidentityassociation -f config.yaml
```

OPPURE (per eliminare una singola associazione) passa i flag `--namespace` e `--service-account-name` tramite CLI:

```
eksctl delete podidentityassociation --cluster my-cluster --namespace default --
service-account-name s3-reader
```

Supporto dei componenti aggiuntivi EKS per le associazioni di identità dei pod

I componenti aggiuntivi EKS supportano anche la ricezione di autorizzazioni IAM tramite EKS Pod Identity Associations. Il file di configurazione espone tre campi che consentono di configurarli:, e. `addon.podIdentityAssociations` `addonsConfig.autoApplyPodIdentityAssociations` `addon.useDefaultPodIdentityAssociations` È possibile configurare in modo esplicito le associazioni di identità del pod desiderate utilizzando `addon.podIdentityAssociations` o far risolvere (e applicare) `eksctl` automaticamente la configurazione di identità del pod consigliata, utilizzando o. `addonsConfig.autoApplyPodIdentityAssociations` `addon.useDefaultPodIdentityAssociations`

> **Note**
>
> Non tutti i componenti aggiuntivi EKS supporteranno le associazioni di identità dei pod al momento del lancio. In questo caso, le autorizzazioni IAM richieste continueranno a essere fornite utilizzando le impostazioni IRSA.

Creazione di componenti aggiuntivi con autorizzazioni IAM

Quando crei un addon che richiede le autorizzazioni IAM, `eksctl` verificherà innanzitutto se le associazioni di identità dei pod o le impostazioni IRSA vengono configurate in modo esplicito come parte del file di configurazione e, in tal caso, utilizzerà una di queste per configurare le autorizzazioni per l'addon, ad es.

```
addons:
- name: vpc-cni
  podIdentityAssociations:
  - serviceAccountName: aws-node
    permissionPolicyARNs: ["arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"]
```

ed esegui

```
eksctl create addon -f config.yaml
2024-05-13 15:38:58 [#] pod identity associations are set for "vpc-cni" addon; will use
 these to configure required IAM permissions
```

> **Note**
>
> L'impostazione contemporanea delle identità dei pod e dell'IRSA non è consentita e genererà un errore di convalida.

Per i componenti aggiuntivi EKS che supportano le identità dei pod, `eksctl` offre la possibilità di configurare automaticamente tutte le autorizzazioni IAM consigliate, al momento della creazione dell'addon. Ciò può essere ottenuto semplicemente impostando il file di `addonsConfig.autoApplyPodIdentityAssociations: true` configurazione. ad es.

```
addonsConfig:
  autoApplyPodIdentityAssociations: true
# bear in mind that if either pod identity or IRSA configuration is explicitly set in
 the config file,
# or if the addon does not support pod identities,
# addonsConfig.autoApplyPodIdentityAssociations won't have any effect.
addons:
- name: vpc-cni
```

ed esegui

```
eksctl create addon -f config.yaml
2024-05-13 15:38:58 [#] "addonsConfig.autoApplyPodIdentityAssociations" is set to true;
 will lookup recommended pod identity configuration for "vpc-cni" addon
```

Equivalentemente, lo stesso può essere fatto tramite i flag CLI, ad es.

```
eksctl create addon --cluster my-cluster --name vpc-cni --auto-apply-pod-identity-
associations
```

Per migrare un addon esistente per utilizzare l'identità del pod con le politiche IAM consigliate, usa

```
addons:
- name: vpc-cni
  useDefaultPodIdentityAssociations: true
```

```
eksctl update addon -f config.yaml
```

Aggiornamento dei componenti aggiuntivi con autorizzazioni IAM

Quando si aggiorna un componente aggiuntivo, la specificazione `addon.PodIdentityAssociations` rappresenterà l'unica fonte di verità sullo stato che dovrà avere l'addon, una volta completata l'operazione di aggiornamento. Dietro le quinte, vengono eseguiti diversi tipi di operazioni per raggiungere lo stato desiderato, ad es.

- crea identità di pod presenti nel file di configurazione, ma mancanti nel cluster
- elimina gli identità dei pod esistenti che sono stati rimossi dal file di configurazione, insieme a tutte le risorse IAM associate
- aggiorna le identità dei pod esistenti che sono presenti anche nel file di configurazione e per le quali è stato modificato il set di autorizzazioni IAM

> **Note**
>
> Il ciclo di vita delle associazioni di identità dei pod di proprietà di EKS Add-ons viene gestito direttamente dall'API EKS Addons.

Non puoi utilizzare `eksctl update podidentityassociation` (per aggiornare le autorizzazioni IAM) o `eksctl delete podidentityassociations` (per rimuovere l'associazione) per le associazioni utilizzate con un componente aggiuntivo Amazon EKS. Invece, `eksctl update addon` o `eksctl delete addon` deve essere usato.

Vediamo un esempio di quanto sopra, iniziando con l'analisi della configurazione iniziale dell'identità del pod per l'addon:

```
eksctl get podidentityassociation --cluster my-cluster --namespace opentelemetry-
operator-system --output json
[
    {
        ...
        "ServiceAccountName": "adot-col-prom-metrics",
        "RoleARN": "arn:aws:iam::111122223333:role/eksctl-my-cluster-addon-adot-
podident-Role1-JwrGA4mn1Ny8",
        # OwnerARN is populated when the pod identity lifecycle is handled by the EKS
 Addons API
        "OwnerARN": "arn:aws:eks:us-west-2:111122223333:addon/my-cluster/adot/
b2c7bb45-4090-bf34-ec78-a2298b8643f6"
    },
    {
        ...
        "ServiceAccountName": "adot-col-otlp-ingest",
        "RoleARN": "arn:aws:iam::111122223333:role/eksctl-my-cluster-addon-adot-
podident-Role1-Xc7qVg5fgCqr",
        "OwnerARN": "arn:aws:eks:us-west-2:111122223333:addon/my-cluster/adot/
b2c7bb45-4090-bf34-ec78-a2298b8643f6"
    }
]
```

Ora usa la configurazione seguente:

```
addons:
- name: adot
  podIdentityAssociations:

  # For the first association, the permissions policy of the role will be updated
  - serviceAccountName: adot-col-prom-metrics
    permissionPolicyARNs:
    #- arn:aws:iam::aws:policy/AmazonPrometheusRemoteWriteAccess
    - arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy

  # The second association will be deleted, as it's been removed from the config file
  #- serviceAccountName: adot-col-otlp-ingest
  #  permissionPolicyARNs:
  #  - arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess

  # The third association will be created, as it's been added to the config file
  - serviceAccountName: adot-col-container-logs
    permissionPolicyARNs:
```

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

ed esegui

```
eksctl update addon -f config.yaml
...
# updating the permission policy for the first association
2024-05-14 13:27:43 [#]  updating IAM resources stack "eksctl-my-cluster-addon-
adot-podidentityrole-adot-col-prom-metrics" for pod identity association "a-
reaxk2uz1iknwazwj"
2024-05-14 13:27:44 [#]  waiting for CloudFormation changeset "eksctl-opentelemetry-
operator-system-adot-col-prom-metrics-update-1715682463" for stack "eksctl-my-cluster-
addon-adot-podidentityrole-adot-col-prom-metrics"
2024-05-14 13:28:47 [#]  waiting for CloudFormation stack "eksctl-my-cluster-addon-
adot-podidentityrole-adot-col-prom-metrics"
2024-05-14 13:28:47 [#]  updated IAM resources stack "eksctl-my-cluster-addon-adot-
podidentityrole-adot-col-prom-metrics" for "a-reaxk2uz1iknwazwj"
# creating the IAM role for the second association
2024-05-14 13:28:48 [#]  deploying stack "eksctl-my-cluster-addon-adot-podidentityrole-
adot-col-container-logs"
2024-05-14 13:28:48 [#]  waiting for CloudFormation stack "eksctl-my-cluster-addon-
adot-podidentityrole-adot-col-container-logs"
2024-05-14 13:29:19 [#]  waiting for CloudFormation stack "eksctl-my-cluster-addon-
adot-podidentityrole-adot-col-container-logs"
# updating the addon, which handles the pod identity config changes behind the scenes
2024-05-14 13:29:19 [#]  updating addon
# deleting the IAM role for the third association
2024-05-14 13:29:19 [#]  deleting IAM resources for pod identity service account adot-
col-otlp-ingest
2024-05-14 13:29:20 [#]  will delete stack "eksctl-my-cluster-addon-adot-
podidentityrole-adot-col-otlp-ingest"
2024-05-14 13:29:20 [#]  waiting for stack "eksctl-my-cluster-addon-adot-
podidentityrole-adot-col-otlp-ingest" to get deleted
2024-05-14 13:29:51 [#]  waiting for CloudFormation stack "eksctl-my-cluster-addon-
adot-podidentityrole-adot-col-otlp-ingest"
2024-05-14 13:29:51 [#]  deleted IAM resources for addon adot
```

ora controlla che la configurazione dell'identità del pod sia stata aggiornata correttamente

```
eksctl get podidentityassociation --cluster my-cluster --output json
[
    {
        ...
```

```
        "ServiceAccountName": "adot-col-prom-metrics",
        "RoleARN": "arn:aws:iam::111122223333:role/eksctl-my-cluster-addon-adot-
podident-Role1-nQAlp0KktS2A",
        "OwnerARN": "arn:aws:eks:us-west-2:111122223333:addon/my-cluster/
adot/1ec7bb63-8c4e-ca0a-f947-310c4b55052e"
    },
    {
        ...
        "ServiceAccountName": "adot-col-otlp-ingest",
        "RoleARN": "arn:aws:iam::111122223333:role/eksctl-my-cluster-addon-adot-
podident-Role1-1k1XhAdziGzX",
        "OwnerARN": "arn:aws:eks:us-west-2:111122223333:addon/my-cluster/
adot/1ec7bb63-8c4e-ca0a-f947-310c4b55052e"
    }
]
```

Per rimuovere tutte le associazioni di identità dei pod da un addon, `addon.PodIdentityAssociations` devono essere impostate esplicitamente su, ad es. `[]`

```
addons:
- name: vpc-cni
  # omitting the `podIdentityAssociations` field from the config file,
  # instead of explicitly setting it to [], will result in a validation error
  podIdentityAssociations: []
```

ed esegui

```
eksctl update addon -f config.yaml
```

Eliminazione di componenti aggiuntivi con autorizzazioni IAM

L'eliminazione di un componente aggiuntivo rimuoverà anche tutte le identità del pod associate all'addon. L'eliminazione del cluster otterrà lo stesso effetto, per tutti i componenti aggiuntivi. Verranno eliminati anche tutti i ruoli IAM per le identità dei pod`eksctl`, creati da.

Migrazione degli account e dei componenti aggiuntivi iamservice esistenti alle associazioni di identità dei pod

Esiste un comando `eksctl` utils per la migrazione dei ruoli IAM esistenti per gli account di servizio alle associazioni di identità dei pod, ad es.

```
eksctl utils migrate-to-pod-identity --cluster my-cluster --approve
```

Dietro le quinte, il comando applicherà i seguenti passaggi:

- installa l'`eks-pod-identity-agent`addon se non è già attivo nel cluster
- identifica tutti i ruoli IAM associati a iamserviceaccounts
- identifica tutti i ruoli IAM associati ai componenti aggiuntivi EKS che supportano le associazioni di identità dei pod
- aggiorna la policy di fiducia IAM di tutti i ruoli identificati, con un'ulteriore entità attendibile, che indichi il nuovo responsabile del servizio EKS (e, facoltativamente, rimuovi la relazione fiduciaria esistente con il provider OIDC)
- crea associazioni di identità pod per i ruoli filtrati associati a iamserviceaccounts
- aggiorna i componenti aggiuntivi EKS con le identità dei pod (l'API EKS creerà le identità dei pod dietro le quinte)

L'esecuzione del comando senza il `--approve` flag produrrà solo un piano costituito da una serie di attività che riflettono i passaggi precedenti, ad es.

```
[#]  (plan) would migrate 2 iamserviceaccount(s) and 2 addon(s) to pod identity
 association(s) by executing the following tasks
[#]  (plan)

3 sequential tasks: { install eks-pod-identity-agent addon,
    ## tasks for migrating the addons
    2 parallel sub-tasks: {
        2 sequential sub-tasks: {
            update trust policy for owned role "eksctl-my-cluster--Role1-DDuMLoeZ8weD",
            migrate addon aws-ebs-csi-driver to pod identity,
        },
        2 sequential sub-tasks: {
            update trust policy for owned role "eksctl-my-cluster--Role1-xYiPFOVp1aeI",
            migrate addon vpc-cni to pod identity,
        },
    },
    ## tasks for migrating the iamserviceaccounts
    2 parallel sub-tasks: {
        2 sequential sub-tasks: {
            update trust policy for owned role "eksctl-my-cluster--Role1-QLXqHcq9O1AR",
            create pod identity association for service account "default/sa1",
```

```
        },
        2 sequential sub-tasks: {
            update trust policy for unowned role "Unowned-Role1",
            create pod identity association for service account "default/sa2",
        },
    }
}
[#]  all tasks were skipped
[!]  no changes were applied, run again with '--approve' to apply the changes
```

La relazione di fiducia esistente con il provider OIDC viene sempre rimossa dai ruoli IAM associati ai componenti aggiuntivi EKS. Inoltre, per rimuovere la relazione di fiducia esistente con il provider OIDC dai ruoli IAM associati a iamserviceaccounts, esegui il comando con flag, ad es. `--remove-oidc-provider-trust-relationship`

```
eksctl utils migrate-to-pod-identity --cluster my-cluster --approve --remove-oidc-
provider-trust-relationship
```

Supporto per Pod Identity su più account

eksctl supporta l'accesso a più account EKS Pod Identity. Questa funzionalità consente ai pod in esecuzione nel cluster EKS di accedere alle risorse AWS in un altro account AWS.

Utilizzo

Per creare un'associazione di identità pod con accesso su più account, configura innanzitutto IAM Roles and Policies che consentano l'accesso da un account AWS di origine (con il cluster) a un account AWS di destinazione (con le risorse a cui il cluster può accedere). Per un esempio, consulta «Amazon EKS Pod Identity semplifica l'accesso tra account».

Una volta configurato un ruolo IAM in ogni account, usa eksctl per creare le associazioni di identità dei pod:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  # The cluster name and service account name should match the target
  # account policy's trust relationship.
  name: my-cluster
  region: us-west-2
  version: "1.32"
```

```
addons:
  - name: vpc-cni
  - name: coredns
  - name: kube-proxy
  - name: eks-pod-identity-agent

iam:
  podIdentityAssociations:
  - namespace: default
    serviceAccountName: demo-app-sa
    createServiceAccount: true
    # The source role in the same account as the cluster
    roleARN: arn:aws:iam::1111111111:role/account-a-role
    # The target role in a different account
    targetRoleARN: arn:aws:iam::2222222222:role/account-b-role
    # Optional: Disable session tags
    disableSessionTags: false

managedNodeGroups:
  - name: my-cluster
    instanceType: m6a.large
    privateNetworking: true
    minSize: 2
    desiredCapacity: 2
    maxSize: 3
```

Ulteriori riferimenti

Supporto ufficiale dei componenti aggiuntivi AWS Userdocs for EKS per le identità dei pod

Post ufficiale del blog AWS sulle associazioni di identità dei pod

Userdocs ufficiali AWS per Pod Identity Associations

Opzioni di implementazione

Questo capitolo tratta l'uso di eksctl per gestire i cluster EKS distribuiti in ambienti alternativi.

Per informazioni più accurate sulle opzioni di implementazione EKS, consulta Distribuire i cluster Amazon EKS in ambienti cloud e locali nella Guida per l'utente EKS.

Argomenti:

- the section called “EKS ovunque”
  - Usa eksctl con i cluster Amazon EKS Anywhere.
  - Amazon EKS Anywhere è un software di gestione dei container creato da AWS che semplifica l'esecuzione e la gestione di Kubernetes in locale e all'edge.
- the section called “Supporto per AWS Outposts”
  - Usa eksctl con i cluster EKS su AWS Outposts.
  - AWS Outposts è una famiglia di soluzioni completamente gestite che fornisce l'infrastruttura e i servizi AWS praticamente a qualsiasi location locale o periferica per un'esperienza ibrida davvero coerente.
  - Il supporto di AWS Outposts in eksctl consente di creare cluster locali con l'intero cluster Kubernetes, inclusi il piano di controllo EKS e i nodi di lavoro, in esecuzione localmente su AWS Outposts.
- the section called “Nodi ibridi EKS”
  - Esegui applicazioni locali e periferiche su un'infrastruttura gestita dal cliente con gli stessi cluster, funzionalità e strumenti AWS EKS che usi nel cloud AWS.

EKS ovunque

`eksctl`fornisce l'accesso alla funzionalità di AWS richiamata `EKS Anywhere` con il comando sub. `eksctl anywhere` Ciò richiede il `eksctl-anywhere` file binario presente su. `PATH` Segui le istruzioni riportate qui Installa eksctl-anywhere per installarlo.

Una volta fatto, esegui i comandi ovunque eseguendo:

```
eksctl anywhere version
v0.5.0
```

Per ulteriori informazioni su EKS Anywhere, visita il sito web EKS Anywhere.

Supporto per AWS Outposts

> **Warning**
>
> I gruppi di nodi gestiti EKS non sono supportati su Outposts.

Estensione dei cluster esistenti ad AWS Outposts

Puoi estendere un cluster EKS esistente in esecuzione in una regione AWS ad AWS Outposts impostando nuovi gruppi di nodi `nodeGroup.outpostARN` per creare gruppi di nodi su Outposts, come in:

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

metadata:
  name: existing-cluster
  region: us-west-2

nodeGroups:
  # Nodegroup will be created in an AWS region.
  - name: ng

  # Nodegroup will be created on the specified Outpost.
  - name: outpost-ng
    privateNetworking: true
    outpostARN: "arn:aws:outposts:us-west-2:1234:outpost/op-1234"
```

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

In questa configurazione, il piano di controllo EKS viene eseguito in una regione AWS mentre i gruppi di nodi con `outpostARN` set vengono eseguiti sull'Outpost specificato. Quando un nodegroup viene creato su Outposts per la prima volta, eksctl estende il VPC creando sottoreti sull'Outpost specificato. Queste sottoreti vengono utilizzate per creare gruppi di nodi che sono stati impostati. `outpostARN`

I clienti con un VPC preesistente devono creare le sottoreti su Outposts e trasferirle, come in: `nodeGroup.subnets`

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

metadata:
  name: extended-cluster-vpc
  region: us-west-2

vpc:
  id: vpc-1234
    subnets:
      private:
        outpost-subnet-1:
          id: subnet-1234

nodeGroups:
  # Nodegroup will be created in an AWS region.
  - name: ng

  # Nodegroup will be created on the specified Outpost.
  - name: outpost-ng
    privateNetworking: true
    # Subnet IDs for subnets created on Outpost.
    subnets: [subnet-5678]
    outpostARN: "arn:aws:outposts:us-west-2:1234:outpost/op-1234"
```

Creazione di un cluster locale su AWS Outposts

> **Note**
>
> I cluster locali supportano solo i rack dell'Outpost.

> **Note**
>
> Solo Amazon Linux 2 è supportato per i gruppi di nodi quando il piano di controllo è su Outposts. Solo i tipi di volume EBS gp2 sono supportati per i gruppi di nodi su Outposts.

Il supporto di AWS Outposts in eksctl consente di creare cluster locali con l'intero cluster Kubernetes, inclusi il piano di controllo EKS e i nodi di lavoro, in esecuzione localmente su AWS Outposts. I clienti possono creare un cluster locale con il piano di controllo EKS e i nodi di lavoro in esecuzione localmente su AWS Outposts oppure estendere un cluster EKS esistente in esecuzione in una regione AWS ad AWS Outposts creando nodi di lavoro su Outposts.

Per creare il piano di controllo EKS e i gruppi di nodi su AWS Outposts, `outpost.controlPlaneOutpostARN` imposta Outpost ARN, come in:

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

metadata:
  name: outpost
  region: us-west-2

outpost:
  # Required.
  controlPlaneOutpostARN: "arn:aws:outposts:us-west-2:1234:outpost/op-1234"
  # Optional, defaults to the smallest available instance type on the Outpost.
  controlPlaneInstanceType: m5d.large
```

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

Questo indica a eksctl di creare il piano di controllo EKS e le sottoreti sull'Outpost specificato. Poiché un rack Outposts esiste in un'unica zona di disponibilità, eksctl crea solo una sottorete pubblica e privata. eksctl non associa il VPC creato a un gateway locale e, come tale, eksctl non avrà la connettività al server API e non sarà in grado di creare gruppi di nodi. Pertanto, se `ClusterConfig` contiene dei gruppi di nodi durante la creazione del cluster, il comando deve essere eseguito con, come in: `--without-nodegroup`

```
eksctl create cluster -f outpost.yaml --without-nodegroup
```

È responsabilità del cliente associare il VPC creato da eksctl al gateway locale dopo la creazione del cluster per abilitare la connettività al server API. Dopo questo passaggio, è possibile creare gruppi di nodi utilizzando. `eksctl create nodegroup`

È possibile specificare facoltativamente il tipo di istanza per i nodi del piano di controllo in `outpost.controlPlaneInstanceType` o per i gruppi di nodi in`nodeGroup.instanceType`, ma il tipo di istanza deve esistere su Outpost o eksctl restituirà un errore. Per impostazione predefinita, eksctl tenta di scegliere il tipo di istanza più piccolo disponibile su Outpost per i nodi e i gruppi di nodi del piano di controllo.

Quando il piano di controllo è su Outposts, i gruppi di nodi vengono creati su quell'Outpost. Facoltativamente, è possibile specificare l'ARN Outpost per il gruppo di nodi in, `nodeGroup.outpostARN` ma deve corrispondere all'ARN Outpost del piano di controllo.

```
# outpost-fully-private.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: outpost-fully-private
  region: us-west-2

privateCluster:
  enabled: true

outpost:
  # Required.
  controlPlaneOutpostARN: "arn:aws:outposts:us-west-2:1234:outpost/op-1234"
  # Optional, defaults to the smallest available instance type on the Outpost.
  controlPlaneInstanceType: m5d.large
```

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

metadata:
```

```
  name: outpost
  region: us-west-2

outpost:
  # Required.
  controlPlaneOutpostARN: "arn:aws:outposts:us-west-2:1234:outpost/op-1234"
  # Optional, defaults to the smallest available instance type on the Outpost.
  controlPlaneInstanceType: m5d.large

  controlPlanePlacement:
    groupName: placement-group-name
```

VPC esistente

I clienti con un VPC esistente possono creare cluster locali su AWS Outposts specificando la configurazione della sottorete in, come in: `vpc.subnets`

```
# outpost-existing-vpc.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: outpost
  region: us-west-2

vpc:
  id: vpc-1234
  subnets:
    private:
      outpost-subnet-1:
        id: subnet-1234

nodeGroups:
  - name: outpost-ng
    privateNetworking: true

outpost:
    # Required.
    controlPlaneOutpostARN: "arn:aws:outposts:us-west-2:1234:outpost/op-1234"
    # Optional, defaults to the smallest available instance type on the Outpost.
    controlPlaneInstanceType: m5d.large
```

```
eksctl create cluster -f outpost-existing-vpc.yaml
```

Le sottoreti devono esistere nell'Outpost specificato in o eksctl restituirà un errore. `outpost.controlPlaneOutpostARN` Puoi anche specificare i gruppi di nodi durante la creazione del cluster se hai accesso al gateway locale per la sottorete o disponi di connettività alle risorse VPC.

Funzionalità non supportate nei cluster locali

- Componenti aggiuntivi
- Ruoli IAM per gli account di servizio
- IPv6
- Fornitori di identità
- Fargate
- Crittografia KMS
- Zone locali
- Karpenter
- Selettore di istanze
- Le zone di disponibilità non possono essere specificate in quanto l'impostazione predefinita è la zona di disponibilità Outpost.
- `vpc.publicAccessCIDRs` e `vpc.autoAllocateIPv6` non sono supportate.
- L'accesso pubblico da endpoint al server API non è supportato in quanto un cluster locale può essere creato solo con un accesso endpoint riservato agli endpoint.

Ulteriori informazioni

- Amazon EKS su AWS Outposts
- Cluster locali per Amazon EKS su AWS Outposts
- Creazione di cluster locali
- Avvio di nodi Amazon Linux autogestiti su un Outpost

Sicurezza

`eksctl`fornisce alcune opzioni che possono migliorare la sicurezza del cluster EKS.

`withOIDC`

Abilita `withOIDC`la creazione automatica di un IRSA per il plug-in Amazon CNI e limita le autorizzazioni concesse ai nodi del cluster, concedendo invece le autorizzazioni necessarie solo all'account del servizio CNI.

Il background è descritto in questa documentazione AWS.

`disablePodIMDS`

Per i gruppi di nodi gestiti e non gestiti, è disponibile `disablePodIMDS`l'opzione che impedisce a tutti i pod di rete non host in esecuzione in questo gruppo di nodi di effettuare richieste IMDS.

> **Note**
>
> Non può essere usato insieme a. `withAddonPolicies`

Crittografia KMS Envelope per cluster EKS

> **Note**
>
> 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. Per ulteriori informazioni, consulta Crittografia a busta predefinita per tutti i dati dell'API Kubernetes nella Guida per l'utente EKS.

EKS supporta l'utilizzo di chiavi AWS KMS per fornire la crittografia in busta dei segreti Kubernetes archiviati in EKS. La crittografia Envelope aggiunge un ulteriore livello di crittografia gestito dal cliente per i segreti delle applicazioni o i dati utente archiviati all'interno di un cluster Kubernetes.

In precedenza, Amazon EKS supportava l'abilitazione della crittografia delle buste utilizzando chiavi KMS solo durante la creazione del cluster. Ora puoi abilitare la crittografia delle buste per i cluster Amazon EKS in qualsiasi momento.

Scopri di più sull'utilizzo del supporto del provider di crittografia EKS per un defense-in-depth post sul blog sui contenitori AWS.

Creazione di un cluster con crittografia KMS abilitata

```
# kms-cluster.yaml
# A cluster with KMS encryption enabled
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: kms-cluster
  region: us-west-2

managedNodeGroups:
- name: ng
# more config

secretsEncryption:
  # KMS key used for envelope encryption of Kubernetes secrets
  keyARN: arn:aws:kms:us-west-2:<account>:key/<key>
```

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

Abilitazione della crittografia KMS su un cluster esistente

Per abilitare la crittografia KMS su un cluster che non l'ha già abilitata, esegui

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

o senza un file di configurazione:

```
eksctl utils enable-secrets-encryption --cluster=kms-cluster --key-arn=arn:aws:kms:us-
west-2:<account>:key/<key> --region=<region>
```

Oltre ad abilitare la crittografia KMS sul cluster EKS, eksctl cripta nuovamente tutti i segreti Kubernetes esistenti utilizzando la nuova chiave KMS aggiornandoli con l'annotazione. `eksctl.io/kms-encryption-timestamp` Questo comportamento può essere disabilitato passando, ad esempio: `--encrypt-existing-secrets=false`

```
eksctl utils enable-secrets-encryption --cluster=kms-cluster --key-arn=arn:aws:kms:us-west-2:<account>:key/<key> --encrypt-existing-secrets=false --region=<region>
```

Se un cluster ha già la crittografia KMS abilitata, eksctl procederà a ricrittografare tutti i segreti esistenti.

> **Note**
>
> Una volta abilitata, la crittografia KMS non può essere disabilitata o aggiornata per utilizzare una chiave KMS diversa.

Risoluzione dei problemi

Questo argomento include istruzioni su come risolvere gli errori più comuni con Eksctl.

Creazione dello stack non riuscita

Puoi usare il `--cfn-disable-rollback` flag per impedire a Cloudformation di ripristinare gli stack falliti per semplificare il debug.

L'ID di sottorete «subnet-" non è lo stesso di «subnet-22222222"

Dato un file di configurazione che specifica le sottoreti per un VPC come il seguente:

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

metadata:
  name: test
  region: us-east-1

vpc:
  subnets:
    public:
      us-east-1a: {id: subnet-11111111}
      us-east-1b: {id: subnet-22222222}
    private:
      us-east-1a: {id: subnet-33333333}
      us-east-1b: {id: subnet-44444444}

nodeGroups: []
```

Un errore `subnet ID "subnet-11111111" is not the same as "subnet-22222222"` indica che le sottoreti specificate non sono posizionate nella zona di disponibilità corretta. Controlla nella console AWS qual è l'ID di sottorete corretto per ogni zona di disponibilità.

In questo esempio, la configurazione corretta per il VPC sarebbe:

```
vpc:
  subnets:
```

```
    public:
      us-east-1a: {id: subnet-22222222}
      us-east-1b: {id: subnet-11111111}
    private:
      us-east-1a: {id: subnet-33333333}
      us-east-1b: {id: subnet-44444444}
```

Problemi di eliminazione

Se l'eliminazione non funziona o dimentichi di aggiungere `--wait` l'eliminazione, potresti dover utilizzare gli altri strumenti di Amazon per eliminare gli stack di cloudformation. Ciò può essere fatto tramite la gui o con la cli di aws.

kubectl si registra e kubectl run fallisce con un errore di autorizzazione

Se i tuoi nodi sono distribuiti in una sottorete privata `kubectl logs` e/o `kubectl run` falliscono con un errore come il seguente:

```
Error attaching, falling back to logs: unable to upgrade connection: Authorization
 error (user=kube-apiserver-kubelet-client, verb=create, resource=nodes,
 subresource=proxy)
```

```
Error from server (InternalError): Internal error occurred: Authorization error
 (user=kube-apiserver-kubelet-client, verb=get, resource=nodes, subresource=proxy)
```

Quindi potrebbe essere necessario impostare. enableDnsHostnames Maggiori dettagli sono disponibili in questo numero.

Annunci

Questo argomento riguarda gli annunci precedenti di nuove funzionalità di Eksctl.

Gruppi di nodi gestiti (impostazione predefinita)

A partire da eksctl v0.58.0, eksctl crea gruppi di nodi gestiti per impostazione predefinita quando un file non è specificato per and. `ClusterConfig` `eksctl create cluster` `eksctl create nodegroup` Per creare un gruppo di nodi `--managed=false` autogestito, pass. Ciò potrebbe interrompere gli script che non utilizzano un file di configurazione se viene utilizzata una funzionalità non supportata nei gruppi di nodi gestiti, ad esempio i gruppi di nodi di Windows. Per risolvere questo problema`--managed=false`, passa o specifica la configurazione del tuo nodegroup in un `ClusterConfig` file usando il campo che crea un gruppo di nodi autogestito. `nodeGroups`

Nodegroup Bootstrap Override For Custom AMIs

Questa modifica è stata annunciata nel numero Breaking: soon... overrideBootstrapCommand Ora, è successo in questo PR. Leggi attentamente il numero allegato sul motivo per cui abbiamo deciso di abbandonare il supporto di script personalizzati AMIs senza bootstrap o con script bootstrap parziali.

Forniamo comunque un aiuto! Speriamo che migrare non sia così doloroso. `eksctl`fornisce comunque uno script che, una volta fornito, esporterà un paio di utili proprietà e impostazioni dell'ambiente. Questo script si trova qui.

Le seguenti proprietà ambientali saranno a tua disposizione:

```
API_SERVER_URL
B64_CLUSTER_CA
INSTANCE_ID
INSTANCE_LIFECYCLE
CLUSTER_DNS
NODE_TAINTS
MAX_PODS
NODE_LABELS
CLUSTER_NAME
CONTAINER_RUNTIME # default is docker
KUBELET_EXTRA_ARGS # for details, look at the script
```

Il minimo che deve essere usato quando si esegue l'override, quindi `eksctl` non fallisce, sono le etichette! `eksctl`si affida a un set specifico di etichette da inserire nel nodo, in modo che possa trovarle. Quando definisci l'override, fornisci questo comando di override minimo indispensabile:

```
    overrideBootstrapCommand: |
      #!/bin/bash

      source /var/lib/cloud/scripts/eksctl/bootstrap.helper.sh

      # Note "--node-labels=${NODE_LABELS}" needs the above helper sourced to work,
 otherwise will have to be defined manually.
      /etc/eks/bootstrap.sh ${CLUSTER_NAME} --container-runtime containerd --kubelet-
extra-args "--node-labels=${NODE_LABELS}"
```

Per i gruppi di nodi che non dispongono di accesso a Internet in uscita, è necessario fornire `--apiserver-endpoint` e inviare lo script bootstrap nel `--b64-cluster-ca` modo seguente:

```
    overrideBootstrapCommand: |
      #!/bin/bash

      source /var/lib/cloud/scripts/eksctl/bootstrap.helper.sh

      # Note "--node-labels=${NODE_LABELS}" needs the above helper sourced to work,
 otherwise will have to be defined manually.
      /etc/eks/bootstrap.sh ${CLUSTER_NAME} --container-runtime containerd --kubelet-
extra-args "--node-labels=${NODE_LABELS}" \
        --apiserver-endpoint ${API_SERVER_URL} --b64-cluster-ca ${B64_CLUSTER_CA}
```

Nota l'impostazione `--node-labels`. Se questo valore non è definito, il nodo entrerà a far parte del cluster, ma alla fine `eksctl` scadrà durante l'ultimo passaggio, quando attende che arrivino i nodi. `Ready` Sta eseguendo una ricerca su Kubernetes per i nodi che hanno l'etichetta. `alpha.eksctl.io/nodegroup-name=<cluster-name>` Questo vale solo per i gruppi di nodi non gestiti. Per gestito utilizza un'etichetta diversa.

Se è possibile passare a gruppi di nodi gestiti per evitare questo sovraccarico, è giunto il momento di farlo. Rende molto più semplice l'override.

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.