Amazon EKS

Eksctl-Benutzerhandbuch

Eksctl-Benutzerhandbuch: Amazon EKS

Table of Contents

Was ist Eksctl?

eksctl ist ein Befehlszeilen-Hilfsprogramm, das den Prozess der Erstellung, Verwaltung und des Betriebs von Amazon Elastic Kubernetes Service (Amazon EKS) -Clustern automatisiert und vereinfacht. eksctl wurde in Go geschrieben und bietet eine deklarative Syntax über YAML-Konfigurationen und CLI-Befehle, um komplexe EKS-Cluster-Operationen zu handhaben, für die andernfalls mehrere manuelle Schritte in verschiedenen AWS-Services erforderlich wären.

eksctl ist besonders nützlich für DevOps Ingenieure, Plattformteams und Kubernetes-Administratoren, die EKS-Cluster konsistent und in großem Umfang bereitstellen und verwalten müssen. Es ist besonders nützlich für Unternehmen, die von selbstverwaltetem Kubernetes zu EKS wechseln oder solche, die Infrastructure-as-Code-Praktiken (IaC) implementieren, da es in bestehende Pipelines und Automatisierungsworkflows integriert werden kann. CI/CD Das Tool abstrahiert viele der komplexen Interaktionen zwischen AWS-Services, die für die Einrichtung eines EKS-Clusters erforderlich sind, wie z. B. die VPC-Konfiguration, die Erstellung von IAM-Rollen und die Verwaltung von Sicherheitsgruppen.

Zu den wichtigsten Funktionen von eksctl gehören die Möglichkeit, voll funktionsfähige EKS-Cluster mit einem einzigen Befehl zu erstellen, Unterstützung für benutzerdefinierte Netzwerkkonfigurationen, automatisiertes Knotengruppenmanagement und Workflow-Integration. GitOps Das Tool verwaltet Cluster-Upgrades, skaliert Knotengruppen und wickelt das Add-On-Management über einen deklarativen Ansatz ab. eksctl bietet auch erweiterte Funktionen wie die Fargate-Profilkonfiguration, die Anpassung verwalteter Knotengruppen und die Spot-Instance-Integration und gewährleistet gleichzeitig die Kompatibilität mit anderen AWS-Tools und -Services durch die native AWS-SDK-Integration.

Features

Die Funktionen, die derzeit implementiert sind, sind:

- Cluster erstellen, abrufen, auflisten und löschen
- Knotengruppen erstellen, entleeren und löschen
- Skalieren Sie eine Knotengruppe
- Aktualisieren eines -Clusters
- Benutzerdefiniert verwenden AMIs

- VPC-Netzwerke konfigurieren
- Konfigurieren Sie den Zugriff auf API-Endpunkte
- Support für GPU-Knotengruppen
- Spot-Instances und gemischte Instances
- IAM-Management und Zusatzrichtlinien
- Listet die Cloudformation-Stacks der Cluster auf
- Installieren Sie Coredns
- Schreiben Sie die kubeconfig-Datei für einen Cluster

Häufig gestellte Fragen zu Eksctl

Allgemeines

Kann ich **`eksctl`** damit Cluster verwalten, die nicht von erstellt wurden? **`eksctl`**

Ja! Ab der Version können `0.40.0` Sie für `eksctl` jeden Cluster arbeiten, unabhängig davon, ob er von `eksctl` oder nicht erstellt wurde. Weitere Informationen finden Sie unter the section called “Nicht von EKSCTL erstellte Cluster”.

Knotengruppen

Wie kann ich den Instanztyp meiner Nodegroup ändern?

Aus der Sicht von sind Knotengruppen `eksctl` unveränderlich. Das bedeutet, dass nach der Erstellung nur noch die `eksctl` Knotengruppe nach oben oder unten skaliert werden kann.

Um den Instanztyp zu ändern, erstellen Sie eine neue Knotengruppe mit dem gewünschten Instanztyp und entleeren Sie sie dann, sodass die Workloads auf die neue verschoben werden. Nachdem dieser Schritt abgeschlossen ist, können Sie die alte Knotengruppe löschen.

Wie kann ich die generierten Benutzerdaten für eine Knotengruppe sehen?

Zunächst benötigen Sie den Namen des Cloudformation-Stacks, der die Knotengruppe verwaltet:

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

Sie werden einen Namen sehen, der ähnlich ist wie. `eksctl-CLUSTER_NAME-nodegroup-NODEGROUP_NAME`

Sie können Folgendes ausführen, um die Benutzerdaten abzurufen. Beachten Sie die letzte Zeile, die aus Base64 dekodiert und die gezippten Daten dekomprimiert.

```
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
```

Ingress

Wie richte ich Ingress mit ein? **`eksctl`**

Wir empfehlen die Verwendung des AWS Load Balancer Controllers. Die Dokumentation zur Bereitstellung des Controllers in Ihrem Cluster sowie zur Migration vom alten ALB Ingress Controller finden Sie hier.

Für den Nginx Ingress Controller wäre die Einrichtung dieselbe wie bei jedem anderen Kubernetes-Cluster.

Kubectl

Ich verwende einen HTTPS-Proxy und die Validierung des Cluster-Zertifikats schlägt fehl. Wie kann ich das System verwenden? CAs

Stellen Sie die Umgebungsvariable so `KUBECONFIG_USE_SYSTEM_CA` ein, dass sie die Zertifizierungsstellen des Systems `kubeconfig` respektiert.

Trockenlauf

Mit der Dry-Run-Funktion können Sie die Instanzen überprüfen und ändern, denen die Instanzauswahl entspricht, bevor Sie mit der Erstellung einer Knotengruppe fortfahren.

Wenn mit den Instanzauswahloptionen und aufgerufen `eksctl create cluster` wird`--dry-run`, gibt eksctl eine ClusterConfig Datei aus, die eine Knotengruppe enthält, die die CLI-Optionen und die Instanztypen darstellt, die für die Instanzen festgelegt sind, denen die Ressourcenkriterien für die Instanzauswahl entsprechen.

```
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
```

Die generierte ClusterConfig Datei kann dann übergeben werden an: `eksctl create cluster`

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

Wenn eine ClusterConfig Datei mit übergeben wird`--dry-run`, gibt eksctl eine ClusterConfig Datei aus, die die in der Datei festgelegten Werte enthält.

Einmalige Optionen in eksctl

Es gibt bestimmte einmalige Optionen, die in der `ClusterConfig` Datei nicht dargestellt werden können, z. B. `--install-vpc-controllers`

Es wird erwartet, dass:

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

gefolgt von:

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

wäre gleichbedeutend mit der Ausführung des ersten Befehls ohne`--dry-run`.

eksctl verbietet daher die Übergabe von Optionen, die nicht in der Konfigurationsdatei dargestellt werden können, wenn sie `--dry-run` übergeben werden.

> **Important**
>
> Wenn Sie ein AWS-Profil übergeben müssen, legen Sie die `AWS_PROFILE` Umgebungsvariable fest, anstatt die `--profile` CLI-Option zu übergeben.

Tutorial

In diesem Thema erfahren Sie, wie Sie eksctl installieren und konfigurieren und anschließend damit einen Amazon EKS-Cluster erstellen.

Schritt 1: Installieren Sie eksctl

Gehen Sie wie folgt vor, um die neueste Version von eksctl herunterzuladen und auf Ihrem Linux- oder macOS-Gerät zu installieren:

Um eksctl mit Homebrew zu installieren

1. (Voraussetzung) Installieren Sie Homebrew.
2. Fügen Sie den AWS-Tap hinzu:

```
brew tap aws/tap
```

3. Installieren Sie eksctl

```
brew install eksctl
```

Bevor Sie eksctl verwenden, führen Sie die folgenden Konfigurationsschritte durch:

1. Voraussetzungen für die Installation:
   - Installieren Sie AWS CLI Version 2.x oder höher.
   - Installieren Sie kubectl mit Homebrew:

     ```
     brew install kubernetes-cli
     ```

2. Konfigurieren Sie AWS-Anmeldeinformationen in Ihrer Umgebung:

```
aws configure
```

3. Überprüfen Sie die AWS-CLI-Konfiguration:

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

Schritt 2: Cluster-Konfigurationsdatei erstellen

Erstellen Sie mit diesen Schritten eine Cluster-Konfigurationsdatei:

1. Erstellen Sie eine neue Datei mit dem Namen`cluster.yaml`:

```
touch cluster.yaml
```

2. Fügen Sie die folgende grundlegende Clusterkonfiguration hinzu:

```
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. Passen Sie die Konfiguration an:
   - Aktualisieren Sie das`region`, damit es Ihrer gewünschten AWS-Region entspricht.
   - Ändern Sie das `instanceType` basierend auf Ihren Workload-Anforderungen.
   - Passen Sie das `desiredCapacityminSize`, und `maxSize` an Ihre Bedürfnisse an.
4. Überprüfen Sie die Konfigurationsdatei:

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

Schritt 3: Cluster erstellen

Gehen Sie wie folgt vor, um Ihren EKS-Cluster zu erstellen:

1. Erstellen Sie den Cluster mithilfe der Konfigurationsdatei:

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

2. Warten Sie auf die Clustererstellung (dies dauert normalerweise 15 bis 20 Minuten).
3. Überprüfen Sie die Clustererstellung:

```
eksctl get cluster
```

4. Konfigurieren Sie kubectl für die Verwendung Ihres neuen Clusters:

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

5. Überprüfen Sie die Cluster-Konnektivität:

```
kubectl get nodes
```

Ihr Cluster ist jetzt einsatzbereit.

Optional: Cluster löschen

Denken Sie daran, den Cluster zu löschen, wenn Sie fertig sind, um unnötige Gebühren zu vermeiden:

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

> **Note**
>
> Für die Clustererstellung können AWS-Gebühren anfallen. Lesen Sie unbedingt die Amazon EKS-Preise, bevor Sie einen Cluster erstellen.

Nächste Schritte

- Konfigurieren Sie Kubectl für die Verbindung mit dem Cluster
- Stellen Sie eine Beispiel-App bereit

Installationsoptionen für Eksctl

`eksctl`kann in offiziellen Versionen installiert werden, wie unten beschrieben. Wir empfehlen, dass Sie nur `eksctl` von den offiziellen GitHub Versionen aus installieren. Sie können sich dafür entscheiden, ein Installationsprogramm eines Drittanbieters zu verwenden. Beachten Sie jedoch, dass AWS diese Installationsmethoden weder unterstützt noch unterstützt. Verwenden Sie sie nach eigenem Ermessen.

Voraussetzung

Sie müssen AWS-API-Anmeldeinformationen konfiguriert haben. Was für AWS CLI oder andere Tools (Kops, Terraform usw.) funktioniert, sollte ausreichen. Sie können `~/.aws/credentials`Datei- oder Umgebungsvariablen verwenden. Weitere Informationen finden Sie in der AWS-CLI-Referenz.

Sie benötigen außerdem den Befehl AWS IAM Authenticator for Kubernetes (entweder `aws-iam-authenticator` oder `aws eks get-token` (verfügbar in Version 1.16.156 oder höher von AWS CLI) in Ihrem. `PATH`

Das für die Erstellung des EKS-Clusters verwendete IAM-Konto sollte über diese minimalen Zugriffsebenen verfügen.

AWS-Service	Zugriffsebene
CloudFormation	Voller Zugriff
EC2	Vollständig: Eingeschränkte Kennzeichnung: Auflisten, Lesen, Schreiben
EC2 Auto Scaling	Eingeschränkt: Auflisten, Schreiben
EKS	Voller Zugriff
IAM	Eingeschränkt: Auflisten, Lesen, Schreiben, Rechteverwaltung
Systems Manager	Limited: List, Read

Für Unix

Um die neueste Version herunterzuladen, führen Sie folgenden Befehl aus:

```
# 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
```

Für Windows

Direkter Download (neueste Version):

- AMD64/x86_64
- ARMv6
- ARMv7
- ARM64

Stellen Sie sicher, dass Sie das Archiv in einen Ordner in der Variablen entpacken. `PATH`

Überprüfen Sie optional die Prüfsumme:

1. Laden Sie die Prüfsummendatei herunter: neueste
2. Verwenden Sie die Befehlszeile, um die Ausgabe manuell mit `CertUtil` der heruntergeladenen Prüfsummendatei zu vergleichen.

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

3. Wird verwendet PowerShell , um die Überprüfung mithilfe des `-eq` Operators zu automatisieren, um ein `True` `False` Oder-Ergebnis zu erhalten:

```
# 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]
```

Mit 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
```

Die `eksctl` ausführbare Datei wird eingefügt`$HOME/bin`, die `$PATH` von Git Bash stammt.

Homebrew

Sie können Homebrew verwenden, um Software auf macOS und Linux zu installieren.

AWS unterhält einen Homebrew-Tap einschließlich eksctl.

Weitere Informationen zum Homebrew-Tap finden Sie im Projekt auf Github und in der Homebrew-Formel für eksctl.

Um eksctl mit Homebrew zu installieren

1. (Voraussetzung) Installieren Sie Homebrew
2. Fügen Sie den AWS-Tap hinzu

```
brew tap aws/tap
```

3. Installieren Sie eksctl

```
brew install eksctl
```

Docker

Für jede Version und jeden RC wird ein Container-Image in das ECR-Repository übertragen. `public.ecr.aws/eksctl/eksctl` Erfahren Sie mehr über die Verwendung auf ECR Public Gallery - eksctl. Zum Beispiel

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

Fertigstellung der Shell

Bash

Um die Bash-Vervollständigung zu aktivieren, führen Sie den folgenden Befehl aus oder geben Sie ihn in `~/.bashrc` oder `~/.profile` ein:

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

Zsh

Um Zsh abzuschließen, führen Sie bitte folgenden Befehl aus:

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

und geben Sie Folgendes ein: `~/.zshrc`

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

Beachten Sie, wenn Sie keine Distribution wie diese ausführen, müssen oh-my-zsh Sie möglicherweise zuerst die Autovervollständigung aktivieren (und einfügen`~/.zshrc`, um sie dauerhaft zu machen):

```
autoload -U compinit
compinit
```

Fische

Die folgenden Befehle können für die auto Vervollständigung von fish verwendet werden:

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

Powershell

Zum Einrichten kann auf den folgenden Befehl verwiesen werden. Bitte beachten Sie, dass der Pfad je nach Ihren Systemeinstellungen unterschiedlich sein kann.

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

Aktualisierungen

> **Important**
>
> Wenn Sie eksctl installieren, indem Sie es direkt herunterladen (ohne einen Paketmanager zu verwenden), müssen Sie es manuell aktualisieren.

Cluster

Dieses Kapitel behandelt die Erstellung und Konfiguration von EKS-Clustern mit eksctl. Es enthält auch Add-Ons und den EKS-Automatikmodus.

Themen:

- the section called “EKS-Zugangseinträge”
  - Vereinfachen Sie das Kubernetes-RBAC-Management, indem Sie aws-auth durch EKS-Zugriffseinträge ConfigMap ersetzen
  - Migrieren Sie bestehende IAM-Identitätszuordnungen von aws-auth zu Access-Einträgen ConfigMap
  - Konfigurieren Sie die Cluster-Authentifizierungsmodi und kontrollieren Sie die Administratorrechte für den Cluster-Ersteller
- the section called “Standard-Add-On-Updates”
  - Sorgen Sie für die Sicherheit Ihrer Cluster, indem Sie standardmäßige EKS-Add-Ons auf älteren Clustern aktualisieren
- the section called “Erweiterungen”
  - Automatisieren Sie Routineaufgaben für die Installation, Aktualisierung und Deinstallation von Add-Ons.
  - Zu den Amazon EKS-Add-Ons gehören AWS-Add-Ons, Open-Source-Community-Add-Ons und Marketplace-Add-Ons.
- the section called “EKS-Automatikmodus”
  - Reduzieren Sie die Betriebskosten, indem Sie AWS Ihre EKS-Infrastruktur verwalten lassen
  - Konfigurieren Sie benutzerdefinierte Knotenpools anstelle von standardmäßigen Allzweck- und Systempools
  - Konvertieren Sie vorhandene EKS-Cluster zur Verwendung des automatischen Modus
- the section called “CloudWatch Protokollierung”
  - Beheben Sie Clusterprobleme, indem Sie Protokolle für bestimmte Komponenten der EKS-Steuerungsebene aktivieren
  - Konfigurieren Sie die Aufbewahrungsfristen für EKS-Clusterprotokolle
  - Ändern Sie die vorhandenen Cluster-Protokollierungseinstellungen mithilfe von eksctl-Befehlen

- the section called “Cluster-Upgrades”
  - Sorgen Sie für Sicherheit und Stabilität, indem Sie die Versionen der EKS-Steuerebene sicher aktualisieren
  - Führen Sie Upgrades über Knotengruppen hinweg durch, indem Sie alte Gruppen durch neue ersetzen
  - Aktualisieren Sie die Standard-Cluster-Add-ons
- the section called “Erstellen und Verwalten von Clustern”
  - Beginnen Sie schnell mit grundlegenden EKS-Clustern unter Verwendung von verwalteten Standardknotengruppen
  - Erstellen Sie benutzerdefinierte Cluster mithilfe von Konfigurationsdateien mit spezifischen Konfigurationen
  - Stellen Sie Cluster in bestehenden Netzwerken VPCs mit privaten Netzwerken und benutzerdefinierten IAM-Richtlinien bereit
- the section called “Kubelet konfigurieren”
  - Beugen Sie dem Mangel an Knotenressourcen vor, indem Sie Kubelet- und System-Daemon-Reservierungen konfigurieren
  - Passen Sie die Schwellenwerte für die Entfernung an die Verfügbarkeit von Speicher und Dateisystem an
  - Aktivieren oder deaktivieren Sie bestimmte Kubelet-Feature-Gates zwischen Knotengruppen
- the section called “EKS-Anschluss”
  - Zentralisieren Sie die Verwaltung hybrider Kubernetes-Bereitstellungen über die EKS-Konsole
  - Konfigurieren Sie IAM-Rollen und -Berechtigungen für den externen Clusterzugriff
  - Externe Cluster entfernen und zugehörige AWS-Ressourcen bereinigen
- the section called “Vollständig privater EKS-Cluster”
  - Erfüllen Sie Sicherheitsanforderungen mit vollständig privaten EKS-Clustern ohne ausgehenden Internetzugang
  - Konfigurieren Sie den privaten Zugriff auf AWS-Services über VPC-Endpunkte
  - Erstellen und verwalten Sie private Knotengruppen mit expliziten Netzwerkeinstellungen
- the section called “Carpenter Support”
  - Automatisieren Sie die Bereitstellung von Knoten
  - Erstellen Sie benutzerdefinierte Karpenter Provisioner-Konfigurationen
  - Richten Sie Karpenter mit der Behandlung von Spot-Instance-Unterbrechungen ein

- the section called “Amazon EMR”
  - Erstellen Sie eine IAM-Identitätszuordnung zwischen EMR und EKS-Cluster
- the section called “EKS Fargate-Unterstützung”
  - Definieren Sie benutzerdefinierte Fargate-Profile für die Pod-Planung
  - Verwaltung von Fargate-Profilen durch Erstellung und Konfigurationsupdates
- the section called “Nicht von EKSCTL erstellte Cluster”
  - Standardisieren Sie die Verwaltung von Clustern, die außerhalb von eksctl erstellt wurden
  - Verwenden Sie eksctl-Befehle auf vorhandenen Nicht-Eksctl-Clustern
- the section called “Aktivieren Sie Zonal Shift”
  - Verbessern Sie die Anwendungsverfügbarkeit, indem Sie schnelle Zonen-Failover-Funktionen aktivieren
  - Konfigurieren Sie Zonal Shift für neue EKS-Cluster-Bereitstellungen
  - Aktivieren Sie Zonal Shift-Funktionen auf vorhandenen EKS-Clustern

Erstellen und Verwalten von Clustern

In diesem Thema wird beschrieben, wie EKS-Cluster mithilfe von Eksctl erstellt und gelöscht werden. Sie können Cluster mit einem CLI-Befehl oder durch Erstellen einer YAML-Datei für die Clusterkonfiguration erstellen.

Einen einfachen Cluster erstellen

Erstellen Sie einen einfachen Cluster mit dem folgenden Befehl:

```
eksctl create cluster
```

Dadurch wird ein EKS-Cluster in Ihrer Standardregion (wie in Ihrer AWS-CLI-Konfiguration angegeben) mit einer verwalteten Knotengruppe erstellt, die zwei m5.large-Knoten enthält.

eksctl erstellt jetzt standardmäßig eine verwaltete Knotengruppe, wenn keine Konfigurationsdatei verwendet wird. Um eine selbstverwaltete Knotengruppe zu erstellen, übergeben Sie an oder. `--managed=false` `eksctl create cluster` `eksctl create nodegroup`

Überlegungen

- Beim Erstellen von Clustern in stoßen `us-east-1` Sie möglicherweise auf einen. `UnsupportedAavailabilityZoneException` In diesem Fall kopieren Sie die vorgeschlagenen Zonen und geben Sie die `--zones` Markierung weiter, zum Beispiel:`eksctl create cluster --region=us-east-1 --zones=us-east-1a,us-east-1b,us-east-1d`. Dieses Problem kann in anderen Regionen auftreten, ist aber seltener. In den meisten Fällen müssen Sie die `--zone` Flagge nicht verwenden.

Erstellen Sie den Cluster mithilfe der Konfigurationsdatei

Sie können einen Cluster mit einer Konfigurationsdatei anstelle von Flags erstellen.

Erstellen Sie zunächst eine `cluster.yaml` Datei:

```
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
```

Führen Sie als Nächstes diesen Befehl aus:

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

Dadurch wird wie beschrieben ein Cluster erstellt.

Wenn Sie eine vorhandene VPC verwenden müssen, können Sie eine Konfigurationsdatei wie diese verwenden:

```
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
```

Der Clustername oder der Knotengruppenname darf nur alphanumerische Zeichen (Groß- und Kleinschreibung beachten) und Bindestriche enthalten. Er muss mit einem alphabetischen Zeichen beginnen und darf 128 Zeichen nicht überschreiten. Andernfalls erhalten Sie einen Validierungsfehler. Weitere Informationen finden Sie unter Erstellen eines Stacks von der CloudFormation Konsole aus im AWS CLoud Formation-Benutzerhandbuch.

Aktualisieren Sie kubeconfig für den neuen Cluster

Nachdem der Cluster erstellt wurde, wird die entsprechende Kubernetes-Konfiguration zu Ihrer kubeconfig-Datei hinzugefügt. Dies ist die Datei, die Sie in der Umgebungsvariablen `KUBECONFIG`

oder standardmäßig konfiguriert haben. `~/.kube/config` Der Pfad zur kubeconfig-Datei kann mit dem Flag überschrieben werden. `--kubeconfig`

Andere Flags, die ändern können, wie die kubeconfig-Datei geschrieben wird:

Flag	Typ	use	Standardwert
--kubeconfig	Zeichenfolge	Pfad zum Schreiben von kubeconfig (nicht kompatibel mit --auto-kubeconfig)	$KUBECONFIG oder ~/.kube/config
--set-kubeconfig-context	bool	wenn wahr, wird der aktuelle Kontext in kubeconfig gesetzt; wenn ein Kontext bereits gesetzt ist, wird er überschrieben	true
--auto-kube-Konfiguration	bool	speichert die kubeconfig-Datei nach dem Clusternamen	true
--write-kubeconfig	bool	schaltet das Schreiben von kubeconfig um	true

Cluster löschen

Um diesen Cluster zu löschen, führen Sie folgenden Befehl aus:

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

> **Warning**
>
> Verwenden Sie das `--wait` Kennzeichen bei Löschvorgängen, um sicherzustellen, dass Löschfehler ordnungsgemäß gemeldet werden.

Ohne das `--wait` Flag führt eksctl nur einen Löschvorgang für den CloudFormation Stack des Clusters durch und wartet nicht auf dessen Löschung. In einigen Fällen können AWS-Ressourcen, die den Cluster oder seine VPC verwenden, dazu führen, dass das Löschen des Clusters fehlschlägt. Wenn Ihr Löschvorgang fehlschlägt oder Sie das Warte-Flag vergessen, müssen Sie möglicherweise zur CloudFormation GUI gehen und die EKS-Stacks von dort löschen.

> **Warning**
>
> PDB-Richtlinien können das Entfernen von Knoten beim Löschen des Clusters blockieren.

Beim Löschen eines Clusters mit Knotengruppen können Pod Disruption Budget (PDB) -Richtlinien verhindern, dass Knoten erfolgreich entfernt werden. Beispielsweise haben `aws-ebs-csi-driver` installierte Cluster in der Regel zwei Pods mit einer PDB-Richtlinie, die es erlaubt, dass jeweils nur ein Pod nicht verfügbar ist, sodass der andere Pod beim Löschen nicht entfernt werden kann. Um den Cluster in diesen Szenarien erfolgreich zu löschen, verwenden Sie das `disable-nodegroup-eviction` Flag, um die PDB-Richtlinienprüfungen zu umgehen:

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

Weitere Beispielkonfigurationsdateien finden Sie im `examples/`Verzeichnis im GitHub eksctl-Repo.

Probelauf

Die Dry-Run-Funktion ermöglicht das Generieren einer ClusterConfig Datei, die die Clustererstellung überspringt und eine ClusterConfig Datei ausgibt, die die bereitgestellten CLI-Optionen darstellt und die von eksctl festgelegten Standardwerte enthält.

Weitere Informationen finden Sie auf der Dry Run-Seite.

EKS-Automatikmodus

eksctl unterstützt EKS Auto Mode, eine Funktion, die das AWS-Management von Kubernetes-Clustern über den Cluster selbst hinaus erweitert, sodass AWS auch die Infrastruktur einrichten und verwalten kann, die den reibungslosen Betrieb Ihrer Workloads ermöglicht. Auf diese Weise können Sie wichtige Infrastrukturentscheidungen delegieren und das Fachwissen von AWS für den day-to-day Betrieb nutzen. Die von AWS verwaltete Cluster-Infrastruktur umfasst viele Kubernetes-

Funktionen als Kernkomponenten, im Gegensatz zu Add-Ons wie Compute-Autoscaling, Pod- und Service-Networking, Anwendungslastausgleich, Cluster-DNS, Blockspeicher und GPU-Unterstützung.

Erstellen eines EKS-Clusters mit aktiviertem Auto-Modus

`eksctl`hat ein neues `autoModeConfig` Feld hinzugefügt, um den Auto-Modus zu aktivieren und zu konfigurieren. Die Form des `autoModeConfig` Feldes ist

```
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
```

Wenn `autoModeConfig.enabled` dies zutrifft, erstellt eksctl einen EKS-Cluster, indem es `computeConfig.enabled: truekubernetesNetworkConfig.elasticLoadBalancing.enabled: true`, und `storageConfig.blockStorage.enabled: true` an die EKS-API weitergibt, wodurch die Verwaltung von Datenebenenkomponenten wie Rechenleistung, Speicher und Netzwerk ermöglicht wird.

Um einen EKS-Cluster mit aktiviertem Auto-Modus zu erstellen`autoModeConfig.enabled: true`, legen Sie Folgendes fest

```
# 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 erstellt eine Knotenrolle, die für Knoten verwendet wird, die im automatischen Modus gestartet wurden. eksctl erstellt auch die Knotenpools und. `general-purpose` `system` Um die Erstellung der Standard-Knotenpools zu deaktivieren, z. B. um Ihre eigenen Knotenpools zu konfigurieren, die einen anderen Satz von Subnetzen verwenden, setzen Sie, wie 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.
```

Aktualisierung eines EKS-Clusters für die Verwendung des automatischen Modus

Führen Sie folgenden Befehl aus, um einen vorhandenen EKS-Cluster für die Verwendung des automatischen Modus zu aktualisieren

```
# 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**
>
> Wenn der Cluster von eksctl erstellt wurde und öffentliche Subnetze als Cluster-Subnetze verwendet, startet der automatische Modus Knoten in öffentlichen Subnetzen. Um private Subnetze für Worker-Knoten zu verwenden, die im automatischen Modus gestartet wurden, aktualisieren Sie den Cluster so, dass er private Subnetze verwendet.

Automatischer Modus wird deaktiviert

Um den Automatikmodus zu deaktivieren, stellen Sie ihn ein `autoModeConfig.enabled: false` und starten Sie ihn

```
# 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
```

Weitere Informationen

- EKS-Automatikmodus

EKS-Zugangseinträge

Sie können eksctl verwenden, um EKS Access-Einträge zu verwalten. Verwenden Sie Zugriffseinträge, um Kubernetes-Berechtigungen für AWS IAM-Identitäten zu erteilen. Sie können beispielsweise einer Entwicklerrolle die Berechtigung erteilen, Kubernetes-Ressourcen in einem Cluster zu lesen.

In diesem Thema wird beschrieben, wie Sie eksctl zur Verwaltung von Zugriffseinträgen verwenden. Allgemeine Informationen zu Zugriffseinträgen finden Sie unter Gewähren Sie IAM-Benutzern Zugriff auf Kubernetes mit EKS-Zugriffseinträgen.

Sie können von AWS definierte Kubernetes-Zugriffsrichtlinien anhängen oder einer Kubernetes-Gruppe eine IAM-Identität zuordnen.

Weitere Informationen zu den verfügbaren vordefinierten Richtlinien finden Sie unter Zugriffsrichtlinien mit Zugriffseinträgen verknüpfen.

Wenn Sie Kunden-Kubernetes-Richtlinien definieren müssen, ordnen Sie die IAM-Identität einer Kubernetes-Gruppe zu und gewähren Sie dieser Gruppe Berechtigungen.

Cluster-Authentifizierungsmodus

Sie können Zugriffseinträge nur verwenden, wenn der Authentifizierungsmodus des Clusters dies zulässt.

Weitere Informationen finden Sie unter Cluster-Authentifizierungsmodus festlegen

Legen Sie den Authentifizierungsmodus mit einer YAML-Datei fest

`eksctl`hat unter ein neues `accessConfig.authenticationMode` Feld hinzugefügt ClusterConfig, das auf einen der folgenden drei Werte gesetzt werden kann:

- `CONFIG_MAP`- Standard in der EKS-API - `aws-auth` ConfigMap wird nur verwendet
- `API`- Es wird nur die API für Zugriffseinträge verwendet
- `API_AND_CONFIG_MAP`- voreingestellt `eksctl` - `aws-auth` ConfigMap sowohl als auch die API für Zugriffseinträge kann verwendet werden

Stellen Sie den Authentifizierungsmodus in ClusterConfig YAML ein:

```
accessConfig:
  authenticationMode: <>
```

Aktualisieren Sie den Authentifizierungsmodus mit einem Befehl

Wenn Sie Zugriffseinträge für einen bereits vorhandenen, nicht von EKSCTL erstellten Cluster verwenden möchten, bei dem die `CONFIG_MAP` Option verwendet wird, muss der Benutzer zuerst auf einstellen. `authenticationMode` `API_AND_CONFIG_MAP` Dafür `eksctl` wurde ein neuer Befehl zur Aktualisierung des Cluster-Authentifizierungsmodus eingeführt, der sowohl mit CLI-Flags funktioniert, z.

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

Zugriff auf Eintragsressourcen

Zugriffseinträge haben einen Typ, z. B. `STANDARD` oder`EC2_LINUX`. Der Typ hängt davon ab, wie Sie den Zugriffseintrag verwenden.

- Der `standard` Typ dient zur Gewährung von Kubernetes-Berechtigungen für IAM-Benutzer und IAM-Rollen.
  - Sie können beispielsweise Kubernetes-Ressourcen in der AWS-Konsole anzeigen, indem Sie der Rolle oder dem Benutzer, den Sie für den Zugriff auf die Konsole verwenden, eine Zugriffsrichtlinie anhängen.
- Die `EC2_WINDOWS` Typen `EC2_LINUX` und dienen dazu, Kubernetes-Berechtigungen für Instanzen zu gewähren. EC2 Instanzen verwenden diese Berechtigungen, um dem Cluster beizutreten.

Weitere Informationen zu den Arten von Zugriffseinträgen finden Sie unter Zugriffseinträge erstellen

IAM-Entitäten

Sie können Zugriffseinträge verwenden, um Kubernetes-Berechtigungen für IAM-Identitäten wie IAM-Benutzer und IAM-Rollen zu gewähren.

Verwenden Sie das `accessConfig.accessEntries` Feld, um den ARN einer IAM-Ressource einer Access Entries EKS-API zuzuordnen. Zum Beispiel:

```
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
```

Zusätzlich zur Zuordnung von EKS-Richtlinien können Sie auch die Kubernetes-Gruppen angeben, zu denen eine IAM-Entität gehört, wodurch Berechtigungen über RBAC gewährt werden.

Verwaltete Knotengruppen und Fargate

Die Integration mit den Zugangseinträgen für diese Ressourcen wird hinter den Kulissen durch die EKS-API erreicht. Neu erstellte verwaltete Knotengruppen und Fargate-Pods erstellen API-Zugriffseinträge, anstatt vorinstallierte RBAC-Ressourcen zu verwenden. Bestehende Knotengruppen und Fargate-Pods werden nicht geändert und verlassen sich weiterhin auf die Einträge in der aws-auth-Konfigurationsübersicht.

Selbstverwaltete Knotengruppen

Jeder Zugriffseintrag hat einen Typ. Für die Autorisierung von selbstverwalteten Knotengruppen `eksctl` wird für jede Knotengruppe ein eindeutiger Zugriffseintrag erstellt, wobei der Haupt-ARN auf die Knotenrolle ARN und der Typ entweder oder `EC2_LINUX` je nach Knotengruppe AMIFamily gesetzt ist. `EC2_WINDOWS`

Wenn Sie Ihre eigenen Zugriffseinträge erstellen, können Sie auch `EC2_LINUX` (für eine IAM-Rolle, die mit selbstverwalteten Linux- oder Bottlerocket-Knoten verwendet wird), `EC2_WINDOWS` (für eine IAM-Rolle, die mit selbstverwalteten Windows-Knoten verwendet wird), `FARGATE_LINUX` (für eine IAM-Rolle, die mit AWS Fargate (Fargate) verwendet wird) oder als Typ angeben. `STANDARD` Wenn Sie keinen Typ angeben, ist der Standardtyp auf festgelegt. `STANDARD`

> **Note**
>
> Beim Löschen einer Knotengruppe, die mit einer bereits vorhandenen Knotengruppe erstellt wurde`instanceRoleARN`, liegt es in der Verantwortung des Benutzers, den entsprechenden Zugriffseintrag zu löschen, wenn ihm keine Knotengruppen mehr zugeordnet sind. Das liegt daran, dass eksctl nicht versucht herauszufinden, ob ein Zugriffseintrag noch von nicht mit eksctl erstellten, selbstverwalteten Knotengruppen verwendet wird, da dies ein komplizierter Vorgang ist.

Zugangseintrag erstellen

Dies kann auf zwei verschiedene Arten geschehen, entweder während der Clustererstellung, indem Sie die gewünschten Zugriffseinträge als Teil der Konfigurationsdatei angeben und Folgendes ausführen:

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

ODER nach der Clustererstellung, indem Sie Folgendes ausführen:

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

Ein Beispiel für eine Konfigurationsdatei zum Erstellen von Zugriffseinträgen finden Sie unter 40-access-entries.yaml im eksctl-Repo. GitHub

Holen Sie sich den Zugangseintrag

Der Benutzer kann alle mit einem bestimmten Cluster verknüpften Zugriffseinträge abrufen, indem er einen der folgenden Schritte ausführt:

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

ODER

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

Alternativ kann man das Flag verwenden, um nur den Zugangseintrag abzurufen, der einer bestimmten IAM-Entität entspricht. z.B. `--principal-arn`

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

Zugangseintrag löschen

Um jeweils einen einzelnen Zugriffseintrag zu löschen, verwenden Sie:

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

Um mehrere Zugriffseinträge zu löschen, verwenden Sie die `--config-file` Markierung und geben Sie alle zu den Zugriffseinträgen `principalARN's` gehörenden Einträge unter dem `accessEntry` Feld der obersten Ebene an, z. B.

```
...
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
```

Migrieren Sie von aws-auth ConfigMap

Der Benutzer kann seine vorhandenen IAM-Identitäten von `aws-auth` Configmap zu Access-Einträgen migrieren, indem er den folgenden Befehl ausführt:

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

Wenn `--target-authentication-mode` Flag auf gesetzt ist`API`, wird der Authentifizierungsmodus in den `API` Modus umgeschaltet (übersprungen, falls der `API` Modus bereits aktiviert ist), die IAM-Identitätszuordnungen werden zu den Zugriffseinträgen migriert und die Configmap wird aus dem Cluster gelöscht. `aws-auth`

Wenn `--target-authentication-mode` Flag auf gesetzt ist`API_AND_CONFIG_MAP`, wird der Authentifizierungsmodus in `API_AND_CONFIG_MAP` den Modus umgeschaltet (wird übersprungen, falls der `API_AND_CONFIG_MAP` Modus bereits aktiviert ist). IAM-Identitätszuordnungen werden zu Zugriffseinträgen migriert, die Configmap bleibt jedoch erhalten. `aws-auth`

> **Note**
>
> Wenn `--target-authentication-mode` Flag auf gesetzt ist`API`, aktualisiert dieser Befehl den Authentifizierungsmodus nicht in den Modus, wenn für die `API` `aws-auth` Configmap eine der folgenden Einschränkungen gilt.

- Es gibt eine Identitätszuweisung auf Kontoebene.

- Eine oder mehrere Roles/Users sind den Kubernetes-Gruppen zugeordnet, die mit einem Präfix `system:` beginnen (mit Ausnahme von EKS-spezifischen Gruppen`system:masters`, d. h.`system:bootstrappers`, `system:nodes` usw.).
- Eine oder mehrere IAM-Identitätszuordnungen gelten für eine [Service Linked Role] (Link: IAM/latest/UserGuide/using - .html). service-linked-roles

Deaktivieren Sie die Administratorrechte für den Clustersteller

`eksctl`hat ein neues Feld hinzugefügt, `accessConfig.bootstrapClusterCreatorAdminPermissions: boolean` das, wenn es auf „False“ gesetzt ist, die Erteilung von Cluster-Admin-Rechten für die IAM-Identität, die den Cluster erstellt, deaktiviert, d. h.

fügt die Option zur Konfigurationsdatei hinzu:

```
accessConfig:
  bootstrapClusterCreatorAdminPermissions: false
```

und führe aus:

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

Nicht von EKSCTL erstellte Cluster

Sie können `eksctl` Befehle für Cluster ausführen, die nicht von erstellt wurden. `eksctl`

> **Note**
>
> Eksctl kann nur Cluster ohne Besitzer unterstützen, deren Namen mit AWS kompatibel sind. CloudFormation Alle Clusternamen, die nicht mit diesen übereinstimmen, werden die CloudFormation API-Validierungsprüfung nicht bestehen.

Unterstützte Befehle

Die folgenden Befehle können für Cluster verwendet werden, die auf andere Weise als erstellt wurden`eksctl`. Die Befehle, Flags und Optionen der Konfigurationsdatei können auf genau dieselbe Weise verwendet werden.

Wenn wir einige Funktionen verpasst haben, teilen Sie uns dies bitte mit.

- ✓ Erstellen:
  - ✓ `eksctl create nodegroup` (siehe Hinweis unten)
  - ✓ `eksctl create fargateprofile`
  - ✓ `eksctl create iamserviceaccount`
  - ✓ `eksctl create iamidentitymapping`
- ✓ Hol dir:
  - ✓ `eksctl get clusters/cluster`
  - ✓ `eksctl get fargateprofile`
  - ✓ `eksctl get nodegroup`
  - ✓ `eksctl get labels`
- ✓ Löschen:
  - ✓ `eksctl delete cluster`
  - ✓ `eksctl delete nodegroup`
  - ✓ `eksctl delete fargateprofile`
  - ✓ `eksctl delete iamserviceaccount`
  - ✓ `eksctl delete iamidentitymapping`
- ✓ Aktualisierung:
  - ✓ `eksctl upgrade cluster`
  - ✓ `eksctl upgrade nodegroup`
- ✓ Ein-/Ausschalten:
  - ✓ `eksctl set labels`
  - ✓ `eksctl unset labels`
- ✓ Maßstab:
  - ✓ `eksctl scale nodegroup`
- ✓ Abfluss:
  - ✓ `eksctl drain nodegroup`
- ✓ Aktivieren:
  - ✓ `eksctl enable profile`
  - ✓ `eksctl enable repo`

- ✓ Hilfsmittel:
  - ✓ `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`

Knotengruppen erstellen

`eksctl create nodegroup`ist der einzige Befehl, der spezifische Eingaben vom Benutzer erfordert.

Da Benutzer ihre Cluster mit jeder beliebigen Netzwerkkonfiguration erstellen können, `eksctl` werden sie vorerst nicht versuchen, diese Werte abzurufen oder zu erraten. Dies könnte sich in future ändern, wenn wir mehr darüber erfahren, wie Benutzer diesen Befehl auf Clustern verwenden, die nicht von Eksctl erstellt wurden.

Das bedeutet, dass zum Erstellen von Knotengruppen oder verwalteten Knotengruppen auf einem Cluster, der nicht von erstellt wurde`eksctl`, eine Konfigurationsdatei mit VPC-Details bereitgestellt werden muss. Zumindest:

```
---
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"

...
```

Weitere Informationen zu VPC-Konfigurationsoptionen finden Sie unter Netzwerke.

Registrierung von Nicht-EKS-Clustern mit EKS Connector

Sie können den EKS-Connector verwenden, um Cluster außerhalb von AWS in der EKS-Konsole anzuzeigen. Dieser Prozess erfordert die Registrierung des Clusters bei EKS und die Ausführung des EKS Connector-Agenten auf dem externen Kubernetes-Cluster.

`eksctl`vereinfacht die Registrierung von Nicht-EKS-Clustern, indem die erforderlichen AWS-Ressourcen erstellt und Kubernetes-Manifeste generiert werden, damit EKS Connector sie auf den externen Cluster anwenden kann.

Cluster registrieren

Führen Sie Folgendes aus, um einen Kubernetes-Cluster zu registrieren oder eine Verbindung herzustellen, der nicht zu EKS gehört

```
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
```

Dieser Befehl registriert den Cluster und schreibt drei Dateien, die die Kubernetes-Manifeste für EKS Connector enthalten. Diese müssen auf den externen Cluster angewendet werden, bevor die Registrierung abläuft.

> **Note**
>
> `eks-connector-clusterrole.yaml`getund `eks-connector-console-dashboard-full-access-clusterrole.yaml` gibt der aufrufenden IAM-Identität `list` Berechtigungen für Kubernetes-Ressourcen in allen Namespaces. Diese müssen bei Bedarf entsprechend bearbeitet werden, bevor sie auf den Cluster angewendet werden. Informationen zur Konfiguration eines stärker eingeschränkten Zugriffs finden Sie unter Einem Benutzer Zugriff auf einen Cluster gewähren.

Um eine bestehende IAM-Rolle zur Verwendung für EKS Connector bereitzustellen, übergeben Sie sie `--role-arn` wie folgt:

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

Wenn der Cluster bereits existiert, gibt eksctl einen Fehler zurück.

Cluster deregistrieren

Um die Registrierung eines registrierten Clusters aufzuheben oder die Verbindung zu trennen, führen Sie folgenden Befehl aus

```
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
```

Mit diesem Befehl wird der externe Cluster deregistriert und die zugehörigen AWS-Ressourcen entfernt. Sie müssen jedoch die Kubernetes-Ressourcen des EKS-Connectors aus dem Cluster entfernen.

Weitere Informationen

- EKS-Anschluss

Anpassen der Kubelet-Konfiguration

Systemressourcen können über die Konfiguration des Kubelets reserviert werden. Dies wird empfohlen, da das Kubelet im Falle eines Ressourcenmangels möglicherweise nicht in der Lage ist, Pods zu entfernen und den Knoten schließlich zu einem Knoten zu machen. `NotReady` Zu diesem Zweck können die Konfigurationsdateien das `kubeletExtraConfig` Feld enthalten, das ein YAML in freier Form akzeptiert, das in die eingebettet wird. `kubelet.yaml`

Einige Felder in der `kubelet.yaml` werden von eksctl gesetzt und können daher nicht überschrieben werden, wie z. B. das,,`address`, `clusterDomain` oder. `authentication` `authorization` `serverTLSBootstrap`

Die folgende Beispielkonfigurationsdatei erstellt eine Knotengruppe, die `300m` vCPU, `300Mi` Arbeitsspeicher und `1Gi` kurzlebigen Speicher für das Kubelet reserviert; `300m` vCPU, `300Mi` Speicher und kurzlebigen Speicher für Betriebssystem-Daemons; und die Räumung `1Gi` von Pods einleitet, wenn weniger als Speicher verfügbar ist oder weniger als 10% des Root-Dateisystems verfügbar sind. `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 diesem Beispiel CPUs würde bei Instanzen des Typs `m5a.xlarge` mit 4 V CPUs und 16 GiB Speicher die Größe von 3,4 und 15,4 GiB Speicher `Allocatable` betragen. Es ist wichtig zu wissen, dass die in der Konfigurationsdatei für die Felder angegebenen Werte die in `kubeletExtraconfig` eksctl angegebenen Standardwerte vollständig überschreiben. Wenn Sie jedoch einen oder mehrere Parameter weglassen, werden die fehlenden `kubeReserved` Parameter standardmäßig auf vernünftige Werte gesetzt, die auf dem verwendeten AWS-Instanztyp basieren.

`kubeReserved`Berechnung

Es wird zwar generell empfohlen, eine gemischte Instanz so NodeGroup zu konfigurieren, dass sie Instanzen mit derselben CPU- und RAM-Konfiguration verwendet, aber das ist keine strikte Anforderung. Daher verwendet die `kubeReserved` Berechnung die kleinste Instanz im `InstanceDistribution.InstanceTypes` Feld. Auf diese Weise werden NodeGroups bei unterschiedlichen Instanztypen nicht zu viele Ressourcen für die kleinste Instanz reserviert. Dies könnte jedoch zu einer Reservierung führen, die für den größten Instance-Typ zu klein ist.

> **Warning**
>
> Standardmäßig `eksctl` festgelegt`featureGates.RotateKubeletServerCertificate=true`, aber wenn benutzerdefinierte Einstellungen bereitgestellt `featureGates` werden, werden sie nicht gesetzt. Sie sollten es immer einschließen`featureGates.RotateKubeletServerCertificate=true`, es sei denn, Sie müssen es deaktivieren.

CloudWatch Protokollierung

In diesem Thema wird erklärt, wie Sie die CloudWatch Amazon-Protokollierung für die Komponenten der Steuerungsebene Ihres EKS-Clusters konfigurieren. CloudWatch Die Protokollierung bietet Einblick in den Betrieb der Kontrollebene Ihres Clusters. Dies ist für die Behebung von Problemen, die Prüfung der Cluster-Aktivitäten und die Überwachung des Zustands Ihrer Kubernetes-Komponenten unerlässlich.

Protokollierung aktivieren CloudWatch

CloudWatch Die Protokollierung für die EKS-Steuerebene ist aufgrund von Datenaufnahme- und Speicherkosten standardmäßig nicht aktiviert.

Um die Protokollierung auf der Kontrollebene bei der Clustererstellung zu aktivieren, müssen Sie eine **`cloudWatch.clusterLogging.enableTypes`**Einstellung in Ihrem definieren `ClusterConfig` (Beispiele finden Sie unten).

Wenn Sie also eine Konfigurationsdatei mit der richtigen **`cloudWatch.clusterLogging.enableTypes`**Einstellung haben, können Sie einen Cluster mit erstellen`eksctl create cluster --config-file=<path>`.

Wenn Sie bereits einen Cluster erstellt haben, können Sie ihn verwenden`eksctl utils update-cluster-logging`.

> **Note**
>
> Dieser Befehl wird standardmäßig im Planmodus ausgeführt. Sie müssen ein `--approve` Flag angeben, um die Änderungen auf Ihren Cluster anzuwenden.

Wenn Sie eine Konfigurationsdatei verwenden, führen Sie Folgendes aus:

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

Alternativ können Sie CLI-Flags verwenden.

Um alle Arten von Protokollen zu aktivieren, führen Sie folgenden Befehl aus:

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

Um `audit` Logs zu aktivieren, führen Sie folgenden Befehl aus:

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

Um alle außer `controllerManager` Logs zu aktivieren, führen Sie folgenden Befehl aus:

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

Wenn die Protokolltypen `api` und die `scheduler` Protokolltypen bereits aktiviert waren, führen Sie folgenden Befehl aus, um sie gleichzeitig zu deaktivieren `scheduler` und zu aktivieren`controllerManager`:

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

Dadurch bleiben `api` und `controllerManager` als einzige Protokolltypen aktiviert.

Um alle Arten von Protokollen zu deaktivieren, führen Sie folgenden Befehl aus:

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

Beispiele für **`ClusterConfig`**

In einem EKS-Cluster `clusterLogging` kann das `enableTypes` Feld darunter eine Liste möglicher Werte enthalten, um die verschiedenen Protokolltypen für die Komponenten der Steuerungsebene zu aktivieren.

Die folgenden Werte sind möglich:

- `api`: Aktiviert die Kubernetes-API-Serverprotokolle.
- `audit`: Aktiviert die Kubernetes-Audit-Logs.
- `authenticator`: Aktiviert die Authenticator-Logs.
- `controllerManager`: Aktiviert die Kubernetes-Controller-Manager-Logs.
- `scheduler`: Aktiviert die Kubernetes-Scheduler-Protokolle.

Weitere Informationen finden Sie in der EKS-Dokumentation.

Deaktivieren Sie alle Protokolle

Um alle Typen zu deaktivieren, verwenden `[]` oder entfernen Sie den `cloudWatch` Abschnitt vollständig.

Alle Protokolle aktivieren

Sie können alle Typen mit `"*"` oder aktivieren`"all"`. Zum Beispiel:

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

Aktivieren Sie ein oder mehrere Protokolle

Sie können eine Teilmenge von Typen aktivieren, indem Sie die Typen auflisten, die Sie aktivieren möchten. Zum Beispiel:

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

Aufbewahrungszeitraum protokollieren

Standardmäßig werden Protokolle auf unbestimmte Zeit in CloudWatch Logs gespeichert. In Logs können Sie die Anzahl der Tage angeben, für die die Protokolle der Kontrollebene aufbewahrt CloudWatch werden sollen. Im folgenden Beispiel werden Protokolle 7 Tage lang aufbewahrt:

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

Vollständiges Beispiel

```
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
```

Vollständig privater EKS-Cluster

eksctl unterstützt die Erstellung von vollständig privaten Clustern, die keinen ausgehenden Internetzugang haben und nur private Subnetze haben. VPC-Endpunkte werden verwendet, um den privaten Zugriff auf AWS-Services zu ermöglichen.

In diesem Handbuch wird beschrieben, wie Sie einen privaten Cluster ohne ausgehenden Internetzugang erstellen.

Einen vollständig privaten Cluster erstellen

Das einzige erforderliche Feld zum Erstellen eines vollständig privaten Clusters ist: `privateCluster.enabled`

```
privateCluster:
  enabled: true
```

Nach der Clustererstellung müssen eksctl-Befehle, die Zugriff auf den Kubernetes-API-Server benötigen, in der VPC des Clusters, einer Peer-VPC oder mit anderen Mitteln wie AWS Direct Connect ausgeführt werden. eksctl-Befehle, die Zugriff auf den EKS benötigen, funktionieren nicht, wenn sie innerhalb der VPC des Clusters ausgeführt APIs werden. Um dieses Problem zu beheben, erstellen Sie einen Schnittstellenendpunkt für Amazon EKS, um APIs von Ihrer Amazon Virtual Private Cloud (VPC) privat auf das Amazon Elastic Kubernetes Service (Amazon EKS) -Management zuzugreifen. In einer future Version wird eksctl Unterstützung für die Erstellung dieses Endpunkts hinzufügen, sodass er nicht manuell erstellt werden muss. Befehle, die Zugriff auf die OpenID Connect-Provider-URL benötigen, müssen von außerhalb der VPC Ihres Clusters ausgeführt werden, sobald Sie AWS PrivateLink für Amazon EKS aktiviert haben.

Das Erstellen verwalteter Knotengruppen wird weiterhin funktionieren, und das Erstellen von selbstverwalteten Knotengruppen funktioniert, da es Zugriff auf den API-Server über den EKS-Schnittstellenendpunkt benötigt, wenn der Befehl von der VPC des Clusters, einer Peer-VPC oder mit anderen Mitteln wie AWS Direct Connect ausgeführt wird.

> **Note**
>
> VPC-Endpoints werden stundenweise und nutzungsabhängig abgerechnet. Weitere Informationen zur Preisgestaltung finden Sie unter PrivateLink AWS-Preise

> **Warning**
>
> Vollständig private Cluster werden in `eu-south-1` nicht unterstützt.

Konfiguration des privaten Zugriffs auf zusätzliche AWS-Services

Damit Worker-Knoten privat auf AWS-Services zugreifen können, erstellt eksctl VPC-Endpunkte für die folgenden Services:

- Schnittstellenendpunkte für ECR (`ecr.api`sowohl als auch`ecr.dkr`) zum Abrufen von Container-Images (AWS CNI-Plugin usw.)
- Ein Gateway-Endpunkt für S3 zum Abrufen der eigentlichen Bildebenen
- Ein Schnittstellenendpunkt, der für die `aws-cloud-provider` Integration EC2 erforderlich ist
- Ein Schnittstellenendpunkt für STS zur Unterstützung von Fargate und IAM Roles for Services Accounts (IRSA)
- Ein Schnittstellenendpunkt für die CloudWatch Protokollierung (`logs`), wenn CloudWatch die Protokollierung aktiviert ist

Diese VPC-Endpunkte sind für einen funktionierenden privaten Cluster unerlässlich, weshalb eksctl ihre Konfiguration oder Deaktivierung nicht unterstützt. Ein Cluster benötigt jedoch möglicherweise privaten Zugriff auf andere AWS-Services (z. B. Autoscaling, das für den Cluster Autoscaler erforderlich ist). Diese Dienste können in angegeben werden`privateCluster.additionalEndpointServices`, wodurch eksctl angewiesen wird, für jeden von ihnen einen VPC-Endpunkt zu erstellen.

Um beispielsweise privaten Zugriff auf Autoscaling und Logging zu ermöglichen: CloudWatch

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

Die in unterstützten Endpunkte `additionalEndpointServices` sind`autoscaling`, und`cloudformation`. `logs`

Endpunkterstellungen werden übersprungen

Wenn bereits eine VPC mit den erforderlichen AWS-Endpunkten erstellt und mit den in der EKS-Dokumentation beschriebenen Subnetzen verknüpft wurde, `eksctl` können Sie deren Erstellung überspringen, indem Sie die folgende Option `skipEndpointCreation` angeben:

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

Diese Einstellung kann nicht zusammen mit verwendet werden. `additionalEndpointServices` Es wird die gesamte Endpunkterstellung übersprungen. Außerdem wird diese Einstellung nur empfohlen, wenn die <# Endpunkt-Subnetz-Topologie korrekt eingerichtet ist. Wenn die Subnetz-IDs korrekt sind, wird das `vpce` Routing mit Präfixadressen eingerichtet, alle erforderlichen EKS-Endpunkte werden erstellt und mit der bereitgestellten VPC verknüpft. `eksctl`wird keine dieser Ressourcen ändern.

Knotengruppen

In einem vollständig privaten Cluster werden nur private Knotengruppen (sowohl verwaltete als auch selbstverwaltete) unterstützt, da die VPC des Clusters ohne öffentliche Subnetze erstellt wird. Das `privateNetworking` Feld (`nodeGroup[].privateNetworking` und`managedNodeGroup[`) muss explizit festgelegt werden. In einem vollständig privaten Cluster ist es ein Fehler, den `privateNetworking` Wert nicht gesetzt zu lassen.

```
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
```

Cluster-Endpunktzugriff

Ein vollständig privater Cluster unterstützt keine Änderungen `clusterEndpointAccess` während der Clustererstellung. Es ist ein Fehler, entweder `clusterEndpoints.publicAccess` oder festzulegen`clusterEndpoints.privateAccess`, da ein vollständig privater Cluster nur privaten Zugriff haben kann und die Änderung dieser Felder den Cluster beschädigen kann.

Vom Benutzer bereitgestellte VPC und Subnetze

eksctl unterstützt die Erstellung vollständig privater Cluster unter Verwendung einer bereits vorhandenen VPC und Subnetze. Es können nur private Subnetze angegeben werden, und es ist ein Fehler, Subnetze unter anzugeben. `vpc.subnets.public`

eksctl erstellt VPC-Endpunkte in der bereitgestellten VPC und ändert Routentabellen für die bereitgestellten Subnetze. Jedem Subnetz sollte eine explizite Routing-Tabelle zugeordnet sein, da eksctl die Haupt-Routing-Tabelle nicht ändert.

```
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
```

Verwaltung eines vollständig privaten Clusters

Damit alle Befehle nach der Clustererstellung funktionieren, benötigt eksctl privaten Zugriff auf den EKS-API-Serverendpunkt und einen ausgehenden Internetzugang (für). `EKS:DescribeCluster` Befehle, die keinen Zugriff auf den API-Server benötigen, werden unterstützt, wenn eksctl über einen ausgehenden Internetzugang verfügt.

Erzwungenes Löschen eines vollständig privaten Clusters

Beim Löschen eines vollständig privaten Clusters über eksctl treten wahrscheinlich Fehler auf, da eksctl nicht automatisch auf alle Ressourcen des Clusters zugreifen kann. `--force`existiert, um dieses Problem zu lösen: Es erzwingt das Löschen des Clusters und fährt fort, wenn Fehler auftreten.

Einschränkungen

Eine Einschränkung der aktuellen Implementierung besteht darin, dass eksctl den Cluster zunächst mit aktiviertem öffentlichem und privatem Endpunktzugriff erstellt und den öffentlichen Endpunktzugriff deaktiviert, nachdem alle Operationen abgeschlossen sind. Dies ist erforderlich, da eksctl Zugriff auf den Kubernetes-API-Server benötigt, damit selbstverwaltete Knoten dem Cluster beitreten und Fargate unterstützen können. GitOps Nachdem diese Operationen abgeschlossen sind, stellt eksctl den Zugriff auf Cluster-Endpunkte auf „Nur privat“ um. Dieses zusätzliche Update bedeutet, dass die Erstellung eines vollständig privaten Clusters länger dauert als die eines Standardclusters. In future könnte eksctl zu einer VPC-fähigen Lambda-Funktion wechseln, um diese API-Operationen durchzuführen.

Ausgehender Zugriff über HTTP-Proxyserver

eksctl kann APIs über einen konfigurierten HTTP (S) -Proxyserver mit AWS kommunizieren. Sie müssen jedoch sicherstellen, dass Sie Ihre Proxy-Ausschlussliste korrekt eingestellt haben.

Im Allgemeinen müssen Sie sicherstellen, dass Anfragen für den VPC-Endpunkt für Ihren Cluster nicht über Ihre Proxys weitergeleitet werden, indem Sie eine entsprechende `no_proxy` Umgebungsvariable einschließlich des Werts festlegen. `.eks.amazonaws.com`

Wenn Ihr Proxyserver „SSL-Interception“ durchführt und Sie IAM Roles for Service Accounts (IRSA) verwenden, müssen Sie sicherstellen, dass Sie SSL für die Domain explizit umgehen. Man-in-the-Middle `oidc.<region>.amazonaws.com` Andernfalls erhält eksctl den falschen Fingerabdruck des Stammzertifikats für den OIDC-Anbieter, und das AWS VPC CNI-Plug-In kann nicht gestartet werden, da keine IAM-Anmeldeinformationen abgerufen werden können, wodurch Ihr Cluster funktionsunfähig wird.

Weitere Informationen

- Private EKS-Cluster

Erweiterungen

In diesem Thema wird beschrieben, wie Sie Amazon EKS Add-Ons für Ihre Amazon EKS-Cluster mithilfe von eksctl verwalten. EKS Add-Ons ist eine Funktion, mit der Sie die Kubernetes-Betriebssoftware über die EKS-API aktivieren und verwalten können, wodurch der Prozess der Installation, Konfiguration und Aktualisierung von Cluster-Add-Ons vereinfacht wird.

> **Warning**
>
> eksctl installiert jetzt Standard-Addons (vpc-cni, coredns, kube-proxy) als EKS-Addons statt als selbstverwaltete Addons. Das bedeutet, dass Sie anstelle von Befehlen Befehle für Cluster verwenden sollten, die mit eksctl v0.184.0 und höher erstellt wurden. `eksctl update addon` `eksctl utils update-*`

Sie können Cluster ohne standardmäßige Netzwerk-Addons erstellen, wenn Sie alternative CNI-Plugins wie Cilium und Calico verwenden möchten.

EKS-Add-ons unterstützen jetzt den Empfang von IAM-Berechtigungen über EKS Pod Identity Associations, sodass sie sich mit AWS-Services außerhalb des Clusters verbinden können

Addons erstellen

Eksctl bietet mehr Flexibilität bei der Verwaltung von Cluster-Addons:

In Ihrer Konfigurationsdatei können Sie die gewünschten Addons und (falls erforderlich) die Rolle oder Richtlinien angeben, die ihnen zugewiesen werden sollen:

```
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: '*'
```

Sie können höchstens eines von`attachPolicy`, `attachPolicyARNs` und `serviceAccountRoleARN` angeben.

Wenn keine dieser Optionen angegeben ist, wird das Addon mit einer Rolle erstellt, der alle empfohlenen Richtlinien zugeordnet sind.

> **Note**
>
> Um Richtlinien an Addons anzuhängen, muss `OIDC` Ihr Cluster aktiviert sein. Wenn es nicht aktiviert ist, ignorieren wir alle angehängten Richtlinien.

Sie können dann entweder diese Addons während der Clustererstellung erstellen lassen:

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

Oder erstellen Sie die Addons explizit nach der Clustererstellung mit der Konfigurationsdatei oder den CLI-Flags:

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

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

Wenn während der Addon-Erstellung bereits eine selbstverwaltete Version des Addons auf dem Cluster existiert, können Sie wählen, wie potenzielle `configMap` Konflikte gelöst werden sollen, indem Sie die `resolveConflicts` Option in der Konfigurationsdatei festlegen, z. B.

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

Bei Addon Create unterstützt das `resolveConflicts` Feld drei unterschiedliche Werte:

- `none`- EKS ändert den Wert nicht. Die Erstellung schlägt möglicherweise fehl.
- `overwrite`- EKS überschreibt alle Konfigurationsänderungen auf die EKS-Standardwerte zurück.
- `preserve`- EKS ändert den Wert nicht. Die Erstellung schlägt möglicherweise fehl. (Ähnlich`none`, aber anders als `preserve`beim Aktualisieren von Addons).

Aktivierte Addons auflisten

Sie können sehen, welche Addons in Ihrem Cluster aktiviert sind, indem Sie Folgendes ausführen:

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

or

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

Die Version des Addons festlegen

Das Einstellen der Version des Addons ist optional. Wenn das `version` Feld leer gelassen wird, `eksctl` wird die Standardversion für das Addon aufgelöst. Weitere Informationen darüber, welche Version die Standardversion für bestimmte Addons ist, finden Sie in der AWS-Dokumentation zu EKS. Beachten Sie, dass die Standardversion möglicherweise nicht unbedingt die neueste verfügbare Version ist.

Die Addon-Version kann auf eingestellt werden. `latest` Alternativ kann die Version mit dem angegebenen EKS-Build-Tag festgelegt werden, z. B. `v1.7.5-eksbuild.1` oder`v1.7.5-`

`eksbuild.2`. Es kann auch auf die Release-Version des Addons gesetzt werden, z. B. `v1.7.5` oder`1.7.5`, und das `eksbuild` Suffix-Tag wird für Sie erkannt und gesetzt.

Im folgenden Abschnitt erfahren Sie, wie Sie verfügbare Addons und ihre Versionen entdecken können.

Addons entdecken

Sie können herausfinden, welche Addons für die Installation in Ihrem Cluster verfügbar sind, indem Sie Folgendes ausführen:

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

Dadurch wird die Kubernetes-Version Ihres Clusters erkannt und danach gefiltert. Wenn Sie sehen möchten, welche Addons für eine bestimmte Kubernetes-Version verfügbar sind, können Sie alternativ Folgendes ausführen:

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

Sie können Addons auch entdecken, indem Sie nach ihren, filtern. `type` `owner` and/or `publisher` Um zum Beispiel Addons für einen bestimmten Besitzer und Typ zu sehen, kannst du Folgendes ausführen:

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

Die `publishers` Flags`types`, `owners` und, sind optional und können zusammen oder einzeln angegeben werden, um die Ergebnisse zu filtern.

Entdecken Sie das Konfigurationsschema für Addons

Nachdem Sie das Addon und die Version entdeckt haben, können Sie sich die Anpassungsoptionen ansehen, indem Sie das JSON-Konfigurationsschema abrufen.

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

Dadurch wird ein JSON-Schema mit den verschiedenen Optionen zurückgegeben, die für dieses Addon verfügbar sind.

Mit Konfigurationswerten arbeiten

`ConfigurationValues`kann während der Erstellung oder Aktualisierung von Addons in der Konfigurationsdatei bereitgestellt werden. Es werden nur die Formate JSON und YAML unterstützt.

Zum Beispiel ,

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

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

> **Note**
>
> Denken Sie daran, dass bei der Änderung von Addon-Konfigurationswerten Konfigurationskonflikte auftreten können.

```
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`.
```

Zusätzlich ruft der Befehl get jetzt auch `ConfigurationValues` für das Addon ab. z.B.

```
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
```

Addons werden aktualisiert

Du kannst deine Addons auf neuere Versionen aktualisieren und ändern, welche Richtlinien angehängt werden, indem du Folgendes ausführst:

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

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

Ähnlich wie bei der Erstellung von Addons haben Sie beim Aktualisieren eines Addons die volle Kontrolle über die Konfigurationsänderungen, die Sie möglicherweise zuvor auf dieses Add-on vorgenommen haben. `configMap` Insbesondere können Sie sie beibehalten oder überschreiben. Diese optionale Funktionalität ist über dasselbe Feld in der Konfigurationsdatei verfügbar`resolveConflicts`. z. B.

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

Für das Addon-Update akzeptiert das `resolveConflicts` Feld drei unterschiedliche Werte:

- `none`- EKS ändert den Wert nicht. Das Update schlägt möglicherweise fehl.
- `overwrite`- EKS überschreibt alle Konfigurationsänderungen auf die EKS-Standardwerte zurück.
- `preserve`- EKS behält den Wert bei. Wenn Sie diese Option wählen, empfehlen wir Ihnen, alle Feld- und Wertänderungen in einem Cluster zu testen, der nicht zur Produktion gehört, bevor Sie das Add-on auf Ihrem Produktionscluster aktualisieren.

Addons löschen

Sie können ein Addon löschen, indem Sie Folgendes ausführen:

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

Dadurch werden das Addon und alle damit verknüpften IAM-Rollen gelöscht.

Wenn Sie Ihren Cluster löschen, werden auch alle mit Addons verknüpften IAM-Rollen gelöscht.

Flexibilität bei der Clustererstellung für Standard-Netzwerk-Addons

Wenn ein Cluster erstellt wird, installiert EKS automatisch VPC CNI, CoreDNS und Kube-Proxy als selbstverwaltete Addons. Um dieses Verhalten zu deaktivieren, um andere CNI-Plugins wie Cilium und Calico zu verwenden, unterstützt eksctl jetzt die Erstellung eines Clusters ohne standardmäßige Netzwerk-Addons. Um einen solchen Cluster zu erstellen, stellen Sie Folgendes ein: `addonsConfig.disableDefaultAddons`

```
addonsConfig:
  disableDefaultAddons: true
```

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

Um einen Cluster nur mit CoreDNS und Kube-Proxy und nicht mit VPC CNI zu erstellen, geben Sie die Addons explizit an und setzen Sie sie wie folgt: `addons` `addonsConfig.disableDefaultAddons`

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

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

Im Rahmen dieser Änderung installiert eksctl bei der Clustererstellung nun Standard-Addons als EKS-Addons statt als selbstverwaltete Addons, sofern sie nicht explizit auf true gesetzt sind. `addonsConfig.disableDefaultAddons` Daher können `eksctl utils update-*` Befehle nicht mehr zum Aktualisieren von Addons für Cluster verwendet werden, die mit eksctl v0.184.0 und höher erstellt wurden:

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

Stattdessen sollte es jetzt verwendet werden. `eksctl update addon`

Weitere Informationen finden Sie unter Amazon EKS bietet Flexibilität bei der Clustererstellung für Netzwerk-Add-Ons.

Zugriff für Amazon EMR aktivieren

Damit EMR Operationen auf der Kubernetes-API ausführen kann, müssen seiner SLR die erforderlichen RBAC-Berechtigungen erteilt werden. eksctl stellt einen Befehl bereit, der die erforderlichen RBAC-Ressourcen für EMR erstellt und aktualisiert, um die Rolle an die SLR für EMR zu binden. `aws-auth` ConfigMap

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

EKS Fargate-Unterstützung

AWS Fargate ist eine verwaltete Compute-Engine für Amazon ECS, die Container ausführen kann. In Fargate müssen Sie keine Server oder Cluster verwalten.

Amazon EKS kann jetzt Pods auf AWS Fargate starten. Dadurch müssen Sie sich keine Gedanken darüber machen, wie Sie die Infrastruktur für Pods bereitstellen oder verwalten, und es wird einfacher, leistungsstarke, hochverfügbare Kubernetes-Anwendungen auf AWS zu erstellen und auszuführen.

Einen Cluster mit Fargate-Unterstützung erstellen

Sie können einen Cluster mit Fargate-Unterstützung hinzufügen mit:

```
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
```

Dieser Befehl hat einen Cluster und ein Fargate-Profil erstellt. Dieses Profil enthält bestimmte Informationen, die AWS benötigt, um Pods in Fargate zu instanziieren. Dies sind:

- Pod-Ausführungsrolle, um die für die Ausführung des Pods erforderlichen Berechtigungen und den Netzwerkstandort (Subnetz) für die Ausführung des Pods zu definieren. Dadurch können dieselben Netzwerk- und Sicherheitsberechtigungen auf mehrere Fargate-Pods angewendet werden, und es wird einfacher, vorhandene Pods auf einem Cluster zu Fargate zu migrieren.

- Selektor, um zu definieren, welche Pods auf Fargate laufen sollen. Dieser besteht aus einem `namespace` und. `labels`

Wenn das Profil nicht angegeben ist, aber die Unterstützung für Fargate aktiviert ist und `--fargate` ein Fargate-Standardprofil erstellt wird. Dieses Profil zielt auf die `default` und die `kube-system` Namespaces ab, sodass Pods in diesen Namespaces auf Fargate laufen.

Das erstellte Fargate-Profil kann mit dem folgenden Befehl überprüft werden:

```
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
```

Weitere Informationen zu Selektoren finden Sie unter Entwerfen von Fargate-Profilen.

Erstellen eines Clusters mit Fargate-Unterstützung mithilfe einer Konfigurationsdatei

Die folgende Konfigurationsdatei deklariert einen EKS-Cluster mit einer Knotengruppe, die aus einer EC2 `m5.large` Instanz und zwei Fargate-Profilen besteht. Alle in den `kube-system` Namespaces `default` und definierten Pods werden auf Fargate ausgeführt. Alle Pods im `dev` Namespace, die auch das Label haben, `dev=passed` laufen auch auf Fargate. Alle anderen Pods werden auf dem Knoten in geplant. `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
```

Entwerfen von Fargate-Profilen

Jeder Selektoreintrag besteht aus bis zu zwei Komponenten, einem Namespace und einer Liste von Schlüssel-Wert-Paaren. Nur die Namespace-Komponente ist erforderlich, um einen Selektoreintrag zu erstellen. Alle Regeln (Namespaces, Schlüssel-Wert-Paare) müssen für einen Pod gelten, damit sie einem Selektoreintrag entsprechen. Ein Pod muss nur einem Selektoreintrag entsprechen, um auf dem Profil ausgeführt zu werden. Jeder Pod, der alle Bedingungen in einem Auswahlfeld erfüllt, würde für die Ausführung auf Fargate geplant. Alle Pods, die keinem der Namespaces auf der Whitelist entsprechen, bei denen der Benutzer jedoch manuell den Scheduler: Fargate-Scheduler

eingegeben hat, würden im Status Ausstehend hängen bleiben, da sie nicht autorisiert waren, auf Fargate ausgeführt zu werden.

Profile müssen die folgenden Anforderungen erfüllen:

- Pro Profil ist ein Selektor erforderlich
- Jeder Selektor muss einen Namespace enthalten; Beschriftungen sind optional

Beispiel: Planung der Arbeitslast in Fargate

Um Pods auf Fargate für das oben erwähnte Beispiel zu planen, könnte man beispielsweise einen Namespace namens erstellen `dev` und den Workload dort bereitstellen:

```
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
```

Aus der Ausgabe des letzten `kubectl get pods` Befehls können wir ersehen, dass der `nginx` Pod in einem Knoten namens bereitgestellt wird. `fargate-ip-192-168-183-140.ap-northeast-1.compute.internal`

Verwaltung von Fargate-Profilen

Um Kubernetes-Workloads auf Fargate bereitzustellen, benötigt EKS ein Fargate-Profil. Kümmert sich beim Erstellen eines Clusters wie in den obigen Beispielen darum, `eksctl` indem es ein Standardprofil erstellt. Bei einem bereits vorhandenen Cluster ist es auch möglich, ein Fargate-Profil mit dem `eksctl create fargateprofile` folgenden Befehl zu erstellen:

> **Note**
>
> Dieser Vorgang wird nur auf Clustern unterstützt, die auf der EKS-Plattformversion `eks.5` oder höher ausgeführt werden.

> **Note**
>
> Wenn das bestehende mit einer Version `eksctl` vor 0.11.0 erstellt wurde, müssen Sie es ausführen, `eksctl upgrade cluster` bevor Sie das Fargate-Profil erstellen.

```
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"
```

Sie können auch den Namen des zu erstellenden Fargate-Profils angeben. Dieser Name darf nicht mit dem Präfix `eks-` beginnen.

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

Wenn Sie diesen Befehl mit CLI-Flags verwenden, kann eksctl nur ein einzelnes Fargate-Profil mit einem einfachen Selektor erstellen. Für komplexere Selektoren, zum Beispiel mit mehr Namespaces, unterstützt eksctl die Verwendung einer Konfigurationsdatei:

```
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
```

Um bestehende Fargate-Profile in einem Cluster zu sehen:

```
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
```

Und um sie im `yaml` Format zu sehen:

```
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
```

Oder im `json` Format:

```
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"
        ]
    }
]
```

Fargate-Profile sind von Natur aus unveränderlich. Um etwas zu ändern, erstellen Sie ein neues Fargate-Profil mit den gewünschten Änderungen und löschen Sie das alte mit dem `eksctl delete fargateprofile` Befehl wie im folgenden Beispiel:

```
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"
}
```

Beachten Sie, dass das Löschen des Profils ein Vorgang ist, der bis zu einigen Minuten dauern kann. Wenn das `--wait` Flag nicht angegeben ist, geht `eksctl` optimistisch davon aus, dass das Profil gelöscht wird, und kehrt zurück, sobald die AWS-API-Anfrage gesendet wurde. Um zu `eksctl` warten, bis das Profil erfolgreich gelöscht wurde, verwenden Sie `--wait` es wie im obigen Beispiel.

Weitere Informationen

- AWS Fargate
- Amazon EKS kann jetzt Pods auf AWS Fargate starten

Cluster-Upgrades

Ein mit `eksctl` verwalteter Cluster kann in 3 einfachen Schritten aktualisiert werden:

1. Aktualisieren Sie die Version der Steuerungsebene mit `eksctl upgrade cluster`
2. Knotengruppen aktualisieren
3. aktualisieren Sie die standardmäßigen Netzwerk-Add-Ons (weitere Informationen finden Sie unterthe section called “Standard-Add-On-Updates”):

Lesen Sie die Ressourcen im Zusammenhang mit dem Cluster-Upgrade sorgfältig durch:

- Aktualisieren Sie den vorhandenen Cluster auf die neue Kubernetes-Version im Amazon EKS-Benutzerhandbuch
- Bewährte Methoden für Cluster-Upgrades finden Sie im EKS-Best-Practices-Leitfaden

> **Note**
>
> Die alte `eksctl update cluster` Version wird veraltet sein. Verwenden Sie stattdessen `eksctl upgrade cluster`.

Die Version der Steuerungsebene wird aktualisiert

Versionsupgrades auf der Kontrollebene müssen jeweils für eine Nebenversion durchgeführt werden.

Um die Steuerungsebene auf die nächste verfügbare Version zu aktualisieren, führen Sie folgenden Befehl aus:

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

Mit diesem Befehl werden die Änderungen nicht sofort übernommen. Sie müssen ihn erneut ausführen, um die Änderungen `--approve` zu übernehmen.

Die Zielversion für das Cluster-Upgrade kann sowohl mit dem CLI-Flag angegeben werden:

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

oder mit der Konfigurationsdatei

```
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**
>
> Die einzigen zulässigen Werte für die `metadata.version` Argumente `--version` und sind die aktuelle Version des Clusters oder eine höhere Version. Upgrades von mehr als einer Kubernetes-Version werden nicht unterstützt.

Standard-Add-On-Updates

In diesem Thema wird erklärt, wie die vorinstallierten Standard-Add-Ons aktualisiert werden, die in EKS-Clustern enthalten sind.

> **Warning**
>
> eksctl installiert jetzt Standard-Addons als EKS-Addons statt als selbstverwaltete Addons. Lesen Sie mehr über die Auswirkungen auf die Flexibilität bei der Clustererstellung für Standard-Netzwerk-Addons.
> Kann zum Aktualisieren von Addons `eksctl utils update-<addon>` nicht für Cluster verwendet werden, die mit eksctl v0.184.0 und höher erstellt wurden. Dieses Handbuch gilt nur für Cluster, die vor dieser Änderung erstellt wurden.

Es gibt 3 Standard-Add-Ons, die in jedem EKS-Cluster enthalten sind:

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

Aktualisieren Sie das vorinstallierte Add-On

Offizielle EKS-Addons, die manuell durch `eksctl create addons` oder bei der Clustererstellung erstellt werden, können wie folgt verwaltet werden. `eksctl create/get/update/delete addon` In solchen Fällen lesen Sie bitte in den Dokumenten zu EKS-Add-Ons nach.

Der Prozess für die Aktualisierung jedes von ihnen ist unterschiedlich, daher gibt es 3 verschiedene Befehle, die Sie ausführen müssen. Alle folgenden Befehle werden akzeptiert`--config-file`. Standardmäßig wird jeder dieser Befehle im Planmodus ausgeführt. Wenn Sie mit den vorgeschlagenen Änderungen zufrieden sind, führen Sie ihn erneut mit `--approve` aus.

Führen Sie zum Aktualisieren `kube-proxy` Folgendes aus:

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

Führen Sie zum Aktualisieren `aws-node` Folgendes aus:

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

Führen Sie zum Aktualisieren `coredns` Folgendes aus:

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

Stellen Sie nach dem Upgrade sicher, dass Sie alle Addon-Pods ausführen `kubectl get pods -n kube-system` und überprüfen, ob sie bereit sind. Sie sollten etwa Folgendes sehen:

```
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 für Zonal Shift in EKS-Clustern

EKS unterstützt jetzt Amazon Application Recovery Controller (ARC) Zonal Shift und Zonal Autoshift, wodurch die Stabilität von Multi-AZ-Cluster-Umgebungen verbessert wird. Mit AWS Zonal Shift können Kunden den Cluster-Traffic von einer beeinträchtigten Availability Zone wegverlagern und so sicherstellen, dass neue Kubernetes-Pods und -Knoten nur in gesunden Verfügbarkeitszonen gestartet werden.

Erstellen eines Clusters mit aktivierter Zonenverschiebung

```
# 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
```

Zonal Shift auf einem vorhandenen Cluster aktivieren

Führen Sie folgenden Befehl aus, um Zonal Shift auf einem vorhandenen Cluster zu aktivieren oder zu deaktivieren

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

oder ohne Konfigurationsdatei:

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

Weitere Informationen

- EKS Zonal Shift

Carpenter Support

`eksctl`unterstützt das Hinzufügen von Karpenter zu einem neu erstellten Cluster. Dadurch werden alle notwendigen Voraussetzungen geschaffen, die im Abschnitt Erste Schritte von Karpenter beschrieben sind, einschließlich der Installation von Karpenter selbst mithilfe von Helm. Wir unterstützen derzeit die Installation von Versionen ab und höher. `0.20.0`

Verwenden Sie das `eksctl` Cluster-Konfigurationsfeld`karpenter`, um es zu installieren und zu konfigurieren.

Das folgende Yaml beschreibt eine typische Installationskonfiguration:

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

metadata:
  name: cluster-with-karpenter
  region: us-west-2
  version: '1.24'
  tags:
    karpenter.sh/discovery: cluster-with-karpenter # here, it is set to the cluster
 name
iam:
  withOIDC: true # required

karpenter:
  version: 'v0.20.0' # Exact version must be specified

managedNodeGroups:
```

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

Die Version ist die Version von Karpenter, wie sie in ihrem Helm-Repository zu finden ist. Die folgenden Optionen können ebenfalls eingestellt werden:

```
karpenter:
  version: 'v0.20.0'
  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
```

OIDC muss definiert sein, um Karpenter zu installieren.

Sobald Karpenter erfolgreich installiert wurde, fügen Sie einen Provisioner hinzu, damit Karpenter beginnen kann, dem Cluster die richtigen Knoten hinzuzufügen.

Der `instanceProfile` Abschnitt des Provisioners muss mit dem Namen der erstellten Rolle übereinstimmen. `NodeInstanceProfile` Zum Beispiel:

```
apiVersion: karpenter.sh/v1alpha5
kind: Provisioner
metadata:
  name: default
spec:
  requirements:
    - key: karpenter.sh/capacity-type
      operator: In
      values: ["on-demand"]
  limits:
    resources:
      cpu: 1000
  provider:
    instanceProfile: eksctl-KarpenterNodeInstanceProfile-${CLUSTER_NAME}
    subnetSelector:
      karpenter.sh/discovery: cluster-with-karpenter # must match the tag set in the
 config file
    securityGroupSelector:
```

```
      karpenter.sh/discovery: cluster-with-karpenter # must match the tag set in the
config file
 ttlSecondsAfterEmpty: 30
```

Beachten Sie, dass der verwendete Name für `defaultInstanceProfile` ist, sofern er nicht definiert `instanceProfile` ist`eksctl-KarpenterNodeInstanceProfile-<cluster-name>`.

Cluster-Konfigurationsschema

> **Note**
>
> Der Speicherort des Schemas wird derzeit migriert.

Sie können eine Yaml-Datei verwenden, um einen Cluster zu erstellen. Sehen Sie sich die Schemareferenz an.

Zum Beispiel:

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

Die Schemareferenz für diese Datei ist unter verfügbar GitHub.

Weitere Hinweise zur Verwendung der Datei finden Sie unterthe section called “Erstellen und Verwalten von Clustern”.

Knotengruppen

Dieses Kapitel enthält Informationen darüber, wie Sie Nodegroups mit Eksctl erstellen und konfigurieren. Knotengruppen sind Gruppen von EC2 Instanzen, die an einen EKS-Cluster angehängt sind.

Themen:

- the section called “Spot-Instances”
  - Erstellen und verwalten Sie EKS-Cluster mit Spot-Instances mithilfe verwalteter Knotengruppen
  - Konfigurieren Sie Spot-Instances für nicht verwaltete Knotengruppen mithilfe der MixedInstancesPolicy
  - Unterscheiden Sie Spot- und On-Demand-Instances anhand des `node-lifecycle` Kubernetes-Labels
- the section called “Auto Scaling”
  - Ermöglichen Sie die automatische Skalierung von Kubernetes-Clusterknoten, indem Sie einen Cluster oder eine Knotengruppe mit IAM-Rolle erstellen, die die Verwendung des Cluster-Autoscalers ermöglicht
  - Konfigurieren Sie die Knotengruppendefinitionen so, dass sie die erforderlichen Tags und Anmerkungen enthalten, damit der Cluster-Autoscaler die Knotengruppe skalieren kann
  - Erstellen Sie separate Knotengruppen für jede Availability Zone, wenn Workloads zonenspezifische Anforderungen wie zonenspezifische Speicher- oder Affinitätsregeln haben
- the section called “Von EKS verwaltete Knotengruppen”
  - Bereitstellen und Verwalten von EC2 Instanzen (Knoten) für Amazon EKS Kubernetes-Cluster
  - Wenden Sie ganz einfach Bugfixes, Sicherheitspatches und Update-Knoten auf die neuesten Kubernetes-Versionen an
- the section called “EKS-Hybridknoten”
  - Ermöglichen Sie die Ausführung von lokalen und Edge-Anwendungen auf einer vom Kunden verwalteten Infrastruktur mit denselben AWS EKS-Clustern, Funktionen und Tools, die in der AWS-Cloud verwendet werden
  - Konfigurieren Sie Netzwerke, um lokale Netzwerke mit einer AWS-VPC zu verbinden, indem Sie Optionen wie AWS Site-to-Site VPN oder AWS Direct Connect verwenden

  - Richten Sie Anmeldeinformationen für Remote-Knoten zur Authentifizierung beim EKS-Cluster ein, indem Sie entweder AWS Systems Manager (SSM) oder AWS IAM Roles Anywhere verwenden
- the section called “Config für die Knotenreparatur”
  - Aktivierung von Node Repair für von EKS verwaltete Knotengruppen, um fehlerhafte Worker-Knoten automatisch zu überwachen und zu ersetzen oder neu zu starten
- the section called “ARM-Unterstützung”
  - Erstellen Sie einen EKS-Cluster mit ARM-basierten Graviton-Instances, um Leistung und Kosteneffizienz zu verbessern
- the section called “Makel”
  - Wenden Sie Taints auf bestimmte Knotengruppen in einem Kubernetes-Cluster an
  - Steuern Sie die Planung und Entfernung von Pods anhand von Taint-Schlüsseln, Werten und Effekten
- the section called “Support für Startvorlagen”
  - Verwaltete Knotengruppen mithilfe einer bereitgestellten Startvorlage starten EC2
  - Aktualisierung einer verwalteten Knotengruppe zur Verwendung einer anderen Version einer Startvorlage
  - Grundlegendes zu Einschränkungen und Überlegungen bei der Verwendung von benutzerdefinierten Vorlagen AMIs und Startvorlagen mit verwalteten Knotengruppen
- the section called “Arbeiten Sie mit Knotengruppen”
  - Aktivieren Sie den SSH-Zugriff auf EC2 Instanzen in der Knotengruppe
  - Skalieren Sie die Anzahl der Knoten in einer Knotengruppe nach oben oder unten
- the section called “Benutzerdefinierte Subnetze”
  - Erweitern Sie eine bestehende VPC um ein neues Subnetz und fügen Sie diesem Subnetz eine Nodegroup hinzu
- the section called “Knoten-Bootstrapping”
  - Machen Sie sich mit dem neuen Knoteninitialisierungsprozess (nodeadm) vertraut, der 2023 eingeführt wurde AmazonLinux
  - Erfahren Sie mehr über die NodeConfig Standardeinstellungen, die von eksctl für selbstverwaltete und EKS-verwaltete Knoten angewendet werden
  - Passen Sie den Knoten-Bootstrapping-Prozess an, indem Sie einen benutzerdefinierten overrideBootstrapCommand NodeConfig

- the section called “Nicht verwaltete Knotengruppen”
  - Erstellen oder aktualisieren Sie nicht verwaltete Knotengruppen in einem EKS-Cluster
  - Aktualisieren Sie Standard-Kubernetes-Add-Ons wie kube-proxy, aws-node und CoreDNS
- the section called “GPU-Unterstützung”
  - Eksctl unterstützt die Auswahl von GPU-Instanztypen für Knotengruppen und ermöglicht so die Verwendung von GPU-beschleunigten Workloads auf EKS-Clustern.
  - Eksctl installiert das NVIDIA Kubernetes-Geräte-Plug-In automatisch, wenn ein GPU-fähiger Instanztyp ausgewählt wird, und erleichtert so die Nutzung von GPU-Ressourcen im Cluster.
  - Benutzer können die automatische Plugin-Installation deaktivieren und mithilfe der bereitgestellten Befehle manuell eine bestimmte Version des NVIDIA Kubernetes-Geräte-Plug-ins installieren.
- the section called “Instanzenauswahl”
  - Generieren Sie automatisch eine Liste geeigneter EC2 Instanztypen auf der Grundlage von Ressourcenkriterien wie V CPUs GPUs, Arbeitsspeicher und CPU-Architektur
  - Erstellen Sie Cluster und Knotengruppen mit den Instanztypen, die den angegebenen Instanzauswahlkriterien entsprechen
  - Führen Sie vor dem Erstellen einer Knotengruppe einen Probelauf durch, um die Instanztypen zu überprüfen und zu ändern, denen der Instanzselektor entspricht
- the section called “Zusätzliche Volume-Zuordnungen”
  - Konfigurieren Sie zusätzliche Volume-Zuordnungen für eine verwaltete Knotengruppe in einem EKS-Cluster
  - Passen Sie Volume-Eigenschaften wie Größe, Typ, Verschlüsselung, IOPS und Durchsatz für die zusätzlichen Volumes an
  - Hängen Sie vorhandene EBS-Snapshots als zusätzliche Volumes an die Knotengruppe an
- the section called “Windows Worker-Knoten”
  - Fügen Sie Windows-Knotengruppen zu einem vorhandenen Linux-Kubernetes-Cluster hinzu, um die Ausführung von Windows-Workloads zu ermöglichen
  - Planen Sie Workloads auf dem entsprechenden Betriebssystem (Windows oder Linux) mithilfe von Knotenselektoren auf der Grundlage der Labels und `kubernetes.io/os` `kubernetes.io/arch`
- the section called “Benutzerdefinierte AMI-Unterstützung”

- Verwenden Sie das `--node-ami` Flag, um ein benutzerdefiniertes AMI für Knotengruppen anzugeben, AWS nach dem neuesten EKS-optimierten AMI abzufragen oder AWS Systems Manager Parameter Store zu verwenden, um das AMI zu finden.
- Setzen Sie das `--node-ami-family` Flag, um die Betriebssystemfamilie für das Knotengruppen-AMI anzugeben, z. B. AmazonLinux 2, Ubuntu2204 oder 2022. WindowsServer CoreContainer
- Geben Sie für Windows-Knotengruppen ein benutzerdefiniertes AMI an und stellen Sie ein PowerShell Bootstrap-Skript über den `overrideBootstrapCommand` bereit.

- the section called “Custom DNS”
  - Überschreiben Sie die IP-Adresse des DNS-Servers, die für interne und externe DNS-Suchvorgänge verwendet wird

Arbeiten Sie mit Knotengruppen

Knotengruppen erstellen

Sie können zusätzlich zu der ursprünglichen Knotengruppe, die zusammen mit dem Cluster erstellt wurde, eine oder mehrere Knotengruppen hinzufügen.

Um eine zusätzliche Knotengruppe zu erstellen, verwenden Sie:

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

> **Note**
>
> `--version`Flag wird für verwaltete Knotengruppen nicht unterstützt. Es erbt immer die Version von der Steuerungsebene.

Standardmäßig erben neue, nicht verwaltete Knotengruppen die Version von der Steuerungsebene (`--version=auto`). Sie können jedoch auch eine andere Version angeben und damit die Verwendung der jeweils neuesten Version `--version=latest` erzwingen.

Darüber hinaus können Sie dieselbe Konfigurationsdatei verwenden, die für Folgendes verwendet wurde: `eksctl create cluster`

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

Eine Knotengruppe aus einer Konfigurationsdatei erstellen

Knotengruppen können auch über eine Clusterdefinition oder eine Konfigurationsdatei erstellt werden. Ausgehend von der folgenden Beispiel-Konfigurationsdatei und einem vorhandenen Cluster mit dem Namen: `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
```

Die Nodegroups `ng-1-workers` und `ng-2-builders` können mit diesem Befehl erstellt werden:

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

Lastausgleich

Wenn Sie sich bereits darauf vorbereitet haben, bestehende Classic Load or/and Balancer-Zielgruppen an die Nodegroups anzuhängen, können Sie diese in der Konfigurationsdatei angeben. Die or/and Zielgruppen der klassischen Load Balancer werden bei der Erstellung von Nodegroups automatisch mit der ASG verknüpft. Dies wird nur für selbstverwaltete Knotengruppen unterstützt, die über das Feld definiert wurden. `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
```

Auswahl der Knotengruppe in den Konfigurationsdateien

Um eine `create` `delete` Oder-Operation nur für eine Teilmenge der in einer Konfigurationsdatei angegebenen Knotengruppen auszuführen, gibt es zwei CLI-Flags, die eine Liste von Globs akzeptieren, `0` und z. B.: `1`

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

Mit der obigen Beispielkonfigurationsdatei kann man mit dem folgenden Befehl alle Worker-Knotengruppen außer der Worker-Knotengruppe erstellen:

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

Oder man könnte die Builder-Knotengruppe löschen mit:

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

In diesem Fall müssen wir auch den `--approve` Befehl angeben, um die Knotengruppe tatsächlich zu löschen.

Regeln einschließen und ausschließen

- wenn kein `--include` oder angegeben `--exclude` ist, ist alles enthalten
- wenn only angegeben `--include` ist, werden nur Knotengruppen eingeschlossen, die diesen Globs entsprechen
- wenn only angegeben `--exclude` ist, werden alle Knotengruppen eingeschlossen, die nicht zu diesen Globs passen
- wenn beide angegeben sind, haben `--exclude` Regeln Vorrang vor `--include` (d. h. Knotengruppen, die den Regeln in beiden Gruppen entsprechen, werden ausgeschlossen)

Knotengruppen auflisten

Um die Details zu einer Knotengruppe oder allen Knotengruppen aufzulisten, verwenden Sie:

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

Um eine oder mehrere Knotengruppen im YAML- oder JSON-Format aufzulisten, das mehr Informationen als die Standard-Logtabelle ausgibt, verwenden Sie:

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

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

Unveränderlichkeit der Knotengruppe

Knotengruppen sind konstruktionsbedingt unveränderlich. Das heißt, wenn Sie etwas (außer Skalierung) ändern müssen, wie das AMI oder den Instance-Typ einer Knotengruppe, müssten Sie eine neue Knotengruppe mit den gewünschten Änderungen erstellen, die Last verschieben und die alte löschen. Weitere Informationen finden Sie im Abschnitt Löschen und Löschen von Knotengruppen.

Skalieren von Knotengruppen

Die Skalierung von Knotengruppen ist ein Vorgang, der bis zu einigen Minuten dauern kann. Wenn das `--wait` Flag nicht angegeben ist, erwartet `eksctl` optimistisch, dass die Knotengruppe skaliert wird, und kehrt zurück, sobald die AWS-API-Anfrage gesendet wurde. Um zu `eksctl` warten, bis die Knoten verfügbar sind, fügen Sie ein `--wait` Flag wie im folgenden Beispiel hinzu.

> **Note**
>
> Die Skalierung einer Knotengruppe down/in (d. h. die Reduzierung der Anzahl der Knoten) kann zu Fehlern führen, da wir uns ausschließlich auf Änderungen an der ASG verlassen. Das bedeutet, dass die Knoten, die gerade bearbeitet werden, removed/terminated nicht explizit entleert werden. Dies könnte in future verbesserungswürdig sein.

Die Skalierung einer verwalteten Knotengruppe erfolgt durch den direkten Aufruf der EKS-API, die die Konfiguration einer verwalteten Knotengruppe aktualisiert.

Skalierung einer einzelnen Knotengruppe

Eine Knotengruppe kann mit dem folgenden Befehl skaliert werden: `eksctl scale nodegroup`

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

Um beispielsweise eine Knotengruppe auf 5 Knoten zu skalieren, führen Sie folgenden `ng-a345f4e1` Befehl `cluster-1` aus:

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

Eine Knotengruppe kann auch skaliert werden, indem eine Konfigurationsdatei verwendet wird, die an die Knotengruppe übergeben wird, `--config-file` und indem der Name der Knotengruppe angegeben wird, mit der skaliert werden soll. `--name` Eksctl durchsucht die Konfigurationsdatei und entdeckt diese Knotengruppe sowie ihre Skalierungskonfigurationswerte.

Wenn die gewünschte Anzahl von Knoten NOT innerhalb des Bereichs der aktuellen minimalen und aktuellen maximalen Anzahl von Knoten liegt, wird ein bestimmter Fehler angezeigt. Diese Werte können auch mit Flags `--nodes-min` und `--nodes-max` entsprechend übergeben werden.

Skalierung mehrerer Knotengruppen

Eksctl kann alle Knotengruppen in einer mitgelieferten Konfigurationsdatei erkennen und skalieren. `--config-file`

Ähnlich wie bei der Skalierung einer einzelnen Knotengruppe gelten für jede Knotengruppe die gleichen Validierungen. Beispielsweise muss die gewünschte Anzahl von Knoten im Bereich der minimalen und maximalen Anzahl von Knoten liegen.

Löschen und Entleeren von Knotengruppen

Um eine Knotengruppe zu löschen, führen Sie folgenden Befehl aus:

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

Mit diesem Befehl können auch Regeln zum Einschließen und Ausschließen verwendet werden.

> **Note**
>
> Dadurch werden alle Pods aus dieser Knotengruppe gelöscht, bevor die Instanzen gelöscht werden.

Um die Räumungsregeln während des Löschvorgangs zu überspringen, führen Sie folgenden Befehl aus:

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

Alle Knoten sind gesperrt und alle Pods werden beim Löschen aus einer Knotengruppe entfernt. Wenn Sie jedoch eine Knotengruppe leeren müssen, ohne sie zu löschen, führen Sie folgenden Befehl aus:

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

Um eine Knotengruppe abzukoppeln, führen Sie folgenden Befehl aus:

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

Um Räumungsregeln wie Einstellungen zu ignorieren, führen Sie folgenden Befehl aus: PodDisruptionBudget

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

Um den Entleerungsprozess zu beschleunigen, können Sie `--parallel <value>` die Anzahl der Knoten angeben, die parallel entleert werden sollen.

Weitere Funktionen

Sie können auch SSH-, ASG-Zugriff und andere Funktionen für eine Knotengruppe aktivieren, z. B.:

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

Beschriftungen aktualisieren

Es gibt keine speziellen Befehle`eksctl`, um die Labels einer Knotengruppe zu aktualisieren, aber es kann leicht erreicht werden`kubectl`, z. B. mit:

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

SSH-Zugang

Sie können den SSH-Zugriff für Knotengruppen aktivieren, indem Sie eine von `publicKeyName` und `publicKeyPath` in Ihrer `publicKey` Knotengruppen-Konfiguration konfigurieren. Alternativ können

Sie AWS Systems Manager (SSM) verwenden, um per SSH auf Knoten zuzugreifen, indem Sie die Knotengruppe wie folgt konfigurieren: `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
```

Nicht verwaltete Knotengruppen

Durch `eksctl` das Setzen `--managed=false` oder Verwenden des `nodeGroups` Felds wird eine nicht verwaltete Knotengruppe erstellt. Beachten Sie, dass nicht verwaltete Knotengruppen nicht in der EKS-Konsole angezeigt werden, die in der Regel nur über EKS-verwaltete Knotengruppen Bescheid weiß.

Sie sollten die Knotengruppen erst aktualisieren, nachdem Sie sie ausgeführt haben. `eksctl upgrade cluster` (Siehe Aktualisieren von Clustern.)

Wenn Sie einen einfachen Cluster mit nur einer anfänglichen Knotengruppe haben (d. h. mit erstellt`eksctl create cluster`), ist der Vorgang sehr einfach:

1. Holen Sie sich den Namen der alten Knotengruppe:

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

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

2. Erstellen Sie eine neue Knotengruppe:

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

3. Lösche die alte Knotengruppe:

```
 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.
> ```

Aktualisierung mehrerer Knotengruppen

Wenn Sie mehrere Knotengruppen haben, liegt es in Ihrer Verantwortung, nachzuverfolgen, wie jede einzelne konfiguriert wurde. Sie können dies mithilfe von Konfigurationsdateien tun, aber wenn Sie sie noch nicht verwendet haben, müssen Sie Ihren Cluster überprüfen, um herauszufinden, wie die einzelnen Knotengruppen konfiguriert wurden.

Im Allgemeinen möchten Sie:

- überprüfen Sie, welche Knotengruppen Sie haben und welche gelöscht werden können oder für die neue Version ersetzt werden müssen
- notieren Sie sich die Konfiguration jeder Knotengruppe. Erwägen Sie, beim nächsten Mal die Konfigurationsdatei zu verwenden, um Upgrades zu vereinfachen

Aktualisierung mit der Konfigurationsdatei

Wenn Sie die Konfigurationsdatei verwenden, müssen Sie wie folgt vorgehen.

Bearbeiten Sie die Konfigurationsdatei, um neue Knotengruppen hinzuzufügen und alte Knotengruppen zu entfernen. Wenn Sie nur Knotengruppen aktualisieren und dieselbe Konfiguration beibehalten möchten, können Sie einfach die Namen der Knotengruppen ändern, z. B. sie an den Namen anhängen. `-v2`

Um alle neuen Knotengruppen zu erstellen, die in der Konfigurationsdatei definiert sind, führen Sie folgenden Befehl aus:

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

Sobald Sie neue Knotengruppen eingerichtet haben, können Sie alte löschen:

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

> **Note**
>
> Die erste Ausführung erfolgt im Planmodus. Wenn Sie mit den vorgeschlagenen Änderungen zufrieden sind, führen Sie die Ausführung erneut mit aus. `--approve`

Standard-Add-Ons werden aktualisiert

Möglicherweise müssen Sie die auf Ihrem Cluster installierten Netzwerk-Add-Ons aktualisieren. Weitere Informationen finden Sie unter the section called “Standard-Add-On-Updates”.

Von EKS verwaltete Knotengruppen

Die von Amazon EKS verwalteten Knotengruppen sind eine Funktion, die die Bereitstellung und das Lebenszyklusmanagement von Knoten (EC2 Instances) für Amazon EKS Kubernetes-Cluster automatisiert. Kunden können optimierte Knotengruppen für ihre Cluster bereitstellen, und EKS hält ihre Knoten mit den neuesten Kubernetes- und Host-Betriebssystemversionen auf dem neuesten Stand.

Eine von EKS verwaltete Knotengruppe ist eine Autoscaling-Gruppe und zugehörige EC2 Instances, die von AWS für einen Amazon EKS-Cluster verwaltet werden. Jede Knotengruppe verwendet das Amazon EKS-optimierte Amazon Linux 2-AMI. Amazon EKS macht es einfach, Bugfixes und Sicherheitspatches auf Knoten anzuwenden und sie auf die neuesten Kubernetes-Versionen zu aktualisieren. Jede Knotengruppe startet eine Autoscaling-Gruppe für Ihren Cluster, die sich über mehrere AWS-VPC-Verfügbarkeitszonen und Subnetze erstrecken kann, um eine hohe Verfügbarkeit zu gewährleisten.

NEU: Launch Template-Unterstützung für verwaltete Knotengruppen

> **Note**
>
> Der Begriff „nicht verwaltete Knotengruppen“ wurde verwendet, um sich auf Knotengruppen zu beziehen, die eksctl von Anfang an unterstützt hat (dargestellt durch das Feld). `nodeGroups` Die `ClusterConfig` Datei verwendet das `nodeGroups` Feld weiterhin zur Definition von nicht verwalteten Knotengruppen, und verwaltete Knotengruppen werden mit dem Feld definiert. `managedNodeGroups`

Verwaltete Knotengruppen erstellen

```
$ eksctl create nodegroup
```

Neue Cluster

Führen Sie folgenden Befehl aus, um einen neuen Cluster mit einer verwalteten Knotengruppe zu erstellen

```
eksctl create cluster
```

Um mehrere verwaltete Knotengruppen zu erstellen und mehr Kontrolle über die Konfiguration zu haben, kann eine Konfigurationsdatei verwendet werden.

> **Note**
>
> Verwaltete Knotengruppen haben keine vollständige Funktionsparität mit nicht verwalteten Knotengruppen.

```
# 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
```

Ein weiteres Beispiel für eine Konfigurationsdatei zum Erstellen einer verwalteten Knotengruppe finden Sie hier.

Es ist möglich, einen Cluster mit verwalteten und nicht verwalteten Knotengruppen zu haben. Nicht verwaltete Knotengruppen `eksctl get nodegroup` werden nicht in der AWS EKS-Konsole angezeigt, sondern listen beide Arten von Knotengruppen auf.

```
# 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
```

NEU Support für benutzerdefiniertes AMI, Sicherheitsgruppen`instancePrefix`,`instanceName`,`ebsOptimized`,`volumeType`,`volumeName`,`volume` und `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'
```

Wenn Sie einen Instance-Typ anfordern, der nur in einer Zone verfügbar ist (und die eksctl-Konfiguration erfordert die Angabe von zwei), stellen Sie sicher, dass Sie die Availability Zone zu Ihrer Knotengruppen-Anfrage hinzufügen:

```
# 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
```

Dies kann für Instance-Typen wie die Hpc6-Familie gelten, die nur in einer Zone verfügbar sind.

Bestehende Cluster

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

Tipp: Wenn Sie eine `ClusterConfig` Datei verwenden, um Ihren gesamten Cluster zu beschreiben, beschreiben Sie Ihre neue verwaltete Knotengruppe im `managedNodeGroups` Feld und führen Sie Folgendes aus:

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

Verwaltete Knotengruppen aktualisieren

Sie können eine Knotengruppe jederzeit auf die neueste EKS-optimierte AMI-Release-Version für den von Ihnen verwendeten AMI-Typ aktualisieren.

Wenn Ihre Knotengruppe dieselbe Kubernetes-Version wie der Cluster hat, können Sie auf die neueste AMI-Release-Version für diese Kubernetes-Version des AMI-Typs aktualisieren, den Sie

verwenden. Wenn Ihre Knotengruppe die vorherige Kubernetes-Version der Kubernetes-Version des Clusters ist, können Sie die Knotengruppe auf die neueste AMI-Release-Version aktualisieren, die der Kubernetes-Version der Knotengruppe entspricht, oder auf die neueste AMI-Release-Version aktualisieren, die der Kubernetes-Version des Clusters entspricht. Sie können eine Knotengruppe nicht auf eine frühere Kubernetes-Version zurücksetzen.

Um eine verwaltete Knotengruppe auf die neueste AMI-Release-Version zu aktualisieren:

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

Die Knotengruppe kann wie folgt auf die neueste AMI-Version für eine bestimmte Kubernetes-Version aktualisiert werden:

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

Um auf eine bestimmte AMI-Release-Version statt auf die neueste Version zu aktualisieren, übergeben Sie`--release-version`:

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

> **Note**
>
> Wenn die verwalteten Knoten benutzerdefiniert bereitgestellt werden AMIs, muss der folgende Workflow befolgt werden, um eine neue Version des benutzerdefinierten AMI bereitzustellen.

- Die anfängliche Bereitstellung der Knotengruppe muss mithilfe einer Startvorlage erfolgen. z. B.

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

- eine neue Version des benutzerdefinierten AMI erstellen (mithilfe der AWS EKS-Konsole).

- eine neue Version der Startvorlage mit der neuen AMI-ID erstellen (mithilfe der AWS EKS-Konsole).
- führen Sie ein Upgrade der Knoten auf die neue Version der Startvorlage durch. z. B.

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

Handhabung parallel Upgrades für Knoten

Mehrere verwaltete Knoten können gleichzeitig aktualisiert werden. Um parallel Upgrades zu konfigurieren, definieren Sie beim Erstellen `updateConfig` der Knotengruppe die einer Knotengruppe. Ein Beispiel finden `updateConfig` Sie hier.

Um Ausfallzeiten Ihrer Workloads aufgrund der gleichzeitigen Aktualisierung mehrerer Knoten zu vermeiden, können Sie die Anzahl der Knoten begrenzen, die während eines Upgrades nicht verfügbar sein können, indem Sie dies im `maxUnavailable` Feld für angeben. `updateConfig` Verwenden Sie alternativ`maxUnavailablePercentage`, wodurch die maximale Anzahl nicht verfügbarer Knoten als Prozentsatz der Gesamtzahl der Knoten definiert wird.

Beachten Sie, dass dieser Wert `maxUnavailable` nicht höher sein kann als`maxSize`. Ebenfalls `maxUnavailable` und `maxUnavailablePercentage` kann nicht gleichzeitig verwendet werden.

Diese Funktion ist nur für verwaltete Knoten verfügbar.

Verwaltete Knotengruppen werden aktualisiert

`eksctl`ermöglicht das Aktualisieren des UpdateConfigAbschnitts einer verwalteten Knotengruppe. Dieser Abschnitt definiert zwei Felder. `MaxUnavailable`und`MaxUnavailablePercentage`. Ihre Knotengruppen sind während des Updates nicht betroffen, sodass mit Ausfallzeiten nicht zu rechnen ist.

Der Befehl `update nodegroup` sollte mit einer Konfigurationsdatei unter Verwendung des Flags verwendet werden. `--config-file` Die Nodegroup sollte einen `nodeGroup.updateConfig` Abschnitt enthalten. Weitere Informationen finden Sie hier.

Gesundheitsprobleme bei Nodegroup

EKS Managed Nodegroups überprüft automatisch die Konfiguration Ihrer Knotengruppe und Knoten auf Integritätsprobleme und meldet diese über die EKS-API und -Konsole. So zeigen Sie Gesundheitsprobleme für eine Knotengruppe an:

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

Labels verwalten

EKS Managed Nodegroups unterstützt das Anhängen von Labels, die auf die Kubernetes-Knoten in der Knotengruppe angewendet werden. Dies wird während der Cluster- oder Knotengruppenerstellung über das `labels` Feld in eksctl angegeben.

Um neue Labels zu setzen oder bestehende Labels für eine Knotengruppe zu aktualisieren:

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

So löschen oder entfernen Sie Labels aus einer Knotengruppe:

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

So zeigen Sie alle Labels an, die für eine Knotengruppe gesetzt sind:

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

Skalierung verwalteter Knotengruppen

`eksctl scale nodegroup`unterstützt auch verwaltete Knotengruppen. Die Syntax für die Skalierung einer verwalteten oder nicht verwalteten Knotengruppe ist dieselbe.

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

Weitere Informationen

- Von EKS verwaltete Knotengruppen

Knoten-Bootstrapping

AmazonLinux2023

AL2023 führte einen neuen Knoteninitialisierungsprozess nodeadm ein, der ein YAML-Konfigurationsschema verwendet und die Verwendung von Skripten überflüssig macht. `/etc/eks/bootstrap.sh`

> **Note**
>
> Mit den Kubernetes-Versionen 1.30 und höher ist Amazon Linux 2023 das Standardbetriebssystem.

Standardeinstellungen für AL2

Für selbstverwaltete Knoten und EKS-verwaltete Knoten, die auf benutzerdefinierten Knoten basieren AMIs, `eksctl` wird ein Standard `NodeConfig` -, Minimalwert erstellt und dieser automatisch in die Benutzerdaten der Startvorlage der Knotengruppen eingefügt, d. h.

```
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

--//--
```

Für EKS-verwaltete Knoten, die auf systemeigenen Knoten basieren AMIs, `NodeConfig` wird der Standard von EKS MNG unter der Haube hinzugefügt und direkt an die Benutzerdaten von angehängt. EC2 Daher muss es in diesem Szenario `eksctl` nicht in die Startvorlage aufgenommen werden.

Konfiguration des Bootstrapping-Prozesses

Um erweiterte Eigenschaften von `NodeConfig` festzulegen oder einfach die Standardwerte zu überschreiben, können Sie mit eksctl ein benutzerdefiniertes via oder z. `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
```

Diese benutzerdefinierte Konfiguration wird den Benutzerdaten von eksctl vorangestellt und mit der Standardkonfiguration zusammengeführt. `nodeadm` Lesen Sie hier mehr über die Fähigkeit`nodeadm`, mehrere Konfigurationsobjekte zusammenzuführen.

Starten Sie die Vorlagenunterstützung für verwaltete Knotengruppen

eksctl unterstützt das Starten verwalteter Knotengruppen mithilfe einer bereitgestellten Startvorlage. EC2 Dies ermöglicht mehrere Anpassungsoptionen für Knotengruppen, einschließlich der Bereitstellung von benutzerdefinierten Gruppen AMIs und Sicherheitsgruppen sowie der Weitergabe von Benutzerdaten für das Knoten-Bootstrapping.

Verwaltete Knotengruppen mithilfe einer bereitgestellten Startvorlage erstellen

```
# 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
```

Aktualisierung einer verwalteten Knotengruppe zur Verwendung einer anderen Version der Startvorlage

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

> **Note**
>
> Wenn eine Startvorlage ein benutzerdefiniertes AMI verwendet, sollte die neue Version auch ein benutzerdefiniertes AMI verwenden, da sonst der Upgrade-Vorgang fehlschlägt

Wenn eine Startvorlage kein benutzerdefiniertes AMI verwendet, kann auch die Kubernetes-Version angegeben werden, auf die das Upgrade durchgeführt werden soll:

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

Hinweise zur Unterstützung von benutzerdefinierten AMIs und Startvorlagen

- Wenn eine Startvorlage bereitgestellt wird, werden die folgenden Felder nicht unterstützt:`instanceType`,,`ami`,`ssh.allow`,`ssh.sourceSecurityGroupIds`,`securityGroups`,`inst`overrideBootstrapCommand` und`disableIMDSv1`.
- Wenn Sie ein benutzerdefiniertes AMI (`ami`) verwenden, `overrideBootstrapCommand` muss es auch so eingestellt werden, dass es das Bootstrapping durchführt.
- `overrideBootstrapCommand`kann nur gesetzt werden, wenn ein benutzerdefiniertes AMI verwendet wird.
- Wenn eine Startvorlage bereitgestellt wird, gelten die in der Nodegroup-Konfiguration angegebenen Tags nur für die EKS-Nodegroup-Ressource und werden nicht an Instances weitergegeben. EC2

Benutzerdefinierte Subnetze

Es ist möglich, eine bestehende VPC um ein neues Subnetz zu erweitern und diesem Subnetz eine Nodegroup hinzuzufügen.

Warum

Sollte der Cluster nicht mehr vorkonfiguriert sein, ist es möglich IPs, die Größe der vorhandenen VPC mit einem neuen CIDR zu ändern, um ihr ein neues Subnetz hinzuzufügen. Um zu erfahren, wie das geht, lesen Sie diesen Leitfaden zu AWS Extending VPCs.

TL; DR

Gehen Sie zur VPC-Konfiguration und fügen Sie sie hinzu, klicken Sie auf Aktionen->Bearbeiten CIDRs und fügen Sie einen neuen Bereich hinzu. Zum Beispiel:

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

Jetzt müssen Sie ein neues Subnetz hinzufügen. Je nachdem, ob es sich um ein neues privates oder ein öffentliches Subnetz handelt, müssen Sie die Routing-Informationen aus einem privaten bzw. einem öffentlichen Subnetz kopieren.

Sobald das Subnetz erstellt ist, fügen Sie Routing hinzu und kopieren Sie entweder die NAT-Gateway-ID oder das Internet Gateway aus einem anderen Subnetz in der VPC. Achten Sie darauf, dass, wenn es sich um ein öffentliches Subnetz handelt, die automatische IP-Zuweisung aktiviert ist. Aktionen->IP-Einstellungen für automatische Zuweisung ändern-> Automatische Zuweisung öffentlicher Adresse aktivieren. IPv4

Vergessen Sie nicht, je nach Konfiguration des öffentlichen oder privaten Subnetzes auch die TAGS der vorhandenen Subnetze zu kopieren. Dies ist wichtig, da das Subnetz sonst nicht Teil des Clusters ist und Instances im Subnetz nicht beitreten können.

Wenn Sie fertig sind, kopieren Sie die ID des neuen Subnetzes. Wiederholen Sie den Vorgang so oft wie nötig.

Wie

Führen Sie den folgenden Befehl aus, um eine Knotengruppe in den erstellten Subnetzen zu erstellen:

```
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
```

Oder verwenden Sie die Konfiguration als solche:

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

Mit einer Konfiguration wie dieser:

```
# 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
```

Warten Sie, bis die Knotengruppe erstellt wurde und die neuen Instanzen die neuen IP-Bereiche der Subnetze haben sollten.

Den Cluster löschen

Da der neue Zusatz die bestehende VPC durch Hinzufügen einer Abhängigkeit außerhalb des CloudFormation Stacks modifiziert hat, CloudFormation kann der Cluster nicht mehr entfernt werden.

Bevor Sie den Cluster löschen, entfernen Sie alle erstellten zusätzlichen Subnetze von Hand und fahren Sie dann fort, indem Sie Folgendes aufrufen: `eksctl`

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

Custom DNS

Es gibt zwei Möglichkeiten, die IP-Adresse des DNS-Servers zu überschreiben, die für alle internen und externen DNS-Suchvorgänge verwendet wird. Dies entspricht dem `--cluster-dns` Flag für. `kubelet`

Die erste ist durch das `clusterDNS` Feld. Die Konfigurationsdateien akzeptieren ein `string` Feld, das `clusterDNS` mit der IP-Adresse des zu verwendenden DNS-Servers aufgerufen wird. Dies

wird an das übergebenkubelet, das es wiederum über die /etc/resolv.conf Datei an die Pods weiterleitet. Weitere Informationen finden Sie im Schema der Konfigurationsdatei.

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

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

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

Beachten Sie, dass diese Konfiguration nur eine IP-Adresse akzeptiert. Verwenden Sie den folgenden kubeletExtraConfigParameter, um mehr als eine Adresse anzugeben:

```
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"]
```

Makel

Um Taints auf eine bestimmte Knotengruppe anzuwenden, verwende den taints Konfigurationsabschnitt wie folgt:

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

```
        effect: NoExecute
```

Ein vollständiges Beispiel finden Sie hier.

Instanzenauswahl

eksctl unterstützt die Angabe mehrerer Instanztypen für verwaltete und selbstverwaltete Knotengruppen. Bei über 270 EC2 Instanztypen müssen Benutzer jedoch Zeit damit verbringen, herauszufinden, welche Instanztypen für ihre Knotengruppe am besten geeignet sind. Bei der Verwendung von Spot-Instances ist es noch schwieriger, da Sie eine Reihe von Instances auswählen müssen, die gut mit dem Cluster Autoscaler zusammenarbeiten.

eksctl ist jetzt in den EC2 Instance-Selektor integriert, der dieses Problem behebt, indem eine Liste von Instance-Typen auf der Grundlage von Ressourcenkriterien generiert wird: vCPUs, memory, # of und CPU-Architektur. GPUs Wenn die Kriterien für die Instanzauswahl erfüllt sind, erstellt eksctl eine Knotengruppe, in der die Instanztypen auf die Instanztypen gesetzt sind, die den angegebenen Kriterien entsprechen.

Erstellen Sie Cluster und Knotengruppen

Führen Sie folgenden Befehl aus, um einen Cluster mit einer einzelnen Knotengruppe zu erstellen, der Instanztypen verwendet, die den an eksctl übergebenen Ressourcenkriterien für die Instanzauswahl entsprechen

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

Dadurch werden ein Cluster und eine verwaltete Knotengruppe erstellt, wobei das `instanceTypes` Feld auf gesetzt ist `[c5.large, c5a.large, c5ad.large, c5d.large, t2.medium, t3.medium, t3a.medium]` (der Satz der zurückgegebenen Instanztypen kann sich ändern).

Für nicht verwaltete Knotengruppen wird das `instancesDistribution.instanceTypes` Feld wie folgt gesetzt:

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

Die Kriterien für die Instanzauswahl können auch angegeben werden 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
```

Die folgenden CLI-Optionen für die Instanzauswahl werden von `eksctl create cluster` und `eksctl create nodegroup` unterstützt:

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

Eine Beispieldatei finden Sie hier.

Probelauf

Mit der Dry-Run-Funktion können Sie die Instanzen überprüfen und ändern, denen die Instanzauswahl entspricht, bevor Sie mit der Erstellung einer Knotengruppe fortfahren.

```
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
```

Die generierten Daten ClusterConfig können dann übergeben werden an: `eksctl create cluster`

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

Das `instanceSelector` Feld, das die CLI-Optionen darstellt, wird der ClusterConfig Datei aus Gründen der Sichtbarkeit und Dokumentation ebenfalls hinzugefügt. Wenn `--dry-run` es weggelassen wird, wird dieses Feld ignoriert und das `instanceTypes` Feld wird verwendet, andernfalls `instanceTypes` würden alle Änderungen an von eksctl überschrieben.

Wenn eine ClusterConfig Datei mit übergeben wird, gibt eksctl eine ClusterConfig Datei aus`--dry-run`, die denselben Satz von Knotengruppen enthält, nachdem die Ressourcenkriterien für die Instanzauswahl jeder Knotengruppe erweitert wurden.

```
# 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
# ...
```

Spot-Instances

Verwaltete Knotengruppen

`eksctl`unterstützt Spot-Worker-Knoten mithilfe von EKS Managed Nodegroups, einer Funktion, die es EKS-Kunden mit fehlertoleranten Anwendungen ermöglicht, EC2 Spot-Instances für ihre EKS-Cluster einfach bereitzustellen und zu verwalten. EKS Managed Nodegroup konfiguriert und startet eine EC2 Autoscaling-Gruppe von Spot-Instances gemäß den Best Practices von Spot und entleert Spot-Worker-Knoten automatisch, bevor die Instances von AWS unterbrochen werden. Für die Nutzung dieser Funktion fallen keine zusätzlichen Gebühren an, und Kunden zahlen nur für die Nutzung der AWS-Ressourcen wie EC2 Spot-Instances und EBS-Volumes.

Um mithilfe von Spot-Instances einen Cluster mit einer verwalteten Knotengruppe zu erstellen, übergeben Sie das `--spot` Flag und eine optionale Liste von Instance-Typen:

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

So erstellen Sie eine verwaltete Knotengruppe mithilfe von Spot-Instances auf einem vorhandenen Cluster:

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

So erstellen Sie Spot-Instances mithilfe verwalteter Knotengruppen über eine Konfigurationsdatei:

```
# 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**
>
> Nicht verwaltete Knotengruppen unterstützen die `instanceTypes` Felder `spot` und nicht, stattdessen wird das `instancesDistribution` Feld zur Konfiguration von Spot-Instances verwendet. Siehe unten

Weitere Informationen

- EKS Spot Nodegroups
- Kapazitätstypen für verwaltete EKS-Knotengruppen

Nicht verwaltete Knotengruppen

`eksctl`unterstützt Spot-Instances über die MixedInstancesPolicy for Auto Scaling Groups.

Hier ist ein Beispiel für eine Knotengruppe, die zu 50% Spot-Instances und zu 50% On-Demand-Instances verwendet:

```
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
```

Beachten Sie, dass das `nodeGroups.X.instanceType` Feld nicht festgelegt werden sollte, wenn Sie das `instancesDistribution` Feld verwenden.

In diesem Beispiel werden GPU-Instanzen verwendet:

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

In diesem Beispiel wird die Strategie zur kapazitätsoptimierten Spot-Allokation verwendet:

```
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"
```

In diesem Beispiel wird die capacity-optimized-prioritized Spot-Allokationsstrategie verwendet:

```
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"
```

Verwenden Sie die `capacity-optimized-prioritized` Zuweisungsstrategie und legen Sie dann die Reihenfolge der Instance-Typen in der Liste der Startvorlagen-Overrides von der höchsten zur niedrigsten Priorität fest (vom ersten zum letzten in der Liste). Amazon EC2 Auto Scaling berücksichtigt die Prioritäten des Instance-Typs nach bestem Wissen und Gewissen, optimiert jedoch zuerst die Kapazität. Dies ist eine gute Option für Workloads, bei denen die Möglichkeit von Unterbrechungen minimiert werden muss, bei denen aber auch die Präferenz für bestimmte Instance-Typen wichtig ist. Weitere Informationen finden Sie unter ASG-Kaufoptionen.

Beachten Sie, dass das `spotInstancePools` Feld nicht festgelegt werden sollte, wenn Sie das Feld verwenden. `spotAllocationStrategy` Wenn das nicht angegeben `spotAllocationStrategy` ist, EC2 wird standardmäßig die `lowest-price` Strategie verwendet.

Hier ist ein minimales Beispiel:

```
nodeGroups:
  - name: ng-1
    instancesDistribution:
      instanceTypes: ["t3.small", "t3.medium"] # At least one instance type should be
 specified
```

Um Knoten zwischen Spot- und On-Demand-Instances zu unterscheiden, können Sie das Kubernetes-Label verwendennode`-lifecycle`, das `on-demand` je nach `spot` Typ den Wert oder hat.

Parameter in InstancesDistribution

Einzelheiten finden Sie im Cluster-Konfigurationsschema.

GPU-Unterstützung

Eksctl unterstützt die Auswahl von GPU-Instanztypen für Knotengruppen. Geben Sie einfach einen kompatiblen Instanztyp für den Befehl create oder über die Konfigurationsdatei an.

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

> **Note**
>
> Es ist nicht mehr erforderlich, das Marketplace-AMI für GPU-Unterstützung auf EKS zu abonnieren.

Die AMI-Resolver (`autoundauto-ssm`) erkennen, dass Sie einen GPU-Instance-Typ verwenden möchten, und wählen das richtige EKS-optimierte beschleunigte AMI aus.

Eksctl erkennt, dass ein AMI mit einem GPU-fähigen Instanztyp ausgewählt wurde, und installiert das NVIDIA Kubernetes-Geräte-Plugin automatisch.

> **Note**
>
> Windows und Ubuntu werden AMIs nicht mit installierten GPU-Treibern ausgeliefert, sodass die Ausführung von GPU-beschleunigten Workloads nicht sofort funktioniert.

Verwenden Sie den Befehl create, um die automatische Plugin-Installation zu deaktivieren und eine bestimmte Version manuell `--install-nvidia-plugin=false` zu installieren. Zum Beispiel:

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

und für die Versionen 0.15.0 und höher

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

oder für ältere Versionen

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

Die Installation des NVIDIA Kubernetes-Geräte-Plugins wird übersprungen, wenn der Cluster nur Bottlerocket-Knotengruppen enthält, da Bottlerocket die Ausführung des Geräte-Plugins bereits übernimmt. Wenn Sie in den Konfigurationen Ihres Clusters unterschiedliche AMI-Familien verwenden, müssen Sie möglicherweise Taints and Tolerations verwenden, um zu verhindern, dass das Geräte-Plugin auf Bottlerocket-Knoten ausgeführt wird.

ARM-Unterstützung

In diesem Thema wird beschrieben, wie Sie einen Cluster mit einer ARM-Knotengruppe erstellen und wie Sie einem vorhandenen Cluster eine ARM-Knotengruppe hinzufügen.

EKS unterstützt die 64-Bit-ARM-Architektur mit seinen Graviton-Prozessoren. Um einen Cluster zu erstellen, wählen Sie einen der Graviton-basierten Instance-Typen (`a1`,,`t4g`,`m6g`,`m7g`,`m6gd`,,`c6g`, `c7g` `c6gd` `r6g` `r7g` `r6gd` `m8gr8g`,`c8g`) aus und führen Sie Folgendes aus:

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

oder verwenden Sie eine Konfigurationsdatei:

```
---
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 wird auch in verwalteten Knotengruppen unterstützt:

```
---
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
```

Die AMI-Resolver `auto` und `auto-ssm` leiten das richtige AMI basierend auf dem ARM-Instance-Typ ab. Nur bei den Familien AmazonLinux 2023, AmazonLinux 2 und Bottlerocket ist EKS für ARM optimiert. AMIs

> **Note**
>
> ARM wird für Cluster mit Version 1.15 und höher unterstützt.

Auto Scaling

Auto Scaling aktivieren

Sie können einen Cluster (oder eine Knotengruppe in einem vorhandenen Cluster) mit einer IAM-Rolle erstellen, die die Verwendung von Cluster-Autoscaler ermöglicht:

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

Dieses Flag setzt auch `k8s.io/cluster-autoscaler/<clusterName>` Tags `k8s.io/cluster-autoscaler/enabled` und markiert sie, sodass die Erkennung von Knotengruppen funktionieren sollte.

Sobald der Cluster läuft, müssen Sie Cluster Autoscaler selbst installieren.

Sie sollten Ihren Definitionen für verwaltete oder nicht verwaltete Knotengruppen außerdem Folgendes hinzufügen, um die Tags hinzuzufügen, die für den Cluster-Autoscaler zur Skalierung der Knotengruppe erforderlich sind:

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

Hochskalierung von 0

Wenn Sie in der Lage sein möchten, Ihre Knotengruppe von 0 nach oben zu skalieren und Sie haben Labels and/or Taints für Ihre Knotengruppen definiert, müssen Sie diese als Tags in Ihren Auto Scaling Groups () propagieren. ASGs

Eine Möglichkeit, dies zu tun, besteht darin, die ASG-Tags im `tags` Feld Ihrer Knotengruppendefinitionen festzulegen. Zum Beispiel bei einer Knotengruppe mit den folgenden Labels und Taints:

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

Sie müssten die folgenden ASG-Tags hinzufügen:

```
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"
```

Sowohl für verwaltete als auch für nicht verwaltete Knotengruppen kann dies automatisch geschehen, indem `propagateASGTags` auf gesetzt wird`true`, wodurch die Labels und Taints als Tags zur Auto Scaling Scaling-Gruppe hinzugefügt werden:

```
nodeGroups:
  - name: ng1-public
    ...
    labels:
      my-cool-label: pizza
```

```
    taints:
      feaster: "true:NoSchedule"
    propagateASGTags: true
```

Zonenorientiertes Auto Scaling

Wenn Ihre Workloads zonenspezifisch sind, müssen Sie für jede Zone separate Knotengruppen erstellen. Das liegt daran, dass `cluster-autoscaler` davon ausgegangen wird, dass alle Knoten in einer Gruppe exakt gleichwertig sind. Wenn also beispielsweise ein Scale-up-Ereignis durch einen Pod ausgelöst wird, der ein zonenspezifisches PVC benötigt (z. B. ein EBS-Volume), wird der neue Knoten möglicherweise in der falschen AZ geplant und der Pod kann nicht gestartet werden.

Sie benötigen keine separate Knotengruppe für jede AZ, wenn Ihre Umgebung die folgenden Kriterien erfüllt:

- Keine zonenspezifischen Speicheranforderungen.
- Keine PodAffinity mit einer anderen Topologie als dem Host erforderlich.
- Keine NodeAffinity auf dem Zonenlabel erforderlich.
- Kein NodeSelector auf einer Zonenbeschriftung.

(Lesen Sie hier und hier mehr.)

Wenn Sie alle oben genannten Anforderungen (und möglicherweise weitere) erfüllen, sollten Sie mit einer einzelnen Knotengruppe, die sich über mehrere erstreckt, auf Nummer sicher gehen. AZs Andernfalls sollten Sie separate Single-AZ-Knotengruppen erstellen:

VORHER:

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

NACH:

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

```
    instanceType: m5.xlarge
    availabilityZones: ["eu-west-2b"]
```

Benutzerdefinierte AMI-Unterstützung

Einstellung der AMI-ID des Knotens

Das `--node-ami` Flag ermöglicht eine Reihe fortgeschrittener Anwendungsfälle, z. B. die Verwendung eines benutzerdefinierten AMI oder die Abfrage von AWS in Echtzeit, um zu bestimmen, welches AMI verwendet werden soll. Das Flag kann sowohl für Nicht-GPU- als auch für GPU-Images verwendet werden.

Das Flag kann die AMI-Image-ID für ein Bild verwenden, um es explizit zu verwenden. Es kann auch die folgenden „speziellen“ Schlüsselwörter enthalten:

Stichwort	Beschreibung
auto	Zeigt an, dass das für die Knoten zu verwenden de AMI durch eine Abfrage von AWS EC2 gefunden werden soll. Dies bezieht sich auf den Autoresolver.
Auto-SSM	Zeigt an, dass das für die Knoten zu verwenden de AMI durch eine Abfrage des AWS SSM Parameter Store gefunden werden soll.

> **Note**
>
> Bei der Einstellung `--node-ami` auf eine ID-Zeichenfolge `eksctl` wird davon ausgegangen, dass ein benutzerdefiniertes AMI angefordert wurde. Für AmazonLinux 2- und Ubuntu-Knoten, die sowohl von EKS verwaltet als auch von EKS selbst verwaltet werden, bedeutet dies, dass dies erforderlich `overrideBootstrapCommand` ist. Für AmazonLinux 2023 wird es nicht unterstützt, da es das `/etc/eks/bootstrap.sh` Skript nicht mehr für das Node-Bootstrapping verwendet und stattdessen einen Nodeadm-Initialisierungsprozess verwendet (weitere Informationen finden Sie in der Dokumentation zum Node-Bootstrapping). `overrideBootstrapCommand`

Beispiele für CLI-Flaggen:

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

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

Beispiel für eine Konfigurationsdatei:

```
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>
```

Die `--node-ami` Flagge kann auch mit verwendet werden`eksctl create nodegroup`.

Einstellung der Knoten-AMI-Familie

Sie `--node-ami-family` kann die folgenden Schlüsselwörter annehmen:

Stichwort	Beschreibung
AmazonLinux2	Zeigt an, dass das auf Amazon Linux 2 basierende EKS-AMI-Image verwendet werden sollte (Standard).

Stichwort	Beschreibung
AmazonLinux2023	Zeigt an, dass das auf Amazon Linux 2023 basierende EKS AMI-Image verwendet werden sollte.
Ubuntu 2004	Zeigt an, dass das auf Ubuntu 20.04 LTS (Focal) basierende EKS AMI-Image verwendet werden sollte (unterstützt für EKS 1.29).
UbuntuPro2004	Zeigt an, dass das auf Ubuntu Pro 20.04 LTS (Focal) basierende EKS AMI-Image verwendet werden sollte (verfügbar für EKS >= 1.27, 1.29).
Ubuntu 2204	Zeigt an, dass das auf Ubuntu 22.04 LTS (Jammy) basierende EKS AMI-Image verwendet werden sollte (verfügbar für EKS >= 1.29).
UbuntuPro2204	Zeigt an, dass das auf Ubuntu Pro 22.04 LTS (Jammy) basierende EKS AMI-Image verwendet werden sollte (verfügbar für EKS >= 1.29).
Ubuntu 2404	Zeigt an, dass das auf Ubuntu 24.04 LTS (Noble) basierende EKS AMI-Image verwendet werden sollte (verfügbar für EKS >= 1.31).
UbuntuPro2404	Zeigt an, dass das auf Ubuntu Pro 24.04 LTS (Noble) basierende EKS AMI-Image verwendet werden sollte (verfügbar für EKS >= 1.31).
Bottlerocket	Zeigt an, dass das auf Bottlerocket basierend e EKS-AMI-Image verwendet werden sollte.
WindowsServer2019 FullContainer	Zeigt an, dass das EKS AMI-Image , das auf Windows Server 2019 Full Container basiert, verwendet werden sollte.

Stichwort	Beschreibung
WindowsServer2019 CoreContainer	Zeigt an, dass das auf Windows Server 2019 Core Container basierende EKS-AMI-Image verwendet werden sollte.
WindowsServer2022 FullContainer	Zeigt an, dass das EKS AMI-Image , das auf Windows Server 2022 Full Container basiert, verwendet werden sollte.
WindowsServer2022 CoreContainer	Zeigt an, dass das auf Windows Server 2022 Core Container basierende EKS-AMI-Image verwendet werden sollte.

Beispiel für eine CLI-Flagge:

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

Beispiel für eine Konfigurationsdatei:

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

Die `--node-ami-family` Flagge kann auch mit verwendet werden`eksctl create nodegroup`. `eksctl`erfordert, dass die AMI-Familie explizit über die Konfigurationsdatei oder über das `--node-ami-family` CLI-Flag festgelegt wird, wenn mit einem benutzerdefinierten AMI gearbeitet wird.

> **Note**
>
> Derzeit unterstützen von EKS verwaltete Knotengruppen nur die folgenden AMI-Familien, wenn sie mit benutzerdefinierten arbeiten AMIs:`AmazonLinux2023`,`AmazonLinux2`, und `Ubuntu1804` `Ubuntu2004` `Ubuntu2204`

Unterstützung für benutzerdefinierte Windows-AMIs

Nur selbstverwaltete Windows-Knotengruppen können ein benutzerdefiniertes AMI angeben. `amiFamily`sollte auf eine gültige Windows AMI-Familie eingestellt sein.

Die folgenden PowerShell Variablen werden für das Bootstrap-Skript verfügbar sein:

```
$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
```

Beispiel für eine Konfigurationsdatei:

```
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
```

Benutzerdefinierte AMI-Unterstützung für Bottlerocket

Für Bottlerocket-Knoten wird das nicht unterstützt. `overrideBootstrapCommand` Um einen eigenen Bootstrap-Container zu definieren, sollte man das `bottlerocket` Feld stattdessen als Teil der Konfigurationsdatei verwenden. z. B.

```
  nodeGroups:
  - name: bottlerocket-ng
    ami: ami-custom1234
    amiFamily: Bottlerocket
    bottlerocket:
      enableAdminContainer: true
```

```
    settings:
      bootstrap-containers:
        bootstrap:
          source: <MY-CONTAINER-URI>
```

Windows Worker-Knoten

Ab Version 1.14 unterstützt Amazon EKS Windows-Knoten, die das Ausführen von Windows-Containern ermöglichen. Zusätzlich zu den Windows-Knoten ist ein Linux-Knoten im Cluster erforderlich, um CoreDNS auszuführen, da Microsoft den Host-Netzwerkmodus noch nicht unterstützt. Somit wird ein Windows EKS-Cluster eine Mischung aus Windows-Knoten und mindestens einem Linux-Knoten sein. Die Linux-Knoten sind für das Funktionieren des Clusters von entscheidender Bedeutung. Daher wird für einen Cluster auf Produktionsniveau empfohlen, mindestens zwei `t2.large` Linux-Knoten für HA zu haben.

> **Note**
>
> Sie müssen den VPC-Ressourcencontroller nicht mehr auf Linux-Worker-Knoten installieren, um Windows-Workloads in EKS-Clustern auszuführen, die nach dem 22. Oktober 2021 erstellt wurden. Sie können die Windows-IP-Adressverwaltung auf der EKS-Steuerebene über eine ConfigMap-Einstellung aktivieren (Einzelheiten finden Sie unter Link: eks/latest/userguide/windows -support.html). eksctl patcht das automatisch, um die Windows-IP-Adressverwaltung ConfigMap zu aktivieren, wenn eine Windows-Knotengruppe erstellt wird.

Einen neuen Cluster mit Windows-Unterstützung erstellen

Die Syntax der Konfigurationsdatei ermöglicht die Erstellung eines voll funktionsfähigen Clusters mit Windows-Unterstützung mit einem einzigen Befehl:

```
# 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
```

Um einen neuen Cluster mit einer nicht verwalteten Windows-Knotengruppe ohne Verwendung einer Konfigurationsdatei zu erstellen, geben Sie die folgenden Befehle ein:

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

Hinzufügen von Windows-Unterstützung zu einem vorhandenen Linux-Cluster

Um die Ausführung von Windows-Workloads auf einem vorhandenen Cluster mit Linux-Knoten (`AmazonLinux2`AMI-Familie) zu ermöglichen, müssen Sie eine Windows-Knotengruppe hinzufügen.

NEU Support für von Windows verwaltete Knotengruppen wurde hinzugefügt (--managed=true oder das Flag weglassen).

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

Um sicherzustellen, dass Workloads auf dem richtigen Betriebssystem geplant werden, müssen sie auf das Betriebssystem `nodeSelector` ausgerichtet sein, auf dem sie ausgeführt werden sollen:

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

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

Wenn Sie einen Cluster verwenden, der älter als `1.19` der ist, müssen die `kubernetes.io/arch` Bezeichnungen `kubernetes.io/os` und `beta.kubernetes.io/arch` jeweils durch `beta.kubernetes.io/os` und ersetzt werden.

Weitere Informationen

- EKS Windows-Unterstützung

Zusätzliche Volume-Zuordnungen

Als zusätzliche Konfigurationsoption ist es beim Umgang mit Volume-Mappings möglich, zusätzliche Mappings zu konfigurieren, wenn die Nodegroup erstellt wird.

Stellen Sie dazu das Feld wie folgt ein: `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'
```

Weitere Informationen zur Auswahl von VolumeNames finden Sie in der Dokumentation zur Gerätebenennung. Weitere Informationen zu EBS-Volumes, Volume-Limits für Instanzen oder Blockgerätezuordnungen finden Sie auf dieser Seite.

EKS-Hybridknoten

Einführung

AWS EKS führt Hybrid Nodes ein, eine neue Funktion, mit der Sie lokale und Edge-Anwendungen auf einer vom Kunden verwalteten Infrastruktur mit denselben AWS EKS-Clustern, Funktionen und Tools ausführen können, die Sie in der AWS-Cloud verwenden. AWS EKS Hybird Nodes bietet ein AWS-verwaltetes Kubernetes-Erlebnis in lokalen Umgebungen, damit Kunden die Ausführung von Anwendungen in lokalen, Edge- und Cloud-Umgebungen vereinfachen und standardisieren können. Lesen Sie mehr unter EKS Hybrid Nodes.

Um die Unterstützung dieser Funktion zu erleichtern, führt eksctl ein neues Top-Level-Feld mit dem Namen ein. `remoteNetworkConfig` Jede Konfiguration im Zusammenhang mit Hybridknoten muss über dieses Feld als Teil der Konfigurationsdatei eingerichtet werden; es gibt keine CLI-Flags-Gegenstücke. Darüber hinaus kann jede Remote-Netzwerkkonfiguration beim Start nur während der Clustererstellung eingerichtet und danach nicht aktualisiert werden. Das bedeutet, dass Sie bestehende Cluster nicht aktualisieren können, um Hybridknoten zu verwenden.

Im `remoteNetworkConfig` Abschnitt der Konfigurationsdatei können Sie die beiden Kernbereiche einrichten, wenn es darum geht, Remoteknoten mit Ihren EKS-Clustern zu verbinden: Netzwerke und Anmeldeinformationen.

Netzwerk

EKS Hybrid Nodes ist flexibel in Bezug auf Ihre bevorzugte Methode zur Verbindung Ihrer lokalen Netzwerke mit einer AWS-VPC. Es stehen mehrere dokumentierte Optionen zur Verfügung, darunter AWS Site-to-Site VPN und AWS Direct Connect, und Sie können die Methode wählen, die am besten zu Ihrem Anwendungsfall passt. Bei den meisten Methoden, die Sie wählen könnten, wird Ihre VPC entweder mit einem Virtual Private Gateway (VGW) oder einem Transit Gateway (TGW) verbunden. Wenn Sie sich darauf verlassen, dass eksctl eine VPC für Sie erstellt, konfiguriert eksctl im Rahmen Ihrer VPC auch alle netzwerkbezogenen Voraussetzungen, um die Kommunikation zwischen Ihrer EKS-Steuerebene und den Remote-Knoten zu erleichtern, d. h.

- SG-Regeln für Eingang/Ausgang
- Routen in den Routentabellen der privaten Subnetze
- die VPC-Gateway-Verbindung zum angegebenen TGW oder VGW

Beispiel für eine Konfigurationsdatei:

```
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"]
```

Wenn Ihre bevorzugte Verbindungsmethode nicht die Verwendung eines TGW oder VGW beinhaltet, dürfen Sie sich nicht darauf verlassen, dass eksctl die VPC für Sie erstellt, sondern stattdessen eine bereits vorhandene bereitstellen. In diesem Zusammenhang: Wenn Sie eine bereits bestehende VPC verwenden, nimmt eksctl keine Änderungen daran vor, und es liegt in Ihrer Verantwortung, sicherzustellen, dass alle Netzwerkanforderungen erfüllt sind.

> **Note**
>
> eksctl richtet keine Netzwerkinfrastruktur außerhalb Ihrer AWS-VPC ein (d. h. jegliche Infrastruktur von VGW/TGW zu den Remote-Netzwerken)

Anmeldeinformationen

EKS-Hybridknoten verwenden den AWS IAM Authenticator und temporäre IAM-Anmeldeinformationen, die entweder von AWS SSM oder AWS IAM Roles Anywhere bereitgestellt werden, um sich beim EKS-Cluster zu authentifizieren. Ähnlich wie bei den selbstverwalteten Knotengruppen erstellt eksctl für Sie, sofern nicht anders angegeben, eine IAM-Rolle für Hybridknoten, die von den Remote-Knoten übernommen wird. Wenn Sie IAM Roles Anywhere als Ihren Anbieter für Anmeldeinformationen verwenden, richtet eksctl außerdem ein Profil und einen Vertrauensanker auf der Grundlage eines bestimmten Zertifizierungsstellenpakets ein, z. B. `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
```

Der ARN der von eksctl erstellten Hybrid Nodes Role wird später beim Hinzufügen Ihrer Remote-Knoten zum Cluster benötigt, um Aktivierungen einzurichten `NodeConfig` und zu erstellen (falls Sie SSM verwenden). `nodeadm` Um ihn abzurufen, verwenden Sie:

```
aws cloudformation describe-stacks \
  --stack-name eksctl-<CLUSTER_NAME>-cluster \
  --query 'Stacks[].Outputs[?OutputKey==`RemoteNodesRoleARN`].[OutputValue]' \
  --output text
```

In ähnlicher Weise können Sie, wenn Sie IAM Roles Anywhere verwenden, den ARN des Vertrauensankers und des von eksctl erstellten Anywhere-Profils abrufen und den vorherigen Befehl ändern, indem Sie ihn jeweils durch oder `RemoteNodesRoleARN` ersetzen`RemoteNodesTrustAnchorARN`. `RemoteNodesAnywhereProfileARN`

Wenn Sie bereits über eine IAM Roles Anywhere-Konfiguration verfügen oder SSM verwenden, können Sie eine IAM-Rolle für Hybridknoten über bereitstellen. `remoteNetworkConfig.iam.roleARN` Denken Sie daran, dass eksctl in diesem Szenario den Trust Anchor und das Anywhere-Profil nicht für Sie erstellt. z. B.

```
remoteNetworkConfig:
  iam:
    roleARN: arn:aws:iam::000011112222:role/HybridNodesRole
```

Um die Rolle einer Kubernetes-Identität zuzuordnen und die Remote-Knoten zu autorisieren, dem EKS-Cluster beizutreten, erstellt eksctl einen Zugriffseintrag mit der Hybrid Nodes IAM-Rolle als Haupt-ARN und vom Typ. d. h. `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"
        ]
    }
]
```

Unterstützung für Add-Ons

Container Networking Interface (CNI): Das AWS VPC CNI kann nicht mit Hybridknoten verwendet werden. Die Kernfunktionen von Cilium und Calico werden für die Verwendung mit Hybridknoten unterstützt. Sie können Ihr CNI mit Tools Ihrer Wahl wie Helm verwalten. Weitere Informationen finden Sie unter Konfiguration eines CNI für Hybridknoten.

> **Note**
>
> Wenn Sie VPC CNI in Ihrem Cluster für Ihre selbst verwalteten oder EKS-verwalteten Knotengruppen installieren, müssen Sie `v1.19.0-eksbuild.1` oder später verwenden, da dies eine Aktualisierung des Daemonsets des Add-ons beinhaltet, um es von der Installation auf Hybridknoten auszuschließen.

Weitere Referenzen

- EKS-Hybridknoten UserDocs

- Ankündigung der Markteinführung

Support für Node Repair Config in EKS Managed Nodegroups

EKS Managed Nodegroups unterstützt jetzt Node Repair, bei dem der Zustand verwalteter Knoten überwacht wird und als Reaktion darauf fehlerhafte Worker-Knoten ersetzt oder neu gestartet werden.

Erstellen eines Clusters, einer verwalteten Knotengruppe mit aktivierter Knotenreparatur

Um mithilfe der Knotenreparatur einen Cluster mit einer verwalteten Knotengruppe zu erstellen, übergeben Sie das Flag: `--enable-node-repair`

```
eksctl create cluster --enable-node-repair
```

Um eine verwaltete Knotengruppe mithilfe von Node Repair auf einem vorhandenen Cluster zu erstellen:

```
eksctl create nodegroup --cluster=<clusterName> --enable-node-repair
```

So erstellen Sie mithilfe der Knotenreparatur über eine Konfigurationsdatei einen Cluster mit einer verwalteten Knotengruppe:

```
# node-repair-nodegroup-cluster.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: cluster-44
  region: us-west-2

managedNodeGroups:
- name: ng-1
  nodeRepairConfig:
    enabled: true
```

```
eksctl create cluster -f node-repair-nodegroup-cluster.yaml
```

Weitere Informationen

- EKS Managed Nodegroup Node Health

Netzwerk

Dieses Kapitel enthält Informationen darüber, wie Eksctl Virtual Private Cloud (VPC) -Netzwerke für EKS-Cluster erstellt.

Themen:

- the section called “VPC-Konfiguration”
  - Ändern Sie den VPC-CIDR-Bereich und konfigurieren Sie die Adressierung IPv6
  - Verwenden Sie eine vorhandene VPC
  - Passen Sie die VPC, Subnetze, Sicherheitsgruppen und NAT-Gateways für den neuen EKS-Cluster an
- the section called “Subnetz-Einstellungen”
  - Verwenden Sie private Subnetze für die erste Knotengruppe, um sie vom öffentlichen Internet zu isolieren
  - Passen Sie die Subnetztopologie an, indem Sie mehrere Subnetze pro Availability Zone auflisten und Subnetze in Knotengruppenkonfigurationen angeben
  - Beschränken Sie Knotengruppen auf bestimmte benannte Subnetze in der VPC-Konfiguration
  - Wenn Sie private Subnetze für Knotengruppen verwenden, legen Sie die Einstellung auf fest `privateNetworking` `true`
  - Stellen Sie eine vollständige Subnetzspezifikation mit beiden `public` `private` Konfigurationen in der VPC-Spezifikation bereit
  - In der Knotengruppenkonfiguration `availabilityZones` kann nur eine von `subnets` oder angegeben werden
- the section called “Cluster-Zugriff”
  - Verwalten Sie den öffentlichen und privaten Zugriff auf die Kubernetes-API-Serverendpunkte in einem EKS-Cluster
  - Beschränken Sie den Zugriff auf den öffentlichen EKS Kubernetes-API-Endpunkt, indem Sie zulässige CIDR-Bereiche angeben
  - Aktualisieren Sie die Konfiguration des API-Server-Endpunktzugriffs und die CIDR-Einschränkungen für den öffentlichen Zugriff für einen vorhandenen Cluster
- the section called “Netzwerke auf Steuerungsebene”

  - Aktualisieren Sie die Subnetze, die von der EKS-Steuerebene für einen Cluster verwendet werden
- the section called “IPv6 Support”
  - Geben Sie die IP-Version (IPv4 oder IPv6) an, die beim Erstellen einer VPC mit EKS-Cluster verwendet werden soll

VPC-Konfiguration

Dedizierte VPC für Cluster

Standardmäßig `eksctl create cluster` wird eine dedizierte VPC für den Cluster erstellt. Dies geschieht, um Interferenzen mit vorhandenen Ressourcen aus verschiedenen Gründen zu vermeiden, unter anderem aus Sicherheitsgründen, aber auch, weil es schwierig ist, alle Einstellungen in einer vorhandenen VPC zu erkennen.

- Das Standard-VPC-CIDR, das von `eksctl` verwendet wird, ist. `192.168.0.0/16`
  - Es ist in 8 (`/19`) Subnetze unterteilt (3 private, 3 öffentliche und 2 reservierte).
- Die erste Knotengruppe wird in öffentlichen Subnetzen erstellt.
- Der SSH-Zugriff ist deaktiviert, sofern nicht anders angegeben. `--allow-ssh`
- Die Knotengruppe lässt standardmäßig eingehenden Datenverkehr von der Sicherheitsgruppe der Steuerungsebene an den Ports 1025 bis 65535 zu.

> **Note**
>
> In `us-east-1` eksctl werden standardmäßig nur 2 öffentliche und 2 private Subnetze erstellt.

VPC CIDR ändern

Wenn Sie Peering mit einer anderen VPC einrichten müssen oder einfach einen größeren oder kleineren Bereich von benötigen IPs, können Sie dies mit `--vpc-cidr` Flag ändern. Anleitungen zur Auswahl von CIDR-Blöcken, die für die Verwendung in einer AWS-VPC zugelassen sind, finden Sie in den AWS-Dokumenten.

Wenn Sie einen IPv6 Cluster erstellen, können Sie ihn `VPC.IPv6Cidr` in der Cluster-Konfigurationsdatei konfigurieren. Diese Einstellung befindet sich nur in der Konfigurationsdatei, nicht in einem CLI-Flag.

Wenn Sie einen IPv6 IP-Adressblock besitzen, können Sie auch Ihren eigenen IPv6 Pool mitbringen. Informationen zum Importieren Ihres eigenen Pools finden Sie unter Bringen Sie Ihre eigenen IP-Adressen (BYOIP) zu EC2 Amazon. Verwenden Sie dann die `VPC.IPv6Cidr` in der Cluster-Konfigurationsdatei enthaltene Konfigurationsdatei, um Eksctl zu konfigurieren.

Verwenden Sie eine vorhandene VPC: gemeinsam mit kops

Sie können die VPC eines vorhandenen Kubernetes-Clusters verwenden, der von kops verwaltet wird. Diese Funktion soll das Peering von Migrationsclustern erleichtern. and/or

Wenn Sie zuvor einen Cluster mit kops erstellt haben, z. B. mit Befehlen, die dem Folgenden ähneln:

```
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
```

Sie können in demselben einen EKS-Cluster AZs mit denselben VPC-Subnetzen erstellen (HINWEIS: mindestens 2 AZs/subnets sind erforderlich):

```
eksctl create cluster --name=cluster-2 --region=us-west-2 --vpc-from-kops-
cluster=cluster-1.k8s.local
```

Bestehende VPC verwenden: andere benutzerdefinierte Konfiguration

`eksctl`bietet eine gewisse, aber nicht vollständige Flexibilität für benutzerdefinierte VPC- und Subnetztopologien.

Sie können eine vorhandene VPC verwenden, indem Sie private and/or öffentliche Subnetze mit den Flags `--vpc-private-subnets` und `--vpc-public-subnets` bereitstellen. Es liegt an Ihnen, sicherzustellen, dass die von Ihnen verwendeten Subnetze korrekt kategorisiert sind, da es keine einfache Möglichkeit gibt, zu überprüfen, ob ein Subnetz tatsächlich privat oder öffentlich ist, da die Konfigurationen variieren.

Anhand dieser Flags `eksctl create cluster` wird die VPC-ID automatisch bestimmt, es werden jedoch keine Routing-Tabellen oder andere Ressourcen wie internet/NAT Gateways

erstellt. Es werden jedoch spezielle Sicherheitsgruppen für die ursprüngliche Knotengruppe und die Kontrollebene erstellt.

Sie müssen sicherstellen, dass mindestens 2 Subnetze in unterschiedlichen AZs Subnetzen bereitgestellt werden. Dieser Zustand wird von EKS überprüft. Wenn Sie eine vorhandene VPC verwenden, werden die folgenden Anforderungen nicht von EKS oder Eksctl durchgesetzt oder überprüft, und EKS erstellt den Cluster. Einige Grundfunktionen des Clusters funktionieren auch ohne diese Anforderungen. (Tagging ist beispielsweise nicht unbedingt erforderlich. Tests haben gezeigt, dass es möglich ist, einen funktionierenden Cluster zu erstellen, ohne dass in den Subnetzen Tags gesetzt werden. Es gibt jedoch keine Garantie dafür, dass dies immer gilt, und Tagging wird empfohlen.)

Standardanforderungen:

- Alle angegebenen Subnetze müssen sich in derselben VPC befinden, innerhalb desselben Blocks von IPs
- je nach Bedarf ist eine ausreichende Anzahl von IP-Adressen verfügbar
- je nach Bedarf eine ausreichende Anzahl von Subnetzen (mindestens 2)
- Subnetze sind mit mindestens den folgenden Tags gekennzeichnet:
  - `kubernetes.io/cluster/<name>`Tag, der auf entweder oder `shared` gesetzt ist `owned`
  - `kubernetes.io/role/internal-elb`Tag, der 1 für private Subnetze auf gesetzt ist
  - `kubernetes.io/role/elb`Tag, der 1 für öffentliche Subnetze auf gesetzt ist
- korrekt konfigurierte and/or Internet-NAT-Gateways
- Routing-Tabellen haben korrekte Einträge und das Netzwerk ist funktionsfähig
- NEU: In allen öffentlichen Subnetzen sollte die Eigenschaft `MapPublicIpOnLaunch` aktiviert sein (d. h. `Auto-assign public IPv4 address` in der AWS-Konsole). Verwaltete Knotengruppen und Fargate weisen keine öffentlichen IPv4 Adressen zu, die Eigenschaft muss im Subnetz festgelegt werden.

Möglicherweise gibt es weitere Anforderungen, die von EKS oder Kubernetes gestellt werden, und es liegt ganz bei Ihnen, diese Anforderungen einzuhalten up-to-date. and/or recommendations, and implement those as needed/possible

Die von `eksctl` angewendeten Standardeinstellungen für Sicherheitsgruppen reichen möglicherweise aus, um den Zugriff mit Ressourcen in anderen Sicherheitsgruppen gemeinsam zu nutzen. Wenn Sie die ingress/egress Regeln der Sicherheitsgruppen ändern möchten, müssen Sie

möglicherweise ein anderes Tool verwenden, um Änderungen zu automatisieren, oder Sie müssen dies über die EC2 Konsole tun.

Verwenden Sie im Zweifelsfall keine benutzerdefinierte VPC. Bei Verwendung `eksctl create cluster` ohne `--vpc-*` Flags wird der Cluster immer mit einer voll funktionsfähigen dedizierten VPC konfiguriert.

Beispiele

Erstellen Sie einen Cluster mit einer benutzerdefinierten VPC mit 2x privaten und 2x öffentlichen Subnetzen:

```
eksctl create cluster \
  --vpc-private-subnets=subnet-0ff156e0c4a6d300c,subnet-0426fb4a607393184 \
  --vpc-public-subnets=subnet-0153e560b3129a696,subnet-009fa0199ec203c37
```

oder verwenden Sie die folgende äquivalente Konfigurationsdatei:

```
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
```

Erstellen Sie einen Cluster mit einer benutzerdefinierten VPC mit drei privaten Subnetzen und sorgen Sie dafür, dass die erste Knotengruppe diese Subnetze verwendet:

```
eksctl create cluster \
  --vpc-private-
subnets=subnet-0ff156e0c4a6d300c,subnet-0549cdab573695c03,subnet-0426fb4a607393184 \
  --node-private-networking
```

oder verwenden Sie die folgende äquivalente Konfigurationsdatei:

```
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
```

Erstellen Sie einen Cluster mit benutzerdefinierten öffentlichen VPC 4x-Subnetzen:

```
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
```

Weitere Beispiele finden Sie im Ordner des Repos: `examples`

- mit einer vorhandenen VPC
- mit einem benutzerdefinierten VPC CIDR

Benutzerdefinierte Sicherheitsgruppe für gemeinsam genutzte Knoten

`eksctl`erstellt und verwaltet eine Sicherheitsgruppe für gemeinsam genutzte Knoten, die die Kommunikation zwischen nicht verwalteten Knoten und der Cluster-Steuerebene und den verwalteten Knoten ermöglicht.

Wenn Sie stattdessen Ihre eigene benutzerdefinierte Sicherheitsgruppe angeben möchten, können Sie das `sharedNodeSecurityGroup` Feld in der Konfigurationsdatei überschreiben:

```
vpc:
  sharedNodeSecurityGroup: sg-0123456789
```

Standardmäßig `eksctl` werden bei der Erstellung des Clusters Regeln zu dieser Sicherheitsgruppe hinzugefügt, um die Kommunikation mit und von der Standard-Cluster-Sicherheitsgruppe zu ermöglichen, die EKS erstellt. Die standardmäßige Clustersicherheitsgruppe wird sowohl von der EKS-Steuerebene als auch von verwalteten Knotengruppen verwendet.

Wenn Sie die Sicherheitsgruppenregeln selbst verwalten möchten, können Sie die Erstellung der Regeln `eksctl` verhindern, indem Sie `false` in der Konfigurationsdatei Folgendes festlegen`manageSharedNodeSecurityGroupRules`:

```
vpc:
  sharedNodeSecurityGroup: sg-0123456789
  manageSharedNodeSecurityGroupRules: false
```

NAT-Gateway

Das NAT-Gateway für einen Cluster kann als`Disable`, `Single` (Standard) oder konfiguriert werden`HighlyAvailable`. Bei dieser `HighlyAvailable` Option wird in jeder Availability Zone der Region ein NAT-Gateway bereitgestellt, sodass auch bei einem Ausfall einer AZ die Knoten in der anderen AZs weiterhin mit dem Internet kommunizieren können.

Es kann über das `--vpc-nat-mode` CLI-Flag oder in der Cluster-Konfigurationsdatei wie im folgenden Beispiel angegeben werden:

```
vpc:
  nat:
    gateway: HighlyAvailable # other options: Disable, Single (default)
```

Das vollständige Beispiel finden Sie hier.

> **Note**
>
> Die Angabe des NAT-Gateways wird nur bei der Clustererstellung unterstützt. Es wird während eines Cluster-Upgrades nicht berührt.

Subnetz-Einstellungen

Verwenden Sie private Subnetze für die erste Knotengruppe

Wenn Sie es vorziehen, die ursprüngliche Knotengruppe vom öffentlichen Internet zu isolieren, können Sie das Flag verwenden. `--node-private-networking` In Verbindung mit dem `--ssh-access` Flag kann auf den SSH-Port nur innerhalb der VPC zugegriffen werden.

> **Note**
>
> Die Verwendung des `--node-private-networking` Flags führt dazu, dass ausgehender Datenverkehr über das NAT-Gateway über dessen Elastic IP geleitet wird. Wenn sich die Knoten dagegen in einem öffentlichen Subnetz befinden, wird der ausgehende Datenverkehr nicht über das NAT-Gateway geleitet, sodass der ausgehende Verkehr die IP jedes einzelnen Knotens hat.

Benutzerdefinierte Subnetztopologie

`eksctl`In dieser Version `0.32.0` wurden weitere Anpassungen der Subnetztopologie eingeführt und es wurden folgende Möglichkeiten eröffnet:

- In der VPC-Konfiguration mehrere Subnetze pro AZ auflisten
- Geben Sie Subnetze in der Knotengruppenkonfiguration an

In früheren Versionen mussten benutzerdefinierte Subnetze nach Availability Zone bereitgestellt werden, was bedeutete, dass nur ein Subnetz pro AZ aufgelistet werden konnte. Aus `0.32.0` den identifizierenden Schlüsseln können beliebig gewählt werden.

```
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**
>
> Wenn die AZ als Identifikationsschlüssel verwendet wird, kann der `az` Wert weggelassen werden.

Wenn Sie eine beliebige Zeichenfolge als identifizierenden Schlüssel verwenden, wie oben, entweder:

- `id`muss gesetzt sein (`az`und `cidr` optional)
- oder `az` muss gesetzt werden (`cidr`optional)

Wenn ein Benutzer ein Subnetz nach AZ angibt, ohne CIDR und ID anzugeben, wird ein Subnetz in dieser AZ willkürlich aus der VPC ausgewählt, falls mehrere solcher Subnetze existieren.

> **Note**
>
> Eine vollständige Subnetzspezifikation muss bereitgestellt werden, d. h. sowohl als auch `public` `private` Konfigurationen, die in der VPC-Spezifikation deklariert sind.

Knotengruppen können über die Konfiguration auf benannte Subnetze beschränkt werden. Verwenden Sie bei der Angabe von Subnetzen in der Knotengruppenkonfiguration den in der VPC-Spezifikation angegebenen Identifikationsschlüssel und nicht die Subnetz-ID. Zum Beispiel:

```
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**
>
> In der Knotengruppenkonfiguration `availabilityZones` kann nur einer von `subnets` oder angegeben werden.

Beim Platzieren von Knotengruppen in einem privaten Subnetz `privateNetworking` muss auf in der Knotengruppe Folgendes gesetzt werden: `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
```

Ein vollständiges Konfigurationsbeispiel finden Sie unter 24-nodegroup-subnets.yaml im eksctl-Repo. GitHub

Cluster-Zugriff

Verwaltung des Zugriffs auf die Kubernetes API-Server-Endpunkte

Standardmäßig macht ein EKS-Cluster den Kubernetes-API-Server öffentlich verfügbar, jedoch nicht direkt innerhalb der VPC-Subnetze (public=true, private=false). Traffic, der innerhalb der VPC für den API-Server bestimmt ist, muss zuerst die VPC-Netzwerke (aber nicht das Netzwerk von Amazon) verlassen und dann erneut eintreten, um den API-Server zu erreichen.

Der Endpunktzugriff auf den Kubernetes-API-Server für einen Cluster kann bei der Erstellung des Clusters mithilfe der Cluster-Konfigurationsdatei für den öffentlichen und privaten Zugriff konfiguriert werden. Beispiel unten:

```
vpc:
  clusterEndpoints:
    publicAccess:  <true|false>
    privateAccess: <true|false>
```

Bei der Konfiguration des Kubernetes-API-Endpunktzugriffs gibt es einige zusätzliche Einschränkungen:

1. EKS erlaubt keine Cluster, für die weder der private noch der öffentliche Zugriff aktiviert ist.
2. EKS ermöglicht zwar die Erstellung einer Konfiguration, bei der nur privater Zugriff aktiviert werden kann, aber eksctl unterstützt dies nicht bei der Clustererstellung, da es verhindert, dass eksctl die Worker-Knoten dem Cluster hinzufügen kann.
3. Wenn ein Cluster so aktualisiert wird, dass er nur privaten Kubernetes-API-Endpunktzugriff hat, bedeutet dies, dass Kubernetes-Befehle standardmäßig (z. B.`kubectl`) sowie `eksctl delete cluster`eksctl utils write-kubeconfig, und möglicherweise der Befehl innerhalb der Cluster-VPC ausgeführt werden `eksctl utils update-kube-proxy` müssen.
   - Dies erfordert einige Änderungen an verschiedenen AWS-Ressourcen. Weitere Informationen finden Sie unter Cluster-API-Serverendpunkt.
   - Sie können angeben`vpc.extraCIDRs`, welche zusätzliche CIDR-Bereiche an die anhängen ControlPlaneSecurityGroup, sodass Subnetze außerhalb der VPC den Kubernetes-API-Endpunkt erreichen können. In ähnlicher Weise können Sie auch angeben, dass CIDR-Bereiche angehängt `vpc.extraIPv6CIDRs` werden. IPv6

Im Folgenden finden Sie ein Beispiel dafür, wie Sie den Kubernetes-API-Endpunktzugriff mit dem Unterbefehl konfigurieren könnten: `utils`

```
eksctl utils update-cluster-vpc-config --cluster=<clustername> --private-access=true --
public-access=false
```

Um die Einstellung mithilfe einer **`ClusterConfig`** Datei zu aktualisieren, verwenden Sie:

```
eksctl utils update-cluster-vpc-config -f config.yaml --approve
```

Beachten Sie, dass der aktuelle Wert beibehalten wird, wenn Sie kein Kennzeichen übergeben. Wenn Sie mit den vorgeschlagenen Änderungen zufrieden sind, fügen Sie das `approve` Kennzeichen hinzu, um die Änderung am laufenden Cluster vorzunehmen.

Beschränkung des Zugriffs auf den EKS Kubernetes Public API-Endpunkt

Bei der Standarderstellung eines EKS-Clusters wird der Kubernetes-API-Server öffentlich verfügbar gemacht.

Diese Funktion gilt nur für den öffentlichen Endpunkt. Die Konfigurationsoptionen für den API-Serverendpunktzugriff werden nicht geändert, und Sie haben weiterhin die Möglichkeit, den öffentlichen Endpunkt zu deaktivieren, sodass Ihr Cluster nicht über das Internet zugänglich ist. (Quelle: https://github.com/aws/containers-roadmap/issues/108 #issuecomment -552766489)

Um den Zugriff auf den öffentlichen API-Endpunkt beim Erstellen eines Clusters auf eine Reihe von zu beschränken, legen Sie das Feld fest: CIDRs **`publicAccessCIDRs`**

```
vpc:
  publicAccessCIDRs: ["1.1.1.1/32", "2.2.2.0/24"]
```

Um die Einschränkungen für einen vorhandenen Cluster zu aktualisieren, verwenden Sie:

```
eksctl utils update-cluster-vpc-config --cluster=<cluster> 1.1.1.1/32,2.2.2.0/24
```

Um die Einschränkungen mithilfe einer **`ClusterConfig`** Datei zu aktualisieren, legen Sie das neue CIDRs in fest **`vpc.publicAccessCIDRs`** und führen Sie Folgendes aus:

```
eksctl utils update-cluster-vpc-config -f config.yaml
```

> **Important**
>
> Beim Setzen `publicAccessCIDRs` und Erstellen von Knotengruppen `privateAccess` sollte entweder auf gesetzt werden `true` oder die Knoten IPs sollten der Liste hinzugefügt werden. `publicAccessCIDRs`

Wenn Knoten aufgrund eines eingeschränkten Zugriffs nicht auf den Cluster-API-Endpunkt zugreifen können, schlägt die Clustererstellung fehl, `context deadline exceeded` da die Knoten nicht auf den öffentlichen Endpunkt zugreifen können und dem Cluster nicht beitreten können.

Um sowohl den API-Server-Endpunktzugriff als auch den öffentlichen Zugriff CIDRs für einen Cluster mit einem einzigen Befehl zu aktualisieren, führen Sie folgenden Befehl aus:

```
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
```

Um die Einstellung mithilfe einer Konfigurationsdatei zu aktualisieren:

```
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
```

Aktualisierung von Subnetzen und Sicherheitsgruppen auf Kontrollebene

In dieser Dokumentation wird erklärt, wie Sie die Netzwerkkonfiguration der Steuerungsebene Ihres EKS-Clusters nach der ersten Erstellung ändern können. Dies beinhaltet die Aktualisierung der Subnetze und Sicherheitsgruppen der Kontrollebene.

Aktualisierung der Subnetze der Steuerungsebene

Wenn ein Cluster mit eksctl erstellt wird, wird eine Reihe von öffentlichen und privaten Subnetzen erstellt und an die EKS-API übergeben. EKS erstellt in diesen Subnetzen 2 bis 4 kontenübergreifende elastische Netzwerkschnittstellen (ENIs), um die Kommunikation zwischen der von EKS verwalteten Kubernetes-Steuerebene und Ihrer VPC zu ermöglichen.

Führen Sie folgenden Befehl aus, um die von der EKS-Steuerebene verwendeten Subnetze zu aktualisieren:

```
eksctl utils update-cluster-vpc-config --cluster=<cluster> --control-plane-subnet-
ids=subnet-1234,subnet-5678
```

Um die Einstellung mithilfe einer Konfigurationsdatei zu aktualisieren:

```
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
```

Ohne das `--approve` Flag protokolliert eksctl nur die vorgeschlagenen Änderungen. Wenn Sie mit den vorgeschlagenen Änderungen zufrieden sind, führen Sie den Befehl erneut mit der Markierung aus. `--approve`

Sicherheitsgruppen der Steuerungsebene werden aktualisiert

Zur Verwaltung des Datenverkehrs zwischen der Steuerungsebene und den Worker-Knoten unterstützt EKS die Weitergabe zusätzlicher Sicherheitsgruppen, die auf die von EKS bereitgestellten kontenübergreifenden Netzwerkschnittstellen angewendet werden. Führen Sie folgenden Befehl aus, um die Sicherheitsgruppen für die EKS-Steuerebene zu aktualisieren:

```
eksctl utils update-cluster-vpc-config --cluster=<cluster> --control-plane-security-
group-ids=sg-1234,sg-5678
```

Um die Einstellung mithilfe einer Konfigurationsdatei zu aktualisieren:

```
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
```

Um sowohl die Subnetze der Steuerungsebene als auch die Sicherheitsgruppen für einen Cluster zu aktualisieren, führen Sie folgenden Befehl aus:

```
eksctl utils update-cluster-vpc-config --cluster=<cluster> --control-plane-subnet-
ids=<> --control-plane-security-group-ids=<>
```

Um beide Felder mithilfe einer Konfigurationsdatei zu aktualisieren:

```
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
```

Ein vollständiges Beispiel finden Sie unter cluster-subnets-sgs.yaml.

Ohne das `--approve` Flag protokolliert eksctl nur die vorgeschlagenen Änderungen. Wenn Sie mit den vorgeschlagenen Änderungen zufrieden sind, führen Sie den Befehl erneut mit der Markierung aus. `--approve`

IPv6 Support

Definieren Sie die IP-Familie

Wenn `eksctl` Sie eine VPC erstellen, können Sie die IP-Version definieren, die verwendet werden soll. Die folgenden Optionen können konfiguriert werden:

- IPv4
- IPv6

Der Standardwert ist `IPv4`.

Verwenden Sie das folgende Beispiel, um es zu definieren:

```
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**
>
> Diese Einstellung befindet sich nur in der Konfigurationsdatei, nicht in einem CLI-Flag.

Wenn Sie verwenden IPv6, müssen Sie die folgenden Anforderungen konfigurieren:

- OIDC ist aktiviert
- verwaltete Addons sind wie oben gezeigt definiert
- Die Cluster-Version muss => 1.21 sein
- Die vpc-cni-Add-on-Version muss => 1.10.0 sein
- selbstverwaltete Knotengruppen werden in Clustern nicht unterstützt IPv6
- verwaltete Knotengruppen werden bei Clustern, deren Eigentümer sie nicht sind, nicht unterstützt IPv6
- `vpc.nat`und `serviceIPv4CIDR` Felder werden von eksctl für IPv6-Cluster erstellt und sind keine unterstützten Konfigurationsoptionen
- AutoAllocateIPv6 wird nicht zusammen mit unterstützt IPv6

Private Netzwerke können auch mit der IPv6 IP-Familie betrieben werden. Bitte folgen Sie den Anweisungen unter EKS Private Cluster.

IAM

Dieses Kapitel enthält Informationen zur Arbeit mit AWS IAM.

Themen:

- the section called “IAM-Benutzer und -Rollen verwalten”
  - Verwalten Sie IAM-Benutzer- und Rollenzuordnungen, um den Zugriff auf einen EKS-Cluster zu kontrollieren
  - Konfigurieren Sie IAM-Identitätszuordnungen über die Cluster-Konfigurationsdatei oder CLI-Befehle
- the section called “IAM-Rollen für Dienstkonten”
  - Verwaltung detaillierter Berechtigungen für Anwendungen, die auf Amazon EKS ausgeführt werden und andere AWS-Services verwenden
  - Erstellen und konfigurieren Sie IAM-Rollen und Kubernetes-Servicekontenpaare mit eksctl
  - Aktivieren Sie den IAM OpenID Connect Provider für einen EKS-Cluster, um IAM-Rollen für Dienstkonten zu aktivieren
- the section called “Grenze der IAM-Berechtigungen”
  - Steuern Sie die maximalen Berechtigungen, die IAM-Entitäten (Benutzern oder Rollen) gewährt werden, indem Sie eine Berechtigungsgrenze festlegen
- the section called “EKS Pod Identity-Assoziationen”
  - Konfigurieren Sie IAM-Berechtigungen für EKS-Add-Ons mithilfe der empfohlenen Pod-Identitätszuordnungen
  - Ermöglichen Sie Kubernetes-Anwendungen, die erforderlichen IAM-Berechtigungen für die Verbindung mit AWS-Services außerhalb des Clusters zu erhalten
  - Vereinfachen Sie den Prozess der Automatisierung von IAM-Rollen und Dienstkonten in mehreren EKS-Clustern
- the section called “IAM-Richtlinien”
  - Verwalten Sie IAM-Richtlinien für EKS-Knotengruppen, einschließlich Unterstützung für verschiedene Zusatzrichtlinien wie Image Builder, Auto Scaler, externes DNS, Cert Manager und mehr.
  - Fügen Sie Knotengruppen benutzerdefinierte Instanzrollen oder Inline-Richtlinien hinzu, um zusätzliche Berechtigungen zu erhalten.

- Ordnen Sie Knotengruppen spezifische AWS-verwaltete Richtlinien per ARN zu und stellen Sie sicher, dass erforderliche Richtlinien wie Amazon EKSWorker NodePolicy und Amazoneks_CNI_Policy enthalten sind.

- the section called “IAM-Mindestrichtlinien”
  - Verwaltung von EC2 AWS-Ressourcen, einschließlich Load Balancers, Auto-Scaling-Gruppen und Überwachung CloudWatch
  - CloudFormation AWS-Stacks erstellen und verwalten
  - Verwalten Sie Amazon Elastic Kubernetes Service (EKS) -Cluster, Knotengruppen und zugehörige Ressourcen wie IAM-Rollen und -Richtlinien

IAM-Mindestrichtlinien

In diesem Dokument werden die Mindestanforderungen an IAM-Richtlinien beschrieben, die für die Ausführung der wichtigsten Anwendungsfälle von eksctl erforderlich sind. Dies sind diejenigen, die für die Durchführung der Integrationstests verwendet werden.

> **Note**
>
> Denken Sie daran, sie `<account_id>` durch Ihre eigenen zu ersetzen.

> **Note**
>
> Eine von AWS verwaltete Richtlinie wird von AWS erstellt und verwaltet. Sie können die in den verwalteten AWS-Richtlinien definierten Berechtigungen nicht ändern.

Amazon EC2 FullAccess (von AWS verwaltete Richtlinie)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "ec2:*",
            "Effect": "Allow",
            "Resource": "*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "autoscaling.amazonaws.com",
                        "ec2scheduled.amazonaws.com",
                        "elasticloadbalancing.amazonaws.com",
                        "spot.amazonaws.com",
                        "spotfleet.amazonaws.com",
                        "transitgateway.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

AWSCloudFormationFullAccess (Von AWS verwaltete Richtlinie)

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

```
                "cloudformation:*"
            ],
            "Resource": "*"
        }
    ]
}
```

EksAllAccess

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        },
        {
            "Action": [
                "ssm:GetParameter",
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:*:<account_id>: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::<account_id>:instance-profile/eksctl-*",
                "arn:aws:iam::<account_id>:role/eksctl-*",
                "arn:aws:iam::<account_id>:policy/eksctl-*",
```

```
                "arn:aws:iam::<account_id>:oidc-provider/*",
                "arn:aws:iam::<account_id>:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "arn:aws:iam::<account_id>:role/eksctl-managed-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetUser"
            ],
            "Resource": [
                "arn:aws:iam::<account_id>:role/*",
                "arn:aws:iam::<account_id>:user/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "eks.amazonaws.com",
                        "eks-nodegroup.amazonaws.com",
                        "eks-fargate.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Grenze der IAM-Berechtigungen

Eine Berechtigungsgrenze ist eine erweiterte AWS IAM-Funktion, in der die maximalen Berechtigungen festgelegt wurden, die eine identitätsbasierte Richtlinie einer IAM-Entität gewähren kann. Dabei handelt es sich bei diesen Entitäten entweder um Benutzer oder Rollen. Wenn für eine Entität eine Berechtigungsgrenze festgelegt ist, kann diese Entität nur die Aktionen ausführen, die

sowohl nach ihren identitätsbasierten Richtlinien als auch nach ihren Berechtigungsgrenzen zulässig sind.

Sie können Ihre Berechtigungsgrenze so angeben, dass alle von eksctl erstellten identitätsbasierten Entitäten innerhalb dieser Grenze erstellt werden. Dieses Beispiel zeigt, wie den verschiedenen identitätsbasierten Entitäten, die von eksctl erstellt wurden, eine Berechtigungsgrenze zugewiesen werden kann:

```
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**
>
> Es ist nicht möglich, sowohl einen Rollen-ARN als auch eine Berechtigungsgrenze anzugeben.

Festlegung der VPC-CNI-Berechtigungsgrenze

Bitte beachten Sie, dass eksctl bei der Erstellung eines Clusters mit aktiviertem OIDC aus Sicherheitsgründen automatisch einen `iamserviceaccount` für das VPC-CNI erstellt. Wenn Sie eine Berechtigungsgrenze hinzufügen möchten, müssen Sie die in Ihrer Konfigurationsdatei manuell angeben: `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"
```

IAM-Richtlinien

Sie können Instance-Rollen an Knotengruppen anhängen. Workloads, die auf dem Knoten ausgeführt werden, erhalten IAM-Berechtigungen vom Knoten. Weitere Informationen finden Sie unter IAM-Rollen für Amazon EC2.

Auf dieser Seite sind die vordefinierten IAM-Richtlinienvorlagen aufgeführt, die in eksctl verfügbar sind. Diese Vorlagen vereinfachen den Prozess, Ihren EKS-Knoten die entsprechenden AWS-Serviceberechtigungen zu erteilen, ohne manuell benutzerdefinierte IAM-Richtlinien erstellen zu müssen.

Unterstützte IAM-Add-On-Richtlinien

Beispiel für alle unterstützten Add-On-Richtlinien:

```
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
```

Image Builder Builder-Richtlinie

Die `imageBuilder` Richtlinie ermöglicht den vollständigen Zugriff auf ECR (Elastic Container Registry). Dies ist nützlich, um beispielsweise einen CI-Server zu erstellen, der Bilder an ECR übertragen muss.

EBS-Richtlinie

Die `ebs` Richtlinie aktiviert den neuen EBS-CSI-Treiber (Elastic Block Store Container Storage Interface).

Cert Manager-Richtlinie

Die `certManager` Richtlinie ermöglicht das Hinzufügen von Datensätzen zu Route 53, um die DNS01-Herausforderung zu lösen. Weitere Informationen finden Sie hier.

Eine benutzerdefinierte Instanzrolle hinzufügen

In diesem Beispiel wird eine Knotengruppe erstellt, die eine bestehende IAM-Instanzrolle aus einem anderen Cluster wiederverwendet:

```
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"
```

Inline-Richtlinien anhängen

```
nodeGroups:
  - name: my-special-nodegroup
    iam:
      attachPolicy:
        Version: "2012-10-17"
        Statement:
        - Effect: Allow
          Action:
          - 's3:GetObject'
          Resource: 'arn:aws:s3:::example-bucket/*'
```

Richtlinien per ARN anhängen

```
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/AmazonEC2ContainerRegistryReadOnly
        - arn:aws:iam::aws:policy/ElasticLoadBalancingFullAccess
        - arn:aws:iam::1111111111:policy/kube2iam
      withAddonPolicies:
        autoScaler: true
        imageBuilder: true
```

> **Warning**
>
> Wenn eine Knotengruppe die enthält, muss **`attachPolicyARNs`** sie auch die Standard-Knotenrichtlinien enthalten, wie`AmazonEKSWorkerNodePolicy`, `AmazonEKS_CNI_Policy` und `AmazonEC2ContainerRegistryReadOnly` in diesem Beispiel.

IAM-Benutzer und -Rollen verwalten

> **Note**
>
> AWS schlägt vor, auf das Formular zu the section called “EKS Pod Identity-Assoziationen” migrieren. `aws-auth` ConfigMap

EKS-Cluster verwenden IAM-Benutzer und -Rollen, um den Zugriff auf den Cluster zu steuern. Die Regeln sind in einer Konfigurationsübersicht implementiert

ConfigMap Mit einem CLI-Befehl bearbeiten

angerufen`aws-auth`. `eksctl`stellt Befehle zum Lesen und Bearbeiten dieser Config-Map bereit.

Ruft alle Identitätszuordnungen ab:

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

Ruft alle Identitätszuordnungen ab, die einem ARN entsprechen:

```
eksctl get iamidentitymapping --cluster <clusterName> --region=<region> --arn
 arn:aws:iam::123456:role/testing-role
```

Erstellen Sie eine Identitätszuweisung:

```
 eksctl create iamidentitymapping --cluster  <clusterName> --region=<region> --arn
 arn:aws:iam::123456:role/testing --group system:masters --username admin
```

Löschen Sie eine Identitätszuweisung:

```
eksctl delete iamidentitymapping --cluster  <clusterName> --region=<region> --arn
 arn:aws:iam::123456:role/testing
```

> **Note**
>
> Der obige Befehl löscht ein einzelnes Zuordnungs-FIFO, sofern nicht `--all` angegeben. In diesem Fall werden alle Übereinstimmungen entfernt. Warnt, wenn weitere Zuordnungen gefunden werden, die dieser Rolle entsprechen.

Erstellen Sie eine Kontozuweisung:

```
eksctl create iamidentitymapping --cluster  <clusterName> --region=<region> --account
user-account
```

Löschen Sie eine Kontozuweisung:

```
eksctl delete iamidentitymapping --cluster  <clusterName> --region=<region> --account
user-account
```

ConfigMap Mit einer ClusterConfig Datei bearbeiten

Die Identitätszuordnungen können auch angegeben werden 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
```

IAM-Rollen für Dienstkonten

> **Tip**
>
> `eksctl`unterstützt die Konfiguration detaillierter Berechtigungen für EKS-Anwendungen, die über EKS Pod Identity Associations ausgeführt werden

Amazon EKS unterstützt hier Rollen für Service Accounts (IRSA)], wodurch Cluster-Betreiber AWS IAM-Rollen Kubernetes-Servicekonten zuordnen können.

Dies bietet ein detailliertes Berechtigungsmanagement für Apps, die auf EKS ausgeführt werden und andere AWS-Services verwenden. Dies können Apps sein, die S3, andere Datendienste (RDS, MQ, STS, DynamoDB) oder Kubernetes-Komponenten wie den AWS Load Balancer-Controller oder ExternalDNS verwenden.

Sie können ganz einfach IAM-Rollen- und Dienstkontopaare mit erstellen. `eksctl`

> **Note**
>
> Wenn Sie Instanzrollen verwendet haben und erwägen, stattdessen IRSA zu verwenden, sollten Sie die beiden Rollen nicht mischen.

Funktionsweise

Es funktioniert über den IAM OpenID Connect Provider (OIDC), den EKS verfügbar macht, und IAM-Rollen müssen mit Bezug auf den IAM OIDC-Anbieter (spezifisch für einen bestimmten EKS-Cluster) und einem Verweis auf das Kubernetes-Dienstkonto, an das sie gebunden werden, erstellt werden. Sobald eine IAM-Rolle erstellt wurde, sollte ein Dienstkonto den ARN dieser Rolle als Anmerkung (`eks.amazonaws.com/role-arn`) enthalten. Standardmäßig wird das Dienstkonto so erstellt oder aktualisiert, dass es die Rollenanmerkung enthält. Diese kann mithilfe des Flags `--role-only` deaktiviert werden.

In EKS gibt es einen Zugangscontroller, der AWS-Sitzungsanmeldedaten in Pods bzw. Rollen einfügt, basierend auf der Anmerkung zu dem vom Pod verwendeten Dienstkonto. Die Anmeldeinformationen

werden durch `AWS_ROLE_ARN` `AWS_WEB_IDENTITY_TOKEN_FILE` &-Umgebungsvariablen verfügbar gemacht. Wenn eine aktuelle Version des AWS-SDK verwendet wird (Einzelheiten zur genauen Version finden Sie hier), verwendet die Anwendung diese Anmeldeinformationen.

Im Namen `eksctl` der Ressource steht iamserviceaccount, was ein Paar aus IAM-Rolle und Dienstkonto darstellt.

Verwendung über CLI

> **Note**
>
> IAM-Rollen für Dienstkonten erfordern Kubernetes Version 1.13 oder höher.

Der IAM OIDC Provider ist standardmäßig nicht aktiviert. Sie können ihn mit dem folgenden Befehl aktivieren oder die Konfigurationsdatei verwenden (siehe unten):

```
eksctl utils associate-iam-oidc-provider --cluster=<clusterName>
```

Sobald Sie den IAM-OIDC-Anbieter mit dem Cluster verknüpft haben, führen Sie folgenden Befehl aus, um eine an ein Dienstkonto gebundene IAM-Rolle zu erstellen:

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=<serviceAccountName> --
namespace=<serviceAccountNamespace> --attach-policy-arn=<policyARN>
```

> **Note**
>
> Sie können `--attach-policy-arn` mehrfach angeben, dass mehr als eine Richtlinie verwendet werden soll.

Insbesondere können Sie ein Dienstkonto mit schreibgeschütztem Zugriff auf S3 erstellen, indem Sie Folgendes ausführen:

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=s3-read-only --attach-
policy-arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
```

Standardmäßig wird es im `default` Namespace erstellt, aber Sie können jeden anderen Namespace angeben, z. B.:

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=s3-read-only --
namespace=s3-app --attach-policy-arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess
```

> **Note**
>
> Wenn der Namespace noch nicht existiert, wird er erstellt.

Wenn Sie bereits ein Dienstkonto im Cluster erstellt haben (ohne IAM-Rolle), müssen Sie Flag verwenden`--override-existing-serviceaccounts`.

Benutzerdefiniertes Tagging kann auch auf die IAM-Rolle angewendet werden, indem Folgendes angegeben wird: `--tags`

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=<serviceAccountName> --
tags "Owner=John Doe,Team=Some Team"
```

CloudFormation generiert einen Rollennamen, der eine zufällige Zeichenfolge enthält. Wenn Sie einen vordefinierten Rollennamen bevorzugen, können Sie Folgendes angeben`--role-name`:

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=<serviceAccountName> --
role-name "custom-role-name"
```

Wenn das Dienstkonto mit einem anderen Tool wie Helm erstellt und verwaltet wird, verwenden Sie es, `--role-only` um Konflikte zu vermeiden. Das andere Tool ist dann für die Pflege der ARN-Anmerkung der Rolle verantwortlich. Beachten Sie, dass dies keine Auswirkungen auf die `roleOnly` `--role-only` /-Dienstkonten `--override-existing-serviceaccounts` hat. Die Rolle wird immer erstellt.

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=<serviceAccountName> --
role-only --role-name=<customRoleName>
```

Wenn Sie über eine bestehende Rolle verfügen, die Sie mit einem Dienstkonto verwenden möchten, können Sie anstelle der Richtlinien das `--attach-role-arn` Kennzeichen angeben. Um sicherzustellen, dass die Rolle nur von dem angegebenen Dienstkonto übernommen werden kann, sollten Sie hier ein Dokument zur Beziehungsrichtlinie einrichten].

```
eksctl create iamserviceaccount --cluster=<clusterName> --name=<serviceAccountName> --
attach-role-arn=<customRoleARN>
```

Um die Rollen und Berechtigungen eines Dienstkontos zu aktualisieren, können Sie ausführen`eksctl update iamserviceaccount`.

> **Note**
>
> `eksctl delete iamserviceaccount`löscht Kubernetes, `ServiceAccounts` auch wenn sie nicht von erstellt wurden. `eksctl`

Verwendung mit Konfigurationsdateien

Um die Verwaltung `iamserviceaccounts` mithilfe der Konfigurationsdatei durchzuführen, müssen Sie das gewünschte Konto einrichten `iam.withOIDC: true` und auflisten`iam.serviceAccount`.

Alle Befehle werden unterstützt*`--config-file`*. Sie können iamserviceaccounts genauso verwalten wie Nodegroups. Der `eksctl create iamserviceaccount` Befehl unterstützt `--include` und `--exclude` markiert (weitere Informationen darüber, wie diese funktionieren, finden Sie in diesem Abschnitt). Und der `eksctl delete iamserviceaccount` Befehl unterstützt `--only-missing` auch, sodass Sie Löschungen genauso durchführen können wie Nodegroups.

> **Note**
>
> IAM-Dienstkonten befinden sich innerhalb eines Namespaces, d. h. zwei Dienstkonten mit demselben Namen können in unterschiedlichen Namespaces existieren. Um also ein Dienstkonto als Teil von `--exclude` Flags eindeutig zu definieren`--include`, müssen Sie die Namenszeichenfolge im folgenden Format übergeben. `namespace/name` z. B.

```
eksctl create iamserviceaccount --config-file=<path> --include backend-apps/s3-reader
```

Die Option zum Aktivieren `wellKnownPolicies` ist für die Verwendung von IRSA in bekannten Anwendungsfällen wie `cluster-autoscaler` und `cert-manager` als Abkürzung für Richtlinienlisten enthalten.

Unterstützte bekannte Richtlinien und andere Eigenschaften von `serviceAccounts` sind im Konfigurationsschema dokumentiert.

Sie verwenden das folgende Konfigurationsbeispiel mit`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
```

Wenn Sie einen Cluster erstellen, ohne dass diese Felder gesetzt sind, können Sie die folgenden Befehle verwenden, um alles zu aktivieren, was Sie benötigen:

```
eksctl utils associate-iam-oidc-provider --config-file=<path>
eksctl create iamserviceaccount --config-file=<path>
```

Weitere Informationen

- Einführung detaillierter IAM-Rollen für Dienstkonten
- EKS-Benutzerhandbuch — IAM-Rollen für Dienstkonten
- Zuordnung von IAM-Benutzern und Rollen zu Kubernetes-RBAC-Rollen

EKS Pod Identity-Assoziationen

AWS EKS hat einen neuen erweiterten Mechanismus namens Pod Identity Association eingeführt, mit dem Clusteradministratoren Kubernetes-Anwendungen so konfigurieren können, dass sie IAM-Berechtigungen erhalten, die für die Verbindung mit AWS-Services außerhalb des Clusters erforderlich sind. Pod Identity Association nutzt IRSA, macht es jedoch direkt über die EKS-API konfigurierbar, sodass die Verwendung der IAM-API vollständig entfällt.

Infolgedessen müssen IAM-Rollen nicht mehr auf einen OIDC-Anbieter verweisen und sind daher nicht mehr an einen einzelnen Cluster gebunden. Das bedeutet, dass IAM-Rollen jetzt in mehreren EKS-Clustern verwendet werden können, ohne dass die Rollenvertrauensrichtlinie jedes Mal aktualisiert werden muss, wenn ein neuer Cluster erstellt wird. Dies wiederum macht die Duplizierung von Rollen überflüssig und vereinfacht den Prozess der IRSA-Automatisierung insgesamt.

Voraussetzungen

Hinter den Kulissen wird bei der Implementierung von Pod-Identitätszuordnungen ein Agent als Daemonset auf den Worker-Knoten ausgeführt. Um den erforderlichen Agenten auf dem Cluster auszuführen, bietet EKS ein neues Add-on namens EKS Pod Identity Agent. Daher muss für die Erstellung von Pod-Identitätszuordnungen (im Allgemeinen und mit`eksctl`) das `eks-pod-identity-agent` Addon auf dem Cluster vorinstalliert sein. Dieses Addon kann auf die gleiche Weise wie jedes andere unterstützte Addon erstellt `eksctl` werden.

> **Note**
>
> Wenn Sie den EKS Auto Mode Cluster verwenden, `eks-pod-identity-agent` ist dieser vorinstalliert und Sie können die Erstellung des Addons überspringen.

```
eksctl create addon --cluster my-cluster --name eks-pod-identity-agent
```

Wenn Sie beim Erstellen einer Pod-Identitätszuordnung eine bereits vorhandene IAM-Rolle verwenden, müssen Sie die Rolle außerdem so konfigurieren, dass sie dem neu eingeführten EKS-Dienstprinzipal () vertraut. `pods.eks.amazonaws.com` Ein Beispiel für eine IAM-Vertrauensrichtlinie finden Sie unten:

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

Wenn Sie stattdessen nicht den ARN einer vorhandenen Rolle für den Befehl create angeben, `eksctl` wird hinter den Kulissen eine erstellt und die obige Vertrauensrichtlinie konfiguriert.

Pod-Identitätszuordnungen erstellen

Für die Manipulation von Pod-Identitätszuordnungen `eksctl` wurde ein neues Feld hinzugefügt`iam.podIdentityAssociations`, z. B.

```
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
```

Ein vollständiges Beispiel finden Sie unter pod-identity-associations.yaml.

> **Note**
>
> Abgesehen `permissionPolicy` davon, dass es als Inline-Richtliniendokument verwendet wird, haben alle anderen Felder ein CLI-Flag-Gegenstück.

Das Erstellen von Pod-Identitätszuordnungen kann auf folgende Weise erreicht werden. Während der Clustererstellung, indem Sie die gewünschten Pod-Identitätszuordnungen als Teil der Konfigurationsdatei angeben und Folgendes ausführen:

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

Nach der Cluster-Erstellung entweder mit einer Konfigurationsdatei, z. B.

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

ODER mit CLI-Flags, z. B.

```
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**
>
> Einem Dienstkonto kann jeweils nur eine IAM-Rolle zugeordnet werden. Daher führt der Versuch, eine zweite Pod-Identitätszuordnung für dasselbe Dienstkonto zu erstellen, zu einem Fehler.

Pod-Identitätszuordnungen werden abgerufen

Führen Sie einen der folgenden Befehle aus, um alle Pod-Identitätszuordnungen für einen bestimmten Cluster abzurufen:

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

ODER

```
eksctl get podidentityassociation --cluster my-cluster
```

Um zusätzlich nur die Pod-Identitätszuordnungen innerhalb eines bestimmten Namespace abzurufen, verwenden Sie das `--namespace` Flag, z. B.

```
eksctl get podidentityassociation --cluster my-cluster --namespace default
```

Um schließlich eine einzelne Assoziation abzurufen, die einem bestimmten K8s-Dienstkonto entspricht, fügen Sie auch den obigen Befehl `--service-account-name` hinzu, d. h.

```
eksctl get podidentityassociation --cluster my-cluster --namespace default --service-
account-name s3-reader
```

Aktualisierung der Pod-Identitätszuordnungen

Um die IAM-Rolle einer oder mehrerer Pod-Identitätszuordnungen zu aktualisieren, übergeben Sie entweder die neuen Daten `roleARN(s)` an die Konfigurationsdatei, z. B.

```
iam:
  podIdentityAssociations:
    - namespace: default
      serviceAccountName: s3-reader
      roleARN: new-role-arn-1
    - namespace: dev
      serviceAccountName: app-cache-access
      roleARN: new-role-arn-2
```

und führe Folgendes aus:

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

ODER (um eine einzelne Assoziation zu aktualisieren) übergeben Sie die neuen `--role-arn` über CLI-Flags:

```
eksctl update podidentityassociation --cluster my-cluster --namespace default --
service-account-name s3-reader --role-arn new-role-arn
```

Pod-Identitätszuordnungen löschen

Um eine oder mehrere Pod-Identitätszuordnungen zu löschen, übergeben `namespace(s)` Sie sie entweder `serviceAccountName(s)` an die Konfigurationsdatei, z. B.

```
iam:
  podIdentityAssociations:
    - namespace: default
      serviceAccountName: s3-reader
    - namespace: dev
      serviceAccountName: app-cache-access
```

und führe Folgendes aus:

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

ODER (um eine einzelne Assoziation zu löschen) übergeben Sie die `--namespace` und `--service-account-name` über die CLI-Flags:

```
eksctl delete podidentityassociation --cluster my-cluster --namespace default --
service-account-name s3-reader
```

Unterstützung von EKS-Add-Ons für Pod-Identitätszuordnungen

EKS-Add-Ons unterstützen auch den Empfang von IAM-Berechtigungen über EKS Pod Identity Associations. Die Konfigurationsdatei enthält drei Felder, mit denen diese konfiguriert werden können:`addon.podIdentityAssociations`, und`addonsConfig.autoApplyPodIdentityAssociations`. `addon.useDefaultPodIdentityAssociations` Sie können entweder die gewünschten Pod-Identitätszuordnungen explizit konfigurieren`addon.podIdentityAssociations`, indem Sie entweder `addonsConfig.autoApplyPodIdentityAssociations` oder verwenden, oder die empfohlene Konfiguration der Pod-Identität `eksctl` automatisch auflösen (und anwenden) lassen. `addon.useDefaultPodIdentityAssociations`

> **Note**
>
> Nicht alle EKS-Add-Ons unterstützen Pod-Identitätszuordnungen beim Start. In diesem Fall müssen die erforderlichen IAM-Berechtigungen weiterhin mithilfe der IRSA-Einstellungen bereitgestellt werden.

Addons mit IAM-Berechtigungen erstellen

Wenn Sie ein Addon erstellen, für das IAM-Berechtigungen erforderlich sind, `eksctl` wird zunächst geprüft, ob entweder Pod-Identitätszuordnungen oder IRSA-Einstellungen explizit als Teil der Konfigurationsdatei konfiguriert wurden, und falls ja, verwenden Sie eine davon, um die Berechtigungen für das Addon zu konfigurieren. z.B.

```
addons:
- name: vpc-cni
  podIdentityAssociations:
```

```
  - serviceAccountName: aws-node
    permissionPolicyARNs: ["arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"]
```

und starte

```
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**
>
> Das gleichzeitige Einstellen von Pod-Identitäten und IRSA ist nicht zulässig und führt zu einem Validierungsfehler.

`eksctl`Bietet für EKS-Add-Ons, die Pod-Identitäten unterstützen, die Option, alle empfohlenen IAM-Berechtigungen bei der Addon-Erstellung automatisch zu konfigurieren. Dies kann durch einfaches Einstellen `addonsConfig.autoApplyPodIdentityAssociations: true` in der Konfigurationsdatei erreicht werden. z.B.

```
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
```

und laufe

```
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
```

Entsprechendes kann dasselbe über CLI-Flags erfolgen, z. B.

```
eksctl create addon --cluster my-cluster --name vpc-cni --auto-apply-pod-identity-
associations
```

Um ein vorhandenes Addon zur Verwendung der Pod-Identität mit den empfohlenen IAM-Richtlinien zu migrieren, verwenden Sie

```
addons:
- name: vpc-cni
  useDefaultPodIdentityAssociations: true
```

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

Aktualisierung von Addons mit IAM-Berechtigungen

Bei der Aktualisierung eines Addons `addon.PodIdentityAssociations` stellt die Angabe die einzige Informationsquelle für den Status dar, den das Addon nach Abschluss des Aktualisierungsvorgangs haben soll. Hinter den Kulissen werden verschiedene Arten von Operationen ausgeführt, um den gewünschten Status zu erreichen, d. h.

- Pod-Identitäten erstellen, die in der Konfigurationsdatei vorhanden sind, aber im Cluster fehlen
- löschen Sie vorhandene Pod-Identitäten, die zusammen mit allen zugehörigen IAM-Ressourcen aus der Konfigurationsdatei entfernt wurden
- aktualisieren Sie bestehende Pod-Identitäten, die auch in der Konfigurationsdatei vorhanden sind und für die sich die IAM-Berechtigungen geändert haben

> **Note**
>
> Der Lebenszyklus von Pod-Identitätszuordnungen, die EKS Add-ons gehören, wird direkt von der EKS Addons API verwaltet.

Sie können `eksctl update podidentityassociation` (um IAM-Berechtigungen zu aktualisieren) oder `eksctl delete podidentityassociations` (um die Zuordnung zu entfernen) nicht für Verknüpfungen verwenden, die mit einem Amazon EKS-Add-on verwendet werden. Stattdessen `eksctl delete addon` soll `eksctl update addon` oder verwendet werden.

Sehen wir uns ein Beispiel für das Obige an, beginnend mit der Analyse der anfänglichen Pod-Identitätskonfiguration für das 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"
    }
]
```

Verwenden Sie nun die folgende Konfiguration:

```
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
```

und lauf

```
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
```

Überprüfen Sie nun, ob die Pod-Identitätskonfiguration korrekt aktualisiert wurde

```
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"
    }
]
```

Um alle Pod-Identitätszuordnungen aus einem Addon zu entfernen, `addon.PodIdentityAssociations` muss es explizit auf gesetzt werden`[]`, z. B.

```
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: []
```

und laufe

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

Löschen von Addons mit IAM-Berechtigungen

Durch das Löschen eines Addons werden auch alle mit dem Addon verknüpften Pod-Identitäten entfernt. Durch das Löschen des Clusters wird derselbe Effekt für alle Addons erzielt. Alle IAM-Rollen für Pod-Identitäten, die von erstellt wurden`eksctl`, werden ebenfalls gelöscht.

Migrieren vorhandener IAM-Servicekonten und Addons zu Pod-Identitätszuordnungen

Es gibt einen Befehl `eksctl` utils für die Migration vorhandener IAM-Rollen für Dienstkonten zu Pod-Identitätszuordnungen, d. h.

```
eksctl utils migrate-to-pod-identity --cluster my-cluster --approve
```

Hinter den Kulissen führt der Befehl die folgenden Schritte aus:

- installieren Sie das `eks-pod-identity-agent` Addon, falls es nicht bereits auf dem Cluster aktiv ist
- Identifizieren Sie alle IAM-Rollen, die iamserviceaccounts zugeordnet sind
- Identifizieren Sie alle IAM-Rollen, die mit EKS-Addons verknüpft sind, die Pod-Identitätszuordnungen unterstützen
- aktualisieren Sie die IAM-Vertrauensrichtlinie aller identifizierten Rollen mit einer zusätzlichen vertrauenswürdigen Entität, die auf den neuen EKS-Dienstprinzipal verweist (und entfernen Sie optional die bestehende OIDC-Anbieter-Vertrauensbeziehung)
- Pod-Identitätszuordnungen für gefilterte Rollen erstellen, die iamserviceaccounts zugeordnet sind
- Aktualisieren Sie EKS-Addons mit Pod-Identitäten (die EKS-API erstellt die Pod-Identitäten hinter den Kulissen)

Wenn Sie den Befehl ohne die `--approve` Markierung ausführen, wird nur ein Plan ausgegeben, der aus einer Reihe von Aufgaben besteht, die die obigen Schritte widerspiegeln, z. B.

```
[#]  (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
```

Die bestehende Vertrauensstellung zwischen OIDC-Anbietern wird immer aus den IAM-Rollen entfernt, die mit EKS-Add-ons verknüpft sind. Um außerdem die bestehende Vertrauensstellung eines OIDC-Anbieters aus den IAM-Rollen zu entfernen, die iamserviceaccounts zugeordnet sind, führen Sie den Befehl mit Flag aus, z. B. `--remove-oidc-provider-trust-relationship`

```
eksctl utils migrate-to-pod-identity --cluster my-cluster --approve --remove-oidc-
provider-trust-relationship
```

Weitere Verweise

Offizielle Unterstützung von AWS-Benutzerdokumenten für EKS Add-Ons für Pod-Identitäten

Offizieller AWS-Blogbeitrag zu Pod Identity Associations

Offizielle AWS-Benutzerdokumente für Pod Identity Associations

Optionen für die Bereitstellung

Dieses Kapitel behandelt die Verwendung von eksctl zur Verwaltung von EKS-Clustern, die in alternativen Umgebungen bereitgestellt werden.

Die genauesten Informationen zu den EKS-Bereitstellungsoptionen finden Sie unter Bereitstellen von Amazon EKS-Clustern in Cloud- und lokalen Umgebungen im EKS-Benutzerhandbuch.

Themen:

- the section called “EKS überall”
  - Verwenden Sie eksctl mit Amazon EKS Anywhere Anywhere-Clustern.
  - Amazon EKS Anywhere ist eine von AWS entwickelte Container-Management-Software, die es einfacher macht, Kubernetes vor Ort und am Edge auszuführen und zu verwalten.
- the section called “Support für AWS Outposts”
  - Verwenden Sie eksctl mit EKS-Clustern auf AWS Outposts.
  - AWS Outposts ist eine Familie vollständig verwalteter Lösungen, die AWS-Infrastruktur und -Services für praktisch jeden lokalen oder Edge-Standort bereitstellen und so ein wirklich konsistentes Hybrid-Erlebnis bieten.
  - Durch die Unterstützung von AWS Outposts in eksctl können Sie lokale Cluster erstellen, wobei der gesamte Kubernetes-Cluster, einschließlich der EKS-Steuerebene und der Worker-Knoten, lokal auf AWS Outposts ausgeführt wird.
- the section called “EKS-Hybridknoten”
  - Führen Sie lokale und Edge-Anwendungen auf einer vom Kunden verwalteten Infrastruktur mit denselben AWS EKS-Clustern, Funktionen und Tools aus, die Sie in der AWS-Cloud verwenden.

EKS überall

`eksctl`bietet Zugriff auf die AWS-Funktion, die `EKS Anywhere` mit dem Unterbefehl `eksctl anywhere` aufgerufen wird. Dies erfordert, dass die `eksctl-anywhere` Binärdatei aktiviert `PATH` ist. Bitte folgen Sie den Anweisungen hier Installieren Sie eksctl-anywhere, um es zu installieren.

Wenn Sie fertig sind, führen Sie Anywhere-Befehle aus, indem Sie Folgendes ausführen:

```
eksctl anywhere version
```

```
v0.5.0
```

Weitere Informationen zu EKS Anywhere finden Sie auf der EKS Anywhere-Website.

Support für AWS Outposts

> **Warning**
>
> EKS Managed Nodegroups werden auf Outposts nicht unterstützt.

Erweiterung vorhandener Cluster auf AWS Outposts

Sie können einen vorhandenen EKS-Cluster, der in einer AWS-Region ausgeführt wird, auf AWS Outposts erweitern, indem Sie neue Knotengruppen einrichten`nodeGroup.outpostARN`, um Knotengruppen in Outposts zu erstellen, wie 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 diesem Setup läuft die EKS-Kontrollebene in einer AWS-Region, während Knotengruppen mit `outpostARN` Set auf dem angegebenen Outpost ausgeführt werden. Wenn zum ersten Mal

eine Knotengruppe auf Outposts erstellt wird, erweitert eksctl die VPC, indem Subnetze auf dem angegebenen Outpost erstellt werden. Diese Subnetze werden verwendet, um Knotengruppen zu erstellen, die sich gesetzt haben. `outpostARN`

Kunden mit einer bereits vorhandenen VPC müssen die Subnetze auf Outposts erstellen und sie weitergeben, wie 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"
```

Einen lokalen Cluster auf AWS Outposts erstellen

> **Note**
>
> Lokale Cluster unterstützen nur Outpost-Racks.

> **Note**
>
> Nur Amazon Linux 2 wird für Knotengruppen unterstützt, wenn sich die Steuerungsebene auf Outposts befindet. Für Knotengruppen auf Outposts werden nur EBS-GP2-Volumetypen unterstützt.

Durch die Unterstützung von AWS Outposts in eksctl können Sie lokale Cluster erstellen, wobei der gesamte Kubernetes-Cluster, einschließlich der EKS-Steuerebene und der Worker-Knoten, lokal auf AWS Outposts ausgeführt wird. Kunden können entweder einen lokalen Cluster erstellen, in dem sowohl die EKS-Kontrollebene als auch die Worker-Knoten lokal auf AWS Outposts ausgeführt werden, oder sie können einen vorhandenen EKS-Cluster, der in einer AWS-Region läuft, auf AWS Outposts erweitern, indem sie Worker-Knoten auf Outposts erstellen.

Um die EKS-Kontrollebene und die Knotengruppen auf AWS Outposts `outpost.controlPlaneOutpostARN` zu erstellen, stellen Sie den Outpost-ARN ein, wie 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
```

Dadurch wird eksctl angewiesen, die EKS-Kontrollebene und die Subnetze auf dem angegebenen Outpost zu erstellen. Da ein Outposts-Rack in einer einzigen Availability Zone existiert, erstellt eksctl nur ein öffentliches und ein privates Subnetz. eksctl verknüpft die erstellte VPC nicht mit einem lokalen Gateway, sodass eksctl keine Konnektivität zum API-Server hat und keine Knotengruppen

erstellen kann. Wenn der während der Clustererstellung Knotengruppen `ClusterConfig` enthält, muss der Befehl daher wie folgt ausgeführt werden: `--without-nodegroup`

```
eksctl create cluster -f outpost.yaml --without-nodegroup
```

Es liegt in der Verantwortung des Kunden, die von eksctl erstellte VPC nach der Clustererstellung dem lokalen Gateway zuzuordnen, um die Konnektivität zum API-Server zu ermöglichen. Nach diesem Schritt können Knotengruppen mit erstellt werden. `eksctl create nodegroup`

Sie können optional den Instanztyp für die Knoten der Steuerungsebene in `outpost.controlPlaneInstanceType` oder für die darin enthaltenen Knotengruppen angeben`nodeGroup.instanceType`, aber der Instanztyp muss auf Outpost vorhanden sein, sonst gibt eksctl einen Fehler zurück. Standardmäßig versucht eksctl, den kleinsten verfügbaren Instanztyp auf Outpost für die Knoten und Knotengruppen der Kontrollebene auszuwählen.

Wenn sich die Kontrollebene auf Outposts befindet, werden Knotengruppen auf diesem Außenposten erstellt. Sie können optional den Outpost-ARN für die Nodegroup in angeben, er muss `nodeGroup.outpostARN` jedoch mit dem Outpost-ARN der Kontrollebene übereinstimmen.

```
# 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
```

Vorhandene VPC

Kunden mit einer vorhandenen VPC können lokale Cluster auf AWS Outposts erstellen, indem sie die Subnetzkonfiguration wie folgt angeben: `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
```

Die Subnetze müssen auf dem in angegebenen Outpost vorhanden sein, andernfalls gibt eksctl einen `outpost.controlPlaneOutpostARN` Fehler zurück. Sie können bei der Clustererstellung auch Knotengruppen angeben, wenn Sie Zugriff auf das lokale Gateway für das Subnetz haben oder Konnektivität zu VPC-Ressourcen haben.

Funktionen werden auf lokalen Clustern nicht unterstützt

- Addons
- IAM-Rollen für Dienstkonten
- IPv6
- Identitätsanbieter
- Fargate
- KMS-Verschlüsselung
- Lokale Zonen
- Zimmermann
- Instanzenauswahl
- Availability Zones können nicht angegeben werden, da standardmäßig die Outpost-Verfügbarkeitszone verwendet wird.
- `vpc.publicAccessCIDRs`und `vpc.autoAllocateIPv6` werden nicht unterstützt.
- Der öffentliche Endpunktzugriff auf den API-Server wird nicht unterstützt, da ein lokaler Cluster nur mit privatem Endpunktzugriff erstellt werden kann.

Weitere Informationen

- Amazon EKS auf AWS Outposts
- Lokale Cluster für Amazon EKS auf AWS Outposts
- Lokale Cluster erstellen
- Selbstverwaltete Amazon Linux-Knoten auf einem Outpost starten

Sicherheit

`eksctl`bietet einige Optionen, die die Sicherheit Ihres EKS-Clusters verbessern können.

`withOIDC`

Aktivieren Sie `withOIDC`diese Option, um automatisch eine IRSA für das Amazon CNI-Plugin zu erstellen und die den Knoten in Ihrem Cluster gewährten Berechtigungen einzuschränken, anstatt die erforderlichen Berechtigungen nur dem CNI-Dienstkonto zu gewähren.

Der Hintergrund wird in dieser AWS-Dokumentation beschrieben.

`disablePodIMDS`

Für verwaltete und nicht verwaltete Knotengruppen ist eine `disablePodIMDS`Option verfügbar, die verhindert, dass alle in dieser Knotengruppe ausgeführten Netzwerk-Pods, die keine Host-Netzwerk-Pods sind, IMDS-Anfragen stellen.

> **Note**
>
> Dies kann nicht zusammen mit verwendet werden. `withAddonPolicies`

KMS-Envelope-Verschlüsselung für EKS-Cluster

> **Note**
>
> Amazon Elastic Kubernetes Service (Amazon EKS) bietet eine Standard-Envelope-Verschlüsselung für alle Kubernetes-API-Daten in EKS-Clustern, auf denen Kubernetes Version 1.28 oder höher ausgeführt wird. Weitere Informationen finden Sie unter Standard-Envelope-Verschlüsselung für alle Kubernetes-API-Daten im EKS-Benutzerhandbuch.

EKS unterstützt die Verwendung von AWS-KMS-Schlüsseln zur Envelope-Verschlüsselung von in EKS gespeicherten Kubernetes-Geheimnissen. Die Envelope-Verschlüsselung fügt eine zusätzliche, vom Kunden verwaltete Verschlüsselungsebene für Anwendungsgeheimnisse oder Benutzerdaten hinzu, die in einem Kubernetes-Cluster gespeichert sind.

Bisher unterstützte Amazon EKS die Aktivierung der Umschlagverschlüsselung mit KMS-Schlüsseln nur während der Clustererstellung. Jetzt können Sie die Envelope-Verschlüsselung für Amazon EKS-Cluster jederzeit aktivieren.

Weitere Informationen zur Verwendung der Unterstützung von EKS-Verschlüsselungsanbietern defense-in-depth finden Sie im AWS-Container-Blog.

Einen Cluster mit aktivierter KMS-Verschlüsselung erstellen

```
# 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
```

Aktivieren der KMS-Verschlüsselung auf einem vorhandenen Cluster

Führen Sie folgenden Befehl aus, um die KMS-Verschlüsselung auf einem Cluster zu aktivieren, auf dem sie noch nicht aktiviert ist

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

oder ohne Konfigurationsdatei:

```
eksctl utils enable-secrets-encryption --cluster=kms-cluster --key-arn=arn:aws:kms:us-
west-2:<account>:key/<key> --region=<region>
```

eksctl aktiviert nicht nur die KMS-Verschlüsselung auf dem EKS-Cluster, sondern verschlüsselt auch alle vorhandenen Kubernetes-Geheimnisse mithilfe des neuen KMS-Schlüssels erneut, indem es sie mit der Anmerkung aktualisiert. `eksctl.io/kms-encryption-timestamp` Dieses Verhalten kann durch Übergabe deaktiviert werden, wie in: `--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>
```

Wenn in einem Cluster die KMS-Verschlüsselung bereits aktiviert ist, fährt eksctl mit der erneuten Verschlüsselung aller vorhandenen Geheimnisse fort.

> **Note**
>
> Sobald die KMS-Verschlüsselung aktiviert ist, kann sie nicht mehr deaktiviert oder aktualisiert werden, sodass sie einen anderen KMS-Schlüssel verwendet.

Fehlerbehebung

Dieses Thema enthält Anweisungen zur Behebung häufiger Fehler mit Eksctl.

Die Stack-Erstellung ist fehlgeschlagen

Sie können das `--cfn-disable-rollback` Flag verwenden, um Cloudformation daran zu hindern, fehlgeschlagene Stacks rückgängig zu machen, um das Debuggen zu vereinfachen.

Die Subnetz-ID „Subnetz-11111111" ist nicht dasselbe wie „Subnetz-22222222"

Angesichts einer Konfigurationsdatei, die Subnetze für eine VPC wie folgt spezifiziert:

```
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: []
```

Ein Fehler `subnet ID "subnet-11111111" is not the same as "subnet-22222222"` bedeutet, dass die angegebenen Subnetze nicht in der richtigen Availability Zone platziert wurden. Prüfen Sie in der AWS-Konsole, welche Subnetz-ID für jede Availability Zone die richtige ist.

In diesem Beispiel wäre die richtige Konfiguration für die VPC:

```
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}
```

Probleme beim Löschen

Wenn Ihr Löschvorgang nicht funktioniert oder Sie vergessen, den Löschvorgang hinzuzufügen`--wait`, müssen Sie möglicherweise die anderen Tools von Amazon verwenden, um die Cloudformation-Stacks zu löschen. Dies kann über die GUI oder mit der AWS-CLI erreicht werden.

kubectl loggt und kubectl run schlägt mit einem Autorisierungsfehler fehl

Wenn Ihre Knoten in einem privaten Subnetz bereitgestellt werden `kubectl logs` und/oder mit einem Fehler wie dem folgenden `kubectl run` fehlschlagen:

```
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)
```

Dann müssen Sie möglicherweise einstellen enableDnsHostnames. Weitere Details finden Sie in dieser Ausgabe.

Ankündigungen

Dieses Thema behandelt frühere Ankündigungen neuer Eksctl-Funktionen.

Standard für verwaltete Knotengruppen

Ab eksctl v0.58.0 erstellt eksctl standardmäßig verwaltete Knotengruppen, wenn für und keine Datei angegeben ist. `ClusterConfig` `eksctl create cluster` `eksctl create nodegroup` Um eine selbstverwaltete Knotengruppe zu erstellen, übergeben Sie. `--managed=false` Dies kann dazu führen, dass Skripts, die keine Konfigurationsdatei verwenden, beschädigt werden, wenn eine Funktion verwendet wird, die in verwalteten Knotengruppen nicht unterstützt wird, z. B. Windows-Knotengruppen. Um dieses Problem zu beheben`--managed=false`, übergeben Sie Ihre Knotengruppen-Konfiguration oder geben Sie sie in einer `ClusterConfig` Datei an, indem Sie das `nodeGroups` Feld verwenden, das eine selbstverwaltete Knotengruppe erstellt.

Nodegroup Bootstrap Override für Benutzerdefiniert AMIs

Diese Änderung wurde in der Ausgabe Breaking: overrideBootstrapCommand soon... angekündigt. Jetzt ist es in dieser PR eingetreten. Bitte lesen Sie die beigefügte Ausgabe sorgfältig darüber, warum wir uns entschieden haben, die Unterstützung von benutzerdefinierten Skripten AMIs ohne Bootstrap-Skripte oder mit partiellen Bootstrap-Skripten einzustellen.

Wir stellen immer noch einen Helfer zur Verfügung! Die Migration ist hoffentlich nicht so schmerzhaft. `eksctl`bietet immer noch ein Skript, das, wenn es bereitgestellt wird, einige hilfreiche Umgebungseigenschaften und -einstellungen exportiert. Dieses Skript befindet sich hier.

Die folgenden Umgebungseigenschaften stehen Ihnen zur Verfügung:

```
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
```

Das Minimum, das verwendet werden muss, wenn das Überschreiben `eksctl` nicht fehlschlägt, sind Labels! `eksctl`verlässt sich darauf, dass sich ein bestimmter Satz von Labels auf dem Knoten befindet, damit er sie finden kann. Wenn Sie die Überschreibung definieren, geben Sie bitte mindestens den folgenden Override-Befehl an:

```
    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}"
```

Für Knotengruppen, die keinen ausgehenden Internetzugang haben, müssen Sie das `--apiserver-endpoint` Bootstrap-Skript `--b64-cluster-ca` wie folgt angeben:

```
    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}
```

Beachten Sie die Einstellung `--node-labels`. Wenn dies nicht definiert ist, tritt der Knoten dem Cluster bei, `eksctl` wird aber letztendlich beim letzten Schritt, wenn er darauf wartet, dass die Knoten wieder da sind, eine Zeitüberschreitung eintritt. `Ready` Es führt eine Kubernetes-Suche nach Knoten durch, die das Label tragen. `alpha.eksctl.io/nodegroup-name=<cluster-name>` Dies gilt nur für nicht verwaltete Knotengruppen. Für verwaltet wird ein anderes Label verwendet.

Wenn es überhaupt möglich ist, zu verwalteten Knotengruppen zu wechseln, um diesen Aufwand zu vermeiden, ist jetzt die Zeit dafür gekommen. Macht das ganze Überschreiben viel einfacher.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.