Amazon EKS

# Panduan Pengguna Eksctl

# Panduan Pengguna Eksctl: Amazon EKS

# Table of Contents

# Apa itu Eksctl?

eksctl adalah alat utilitas baris perintah yang mengotomatiskan dan menyederhanakan proses pembuatan, pengelolaan, dan pengoperasian kluster Amazon Elastic Kubernetes Service (Amazon EKS). Ditulis dalam Go, eksctl menyediakan sintaks deklaratif melalui konfigurasi YAMAL dan perintah CLI untuk menangani operasi cluster EKS kompleks yang jika tidak memerlukan beberapa langkah manual di berbagai layanan AWS.

eksctl sangat berharga bagi para DevOps insinyur, tim platform, dan administrator Kubernetes yang perlu secara konsisten menerapkan dan mengelola kluster EKS dalam skala besar. Ini sangat berguna untuk organisasi yang beralih dari Kubernetes yang dikelola sendiri ke EKS, atau mereka yang menerapkan praktik infrastruktur sebagai kode (IAc), karena dapat diintegrasikan ke dalam pipeline dan alur kerja otomatisasi yang ada. CI/CD Alat ini mengabstraksi banyak interaksi kompleks antara layanan AWS yang diperlukan untuk penyiapan kluster EKS, seperti konfigurasi VPC, pembuatan peran IAM, dan manajemen grup keamanan.

Fitur utama eksctl termasuk kemampuan untuk membuat cluster EKS yang berfungsi penuh dengan satu perintah, dukungan untuk konfigurasi jaringan khusus, manajemen grup node otomatis, dan integrasi alur kerja. GitOps Alat ini mengelola peningkatan klaster, menskalakan grup node, dan menangani manajemen add-on melalui pendekatan deklaratif. eksctl juga menyediakan kemampuan canggih seperti konfigurasi profil Fargate, kustomisasi grup node terkelola, dan integrasi instans spot, sambil mempertahankan kompatibilitas dengan alat dan layanan AWS lainnya melalui integrasi AWS SDK asli.

## Fitur

Fitur yang saat ini diterapkan adalah:

- Buat, dapatkan, daftar, dan hapus cluster
- Buat, tiriskan, dan hapus nodegroup
- Skala nodegroup
- Perbarui klaster
- Gunakan kustom AMIs
- Konfigurasikan Jaringan VPC
- Konfigurasikan akses ke titik akhir API
- Support untuk nodegroup GPU

- Instance spot dan instance campuran
- Manajemen IAM dan Kebijakan Add-on
- Daftar cluster Cloudformation tumpukan
- Instal coredns
- Menulis berkas kubeconfig untuk sebuah klaster

# Eksctl FAQ

## Umum

Dapatkah saya menggunakan **`eksctl`** untuk mengelola cluster yang tidak dibuat oleh? **`eksctl`**

Ya! Dari versi `0.40.0` Anda dapat menjalankan `eksctl` terhadap cluster apa pun, apakah itu dibuat oleh `eksctl` atau tidak. Untuk informasi selengkapnya, lihat the section called “Cluster yang tidak dibuat eksctl”.

## Nodegroup

Bagaimana saya bisa mengubah tipe instance nodegroup saya?

Dari sudut pandang`eksctl`, nodegroup tidak dapat diubah. Ini berarti bahwa setelah dibuat, satu-satunya hal yang `eksctl` dapat dilakukan adalah menskalakan nodegroup ke atas atau ke bawah.

Untuk mengubah jenis instance, buat nodegroup baru dengan tipe instance yang diinginkan, lalu tiriskan sehingga beban kerja berpindah ke yang baru. Setelah langkah itu selesai, Anda dapat menghapus nodegroup lama.

Bagaimana saya bisa melihat data pengguna yang dihasilkan untuk nodegroup?

Pertama, Anda memerlukan nama tumpukan Cloudformation yang mengelola nodegroup:

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

Anda akan melihat nama yang mirip dengan`eksctl-CLUSTER_NAME-nodegroup-NODEGROUP_NAME`.

Anda dapat menjalankan yang berikut ini untuk mendapatkan data pengguna. Perhatikan baris terakhir yang menerjemahkan kode dari base64 dan mendekompresi data gzip.

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

## Ingress

Bagaimana cara mengatur ingress dengan? **`eksctl`**

Sebaiknya gunakan AWS Load Balancer Controller. Dokumentasi tentang cara menerapkan controller ke cluster Anda, serta cara bermigrasi dari ALB Ingress Controller lama, dapat ditemukan di sini.

Untuk Nginx Ingress Controller, setup akan sama dengan cluster Kubernetes lainnya.

## Kubectl

Saya menggunakan proxy HTTPS dan validasi sertifikat cluster gagal, bagaimana saya bisa menggunakan sistem? CAs

Tetapkan variabel lingkungan `KUBECONFIG_USE_SYSTEM_CA` untuk `kubeconfig` menghormati otoritas sertifikat sistem.

# Lari Kering

Fitur dry-run memungkinkan Anda untuk memeriksa dan mengubah instance yang cocok dengan pemilih instance sebelum melanjutkan untuk membuat nodegroup.

Kapan `eksctl create cluster` dipanggil dengan opsi pemilih instance dan`--dry-run`, eksctl akan menampilkan ClusterConfig file yang berisi nodegroup yang mewakili opsi CLI dan tipe instance yang disetel ke instance yang cocok dengan kriteria sumber daya pemilih instance.

```
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
```

Yang dihasilkan kemudian ClusterConfig dapat diteruskan ke`eksctl create cluster`:

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

Ketika ClusterConfig file dilewatkan`--dry-run`, eksctl akan menampilkan ClusterConfig file yang berisi nilai-nilai yang ditetapkan dalam file.

## Opsi Satu Kali di eksctl

Ada opsi satu kali tertentu yang tidak dapat direpresentasikan dalam `ClusterConfig` file, misalnya,`--install-vpc-controllers`.

Diharapkan bahwa:

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

diikuti oleh:

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

akan setara dengan menjalankan perintah pertama tanpa`--dry-run`.

oleh karena itu eksctl melarang opsi lewat yang tidak dapat direpresentasikan dalam file konfigurasi saat diteruskan. `--dry-run`

> **Important**
>
> Jika Anda perlu meneruskan profil AWS, atur variabel `AWS_PROFILE` lingkungan, alih-alih meneruskan opsi `--profile` CLI.

# Tutorial

Topik ini memandu Anda melalui instalasi dan konfigurasi eksctl, kemudian menggunakannya untuk membuat cluster Amazon EKS.

## Langkah 1: Instal eksctl

Selesaikan langkah-langkah berikut untuk mengunduh dan menginstal eksctl versi terbaru di perangkat Linux atau macOS Anda:

Untuk menginstal eksctl dengan Homebrew

1. (Prasyarat) Instal Homebrew.
2. Tambahkan ketukan AWS:

```
brew tap aws/tap
```

3. Instal eksctl

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

Sebelum menggunakan eksctl, selesaikan langkah-langkah konfigurasi ini:

1. Instal prasyarat:
   - Instal AWS CLI versi 2.x atau yang lebih baru.
   - Instal kubectl menggunakan Homebrew:

```
brew install kubernetes-cli
```

2. Konfigurasikan kredensil AWS di lingkungan Anda:

```
aws configure
```

3. Verifikasi konfigurasi AWS CLI:

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

# Langkah 2: Buat file konfigurasi cluster

Buat file konfigurasi cluster menggunakan langkah-langkah ini:

1. Buat file baru bernama`cluster.yaml`:

```
touch cluster.yaml
```

2. Tambahkan konfigurasi cluster dasar berikut:

```
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. Sesuaikan konfigurasi:
   - Perbarui `region` agar sesuai dengan wilayah AWS yang Anda inginkan.
   - Ubah `instanceType` berdasarkan persyaratan beban kerja Anda.
   - Sesuaikan`desiredCapacity`,`minSize`, dan `maxSize` sesuai dengan kebutuhan Anda.
4. Validasi file konfigurasi:

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

# Langkah 3: Buat cluster

Ikuti langkah-langkah berikut untuk membuat kluster EKS Anda:

1. Buat cluster menggunakan file konfigurasi:

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

2. Tunggu pembuatan cluster (ini biasanya memakan waktu 15-20 menit).
3. Verifikasi pembuatan cluster:

```
eksctl get cluster
```

4. Konfigurasikan kubectl untuk menggunakan cluster baru Anda:

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

5. Verifikasi konektivitas cluster:

```
kubectl get nodes
```

Cluster Anda sekarang siap digunakan.

## Opsional: Hapus Cluster

Ingatlah untuk menghapus klaster setelah selesai untuk menghindari tagihan yang tidak perlu:

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

> **Note**
>
> Pembuatan klaster dapat dikenakan biaya AWS. Pastikan untuk meninjau harga Amazon EKS sebelum membuat cluster.

# Langkah Berikutnya

- Konfigurasikan Kubectl untuk terhubung ke cluster
- Menerapkan aplikasi sampel

# Opsi instalasi untuk Eksctl

`eksctl`tersedia untuk diinstal dari rilis resmi seperti yang dijelaskan di bawah ini. Kami menyarankan Anda menginstal hanya `eksctl` dari GitHub rilis resmi. Anda dapat memilih untuk menggunakan penginstal pihak ketiga, tetapi harap diperhatikan bahwa AWS tidak memelihara atau mendukung metode penginstalan ini. Gunakan mereka atas kebijaksanaan Anda sendiri.

## Prasyarat

Anda harus memiliki kredenal AWS API yang dikonfigurasi. Apa yang berfungsi untuk AWS CLI atau alat lainnya (kops, Terraform, dll.) Sudah cukup. Anda dapat menggunakan variabel `~/.aws/credentials`file atau lingkungan. Untuk informasi selengkapnya, lihat Referensi AWS CLI.

Anda juga memerlukan AWS IAM Authenticator for Kubernetes command (`aws-iam-authenticator`baik `aws eks get-token` atau (tersedia dalam versi 1.16.156 atau lebih tinggi dari AWS CLI). `PATH`

Akun IAM yang digunakan untuk pembuatan cluster EKS harus memiliki tingkat akses minimal ini.

| Layanan AWS | Tingkat Akses |
|---|---|
| CloudFormation | Akses Penuh |
| EC2 | Lengkap: Menandai Limited: Daftar, Baca, Tulis |
| EC2 Auto Scaling | Terbatas: Daftar, Tulis |
| EKS | Akses Penuh |
| IAM | Terbatas: Daftar, Baca, Tulis, Manajemen Izin |
| Systems Manager | Terbatas: Daftar, Baca |

## Untuk Unix

Untuk mengunduh rilis terbaru, jalankan:

```
# 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
```

## Untuk Windows

Unduhan langsung (rilis terbaru):

- AMD64/x86_64
- ARMv6
- ARMv7
- ARM64

Pastikan untuk membuka zip arsip ke folder dalam PATH variabel.

Secara opsional, verifikasi checksum:

1. Unduh file checksum: latest
2. Gunakan Command Prompt untuk membandingkan `CertUtil` output secara manual dengan file checksum yang diunduh.

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

3. Menggunakan PowerShell untuk mengotomatiskan verifikasi menggunakan `-eq` operator untuk mendapatkan `False` hasil `True` atau:

   ```
   # 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]
```

## Menggunakan 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
```

`eksctl`Executable ditempatkan di`$HOME/bin`, yang berasal dari `$PATH` Git Bash.

# Homebrew

Anda dapat menggunakan Homebrew untuk menginstal perangkat lunak di macOS dan Linux.

AWS mempertahankan keran Homebrew termasuk eksctl.

Untuk informasi lebih lanjut tentang ketukan Homebrew, lihat proyek di Github dan rumus Homebrew untuk eksctl.

Untuk menginstal eksctl dengan Homebrew

1. (Prasyarat) Instal Homebrew
2. Tambahkan ketukan AWS

```
brew tap aws/tap
```

3. Instal eksctl

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

# Docker

Untuk setiap rilis dan RC, gambar kontainer didorong ke repositori ECR. `public.ecr.aws/eksctl/eksctl` Pelajari lebih lanjut tentang penggunaan di Galeri Publik ECR - eksctl. Misalnya,

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

# Penyelesaian Shell

## Bash

Untuk mengaktifkan penyelesaian bash, jalankan yang berikut ini, atau masukkan `~/.bashrc` atau`~/.profile`:

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

## Zsh

Untuk penyelesaian zsh, jalankan:

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

dan masukkan yang berikut ini`~/.zshrc`:

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

Perhatikan jika Anda tidak menjalankan distribusi seperti oh-my-zsh Anda mungkin harus mengaktifkan pelengkapan otomatis terlebih dahulu (dan memasukkannya `~/.zshrc` untuk membuatnya persisten):

```
autoload -U compinit
compinit
```

## Ikan

Perintah di bawah ini dapat digunakan untuk penyelesaian otomatis fish:

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

## Powershell

Perintah di bawah ini dapat dirujuk untuk mengaturnya. Harap dicatat bahwa jalurnya mungkin berbeda tergantung pada pengaturan sistem Anda.

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

# Pembaruan

> **Important**
>
> Jika Anda menginstal eksctl dengan langsung mengunduhnya (tidak menggunakan manajer paket), Anda perlu memperbaruinya secara manual.

# klaster

Bab ini mencakup pembuatan dan konfigurasi kluster EKS menggunakan eksctl. Ini juga termasuk add-on dan Mode Otomatis EKS.

## Topik:

- the section called “Entri Akses EKS”
  - Sederhanakan manajemen RBAC Kubernetes dengan mengganti aws-auth dengan entri akses EKS ConfigMap
  - Migrasikan pemetaan identitas IAM yang ada dari aws-auth untuk mengakses entri ConfigMap
  - Konfigurasikan mode otentikasi cluster dan kontrol izin admin pembuat klaster
- the section called “Pembaruan add-on default”
  - Jaga agar cluster tetap aman dengan memperbarui add-on EKS default pada cluster yang lebih lama
- the section called “Addons”
  - Otomatiskan tugas rutin untuk menginstal, memperbarui, dan menghapus add-on.
  - Pengaya Amazon EKS mencakup add-on AWS, add-on komunitas open source, dan add-on marketplace.
- the section called “Mode Otomatis EKS”
  - Kurangi overhead operasional dengan membiarkan AWS mengelola infrastruktur EKS Anda
  - Konfigurasikan kumpulan node kustom alih-alih kumpulan tujuan umum dan sistem default
  - Konversi kluster EKS yang ada untuk menggunakan Mode Otomatis
- the section called “CloudWatch penebangan”
  - Memecahkan masalah klaster dengan mengaktifkan log untuk komponen bidang kontrol EKS tertentu
  - Konfigurasikan periode retensi log untuk log klaster EKS
  - Ubah pengaturan logging cluster yang ada menggunakan perintah eksctl
- the section called “Upgrade cluster”
  - Menjaga keamanan dan stabilitas dengan memutakhirkan versi pesawat kontrol EKS dengan aman
  - Meluncurkan upgrade di seluruh nodegroup dengan mengganti grup lama dengan yang baru

  - Perbarui add-on klaster default
- the section called “Membuat dan mengelola cluster”
  - Mulai dengan cepat dengan cluster EKS dasar menggunakan nodegroup terkelola default
  - Buat cluster yang disesuaikan menggunakan file konfigurasi dengan konfigurasi tertentu
  - Menyebarkan cluster yang ada VPCs dengan jaringan pribadi dan kebijakan IAM khusus
- the section called “Konfigurasikan kubelet”
  - Cegah kelaparan sumber daya node dengan mengonfigurasi reservasi kubelet dan daemon sistem
  - Sesuaikan ambang penggusuran untuk ketersediaan memori dan sistem file
  - Mengaktifkan atau menonaktifkan gerbang fitur kubelet tertentu di seluruh grup node
- the section called “EKS Konektor”
  - Memusatkan pengelolaan penerapan Kubernetes hybrid melalui Konsol EKS
  - Konfigurasikan peran dan izin IAM untuk akses klaster eksternal
  - Hapus klaster eksternal dan bersihkan sumber daya AWS terkait
- the section called “EKS Cluster Sepenuhnya Pribadi”
  - Memenuhi persyaratan keamanan dengan kluster EKS yang sepenuhnya pribadi yang tidak memiliki akses internet keluar
  - Konfigurasikan akses pribadi ke layanan AWS melalui titik akhir VPC
  - Membuat dan mengelola nodegroup pribadi dengan pengaturan jaringan eksplisit
- the section called “Support Karpenter”
  - Mengotomatiskan penyediaan node
  - Buat konfigurasi penyedia Karpenter kustom
  - Siapkan Karpenter dengan penanganan interupsi instans spot
- the section called “Amazon EMR”
  - Buat pemetaan identitas IAM antara EMR dan kluster EKS
- the section called “Dukungan EKS Fargate”
  - Tentukan profil Fargate khusus untuk penjadwalan pod
  - Kelola profil Fargate melalui pembaruan pembuatan dan konfigurasi
- the section called “Cluster yang tidak dibuat eksctl”
  - Standarisasi manajemen cluster yang dibuat di luar eksctl
  - Gunakan perintah eksctl pada cluster non-eksctl yang ada

- the section called “Aktifkan Zonal Shift”
  - Tingkatkan ketersediaan aplikasi dengan mengaktifkan kemampuan failover zona cepat
  - Konfigurasikan pergeseran zona pada penerapan kluster EKS baru
  - Aktifkan fitur pergeseran zona pada kluster EKS yang ada

# Membuat dan mengelola cluster

Topik ini mencakup cara membuat dan menghapus kluster EKS menggunakan Eksctl. Anda dapat membuat cluster dengan perintah CLI, atau dengan membuat file YAMM konfigurasi cluster.

## Membuat cluster sederhana

Buat cluster sederhana dengan perintah berikut:

```
eksctl create cluster
```

Itu akan membuat cluster EKS di wilayah default Anda (seperti yang ditentukan oleh konfigurasi AWS CLI Anda) dengan satu nodegroup terkelola yang berisi dua node m5.large.

eksctl sekarang membuat nodegroup terkelola secara default ketika file konfigurasi tidak digunakan. Untuk membuat nodegroup yang dikelola sendiri, teruskan `--managed=false` ke atau. `eksctl create cluster` `eksctl create nodegroup`

### Pertimbangan-pertimbangan

- Saat membuat cluster di`us-east-1`, Anda mungkin menemukan file`UnsupportedAavailabilityZoneException`. Jika ini terjadi, salin zona yang disarankan dan berikan `--zones` bendera, misalnya:`eksctl create cluster --region=us-east-1 --zones=us-east-1a,us-east-1b,us-east-1d`. Masalah ini dapat terjadi di daerah lain tetapi kurang umum. Dalam kebanyakan kasus, Anda tidak perlu menggunakan `--zone` bendera.

## Buat cluster menggunakan file konfigurasi

Anda dapat membuat cluster menggunakan file konfigurasi alih-alih flag.

Pertama, buat `cluster.yaml` file:

```
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
```

Selanjutnya, jalankan perintah ini:

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

Ini akan membuat cluster seperti yang dijelaskan.

Jika Anda perlu menggunakan VPC yang ada, Anda dapat menggunakan file konfigurasi seperti ini:

```
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
```

Nama cluster atau nama nodegroup harus hanya berisi karakter alfanumerik (peka huruf besar/kecil) dan tanda hubung. Itu harus dimulai dengan karakter alfabet dan tidak dapat melebihi 128 karakter, atau Anda akan menerima kesalahan validasi. Untuk informasi selengkapnya, lihat Membuat tumpukan dari CloudFormation konsol di panduan CloudFormation pengguna AWS.

## Perbarui kubeconfig untuk cluster baru

Setelah cluster dibuat, konfigurasi kubernetes yang sesuai akan ditambahkan ke file kubeconfig Anda. Ini adalah, file yang telah Anda konfigurasikan dalam variabel lingkungan `KUBECONFIG` atau secara `~/.kube/config` default. Path ke file kubeconfig dapat diganti menggunakan flag. `--kubeconfig`

Bendera lain yang dapat mengubah cara penulisan file kubeconfig:

| bendera | jenis | menggunakan | nilai default |
|---|---|---|---|
| --kubeconfig | string | jalur untuk menulis kubeconfig (tidak kompatibel dengan --auto-kubeconfig) | $KUBECONFIG atau ~/.kube/config |
| --set-kubeconfig-context | bool | jika true maka konteks saat ini akan disetel di kubeconfig; jika | true |

| bendera | jenis | menggunakan | nilai default |
| --- | --- | --- | --- |
| | | konteks sudah disetel maka itu akan ditimpa | |
| --auto-kubeconfig | bool | simpan file kubeconfig dengan nama cluster | true |
| --write-kubeconfig | bool | alihkan penulisan kubeconfig | true |

## Hapus cluster

Untuk menghapus klaster ini, jalankan:

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

> **Warning**
>
> Gunakan `--wait` tanda dengan operasi penghapusan untuk memastikan kesalahan penghapusan dilaporkan dengan benar.

Tanpa `--wait` flag, eksctl hanya akan mengeluarkan operasi penghapusan ke CloudFormation tumpukan cluster dan tidak akan menunggu penghapusannya. Dalam beberapa kasus, sumber daya AWS yang menggunakan cluster atau VPC-nya dapat menyebabkan penghapusan klaster gagal. Jika penghapusan Anda gagal atau Anda lupa tanda tunggu, Anda mungkin harus pergi ke CloudFormation GUI dan menghapus tumpukan eks dari sana.

> **Warning**
>
> Kebijakan PDB dapat memblokir penghapusan node selama penghapusan klaster.

Saat menghapus klaster dengan nodegroup, kebijakan Pod Disruption Budget (PDB) dapat mencegah node berhasil dihapus. Misalnya, cluster yang `aws-ebs-csi-driver` diinstal biasanya memiliki dua pod dengan kebijakan PDB yang memungkinkan hanya satu pod yang tidak tersedia pada satu waktu, membuat pod lainnya tidak dapat dievictable selama penghapusan. Agar berhasil

menghapus klaster dalam skenario ini, gunakan `disable-nodegroup-eviction` tanda untuk melewati pemeriksaan kebijakan PDB:

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

Lihat `examples/`direktori di GitHub repo eksctl untuk file konfigurasi sampel lainnya.

### Lari Kering

Fitur dry-run memungkinkan menghasilkan ClusterConfig file yang melewatkan pembuatan cluster dan mengeluarkan ClusterConfig file yang mewakili opsi CLI yang disediakan dan berisi nilai default yang ditetapkan oleh eksctl.

Info lebih lanjut dapat ditemukan di halaman Dry Run.

## Mode Otomatis EKS

eksctl mendukung EKS Auto Mode, fitur yang memperluas pengelolaan AWS klaster Kubernetes di luar klaster itu sendiri, untuk memungkinkan AWS juga menyiapkan dan mengelola infrastruktur yang memungkinkan kelancaran pengoperasian beban kerja Anda. Ini memungkinkan Anda untuk mendelegasikan keputusan infrastruktur utama dan memanfaatkan keahlian AWS untuk day-to-day operasi. Infrastruktur klaster yang dikelola AWS mencakup banyak kemampuan Kubernetes sebagai komponen inti, sebagai lawan dari add-on, seperti compute autoscaling, pod dan service networking, application load balancing, cluster DNS, block storage, dan dukungan GPU.

### Membuat cluster EKS dengan Mode Otomatis diaktifkan

`eksctl`telah menambahkan `autoModeConfig` bidang baru untuk mengaktifkan dan mengkonfigurasi Mode Otomatis. Bentuk `autoModeConfig` lapangan adalah

```
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
```

Jika `autoModeConfig.enabled` benar, eksctl membuat cluster EKS dengan meneruskan`computeConfig.enabled: true`,`kubernetesNetworkConfig.elasticLoadBalancing.enabled: true`, dan `storageConfig.blockStorage.enabled: true` ke EKS API, memungkinkan pengelolaan komponen bidang data seperti komputasi, penyimpanan, dan jaringan.

Untuk membuat kluster EKS dengan Mode Otomatis diaktifkan, atur`autoModeConfig.enabled: true`, seperti pada

```
# 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 membuat peran node untuk digunakan untuk node yang diluncurkan oleh Mode Otomatis. eksctl juga membuat kumpulan dan node. `general-purpose` `system` Untuk menonaktifkan pembuatan kumpulan node default, misalnya, untuk mengonfigurasi kumpulan node Anda sendiri yang menggunakan kumpulan subnet yang berbeda, setel`nodePools: []`, seperti pada

```
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.
```

## Memperbarui cluster EKS untuk menggunakan Mode Otomatis

Untuk memperbarui cluster EKS yang ada untuk menggunakan Mode Otomatis, jalankan

```
# 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**
>
> Jika cluster dibuat oleh eksctl, dan menggunakan subnet publik sebagai subnet cluster, Mode Otomatis akan meluncurkan node di subnet publik. Untuk menggunakan subnet pribadi untuk node pekerja yang diluncurkan oleh Mode Otomatis, perbarui cluster untuk menggunakan subnet pribadi.

## Menonaktifkan Mode Otomatis

Untuk menonaktifkan Mode Otomatis, atur `autoModeConfig.enabled: false` dan jalankan

```
# 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
```

## Informasi lebih lanjut

- Mode Otomatis EKS

# Entri Akses EKS

Anda dapat menggunakan eksctl untuk mengelola Entri Akses EKS. Gunakan entri akses untuk memberikan izin Kubernetes ke AWS IAM Identities. Misalnya, Anda mungkin memberikan izin peran pengembang untuk membaca sumber daya Kubernetes di klaster.

Topik ini mencakup cara menggunakan eksctl untuk mengelola entri akses. Untuk informasi umum tentang entri akses, lihat Berikan akses kepada pengguna IAM ke Kubernetes dengan entri akses EKS.

Anda dapat melampirkan kebijakan akses Kubernetes yang ditentukan oleh AWS, atau mengaitkan Identitas IAM dengan grup Kubernetes.

Untuk informasi selengkapnya tentang kebijakan yang telah ditentukan sebelumnya, lihat Mengaitkan kebijakan akses dengan entri akses.

Jika Anda perlu mendefinisikan kebijakan Kubernetes kustom, kaitkan Identitas IAM dengan grup Kubernetes, dan berikan izin ke grup tersebut.

## Mode otentikasi cluster

Anda hanya dapat menggunakan entri akses jika mode otentikasi klaster mengizinkannya.

Untuk informasi selengkapnya, lihat Mengatur Mode Otentikasi Cluster

### Mengatur mode otentikasi dengan file YAMG

`eksctl`telah menambahkan `accessConfig.authenticationMode` bidang baru di bawah ClusterConfig, yang dapat diatur ke salah satu dari tiga nilai berikut:

- `CONFIG_MAP`- default di EKS API - hanya `aws-auth` ConfigMap akan digunakan
- `API`- hanya akses entri API yang akan digunakan
- `API_AND_CONFIG_MAP`- default di `eksctl` - keduanya `aws-auth` ConfigMap dan akses entri API dapat digunakan

Setel mode otentikasi di ClusterConfig YAMB:

```
accessConfig:
```

```
  authenticationMode: <>
```

### Perbarui mode otentikasi dengan perintah

Jika Anda ingin menggunakan entri akses pada cluster yang sudah ada, non-eksctl dibuat, di mana `CONFIG_MAP` opsi digunakan, pengguna harus terlebih dahulu mengatur ke. `authenticationMode` `API_AND_CONFIG_MAP` Untuk itu, `eksctl` telah memperkenalkan perintah baru untuk memperbarui mode otentikasi cluster, yang berfungsi baik dengan flag CLI mis.

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

## Akses Sumber Daya Entri

Entri akses memiliki tipe, seperti `STANDARD` atau`EC2_LINUX`. Jenisnya tergantung pada bagaimana Anda menggunakan entri akses.

- `standard`Jenisnya adalah untuk memberikan izin Kubernetes kepada Pengguna IAM dan Peran IAM.
  - Misalnya, Anda dapat melihat sumber daya Kubernetes di konsol AWS dengan melampirkan kebijakan akses ke Peran atau Pengguna yang Anda gunakan untuk mengakses konsol.
- `EC2_WINDOWS`Tipe `EC2_LINUX` dan adalah untuk memberikan izin Kubernetes ke instans EC2. Instance menggunakan izin ini untuk bergabung dengan cluster.

Untuk informasi selengkapnya tentang jenis entri akses, lihat Membuat entri akses

### Entitas IAM

Anda dapat menggunakan entri akses untuk memberikan izin Kubernetes ke Identitas IAM seperti Pengguna IAM dan Peran IAM.

Gunakan `accessConfig.accessEntries` bidang untuk mengaitkan ARN sumber daya IAM dengan API EKS Entri Akses. Contoh:

```
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
```

Selain mengaitkan kebijakan EKS, seseorang juga dapat menentukan grup Kubernetes yang menjadi milik entitas IAM, sehingga memberikan izin melalui RBAC.

## Grup nodegroup dan Fargate yang dikelola

Integrasi dengan entri akses untuk sumber daya ini akan dicapai di belakang layar, oleh EKS API. Grup node terkelola yang baru dibuat dan pod Fargate akan membuat entri akses API, daripada menggunakan sumber daya RBAC yang dimuat sebelumnya. Grup node dan pod Fargate yang ada tidak akan diubah, dan terus mengandalkan entri di peta konfigurasi aws-auth.

## Nodegroup yang dikelola sendiri

Setiap entri akses memiliki tipe. Untuk mengotorisasi nodegroup yang dikelola sendiri, `eksctl` akan membuat entri akses unik untuk setiap nodegroup dengan ARN utama disetel ke ARN peran node dan mengetik disetel ke salah satu atau tergantung pada nodegroup AMIFamily. `EC2_LINUX` `EC2_WINDOWS`

Saat membuat entri akses sendiri, Anda juga dapat menentukan `EC2_LINUX` (untuk peran IAM yang digunakan dengan node yang dikelola sendiri oleh Linux atau Bottlerocket), (untuk peran IAM yang

digunakan dengan node yang dikelola sendiri Windows), `EC2_WINDOWS` (untuk peran IAM yang digunakan dengan AWS Fargate (Fargate)), atau sebagai tipe. `FARGATE_LINUX` `STANDARD` Jika Anda tidak menentukan tipe, tipe default diatur ke`STANDARD`.

> **Note**
>
> Saat menghapus nodegroup yang dibuat dengan yang sudah ada sebelumnya`instanceRoleARN`, adalah tanggung jawab pengguna untuk menghapus entri akses yang sesuai ketika tidak ada lagi nodegroup yang terkait dengannya. Ini karena eksctl tidak berusaha mencari tahu apakah entri akses masih digunakan oleh nodegroup yang dikelola sendiri non-eksctl karena ini adalah proses yang rumit.

## Buat entri akses

Ini dapat dilakukan dengan dua cara berbeda, baik selama pembuatan cluster, menentukan entri akses yang diinginkan sebagai bagian dari file konfigurasi dan berjalan:

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

ATAU posting pembuatan cluster, dengan menjalankan:

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

Untuk contoh file konfigurasi untuk membuat entri akses, lihat 40-access-entries.yaml di repo eksctl. GitHub

## Dapatkan entri akses

Pengguna dapat mengambil semua entri akses yang terkait dengan cluster tertentu dengan menjalankan salah satu dari berikut ini:

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

ATAU

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

Atau, untuk mengambil hanya entri akses yang sesuai dengan entitas IAM tertentu, seseorang harus menggunakan `--principal-arn` bendera tersebut. misalnya

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

## Hapus entri akses

Untuk menghapus entri akses tunggal pada satu waktu, gunakan:

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

Untuk menghapus beberapa entri akses, gunakan `--config-file` bendera dan tentukan semua `principalARN's` yang sesuai dengan entri akses, di bawah bidang tingkat atas`accessEntry`, mis.

```
...
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
```

## Bermigrasi dari aws-auth ConfigMap

Pengguna dapat memigrasikan identitas IAM mereka yang ada dari `aws-auth` configmap untuk mengakses entri dengan menjalankan yang berikut:

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

Ketika `--target-authentication-mode` bendera disetel ke`API`, mode otentikasi dialihkan ke `API` mode (dilewati jika sudah dalam `API` mode), pemetaan identitas IAM akan dimigrasikan untuk mengakses entri, dan configmap dihapus dari cluster. `aws-auth`

Ketika `--target-authentication-mode` bendera disetel ke`API_AND_CONFIG_MAP`, mode otentikasi dialihkan ke mode (dilewati jika sudah dalam `API_AND_CONFIG_MAP`

API_AND_CONFIG_MAP mode), pemetaan identitas IAM akan dimigrasikan untuk mengakses entri, tetapi configmap dipertahankan. aws-auth

> **Note**
>
> Ketika --target-authentication-mode flag diatur keAPI, perintah ini tidak akan memperbarui mode otentikasi ke API mode jika aws-auth configmap memiliki salah satu kendala di bawah ini.

- Ada pemetaan identitas tingkat Akun.
- Satu atau lebih Roles/Users dipetakan ke grup kubernetes yang dimulai dengan awalan system: (kecuali untuk grup khusus EKS yaitusystem:masters,, dll). system:bootstrappers system:nodes
- Satu atau lebih pemetaan identitas IAM adalah untuk [Service Linked Role] (link: IAM/latest/UserGuide/using - service-linked-roles .html).

## Nonaktifkan izin admin pembuat klaster

eksctltelah menambahkan bidang baru accessConfig.bootstrapClusterCreatorAdminPermissions: boolean yang, ketika disetel ke false, menonaktifkan pemberian izin cluster-admin ke identitas IAM yang membuat klaster. yaitu

tambahkan opsi ke file konfigurasi:

```
accessConfig:
  bootstrapClusterCreatorAdminPermissions: false
```

dan jalankan:

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

# Cluster yang tidak dibuat eksctl

Anda dapat menjalankan eksctl perintah terhadap cluster yang tidak dibuat oleheksctl.

> **Note**
>
> Eksctl hanya dapat mendukung cluster yang tidak dimiliki dengan nama yang kompatibel dengan AWS. CloudFormation Nama cluster apa pun yang tidak cocok dengan ini akan gagal dalam pemeriksaan validasi CloudFormation API.

## Perintah yang Didukung

Perintah berikut dapat digunakan terhadap cluster yang dibuat dengan cara apa pun selain`eksctl`. Opsi perintah, bendera, dan file konfigurasi dapat digunakan dengan cara yang persis sama.

Jika kami melewatkan beberapa fungsi, beri tahu kami.

- ✓ Buat:
  - ✓ `eksctl create nodegroup` (lihat catatan di bawah)
  - ✓ `eksctl create fargateprofile`
  - ✓ `eksctl create iamserviceaccount`
  - ✓ `eksctl create iamidentitymapping`
- ✓ Dapatkan:
  - ✓ `eksctl get clusters/cluster`
  - ✓ `eksctl get fargateprofile`
  - ✓ `eksctl get nodegroup`
  - ✓ `eksctl get labels`
- ✓ Hapus:
  - ✓ `eksctl delete cluster`
  - ✓ `eksctl delete nodegroup`
  - ✓ `eksctl delete fargateprofile`
  - ✓ `eksctl delete iamserviceaccount`
  - ✓ `eksctl delete iamidentitymapping`
- ✓ Tingkatkan:
  - ✓ `eksctl upgrade cluster`
  - ✓ `eksctl upgrade nodegroup`
- ✓ Set/Hapus:

- ✓ `eksctl set labels`
- ✓ `eksctl unset labels`

✓ Skala:

- ✓ `eksctl scale nodegroup`

✓ Tiriskan:

- ✓ `eksctl drain nodegroup`

✓ Aktifkan:

- ✓ `eksctl enable profile`
- ✓ `eksctl enable repo`

✓ Utils:

- ✓ `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`

## Membuat nodegroup

`eksctl create nodegroup`adalah satu-satunya perintah yang membutuhkan input khusus dari pengguna.

Karena pengguna dapat membuat cluster mereka dengan konfigurasi jaringan apa pun yang mereka sukai, untuk saat ini, tidak `eksctl` akan mencoba untuk mengambil atau menebak nilai-nilai ini. Ini mungkin berubah di masa depan saat kita mempelajari lebih lanjut tentang bagaimana orang menggunakan perintah ini pada cluster yang tidak dibuat eksctl.

Ini berarti bahwa untuk membuat nodegroup atau nodegroup terkelola pada cluster yang tidak dibuat oleh`eksctl`, file konfigurasi yang berisi detail VPC harus disediakan. Minimal:

```
---
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"

...
```

Untuk informasi selengkapnya tentang opsi konfigurasi VPC, lihat Jaringan.

## Mendaftarkan cluster non-EKS dengan Konektor EKS

Anda dapat menggunakan Konektor EKS untuk melihat cluster di luar AWS di Konsol EKS. Proses ini membutuhkan registrasi cluster dengan EKS dan menjalankan agen EKS Connector pada cluster Kubernetes eksternal.

`eksctl`menyederhanakan pendaftaran klaster non-EKS dengan membuat sumber daya AWS yang diperlukan dan menghasilkan manifes Kubernetes agar Konektor EKS diterapkan ke cluster eksternal.

# Daftar Cluster

Untuk mendaftarkan atau menghubungkan klaster Kubernetes non-EKS, jalankan

```
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
```

Perintah ini akan mendaftarkan cluster dan menulis tiga file yang berisi manifes Kubernetes untuk EKS Connector yang harus diterapkan ke cluster eksternal sebelum pendaftaran berakhir.

> **Note**
>
> `eks-connector-clusterrole.yaml`dan `eks-connector-console-dashboard-full-access-clusterrole.yaml` memberikan `get` dan `list` izin untuk sumber daya Kubernetes di semua namespace ke identitas IAM yang memanggil dan harus diedit sesuai jika diperlukan sebelum menerapkannya ke cluster. Untuk mengonfigurasi akses yang lebih terbatas, lihat Memberikan akses kepada pengguna untuk melihat klaster.

Untuk menyediakan peran IAM yang ada untuk digunakan untuk Konektor EKS, berikan melalui `--role-arn` seperti pada:

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

Jika cluster sudah ada, eksctl akan mengembalikan kesalahan.

## Cluster deregister

Untuk membatalkan pendaftaran atau memutuskan kluster terdaftar, jalankan

```
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
```

Perintah ini akan membatalkan pendaftaran cluster eksternal dan menghapus sumber daya AWS yang terkait, tetapi Anda diharuskan untuk menghapus sumber daya konektor EKS Kubernetes dari cluster.

## Informasi lebih lanjut

- EKS Konektor

# Menyesuaikan konfigurasi kubelet

Sumber daya sistem dapat dipesan melalui konfigurasi kubelet. Ini direkomendasikan, karena dalam kasus kelaparan sumber daya kubelet mungkin tidak dapat mengusir pod dan akhirnya membuat node menjadi. `NotReady` Untuk melakukan ini, file konfigurasi dapat menyertakan `kubeletExtraConfig` bidang yang menerima formulir gratis yaml yang akan disematkan ke dalam file. `kubelet.yaml`

Beberapa bidang di `kubelet.yaml` diatur oleh eksctl dan oleh karena itu tidak dapat ditimpa, seperti,,,`address`, `clusterDomain` atau. `authentication` `authorization` `serverTLSBootstrap`

Contoh file konfigurasi berikut membuat nodegroup yang menyimpan `300m` vCPU, `300Mi` memori dan `1Gi` penyimpanan ephemeral untuk kubelet; `300m` vCPU`300Mi`, memori dan penyimpanan sementara untuk daemon sistem OS; `1Gi` dan menendang penggusuran pod ketika ada kurang dari memori yang tersedia atau kurang dari 10% dari sistem file root. `200Mi`

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

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

nodeGroups:
  - name: ng-1
    instanceType: m5a.xlarge
    desiredCapacity: 1
    kubeletExtraConfig:
        kubeReserved:
            cpu: "300m"
            memory: "300Mi"
            ephemeral-storage: "1Gi"
        kubeReservedCgroup: "/kube-reserved"
        systemReserved:
            cpu: "300m"
            memory: "300Mi"
            ephemeral-storage: "1Gi"
        evictionHard:
            memory.available:  "200Mi"
            nodefs.available: "10%"
        featureGates:
            RotateKubeletServerCertificate: true # has to be enabled, otherwise it will
 be disabled
```

Dalam contoh ini, diberikan contoh tipe `m5a.xlarge` yang memiliki 4 v CPUs dan 16GiB memori, `Allocatable` jumlah CPUs akan menjadi 3,4 dan 15,4 GiB memori. Penting untuk diketahui bahwa nilai yang ditentukan dalam file konfigurasi untuk bidang di `kubeletExtraConfig` akan sepenuhnya menimpa nilai default yang ditentukan oleh eksctl. Namun, menghilangkan satu atau lebih `kubeReserved` parameter akan menyebabkan parameter yang hilang menjadi default ke nilai waras berdasarkan jenis instance aws yang digunakan.

## **`kubeReserved`**perhitungan

Meskipun umumnya disarankan untuk mengkonfigurasi instance campuran NodeGroup untuk menggunakan instance dengan konfigurasi CPU dan RAM yang sama; itu bukan persyaratan yang ketat. Oleh karena itu `kubeReserved` perhitungan menggunakan contoh terkecil di `InstanceDistribution.InstanceTypes` lapangan. Dengan cara ini NodeGroups dengan tipe instance yang berbeda tidak akan menyimpan terlalu banyak sumber daya pada instance terkecil. Namun, ini dapat menyebabkan reservasi yang terlalu kecil untuk jenis instans terbesar.

> **Warning**
>
> Secara default `eksctl` set`featureGates.RotateKubeletServerCertificate=true`, tetapi ketika kustom `featureGates` disediakan, itu akan tidak disetel. Anda harus selalu menyertakan`featureGates.RotateKubeletServerCertificate=true`, kecuali Anda harus menonaktifkannya.

# CloudWatch penebangan

Topik ini menjelaskan cara mengonfigurasi CloudWatch pencatatan Amazon untuk komponen bidang kontrol kluster EKS Anda. CloudWatch logging memberikan visibilitas ke dalam operasi bidang kontrol klaster Anda, yang penting untuk memecahkan masalah, mengaudit aktivitas klaster, dan memantau kesehatan komponen Kubernetes Anda.

## Mengaktifkan pencatatan CloudWatch

CloudWatch logging untuk bidang kontrol EKS tidak diaktifkan secara default karena konsumsi data dan biaya penyimpanan.

Untuk mengaktifkan logging bidang kontrol saat cluster dibuat, Anda perlu menentukan **`cloudWatch.clusterLogging.enableTypes`**pengaturan di `ClusterConfig` (lihat di bawah untuk contoh).

Jadi jika Anda memiliki file konfigurasi dengan **`cloudWatch.clusterLogging.enableTypes`**pengaturan yang benar, Anda dapat membuat cluster dengan`eksctl create cluster --config-file=<path>`.

Jika Anda sudah membuat cluster, Anda dapat menggunakannya`eksctl utils update-cluster-logging`.

> **Note**
>
> perintah ini berjalan dalam mode rencana secara default, Anda harus menentukan `--approve` bendera untuk menerapkan perubahan ke cluster Anda.

Jika Anda menggunakan file konfigurasi, jalankan:

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

Atau, Anda dapat menggunakan bendera CLI.

Untuk mengaktifkan semua jenis log, jalankan:

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

Untuk mengaktifkan `audit` log, jalankan:

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

Untuk mengaktifkan semua kecuali `controllerManager` log, jalankan:

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

Jika jenis `api` dan `scheduler` log sudah diaktifkan, untuk menonaktifkan `scheduler` dan mengaktifkan `controllerManager` pada saat yang sama, jalankan:

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

Ini akan meninggalkan `api` dan `controllerManager` sebagai satu-satunya jenis log diaktifkan.

Untuk menonaktifkan semua jenis log, jalankan:

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

## Contoh **`ClusterConfig`**

Dalam kluster EKS, `enableTypes` bidang di bawah `clusterLogging` dapat mengambil daftar nilai yang mungkin untuk mengaktifkan berbagai jenis log untuk komponen bidang kontrol.

Berikut ini adalah nilai yang mungkin:

- `api`: Mengaktifkan log server API Kubernetes.

- `audit`: Mengaktifkan log audit Kubernetes.
- `authenticator`: Mengaktifkan log authenticator.
- `controllerManager`: Mengaktifkan log manajer pengontrol Kubernetes.
- `scheduler`: Mengaktifkan log penjadwal Kubernetes.

Untuk mempelajari lebih lanjut, lihat dokumentasi EKS.

### Nonaktifkan semua log

Untuk menonaktifkan semua jenis, gunakan `[]` atau hapus `cloudWatch` bagian sepenuhnya.

### Aktifkan semua log

Anda dapat mengaktifkan semua jenis dengan `"*"` atau`"all"`. Contoh:

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

### Aktifkan satu atau lebih log

Anda dapat mengaktifkan subset tipe dengan mencantumkan jenis yang ingin Anda aktifkan. Contoh:

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

### Periode retensi log

Secara default, log disimpan di CloudWatch Log, tanpa batas waktu. Anda dapat menentukan jumlah hari di mana log pesawat kontrol harus disimpan di CloudWatch Log. Contoh berikut mempertahankan log selama 7 hari:

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

### Contoh lengkap

```
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
```

## EKS Cluster Sepenuhnya Pribadi

eksctl mendukung pembuatan cluster yang sepenuhnya pribadi yang tidak memiliki akses internet keluar dan hanya memiliki subnet pribadi. Titik akhir VPC digunakan untuk mengaktifkan akses pribadi ke layanan AWS.

Panduan ini menjelaskan cara membuat cluster pribadi tanpa akses internet keluar.

### Membuat cluster yang sepenuhnya pribadi

Satu-satunya bidang yang diperlukan untuk membuat klaster yang sepenuhnya pribadi adalah: `privateCluster.enabled`

```
privateCluster:
  enabled: true
```

Pasca pembuatan cluster, perintah eksctl yang memerlukan akses ke server API Kubernetes harus dijalankan dari dalam VPC cluster, VPC peered atau menggunakan beberapa cara lain seperti AWS Direct Connect. perintah eksctl yang memerlukan akses ke EKS tidak APIs akan berfungsi jika dijalankan dari dalam VPC cluster. Untuk memperbaikinya, buat endpoint antarmuka untuk Amazon EKS untuk mengakses pengelolaan Amazon Elastic Kubernetes Service (Amazon APIs EKS) secara

pribadi dari Amazon Virtual Private Cloud (VPC) Anda. Dalam rilis future, eksctl akan menambahkan dukungan untuk membuat endpoint ini sehingga tidak perlu dibuat secara manual. Perintah yang memerlukan akses ke URL penyedia OpenID Connect harus dijalankan dari luar VPC klaster Anda setelah Anda mengaktifkan AWS untuk PrivateLink Amazon EKS.

Membuat nodegroup terkelola akan terus berfungsi, dan membuat nodegroup yang dikelola sendiri akan berfungsi karena memerlukan akses ke server API melalui titik akhir antarmuka EKS jika perintah dijalankan dari dalam VPC cluster, VPC peered atau menggunakan beberapa cara lain seperti AWS Direct Connect.

> **Note**
>
> Titik akhir VPC dikenakan biaya per jam dan berdasarkan penggunaan. Detail lebih lanjut tentang harga dapat ditemukan di PrivateLink harga AWS

> **Warning**
>
> Cluster yang sepenuhnya pribadi tidak didukung di. `eu-south-1`

## Mengkonfigurasi akses pribadi ke layanan AWS tambahan

Untuk mengaktifkan node pekerja mengakses layanan AWS secara pribadi, eksctl membuat titik akhir VPC untuk layanan berikut:

- Titik akhir antarmuka untuk ECR (keduanya `ecr.api` dan`ecr.dkr`) untuk menarik gambar kontainer (plugin AWS CNI dll)
- Titik akhir gateway untuk S3 untuk menarik lapisan gambar yang sebenarnya
- Titik akhir antarmuka untuk EC2 yang diperlukan oleh integrasi `aws-cloud-provider`
- Titik akhir antarmuka untuk STS untuk mendukung Peran Fargate dan IAM untuk Akun Layanan (IRSA)
- Titik akhir antarmuka untuk CloudWatch logging (`logs`) jika CloudWatch logging diaktifkan

Titik akhir VPC ini penting untuk cluster pribadi fungsional, dan dengan demikian, eksctl tidak mendukung konfigurasi atau menonaktifkannya. Namun, klaster mungkin memerlukan akses pribadi ke layanan AWS lainnya (misalnya, Penskalaan Otomatis yang diperlukan oleh Cluster Autoscaler).

Layanan ini dapat ditentukan dalam`privateCluster.additionalEndpointServices`, yang menginstruksikan eksctl untuk membuat titik akhir VPC untuk masing-masingnya.

Misalnya, untuk mengizinkan akses pribadi ke Penskalaan Otomatis dan pencatatan: CloudWatch

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

Titik akhir yang didukung `additionalEndpointServices` adalah`autoscaling`, `cloudformation` dan`logs`.

### Melewatkan kreasi titik akhir

Jika VPC telah dibuat dengan titik akhir AWS yang diperlukan yang disiapkan dan ditautkan ke subnet yang dijelaskan dalam dokumentasi EKS, `eksctl` dapat melewati pembuatannya dengan memberikan opsi seperti ini: `skipEndpointCreation`

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

Pengaturan ini tidak dapat digunakan bersama dengan`additionalEndpointServices`. Ini akan melewati semua pembuatan titik akhir. Selain itu, pengaturan ini hanya disarankan jika topologi <# subnet endpoint diatur dengan benar. Jika id subnet benar, `vpce` perutean diatur dengan alamat awalan, semua titik akhir EKS yang diperlukan dibuat dan ditautkan ke VPC yang disediakan. `eksctl`tidak akan mengubah sumber daya ini.

## Nodegroup

Hanya nodegroup pribadi (baik yang dikelola maupun dikelola sendiri) yang didukung dalam klaster yang sepenuhnya pribadi karena VPC klaster dibuat tanpa subnet publik apa pun. `privateNetworking`Bidang (`nodeGroup[]`**`.privateNetworking`** dan`managedNodeGroup[`) harus ditetapkan secara eksplisit. Merupakan kesalahan jika `privateNetworking` tidak disetel di cluster yang sepenuhnya pribadi.

```
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
```

## Akses Titik Akhir Cluster

Cluster yang sepenuhnya pribadi tidak mendukung modifikasi `clusterEndpointAccess` selama pembuatan klaster. Ini adalah kesalahan untuk mengatur salah satu `clusterEndpoints.publicAccess` atau`clusterEndpoints.privateAccess`, karena cluster yang sepenuhnya pribadi hanya dapat memiliki akses pribadi, dan memungkinkan modifikasi bidang ini dapat merusak cluster.

## VPC dan subnet yang disediakan pengguna

eksctl mendukung pembuatan cluster yang sepenuhnya pribadi menggunakan VPC dan subnet yang sudah ada sebelumnya. Hanya subnet pribadi yang dapat ditentukan dan itu adalah kesalahan untuk menentukan subnet di bawah. `vpc.subnets.public`

eksctl membuat titik akhir VPC di VPC yang disediakan dan memodifikasi tabel rute untuk subnet yang disediakan. Setiap subnet harus memiliki tabel rute eksplisit yang terkait dengannya karena eksctl tidak memodifikasi tabel rute utama.

```
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
```

## Mengelola klaster yang sepenuhnya pribadi

Agar semua perintah berfungsi setelah pembuatan cluster, eksctl akan memerlukan akses pribadi ke titik akhir server EKS API, dan akses internet keluar (untuk). `EKS:DescribeCluster` Perintah yang tidak memerlukan akses ke server API akan didukung jika eksctl memiliki akses internet keluar.

## Hapus paksa klaster yang sepenuhnya pribadi

Kesalahan kemungkinan terjadi saat menghapus cluster yang sepenuhnya pribadi melalui eksctl karena eksctl tidak secara otomatis memiliki akses ke semua sumber daya cluster. `--forceada`

untuk menyelesaikan ini: itu akan memaksa menghapus cluster dan melanjutkan ketika kesalahan terjadi.

## Batasan

Keterbatasan implementasi saat ini adalah bahwa eksctl awalnya membuat cluster dengan akses titik akhir publik dan pribadi diaktifkan, dan menonaktifkan akses titik akhir publik setelah semua operasi selesai. Ini diperlukan karena eksctl membutuhkan akses ke server API Kubernetes untuk memungkinkan node yang dikelola sendiri untuk bergabung dengan cluster dan untuk mendukung dan Fargate. GitOps Setelah operasi ini selesai, eksctl mengalihkan akses titik akhir cluster ke private-only. Pembaruan tambahan ini berarti bahwa pembuatan klaster yang sepenuhnya pribadi akan memakan waktu lebih lama daripada klaster standar. Di masa depan, eksctl dapat beralih ke fungsi Lambda berkemampuan VPC untuk melakukan operasi API ini.

## Akses keluar melalui server proxy HTTP

eksctl dapat berbicara dengan AWS APIs melalui server proxy HTTP (S) yang dikonfigurasi, namun Anda harus memastikan bahwa Anda mengatur daftar pengecualian proxy dengan benar.

Umumnya, Anda perlu memastikan bahwa permintaan untuk titik akhir VPC untuk klaster Anda tidak dirutekan melalui proxy Anda dengan menyetel variabel lingkungan yang sesuai `no_proxy` termasuk nilainya. `.eks.amazonaws.com`

Jika server proxy Anda melakukan “intersepsi SSL” dan Anda menggunakan Peran IAM untuk Akun Layanan (IRSA), Anda perlu memastikan bahwa Anda secara eksplisit melewati SSL untuk domain tersebut. Man-in-the-Middle `oidc.<region>.amazonaws.com` Kegagalan untuk melakukannya akan mengakibatkan eksctl mendapatkan cap jempol sertifikat root yang salah untuk penyedia OIDC, dan plugin AWS VPC CNI akan gagal memulai karena tidak dapat memperoleh kredensyal IAM, membuat cluster Anda tidak beroperasi.

## Informasi lebih lanjut

- Kluster Pribadi EKS

# Addons

Topik ini menjelaskan cara mengelola Add-On Amazon EKS untuk kluster Amazon EKS Anda menggunakan eksctl. EKS Add-Ons adalah fitur yang memungkinkan Anda mengaktifkan dan

mengelola perangkat lunak operasional Kubernetes melalui EKS API, menyederhanakan proses menginstal, mengonfigurasi, dan memperbarui add-on klaster.

> **Warning**
>
> eksctl sekarang menginstal addons default (vpc-cni, coredns, kube-proxy) sebagai addons EKS alih-alih addon yang dikelola sendiri. Ini berarti Anda harus menggunakan `eksctl update addon` alih-alih `eksctl utils update-*` perintah untuk cluster yang dibuat dengan eksctl v0.184.0 dan di atasnya.

Anda dapat membuat cluster tanpa addons jaringan default ketika Anda ingin menggunakan plugin CNI alternatif seperti Cilium dan Calico.

Eks Add-on sekarang mendukung penerimaan izin IAM melalui Asosiasi Identitas Pod EKS, memungkinkan mereka untuk terhubung dengan layanan AWS di luar cluster

## Membuat addons

Eksctl memberikan lebih banyak fleksibilitas untuk mengelola addons cluster:

Di file konfigurasi, Anda dapat menentukan addon yang Anda inginkan dan (jika diperlukan) peran atau kebijakan yang akan dilampirkan padanya:

```
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: '*'
```

Anda dapat menentukan paling banyak salah satu`attachPolicy`, `attachPolicyARN`s dan`serviceAccountRoleARN`.

Jika tidak ada yang ditentukan, addon akan dibuat dengan peran yang memiliki semua kebijakan yang direkomendasikan terlampir.

> **Note**
>
> Untuk melampirkan kebijakan ke addons, klaster Anda harus telah `OIDC` diaktifkan. Jika tidak diaktifkan, kami mengabaikan kebijakan apa pun yang dilampirkan.

Anda kemudian dapat membuat addon ini selama proses pembuatan cluster:

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

Atau buat addons secara eksplisit setelah pembuatan cluster menggunakan file konfigurasi atau flag CLI:

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

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

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

> **Tip**
>
> Gunakan `--namespace-config` bendera untuk menyebarkan addons ke namespace khusus, bukan namespace default.

Selama pembuatan addon, jika versi addon yang dikelola sendiri sudah ada di cluster, Anda dapat memilih bagaimana potensi `configMap` konflik akan diselesaikan dengan menyetel `resolveConflicts` opsi melalui file konfigurasi, mis.

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

Untuk addon create, `resolveConflicts` bidang ini mendukung tiga nilai yang berbeda:

- none- EKS tidak mengubah nilainya. Create mungkin gagal.
- `overwrite`- EKS menimpa perubahan konfigurasi apa pun kembali ke nilai default EKS.
- `preserve`- EKS tidak mengubah nilainya. Create mungkin gagal. (Mirip dengannone, tetapi berbeda dari `preservedalam memperbarui addons`).

## Daftar addons yang diaktifkan

Anda dapat melihat addons apa yang diaktifkan di cluster Anda dengan menjalankan:

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

atau

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

## Mengatur versi addon

Mengatur versi addon adalah opsional. Jika `version` bidang dibiarkan kosong `eksctl` akan menyelesaikan versi default untuk addon. Informasi lebih lanjut tentang versi mana yang merupakan versi default untuk addon tertentu dapat ditemukan di dokumentasi AWS tentang EKS. Perhatikan bahwa versi default mungkin belum tentu versi terbaru yang tersedia.

Versi addon dapat diatur ke`latest`. Atau, versi dapat diatur dengan tag build EKS yang ditentukan, seperti `v1.7.5-eksbuild.1` atau`v1.7.5-eksbuild.2`. Ini juga dapat diatur ke versi rilis addon, seperti `v1.7.5` atau`1.7.5`, dan tag `eksbuild` akhiran akan ditemukan dan diatur untuk Anda.

Lihat bagian di bawah ini tentang cara menemukan addons yang tersedia dan versinya.

## Menemukan addons

Anda dapat menemukan addons apa yang tersedia untuk diinstal pada cluster Anda dengan menjalankan:

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

Ini akan menemukan versi kubernetes cluster Anda dan memfilternya. Atau jika Anda ingin melihat addon apa yang tersedia untuk versi kubernetes tertentu, Anda dapat menjalankan:

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

Anda juga dapat menemukan addons dengan memfilternya`type`,. `owner` and/or `publisher` Misalnya, untuk melihat addons untuk pemilik dan jenis tertentu yang dapat Anda jalankan:

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

Bendera`types`, `owners` dan `publishers` bendera adalah opsional dan dapat ditentukan bersama-sama atau secara individual untuk memfilter hasilnya.

## Menemukan skema konfigurasi untuk addons

Setelah menemukan addon dan versi, Anda dapat melihat opsi penyesuaian dengan mengambil skema konfigurasi JSON-nya.

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

Ini mengembalikan skema JSON dari berbagai opsi yang tersedia untuk addon ini.

## Bekerja dengan nilai konfigurasi

`ConfigurationValues`dapat disediakan dalam file konfigurasi selama pembuatan atau pembaruan addons. Hanya format JSON dan YAMAL yang didukung.

Untuk misalnya. ,

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

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

> **Note**
>
> Ingatlah bahwa ketika nilai konfigurasi addon sedang dimodifikasi, konflik konfigurasi akan muncul.

```
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`.
```

Selain itu, perintah get sekarang juga akan mengambil `ConfigurationValues` untuk addon. misalnya

```
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
```

## Menggunakan namespace khusus

Namespace khusus dapat disediakan dalam file konfigurasi selama pembuatan addons. Namespace tidak dapat diperbarui setelah addon dibuat.

### Menggunakan file konfigurasi

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

### Menggunakan bendera CLI

Atau, Anda dapat menentukan namespace kustom menggunakan bendera: `--namespace-config`

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

Perintah get juga akan mengambil nilai namespace untuk addon

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

```
  PodIdentityAssociations: null
  Status: ACTIVE
  Version: v1.47.0-eksbuild.1
```

## Memperbarui addons

Anda dapat memperbarui addon ke versi yang lebih baru dan mengubah kebijakan apa yang dilampirkan dengan menjalankan:

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

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

> **Note**
>
> Konfigurasi namespace tidak dapat diperbarui setelah addon dibuat. `--namespace-config`Bendera hanya tersedia selama pembuatan addon.

Demikian pula dengan pembuatan addon, Saat memperbarui addon, Anda memiliki kontrol penuh atas perubahan konfigurasi yang mungkin telah Anda terapkan sebelumnya pada add-on itu. `configMap` Secara khusus, Anda dapat melestarikan, atau menimpa mereka. Fungsionalitas opsional ini tersedia melalui bidang file konfigurasi yang `resolveConflicts` sama. misalnya,

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

Untuk pembaruan addon, `resolveConflicts` bidang menerima tiga nilai berbeda:

- none- EKS tidak mengubah nilainya. Pembaruan mungkin gagal.
- `overwrite`- EKS menimpa perubahan konfigurasi apa pun kembali ke nilai default EKS.
- `preserve`- EKS mempertahankan nilainya. Jika Anda memilih opsi ini, kami sarankan Anda menguji setiap bidang dan perubahan nilai pada klaster non-produksi sebelum memperbarui add-on pada cluster produksi Anda.

## Menghapus addons

Anda dapat menghapus addon dengan menjalankan:

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

Ini akan menghapus addon dan peran IAM apa pun yang terkait dengannya.

Saat Anda menghapus klaster, semua peran IAM yang terkait dengan addons juga akan dihapus.

## Fleksibilitas pembuatan cluster untuk addons jaringan default

Ketika sebuah cluster dibuat, EKS secara otomatis menginstal VPC CNI, CoreDNS dan kube-proxy sebagai addon yang dikelola sendiri. Untuk menonaktifkan perilaku ini agar dapat menggunakan plugin CNI lain seperti Cilium dan Calico, eksctl sekarang mendukung pembuatan cluster tanpa addon jaringan default. Untuk membuat cluster seperti itu, atur`addonsConfig.disableDefaultAddons`, seperti pada:

```
addonsConfig:
  disableDefaultAddons: true
```

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

Untuk membuat cluster hanya dengan CoreDNS dan kube-proxy dan bukan VPC CNI, tentukan addon secara eksplisit di dan atur, seperti pada: `addons` `addonsConfig.disableDefaultAddons`

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

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

Sebagai bagian dari perubahan ini, eksctl sekarang menginstal addon default sebagai addon EKS alih-alih addon yang dikelola sendiri selama pembuatan cluster jika tidak secara eksplisit disetel ke true. `addonsConfig.disableDefaultAddons` Dengan demikian, `eksctl utils update-*`

perintah tidak dapat lagi digunakan untuk memperbarui addons untuk cluster yang dibuat dengan eksctl v0.184.0 dan di atasnya:

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

Sebaliknya, `eksctl update addon` harus digunakan sekarang.

Untuk mempelajari lebih lanjut, lihat Amazon EKS memperkenalkan fleksibilitas pembuatan klaster untuk add-on jaringan.

## Mengaktifkan Akses untuk Amazon EMR

Untuk memungkinkan EMR melakukan operasi pada Kubernetes API, SLR-nya perlu diberikan izin RBAC yang diperlukan. eksctl menyediakan perintah yang membuat sumber daya RBAC yang diperlukan untuk EMR, dan memperbarui untuk mengikat peran dengan SLR untuk EMR. `aws-auth` ConfigMap

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

## Dukungan EKS Fargate

AWS Fargate adalah mesin komputasi terkelola untuk Amazon ECS yang dapat menjalankan kontainer. Di Fargate Anda tidak perlu mengelola server atau cluster.

Amazon EKS sekarang dapat meluncurkan pod ke AWS Fargate. Ini menghilangkan kebutuhan untuk khawatir tentang bagaimana Anda menyediakan atau mengelola infrastruktur untuk pod dan membuatnya lebih mudah untuk membangun dan menjalankan aplikasi Kubernetes yang berkinerja tinggi dan sangat tersedia di AWS.

### Membuat cluster dengan dukungan Fargate

Anda dapat menambahkan cluster dengan dukungan Fargate dengan:

```
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
```

Perintah ini akan membuat cluster dan profil Fargate. Profil ini berisi informasi tertentu yang dibutuhkan AWS untuk membuat instance pod di Fargate. Ini adalah:

- peran eksekusi pod untuk menentukan izin yang diperlukan untuk menjalankan pod dan lokasi jaringan (subnet) untuk menjalankan pod. Hal ini memungkinkan izin jaringan dan keamanan yang sama diterapkan ke beberapa Pod Fargate dan membuatnya lebih mudah untuk memigrasi pod yang ada di klaster ke Fargate.
- Selector untuk menentukan pod mana yang harus dijalankan di Fargate. Ini disusun oleh a `namespace` dan`labels`.

Ketika profil tidak ditentukan tetapi dukungan untuk Fargate diaktifkan dengan `--fargate` profil Fargate default dibuat. Profil ini menargetkan `default` dan `kube-system` namespace sehingga pod di namespace tersebut akan berjalan di Fargate.

Profil Fargate yang dibuat dapat diperiksa dengan perintah berikut:

```
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
```

Untuk mempelajari selengkapnya tentang penyeleksi, lihat Merancang profil Fargate.

## Membuat cluster dengan dukungan Fargate menggunakan file konfigurasi

File konfigurasi berikut mendeklarasikan cluster EKS dengan nodegroup yang terdiri dari satu instance EC2 `m5.large` dan dua profil Fargate. Semua pod yang didefinisikan dalam `default` dan `kube-system` namespace akan berjalan di Fargate. Semua pod di `dev` namespace yang juga memiliki label juga `dev=passed` akan berjalan di Fargate. Pod lain akan dijadwalkan pada node ding-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
```

# Merancang profil Fargate

Setiap entri pemilih memiliki hingga dua komponen, namespace dan daftar pasangan kunci-nilai. Hanya komponen namespace yang diperlukan untuk membuat entri pemilih. Semua aturan (namespace, pasangan nilai kunci) harus diterapkan ke pod agar sesuai dengan entri pemilih. Sebuah pod hanya perlu mencocokkan satu entri pemilih untuk dijalankan di profil. Pod apa pun yang cocok dengan semua kondisi di bidang pemilih akan dijadwalkan untuk dijalankan di Fargate. Pod apa pun yang tidak cocok dengan Namespace yang masuk daftar putih tetapi di mana pengguna secara manual menyetel bidang scheduler: fargate-scheduler akan macet dalam status Pending, karena tidak diizinkan untuk dijalankan di Fargate.

Profil harus memenuhi persyaratan berikut:

- Satu pemilih wajib per profil
- Setiap pemilih harus menyertakan namespace; label bersifat opsional

## Contoh: penjadwalan beban kerja di Fargate

Untuk menjadwalkan pod di Fargate untuk contoh yang disebutkan di atas, seseorang dapat, misalnya, membuat namespace yang dipanggil `dev` dan menerapkan beban kerja di sana:

```
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
```

Dari output dari `kubectl get pods` perintah terakhir kita dapat melihat bahwa `nginx` pod dideploy dalam sebuah node yang disebut`fargate-ip-192-168-183-140.ap-northeast-1.compute.internal`.

## Mengelola profil Fargate

Untuk menerapkan beban kerja Kubernetes di Fargate, EKS membutuhkan profil Fargate. Saat membuat cluster seperti pada contoh di atas`eksctl`, atasi ini dengan membuat profil default. Mengingat cluster yang sudah ada, dimungkinkan juga untuk membuat profil Fargate dengan perintah: `eksctl create fargateprofile`

> **Note**
>
> Operasi ini hanya didukung pada cluster yang berjalan pada versi platform EKS `eks.5` atau lebih tinggi.

> **Note**
>
> Jika cluster yang ada dibuat dengan versi `eksctl` sebelum 0.11.0, Anda harus menjalankan `eksctl upgrade cluster` sebelum membuat profil Fargate.

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

Anda juga dapat menentukan nama profil Fargate yang akan dibuat. Nama ini tidak boleh dimulai dengan awalan`eks-`.

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

Menggunakan perintah ini dengan bendera CLI eksctl hanya dapat membuat profil Fargate tunggal dengan pemilih sederhana. Untuk pemilih yang lebih kompleks, misalnya dengan lebih banyak ruang nama, eksctl mendukung penggunaan file konfigurasi:

```
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
```

Untuk melihat profil Fargate yang ada dalam sebuah cluster:

```
eksctl get fargateprofile --cluster fargate-example-cluster
NAME         SELECTOR_NAMESPACE  SELECTOR_LABELS  POD_EXECUTION_ROLE_ARN
                                                  SUBNETS
```

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

Dan untuk melihatnya dalam `yaml` format:

```
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
```

Atau dalam `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"
        ]
    }
]
```

Profil Fargate tidak dapat diubah menurut desain. Untuk mengubah sesuatu, buat profil Fargate baru dengan perubahan yang diinginkan dan hapus yang lama dengan `eksctl delete fargateprofile` perintah seperti pada contoh berikut:

```
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"
}
```

Perhatikan bahwa penghapusan profil adalah proses yang dapat memakan waktu hingga beberapa menit. Ketika `--wait` flag tidak ditentukan, `eksctl` secara optimis mengharapkan profil dihapus dan kembali segera setelah permintaan AWS API dikirim. Untuk membuat `eksctl` menunggu sampai profil telah berhasil dihapus, gunakan `--wait` seperti pada contoh di atas.

### Sumber bacaan lebih lanjut

- AWS Fargate
- Amazon EKS sekarang dapat meluncurkan pod ke AWS Fargate

## Upgrade cluster

Cluster yang dikelola `eksctl` dapat ditingkatkan dalam 3 langkah mudah:

1. tingkatkan versi pesawat kontrol dengan `eksctl upgrade cluster`
2. tingkatkan nodegroup
3. memperbarui add-on jaringan default (Untuk informasi selengkapnya, lihatthe section called “Pembaruan add-on default”):

Tinjau sumber daya terkait peningkatan klaster dengan cermat:

- Perbarui klaster yang ada ke versi Kubernetes baru di Panduan Pengguna Amazon EKS
- Praktik Terbaik untuk Upgrade Cluster dalam Panduan Praktik Terbaik EKS

> **Note**
>
> Yang lama `eksctl update cluster` akan usang. Gunakan `eksctl upgrade cluster` sebagai gantinya.

## Memperbarui versi bidang kontrol

Upgrade versi pesawat kontrol harus dilakukan untuk satu versi minor pada satu waktu.

Untuk memutakhirkan bidang kontrol ke versi berikutnya yang tersedia, jalankan:

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

Perintah ini tidak akan segera menerapkan perubahan apa pun, Anda harus menjalankannya kembali `--approve` untuk menerapkan perubahan.

Versi target untuk upgrade cluster dapat ditentukan keduanya dengan flag CLI:

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

atau dengan file konfigurasi

```
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**
>
> Satu-satunya nilai yang diizinkan untuk `metadata.version` argumen `--version` dan adalah versi cluster saat ini atau satu versi yang lebih tinggi. Upgrade lebih dari satu versi Kubernetes tidak didukung.

# Pembaruan add-on default

Topik ini menjelaskan cara memperbarui add-on pra-instal default yang disertakan pada kluster EKS.

> **Warning**
>
> eksctl sekarang menginstal addon default sebagai addons EKS alih-alih addon yang dikelola sendiri. Baca lebih lanjut tentang implikasinya dalam fleksibilitas pembuatan Cluster untuk addon jaringan default.
> Untuk memperbarui addons, `eksctl utils update-<addon>` tidak dapat digunakan untuk cluster yang dibuat dengan eksctl v0.184.0 dan di atasnya. Panduan ini hanya berlaku untuk cluster yang dibuat sebelum perubahan ini.

Ada 3 add-on default yang disertakan dalam setiap cluster EKS:

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

## Perbarui add-on yang sudah diinstal sebelumnya

Untuk addon EKS resmi yang dibuat secara manual melalui `eksctl create addons` atau setelah pembuatan cluster, cara mengelolanya adalah melalui`eksctl create/get/update/delete addon`. Dalam kasus seperti itu, silakan merujuk ke dokumen tentang EKS Add-Ons.

Proses untuk memperbarui masing-masing berbeda, maka ada 3 perintah berbeda yang perlu Anda jalankan. Semua perintah berikut menerima`--config-file`. Secara default masing-masing perintah ini berjalan dalam mode rencana, jika Anda puas dengan perubahan yang diusulkan, jalankan kembali dengan`--approve`.

Untuk memperbarui`kube-proxy`, jalankan:

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

Untuk memperbarui`aws-node`, jalankan:

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

Untuk memperbarui`coredns`, jalankan:

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

Setelah ditingkatkan, pastikan untuk menjalankan `kubectl get pods -n kube-system` dan memeriksa apakah semua pod addon dalam keadaan siap, Anda akan melihat sesuatu seperti ini:

```
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 untuk Zonal Shift di kluster EKS

EKS sekarang mendukung pergeseran zona Amazon Application Recovery Controller (ARC) dan pergeseran otomatis zona yang meningkatkan ketahanan lingkungan cluster multi-AZ. Dengan AWS Zonal Shift, pelanggan dapat mengalihkan lalu lintas dalam klaster dari zona ketersediaan yang terganggu, memastikan pod dan node Kubernetes baru diluncurkan hanya di zona ketersediaan yang sehat.

## Membuat cluster dengan pergeseran zona diaktifkan

```
# 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
```

## Mengaktifkan pergeseran zona pada cluster yang ada

Untuk mengaktifkan atau menonaktifkan pergeseran zona pada cluster yang ada, jalankan

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

atau tanpa file konfigurasi:

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

### Informasi lebih lanjut

- Pergeseran Zonal EKS

## Support Karpenter

`eksctl`memberikan dukungan untuk menambahkan Karpenter ke cluster yang baru dibuat. Ini akan menciptakan semua prasyarat yang diperlukan yang diuraikan dalam bagian Memulai Karpenter termasuk menginstal Karpenter sendiri menggunakan Helm. Saat ini kami mendukung pemasangan versi`0.28.0`+. Lihat bagian kompatibilitas Karpenter untuk detail lebih lanjut.

Konfigurasi cluster berikut menguraikan instalasi Karpenter yang khas:

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

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

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

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

```
    desiredCapacity: 1
```

Versi ini adalah versi Karpenter karena dapat ditemukan di Repositori Helm mereka. Opsi berikut juga tersedia untuk diatur:

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

OIDC harus didefinisikan untuk menginstal Karpenter.

Setelah Karpenter berhasil diinstal, tambahkan NodePool(s) dan NodeClass(es) untuk memungkinkan Karpenter untuk mulai menambahkan node ke cluster.

`nodeClassRef`Bagian NodePool ini harus cocok dengan nama sebuah`EC2NodeClass`. Contoh:

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: example
  annotations:
    kubernetes.io/description: "Example NodePool"
spec:
  template:
    spec:
      requirements:
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64"]
        - key: kubernetes.io/os
          operator: In
          values: ["linux"]
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["on-demand"]
        - key: karpenter.k8s.aws/instance-category
          operator: In
          values: ["c", "m", "r"]
```

```
        - key: karpenter.k8s.aws/instance-generation
          operator: Gt
          values: ["2"]
      nodeClassRef:
        group: karpenter.k8s.aws
        kind: EC2NodeClass
        name: example # must match the name of an EC2NodeClass
```

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

Perhatikan bahwa Anda harus menentukan salah satu `role` atau `instanceProfile` untuk node peluncuran. Jika Anda memilih untuk menggunakan `instanceProfile` nama profil yang dibuat dengan `eksctl` mengikuti pola:`eksctl-KarpenterNodeInstanceProfile-<cluster-name>`.

## Penandaan Grup Keamanan Otomatis

`eksctl`secara otomatis menandai grup keamanan simpul bersama cluster dengan `karpenter.sh/discovery` ketika kedua Karpenter diaktifkan (`karpenter.version`ditentukan) dan `karpenter.sh/discovery` tag ada di. `metadata.tags` Ini memungkinkan kompatibilitas AWS Load Balancer Controller.

Catatan dengan karpenter 0.32.0+, Penyedia telah usang dan diganti dengan. NodePool

# Skema Konfigurasi Cluster

> **Note**
>
> Lokasi skema saat ini sedang dimigrasikan.

Anda dapat menggunakan file yaml untuk membuat cluster. Lihat referensi skema.

Contoh:

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

Referensi skema untuk file ini tersedia di GitHub.

Untuk informasi selengkapnya tentang menggunakan file, lihatthe section called “Membuat dan mengelola cluster”.

# Nodegroup

Bab ini mencakup informasi tentang bagaimana Anda membuat dan mengkonfigurasi Nodegroups dengan Eksctl. Nodegroups adalah grup instans EC2 yang dilampirkan ke cluster EKS.

## Topik:

- the section called “Instans Spot”
  - Buat dan kelola kluster EKS dengan instance Spot menggunakan grup node terkelola
  - Konfigurasikan instance Spot untuk grup node yang tidak dikelola menggunakan MixedInstancesPolicy
  - Membedakan instans Spot dan On-Demand menggunakan label Kubernetes `node-lifecycle`
- the section called “Auto Scaling”
  - Aktifkan penskalaan otomatis node cluster Kubernetes dengan membuat klaster atau nodegroup dengan peran IAM yang memungkinkan penggunaan autoscaler cluster
  - Konfigurasikan definisi nodegroup untuk menyertakan tag dan anotasi yang diperlukan untuk penskalaan otomatis cluster untuk menskalakan nodegroup
  - Buat nodegroup terpisah untuk setiap zona ketersediaan jika beban kerja memiliki persyaratan khusus zona, seperti penyimpanan khusus zona atau aturan afinitas
- the section called “Grup nodegroup yang dikelola EKS”
  - Menyediakan dan mengelola instans (node) EC2 untuk klaster Amazon EKS Kubernetes
  - Terapkan perbaikan bug, patch keamanan, dan perbarui node dengan mudah ke versi Kubernetes terbaru
- the section called “Node Hibrida EKS”
  - Aktifkan menjalankan aplikasi lokal dan edge pada infrastruktur yang dikelola pelanggan dengan kluster, fitur, dan alat Amazon EKS yang sama yang digunakan di AWS
  - Konfigurasikan jaringan untuk menghubungkan jaringan lokal ke VPC, menggunakan opsi seperti AWS VPN atau Site-to-Site AWS Direct Connect
  - Siapkan kredensional untuk node jarak jauh untuk mengautentikasi dengan kluster EKS, menggunakan AWS Systems Manager (SSM) atau AWS IAM Roles Anywhere
- the section called “Konfigurasi Perbaikan Node”
  - Mengaktifkan Perbaikan Node untuk Grup Node Terkelola EKS untuk secara otomatis memantau dan mengganti atau me-reboot node pekerja yang tidak sehat

- the section called “Dukungan ARM”
  - Buat cluster EKS dengan instans Graviton berbasis ARM untuk meningkatkan kinerja dan efisiensi biaya
- the section called “Noda”
  - Menerapkan taints ke grup node tertentu di klaster Kubernetes
  - Kontrol penjadwalan dan penggusuran pod berdasarkan kunci, nilai, dan efek taint
- the section called “Luncurkan dukungan templat”
  - Meluncurkan grup node terkelola menggunakan Template Peluncuran EC2 yang disediakan
  - Memutakhirkan grup node terkelola untuk menggunakan versi Template Peluncuran yang berbeda
  - Memahami batasan dan pertimbangan saat menggunakan kustom AMIs dan Luncurkan Template dengan grup node terkelola
- the section called “Bekerja dengan grup simpul”
  - Aktifkan akses SSH ke instans EC2 di grup node
  - Skala jumlah node dalam grup node ke atas atau ke bawah
- the section called “Subnet kustom”
  - Perluas VPC yang ada dengan subnet baru dan tambahkan Nodegroup ke subnet itu
- the section called “Bootstrap simpul”
  - Memahami proses inisialisasi node baru (nodeadm) yang diperkenalkan pada tahun 2023 AmazonLinux
  - Pelajari tentang NodeConfig pengaturan default yang diterapkan oleh eksctl untuk node yang dikelola sendiri dan dikelola EKS
  - Sesuaikan proses bootstrapping node dengan menyediakan kustom overrideBootstrapCommand NodeConfig
- the section called “Nodegroup tidak terkelola”
  - Membuat atau memperbarui grup node yang tidak terkelola dalam kluster EKS
  - Perbarui add-on Kubernetes default seperti kube-proxy, aws-node, dan CoreDNS
- the section called “Dukungan GPU”
  - Eksctl mendukung pemilihan jenis instans GPU untuk nodegroup, memungkinkan penggunaan beban kerja yang dipercepat GPU pada kluster EKS.
  - Eksctl secara otomatis menginstal plugin perangkat NVIDIA Kubernetes ketika jenis instans berkemampuan GPU dipilih, memfasilitasi penggunaan sumber daya GPU di cluster.

  - Pengguna dapat menonaktifkan instalasi plugin otomatis dan menginstal versi tertentu dari plugin perangkat NVIDIA Kubernetes secara manual menggunakan perintah yang disediakan.
- the section called “Pemilih Instance”
  - Secara otomatis menghasilkan daftar jenis instans EC2 yang sesuai berdasarkan kriteria sumber daya seperti vCPUs, memori GPUs, dan arsitektur CPU
  - Buat cluster dan grup node dengan tipe instance yang cocok dengan kriteria pemilih instance yang ditentukan
  - Lakukan dry run untuk memeriksa dan memodifikasi tipe instance yang cocok dengan pemilih instance sebelum membuat grup node
- the section called “Pemetaan Volume Tambahan”
  - Konfigurasikan pemetaan volume tambahan untuk grup node terkelola dalam kluster EKS
  - Sesuaikan properti volume seperti ukuran, jenis, enkripsi, IOPS, dan throughput untuk volume tambahan
  - Lampirkan snapshot EBS yang ada sebagai volume tambahan ke grup node
- the section called “Node Pekerja Windows”
  - Tambahkan grup node Windows ke cluster Linux Kubernetes yang ada untuk mengaktifkan menjalankan beban kerja Windows
  - Jadwalkan beban kerja pada sistem operasi yang sesuai (Windows atau Linux) menggunakan pemilih node berdasarkan label dan `kubernetes.io/os` `kubernetes.io/arch`
- the section called “Dukungan AMI kustom”
  - Gunakan `--node-ami` flag untuk menentukan AMI kustom untuk grup node, kueri AWS untuk AMI terbaru yang dioptimalkan EKS, atau gunakan AWS Systems Manager Parameter Store untuk menemukan AMI.
  - Atur `--node-ami-family` flag untuk menentukan keluarga sistem operasi untuk grup node AMI, seperti AmazonLinux 2, Ubuntu2204, atau 2022. WindowsServer CoreContainer
  - Untuk grup node Windows, tentukan AMI khusus dan berikan skrip PowerShell bootstrap melalui file`overrideBootstrapCommand`.
- the section called “Kustom DNS”
  - Timpa alamat IP server DNS yang digunakan untuk pencarian DNS internal dan eksternal

# Bekerja dengan grup simpul

## Membuat nodegroup

Anda dapat menambahkan satu atau beberapa nodegroup selain nodegroup awal yang dibuat bersama dengan cluster.

Untuk membuat nodegroup tambahan, gunakan:

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

> **Note**
>
> `--version`flag tidak didukung untuk nodegroup terkelola. Itu selalu mewarisi versi dari bidang kontrol.

Secara default, nodegroup baru yang tidak dikelola mewarisi versi dari control plane (`--version=auto`), tetapi Anda dapat menentukan versi yang berbeda, Anda juga dapat menggunakan `--version=latest` untuk memaksa penggunaan versi mana pun yang terbaru.

Selain itu, Anda dapat menggunakan file konfigurasi yang sama yang digunakan untuk`eksctl create cluster`:

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

### Membuat nodegroup dari file konfigurasi

Nodegroups juga dapat dibuat melalui definisi cluster atau file konfigurasi. Diberikan contoh file konfigurasi berikut dan cluster yang ada disebut`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
```

Nodegroups `ng-1-workers` dan `ng-2-builders` dapat dibuat dengan perintah ini:

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

Penyeimbangan Beban

Jika Anda sudah siap untuk melampirkan grup or/and target penyeimbang beban klasik yang ada ke nodegroups, Anda dapat menentukannya di file konfigurasi. Grup or/and target penyeimbang beban klasik secara otomatis dikaitkan dengan ASG saat membuat nodegroup. Ini hanya didukung untuk nodegroup yang dikelola sendiri yang ditentukan melalui bidang. `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
```

## Pemilihan nodegroup dalam file konfigurasi

Untuk melakukan `delete` operasi `create` atau hanya pada subset dari nodegroup yang ditentukan dalam file konfigurasi, ada dua flag CLI yang menerima daftar gumpalan, dan, misalnya: `0` `1`

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

Dengan menggunakan contoh file konfigurasi di atas, seseorang dapat membuat semua nodegroup pekerja kecuali yang pekerja dengan perintah berikut:

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

Atau seseorang dapat menghapus nodegroup pembangun dengan:

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

Dalam hal ini, kita juga perlu menyediakan `--approve` perintah untuk benar-benar menghapus nodegroup.

### Sertakan dan kecualikan aturan

- jika tidak `--exclude` ada `--include` atau ditentukan semuanya disertakan
- jika hanya `--include` ditentukan, hanya grup simpul yang cocok dengan gumpalan tersebut yang akan disertakan
- jika hanya `--exclude` ditentukan, semua nodegroup yang tidak cocok dengan gumpalan tersebut disertakan
- jika keduanya ditentukan maka `--exclude` aturan lebih diutamakan `--include` (yaitu nodegroup yang cocok dengan aturan di kedua grup akan dikecualikan)

## Daftar nodegroup

Untuk mencantumkan detail tentang nodegroup atau semua nodegroup, gunakan:

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

Untuk mencantumkan satu atau beberapa nodegroup dalam format YAMB atau JSON, yang menampilkan lebih banyak info daripada tabel log default, gunakan:

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

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

## Kekekalan nodegroup

Secara desain, nodegroup tidak dapat diubah. Ini berarti bahwa jika Anda perlu mengubah sesuatu (selain penskalaan) seperti AMI atau tipe instance nodegroup, Anda perlu membuat nodegroup baru dengan perubahan yang diinginkan, memindahkan beban dan menghapus yang lama. Lihat bagian Menghapus dan menguras nodegroups.

## Penskalaan nodegroup

Penskalaan nodegroup adalah proses yang dapat memakan waktu hingga beberapa menit. Ketika `--wait` flag tidak ditentukan, `eksctl` secara optimis mengharapkan nodegroup untuk diskalakan dan kembali segera setelah permintaan AWS API dikirim. Untuk `eksctl` menunggu sampai node tersedia, tambahkan `--wait` bendera seperti contoh di bawah ini.

> **Note**
>
> Penskalaan nodegroup down/in (yaitu mengurangi jumlah node) dapat mengakibatkan kesalahan karena kami hanya mengandalkan perubahan pada ASG. Ini berarti bahwa node yang removed/terminated tidak terkuras secara eksplisit. Ini mungkin area untuk perbaikan di masa depan.

Penskalaan nodegroup terkelola dicapai dengan langsung memanggil EKS API yang memperbarui konfigurasi grup node terkelola.

## Menskalakan satu nodegroup

Sebuah nodegroup dapat diskalakan dengan menggunakan perintah: `eksctl scale nodegroup`

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

Misalnya, untuk menskalakan nodegroup `ng-a345f4e1` `cluster-1` menjadi 5 node, jalankan:

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

Nodegroup juga dapat diskalakan dengan menggunakan file konfigurasi yang diteruskan ke `--config-file` dan menentukan nama nodegroup yang harus diskalakan. `--name` Eksctl akan mencari file konfigurasi dan menemukan nodegroup serta nilai konfigurasi penskalaannya.

Jika jumlah node yang diinginkan `NOT` berada dalam kisaran minimum saat ini dan jumlah maksimum node saat ini, satu kesalahan spesifik akan ditampilkan. Nilai-nilai ini juga dapat diteruskan dengan flag `--nodes-min` dan `--nodes-max` masing-masing.

## Menskalakan beberapa nodegroup

Eksctl dapat menemukan dan menskalakan semua nodegroup yang ditemukan dalam file konfigurasi yang diteruskan. `--config-file`

Demikian pula dengan penskalaan satu nodegroup, kumpulan validasi yang sama berlaku untuk setiap nodegroup. Misalnya, jumlah node yang diinginkan harus berada dalam kisaran jumlah node minimum dan maksimum.

## Menghapus dan menguras nodegroups

Untuk menghapus nodegroup, jalankan:

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

Termasuk dan kecualikan aturan juga dapat digunakan dengan perintah ini.

> **Note**
>
> Ini akan menguras semua pod dari nodegroup tersebut sebelum instance dihapus.

Untuk melewati aturan penggusuran selama proses pembuangan, jalankan:

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

Semua node dikepung dan semua pod dikeluarkan dari nodegroup saat dihapus, tetapi jika Anda perlu menguras nodegroup tanpa menghapusnya, jalankan:

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

Untuk membuka kordon nodegroup, jalankan:

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

Untuk mengabaikan aturan penggusuran seperti PodDisruptionBudget pengaturan, jalankan:

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

Untuk mempercepat proses pembuangan, Anda dapat menentukan `--parallel <value>` jumlah node yang akan mengalir secara paralel.

## Fitur lainnya

Anda juga dapat mengaktifkan SSH, akses ASG, dan fitur lainnya untuk nodegroup, misalnya:

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

## Perbarui label

Tidak ada perintah khusus `eksctl` untuk memperbarui label nodegroup, tetapi dapat dengan mudah dicapai dengan menggunakan`kubectl`, misalnya:

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

## Akses SSH

Anda dapat mengaktifkan akses SSH untuk nodegroups dengan mengonfigurasi salah satu dari`publicKey`, `publicKeyName` dan `publicKeyPath` dalam konfigurasi nodegroup Anda. Atau Anda dapat menggunakan AWS Systems Manager (SSM) ke SSH ke node, dengan mengonfigurasi nodegroup dengan: `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
```

# Nodegroup tidak terkelola

Dieksctl, menyetel `--managed=false` atau menggunakan `nodeGroups` bidang akan membuat nodegroup yang tidak dikelola. Ingatlah bahwa nodegroup yang tidak dikelola tidak muncul di konsol EKS, yang sebagai aturan umum hanya mengetahui tentang nodegroup yang dikelola EKS.

Anda harus memutakhirkan nodegroups hanya setelah Anda menjalankan. `eksctl upgrade cluster` (Lihat Memutakhirkan cluster.)

Jika Anda memiliki cluster sederhana hanya dengan nodegroup awal (yaitu dibuat dengan`eksctl create cluster`), prosesnya sangat sederhana:

1. Dapatkan nama nodegroup lama:

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

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

2. Buat nodegroup baru:

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

3. Hapus nodegroup lama:

```
 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.
> ```

## Memperbarui beberapa nodegroup

Jika Anda memiliki beberapa nodegroup, Anda bertanggung jawab untuk melacak bagaimana masing-masing grup dikonfigurasi. Anda dapat melakukan ini dengan menggunakan file konfigurasi, tetapi jika Anda belum menggunakannya, Anda perlu memeriksa cluster Anda untuk mengetahui bagaimana setiap nodegroup dikonfigurasi.

Secara umum, Anda mencari untuk:

- tinjau nodegroup mana yang Anda miliki dan mana yang dapat dihapus atau harus diganti untuk versi baru
- catat konfigurasi setiap nodegroup, pertimbangkan untuk menggunakan file konfigurasi untuk memudahkan peningkatan di lain waktu

### Memperbarui dengan file konfigurasi

Jika Anda menggunakan file konfigurasi, Anda perlu melakukan hal berikut.

Edit file konfigurasi untuk menambahkan nodegroup baru, dan hapus nodegroup lama. Jika Anda hanya ingin memutakhirkan nodegroups dan mempertahankan konfigurasi yang sama, Anda bisa mengubah nama nodegroup, misalnya menambahkan `-v2` nama.

Untuk membuat semua nodegroup baru yang ditentukan dalam file konfigurasi, jalankan:

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

Setelah Anda memiliki nodegroup baru di tempat, Anda dapat menghapus yang lama:

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

> **Note**
>
> Jalankan pertama dalam mode rencana, jika Anda puas dengan perubahan yang diusulkan, jalankan kembali dengan`--approve`.

## Memperbarui add-on default

Anda mungkin perlu memperbarui add-on jaringan yang diinstal pada cluster Anda. Untuk informasi selengkapnya, lihat the section called “Pembaruan add-on default”.

# Grup nodegroup yang dikelola EKS

Amazon EKS managed nodegroups adalah fitur yang mengotomatiskan penyediaan dan pengelolaan siklus hidup node (instans EC2) untuk klaster Amazon EKS Kubernetes. Pelanggan dapat menyediakan grup node yang dioptimalkan untuk klaster mereka dan EKS akan memperbarui node mereka dengan versi Kubernetes dan OS host terbaru.

Grup node terkelola EKS adalah grup penskalaan otomatis dan instans EC2 terkait yang dikelola oleh AWS untuk kluster Amazon EKS. Setiap grup node menggunakan Amazon Amazon Linux 2 AMI yang dioptimalkan Amazon EKS. Amazon EKS memudahkan penerapan perbaikan bug dan patch keamanan ke node, serta memperbaruinya ke versi Kubernetes terbaru. Setiap grup node meluncurkan grup penskalaan otomatis untuk klaster Anda, yang dapat menjangkau beberapa zona ketersediaan dan subnet AWS VPC untuk ketersediaan tinggi.

Dukungan Template Peluncuran BARU untuk nodegroup terkelola

> **Note**
>
> Istilah “nodegroup tidak terkelola” telah digunakan untuk merujuk pada nodegroup yang telah didukung eksctl sejak awal (diwakili melalui bidang). `nodeGroups` `ClusterConfig`File terus menggunakan `nodeGroups` bidang untuk mendefinisikan nodegroup yang tidak dikelola, dan nodegroup terkelola didefinisikan dengan bidang. `managedNodeGroups`

## Membuat nodegroup terkelola

```
$ eksctl create nodegroup
```

## Cluster baru

Untuk membuat cluster baru dengan nodegroup terkelola, jalankan

```
eksctl create cluster
```

Untuk membuat beberapa nodegroup terkelola dan memiliki kontrol lebih besar atas konfigurasi, file konfigurasi dapat digunakan.

> **Note**
>
> Nodegroup terkelola tidak memiliki paritas fitur lengkap dengan nodegroup yang tidak dikelola.

```
# 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
```

Contoh lain dari file konfigurasi untuk membuat nodegroup terkelola dapat ditemukan di sini.

Dimungkinkan untuk memiliki cluster dengan nodegroup terkelola dan tidak terkelola. Nodegroup yang tidak dikelola tidak muncul di konsol Amazon EKS tetapi `eksctl get nodegroup` akan mencantumkan kedua jenis nodegroup.

```
# 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
```

Dukungan BARU untuk AMI kustom, grup keamananinstancePrefix,,instanceName,ebsOptimized,volumeType,volumeName,volumeEncryp dan 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'
```

Jika Anda meminta jenis instance yang hanya tersedia di satu zona (dan konfigurasi eksctl memerlukan spesifikasi dua) pastikan untuk menambahkan zona ketersediaan ke permintaan grup node Anda:

```
# 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
```

Ini bisa benar untuk tipe contoh seperti keluarga Hpc6 yang hanya tersedia di satu zona.

## Cluster yang ada

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

Tip: jika Anda menggunakan `ClusterConfig` file untuk mendeskripsikan seluruh cluster Anda, jelaskan grup node terkelola baru Anda di `managedNodeGroups` bidang dan jalankan:

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

# Memutakhirkan nodegroup terkelola

Anda dapat memperbarui nodegroup ke versi rilis AMI terbaru yang dioptimalkan EKS untuk tipe AMI yang Anda gunakan kapan saja.

Jika nodegroup Anda adalah versi Kubernetes yang sama dengan cluster, Anda dapat memperbarui ke versi rilis AMI terbaru untuk versi Kubernetes dari tipe AMI yang Anda gunakan. Jika nodegroup Anda adalah versi Kubernetes sebelumnya dari versi Kubernetes cluster, Anda dapat memperbarui nodegroup ke versi rilis AMI terbaru yang cocok dengan versi Kubernetes nodegroup, atau memperbarui ke versi rilis AMI terbaru yang cocok dengan cluster versi Kubernetes. Anda tidak dapat memutar kembali nodegroup ke versi Kubernetes sebelumnya.

Untuk memutakhirkan nodegroup terkelola ke versi rilis AMI terbaru:

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

Nodegroup dapat ditingkatkan ke rilis AMI terbaru untuk versi Kubernetes tertentu menggunakan:

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

Untuk memutakhirkan ke versi rilis AMI tertentu alih-alih versi terbaru, teruskan`--release-version`:

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

> **Note**
>
> Jika node terkelola disebarkan menggunakan kustom AMIs, alur kerja berikut harus diikuti untuk menerapkan versi baru AMI kustom.

- penyebaran awal nodegroup harus dilakukan menggunakan template peluncuran. misalnya

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

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

- buat versi baru AMI kustom (menggunakan konsol Amazon EKS).
- buat versi template peluncuran baru dengan ID AMI baru (menggunakan konsol Amazon EKS).
- tingkatkan node ke versi baru dari template peluncuran. misalnya

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

## Menangani peningkatan paralel untuk node

Beberapa node terkelola dapat ditingkatkan secara bersamaan. Untuk mengonfigurasi peningkatan paralel, tentukan nodegroup saat membuat nodegroup. `updateConfig` Contohnya `updateConfig` dapat ditemukan di sini.

Untuk menghindari downtime pada beban kerja Anda karena memutakhirkan beberapa node sekaligus, Anda dapat membatasi jumlah node yang dapat menjadi tidak tersedia selama upgrade dengan menentukan ini di bidang. `maxUnavailable` `updateConfig` Atau, gunakan`maxUnavailablePercentage`, yang mendefinisikan jumlah maksimum node yang tidak tersedia sebagai persentase dari jumlah total node.

Perhatikan bahwa `maxUnavailable` tidak boleh lebih tinggi dari`maxSize`. Juga, `maxUnavailable` dan `maxUnavailablePercentage` tidak dapat digunakan secara bersamaan.

Fitur ini hanya tersedia untuk node terkelola.

## Memperbarui nodegroup terkelola

`eksctl`memungkinkan memperbarui UpdateConfigbagian dari nodegroup terkelola. Bagian ini mendefinisikan dua bidang. `MaxUnavailable`dan`MaxUnavailablePercentage`. Nodegroup Anda tidak terpengaruh selama pembaruan, sehingga waktu henti seharusnya tidak diharapkan.

Perintah `update nodegroup` harus digunakan dengan file konfigurasi menggunakan `--config-file` bendera. Nodegroup harus berisi bagian. `nodeGroup.updateConfig` Informasi lebih lanjut dapat ditemukan di sini.

## Masalah Kesehatan Nodegroup

EKS Managed Nodegroups secara otomatis memeriksa konfigurasi nodegroup dan node Anda untuk masalah kesehatan dan melaporkannya melalui EKS API dan konsol. Untuk melihat masalah kesehatan untuk nodegroup:

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

## Mengelola Label

EKS Managed Nodegroups mendukung melampirkan label yang diterapkan ke node Kubernetes di nodegroup. Ini ditentukan melalui `labels` bidang di eksctl selama pembuatan cluster atau nodegroup.

Untuk menyetel label baru atau memperbarui label yang ada pada nodegroup:

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

Untuk menghapus atau menghapus label dari nodegroup:

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

Untuk melihat semua label yang disetel pada nodegroup:

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

## Penskalaan Grup Node Terkelola

`eksctl scale nodegroup`juga mendukung nodegroup terkelola. Sintaks untuk penskalaan nodegroup terkelola atau tidak terkelola adalah sama.

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

## Informasi lebih lanjut

- Grup Nodegroup yang Dikelola EKS

# Bootstrap simpul

## AmazonLinux2023

AL2023 memperkenalkan proses inisialisasi node baru nodeadm yang menggunakan skema konfigurasi YAMB, menjatuhkan penggunaan skrip. `/etc/eks/bootstrap.sh`

> **Note**
>
> Dengan Kubernetes versi 1.30 ke atas, Amazon Linux 2023 adalah OS default.

### Pengaturan default untuk AL2

Untuk node yang dikelola sendiri dan node yang dikelola EKS berdasarkan kustom AMIs, `eksctl` buat default, minimal, `NodeConfig` dan secara otomatis menyuntikkannya ke dalam data pengguna template peluncuran nodegroup. yaitu

```
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
```

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

Untuk node yang dikelola EKS berdasarkan native AMIs, default `NodeConfig` ditambahkan oleh EKS MNG di bawah tenda, ditambahkan langsung ke data pengguna EC2. Jadi, dalam skenario ini, `eksctl` tidak perlu memasukkannya ke dalam template peluncuran.

### Mengkonfigurasi proses bootstrap

Untuk mengatur properti lanjutan`NodeConfig`, atau hanya mengganti nilai default, eksctl memungkinkan Anda untuk menentukan kustom melalui atau mis. `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
```

Konfigurasi khusus ini akan ditambahkan ke data pengguna oleh eksctl, dan digabungkan dengan konfigurasi default. `nodeadm` Baca lebih lanjut `nodeadm` tentang kemampuan menggabungkan beberapa objek konfigurasi di sini.

## Luncurkan dukungan Template untuk Managed Nodegroups

eksctl mendukung peluncuran nodegroup terkelola menggunakan Template Peluncuran EC2 yang disediakan. Ini memungkinkan beberapa opsi penyesuaian untuk nodegroup termasuk menyediakan grup kustom AMIs dan keamanan, dan meneruskan data pengguna untuk bootstrap node.

### Membuat nodegroup terkelola menggunakan template peluncuran yang disediakan

```
# 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
```

## Memutakhirkan nodegroup terkelola untuk menggunakan versi template peluncuran yang berbeda

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

> **Note**
>
> Jika template peluncuran menggunakan AMI kustom, maka versi baru juga harus menggunakan AMI khusus atau operasi pemutakhiran akan gagal

Jika template peluncuran tidak menggunakan AMI kustom, versi Kubernetes untuk ditingkatkan juga dapat ditentukan:

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

### Catatan tentang AMI kustom dan dukungan template peluncuran

- Ketika template peluncuran disediakan, bidang berikut tidak didukung:`instanceType`,,,,`ami`,`ssh.allow`,`ssh.sourceSecurityGroupIds`,`securityGroups`,`inst`instanceName`eebsOptimized`,`volumeEncrypted`,`volumeKmsKeyID`,`volumeIOPS`,`maxPodsPerNode`overrideBootstrapCommand` dan`disableIMDSv1`.
- Saat menggunakan AMI kustom (`ami`), juga `overrideBootstrapCommand` harus diatur untuk melakukan bootstrap.
- `overrideBootstrapCommand`hanya dapat diatur saat menggunakan AMI khusus.
- Saat template peluncuran disediakan, tag yang ditentukan dalam konfigurasi nodegroup hanya berlaku untuk sumber daya EKS Nodegroup dan tidak disebarkan ke instans EC2.

## Subnet kustom

Dimungkinkan untuk memperluas VPC yang ada dengan subnet baru dan menambahkan Nodegroup ke subnet itu.

### Mengapa

Jika cluster kehabisan pra-konfigurasi IPs, dimungkinkan untuk mengubah ukuran VPC yang ada dengan CIDR baru untuk menambahkan subnet baru ke dalamnya. Untuk melihat cara melakukannya, baca panduan ini tentang AWS Extending. VPCs

#### TL; DR

Buka konfigurasi VPC dan tambahkan klik Aksi-> Edit CIDRs dan tambahkan rentang baru. Contoh:

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

Sekarang Anda perlu menambahkan Subnet baru. Bergantung pada apakah itu subnet Private atau Public baru, Anda harus menyalin informasi perutean dari subnet pribadi atau publik masing-masing.

Setelah subnet dibuat, tambahkan routing, dan salin ID gateway NAT atau Internet Gateway dari subnet lain di VPC. Berhati-hatilah jika itu adalah subnet publik Aktifkan Penugasan IP Otomatis.

Tindakan-> Ubah pengaturan IP tetapkan otomatis -> Aktifkan alamat publik penetapan otomatis. IPv4

Jangan lupa juga untuk menyalin TAGS dari subnet yang ada tergantung pada konfigurasi subnet Publik atau Pribadi. Ini penting, jika tidak, subnet tidak akan menjadi bagian dari cluster dan instance di subnet tidak akan dapat bergabung.

Setelah selesai, salin ID subnet baru. Ulangi sesering yang diperlukan.

## Bagaimana

Untuk membuat nodegroup di subnet yang dibuat, jalankan perintah berikut:

```
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
```

Atau, gunakan konfigurasi seperti itu:

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

Dengan konfigurasi seperti ini:

```
# 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
```

Tunggu nodegroup dibuat dan instance baru harus memiliki rentang IP baru dari subnet.

### Menghapus cluster

Karena penambahan baru memodifikasi VPC yang ada dengan menambahkan ketergantungan di luar CloudFormation tumpukan, tidak CloudFormation dapat lagi menghapus cluster.

Sebelum menghapus cluster, hapus semua subnet tambahan yang dibuat dengan tangan, lalu lanjutkan dengan memanggil: `eksctl`

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

## Kustom DNS

Ada dua cara untuk menimpa alamat IP server DNS yang digunakan untuk semua pencarian DNS internal dan eksternal. Ini setara dengan `--cluster-dns` bendera untuk`kubelet`.

Yang pertama adalah melalui `clusterDNS` lapangan. File Config menerima `string` bidang yang disebut `clusterDNS` dengan alamat IP server DNS untuk digunakan. Ini akan diteruskan ke `kubelet` yang pada gilirannya akan meneruskannya ke pod melalui `/etc/resolv.conf` file. Untuk informasi selengkapnya, lihat skema file konfigurasi.

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

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

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

Perhatikan bahwa konfigurasi ini hanya menerima satu alamat IP. Untuk menentukan lebih dari satu alamat, gunakan `kubeletExtraConfig`parameter:

```
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"]
```

## Noda

Untuk menerapkan taints ke nodegroup tertentu gunakan bagian `taints` konfigurasi seperti ini:

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

Contoh lengkap dapat ditemukan di sini.

## Pemilih Instance

eksctl mendukung penentuan beberapa tipe instans untuk nodegroup yang dikelola dan dikelola sendiri, tetapi dengan lebih dari 270 tipe instans EC2, pengguna harus meluangkan waktu untuk mencari tahu tipe instance mana yang cocok untuk nodegroup mereka. Ini bahkan lebih sulit ketika menggunakan instance Spot karena Anda perlu memilih satu set instance yang bekerja sama dengan baik dengan Cluster Autoscaler.

eksctl sekarang terintegrasi dengan pemilih instans EC2, yang mengatasi masalah ini dengan membuat daftar jenis instance berdasarkan kriteria sumber daya: vCPUs, memori, # of dan arsitektur CPU. GPUs Ketika kriteria pemilih instance diteruskan, eksctl membuat nodegroup dengan tipe instance yang disetel ke tipe instance yang cocok dengan kriteria yang disediakan.

### Buat cluster dan nodegroup

Untuk membuat cluster dengan nodegroup tunggal yang menggunakan tipe instance yang cocok dengan kriteria sumber daya pemilih instance yang diteruskan ke eksctl, jalankan

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

Ini akan membuat cluster dan nodegroup terkelola dengan `instanceTypes` bidang disetel ke `[c5.large, c5a.large, c5ad.large, c5d.large, t2.medium, t3.medium, t3a.medium]` (kumpulan tipe instance yang dikembalikan dapat berubah).

Untuk nodegroup yang tidak dikelola, `instancesDistribution.instanceTypes` bidang akan disetel:

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

Kriteria pemilih instance juga dapat ditentukan dalam 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
```

Opsi CLI pemilih contoh berikut didukung oleh `eksctl create cluster` dan: `eksctl create nodegroup`

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

Contoh file dapat ditemukan di sini.

## Lari Kering

Fitur dry-run memungkinkan Anda untuk memeriksa dan mengubah instance yang cocok dengan pemilih instance sebelum melanjutkan untuk membuat nodegroup.

```
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
```

Yang dihasilkan kemudian ClusterConfig dapat diteruskan ke`eksctl create cluster`:

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

`instanceSelector`Bidang yang mewakili opsi CLI juga akan ditambahkan ke ClusterConfig file untuk tujuan visibilitas dan dokumentasi. Kapan `--dry-run` dihilangkan, bidang ini akan diabaikan dan bidang akan digunakan, jika tidak, perubahan apa pun akan diganti oleh `instanceTypes` eksctl. `instanceTypes`

Saat ClusterConfig file diteruskan`--dry-run`, eksctl akan menampilkan ClusterConfig file yang berisi kumpulan nodegroup yang sama setelah memperluas kriteria sumber daya pemilih instance setiap nodegroup.

```
# 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
# ...
```

# Instans Spot

## Grup Nodegroup yang Dikelola

`eksctl`mendukung node pekerja Spot menggunakan EKS Managed Nodegroups, sebuah fitur yang memungkinkan pelanggan EKS dengan aplikasi toleran kesalahan untuk dengan mudah menyediakan dan mengelola Instans Spot EC2 untuk kluster EKS mereka. EKS Managed Nodegroup akan mengonfigurasi dan meluncurkan grup Instans Spot Penskalaan Otomatis EC2 mengikuti praktik terbaik Spot dan menguras node pekerja Spot secara otomatis sebelum instans terganggu oleh AWS. Tidak ada biaya tambahan untuk menggunakan fitur ini dan pelanggan hanya membayar untuk menggunakan sumber daya AWS, seperti Instans Spot EC2 dan volume EBS.

Untuk membuat klaster dengan nodegroup terkelola menggunakan instance Spot, berikan `--spot` tanda dan daftar opsional tipe instance:

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

Untuk membuat nodegroup terkelola menggunakan instance Spot pada klaster yang ada:

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

Untuk membuat instance Spot menggunakan nodegroup terkelola melalui file konfigurasi:

```
# 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**
>
> Nodegroup yang tidak dikelola tidak mendukung `instanceTypes` bidang `spot` and, sebagai gantinya bidang digunakan untuk mengonfigurasi instance Spot. `instancesDistribution` Lihat di bawah

## Informasi lebih lanjut

- Grup Nodegroup Titik EKS
- Jenis Kapasitas Nodegroup Terkelola EKS

# Nodegroup Tidak Dikelola

`eksctl`memiliki dukungan untuk instans spot melalui MixedInstancesPolicy untuk Grup Auto Scaling.

Berikut adalah contoh nodegroup yang menggunakan 50% instans spot dan 50% instans on demand:

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

Perhatikan bahwa `nodeGroups.X.instanceType` bidang tidak boleh disetel saat menggunakan `instancesDistribution` bidang.

Contoh ini menggunakan instance GPU:

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

Contoh ini menggunakan strategi alokasi spot yang dioptimalkan kapasitas:

```
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"
```

Contoh ini menggunakan strategi alokasi capacity-optimized-prioritized spot:

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

Gunakan strategi `capacity-optimized-prioritized` alokasi dan kemudian atur urutan jenis instance dalam daftar penggantian template peluncuran dari prioritas tertinggi ke prioritas terendah (pertama hingga terakhir dalam daftar). Amazon EC2 Auto Scaling patuh pada prioritas tipe instans berdasarkan upaya terbaik tetapi mengoptimalkan kapasitas terlebih dahulu. Ini adalah opsi yang baik untuk beban kerja di mana kemungkinan gangguan harus diminimalkan, tetapi juga preferensi untuk jenis instance tertentu penting.Untuk informasi lebih lanjut, lihat Opsi Pembelian ASG.

Perhatikan bahwa `spotInstancePools` bidang tidak boleh disetel saat menggunakan `spotAllocationStrategy` bidang. Jika tidak `spotAllocationStrategy` ditentukan, EC2 akan default untuk menggunakan `lowest-price` strategi.

Berikut adalah contoh minimal:

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

Untuk membedakan node antara instans spot atau on-demand, Anda dapat menggunakan label kubernetes `node-lifecycle` yang akan memiliki nilai `spot` atau `on-demand` tergantung pada jenisnya.

### Parameter dalam InstancesDistribution

Silakan lihat skema konfigurasi cluster untuk detailnya.

## Dukungan GPU

Eksctl mendukung pemilihan jenis instans GPU untuk nodegroups. Cukup berikan jenis instance yang kompatibel ke perintah create, atau melalui file konfigurasi.

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

> **Note**
>
> Tidak perlu lagi berlangganan AMI pasar untuk dukungan GPU di EKS.

Penyelesai AMI (`autodanauto-ssm`) akan melihat bahwa Anda ingin menggunakan jenis instans GPU dan mereka akan memilih AMI akselerasi EKS yang dioptimalkan dengan benar.

Eksctl akan mendeteksi bahwa AMI dengan tipe instans berkemampuan GPU telah dipilih dan akan menginstal plugin perangkat NVIDIA Kubernetes secara otomatis.

> **Note**
>
> Windows dan Ubuntu AMIs tidak dikirimkan dengan driver GPU yang diinstal, maka menjalankan beban kerja yang dipercepat GPU tidak akan berfungsi di luar kotak.

Untuk menonaktifkan instalasi plugin otomatis, dan menginstal versi tertentu secara manual, gunakan `--install-nvidia-plugin=false` dengan perintah create. Contoh:

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

dan, untuk versi 0.15.0 dan di atasnya,

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

atau, untuk versi yang lebih lama,

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

Instalasi plugin perangkat NVIDIA Kubernetes akan dilewati jika cluster hanya menyertakan nodegroup Bottlerocket, karena Bottlerocket sudah menangani eksekusi plugin perangkat. Jika Anda menggunakan keluarga AMI yang berbeda dalam konfigurasi klaster Anda, Anda mungkin perlu menggunakan taints dan toleransi agar plugin perangkat tidak berjalan di node Bottlerocket.

## Dukungan ARM

Topik ini mencakup cara membuat cluster dengan grup node ARM, dan cara menambahkan grup node ARM ke cluster yang ada.

EKS mendukung arsitektur ARM 64-bit dengan prosesor Graviton. Untuk membuat cluster, pilih salah satu jenis instance berbasis Graviton (`a1`,`t4g`,`m6g`,`m7g`,`m6gd`,`c6g`,`c7g`,`c6gd`,`r6g`, `r7g` `r6gd` `m8gr8g`,`c8g`) dan jalankan:

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

atau gunakan file konfigurasi:

```
---
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 juga didukung di nodegroup terkelola:

```
---
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
```

Resolver AMI, `auto` dan`auto-ssm`, akan menyimpulkan AMI yang benar berdasarkan tipe instans ARM. Hanya keluarga AmazonLinux 2023, AmazonLinux 2 dan Bottlerocket yang memiliki EKS yang dioptimalkan untuk ARM. AMIs

> **Note**
>
> ARM didukung untuk cluster dengan versi 1.15 dan lebih tinggi.

# Auto Scaling

## Aktifkan Auto Scaling

Anda dapat membuat cluster (atau nodegroup di cluster yang ada) dengan peran IAM yang memungkinkan penggunaan autoscaler cluster:

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

Bendera ini juga menetapkan `k8s.io/cluster-autoscaler/enabled` dan memberi `k8s.io/cluster-autoscaler/<clusterName>` tag, jadi penemuan nodegroup akan berfungsi.

Setelah cluster berjalan, Anda harus menginstal Cluster Autoscaler itu sendiri.

Anda juga harus menambahkan yang berikut ini ke definisi nodegroup terkelola atau tidak terkelola untuk menambahkan tag yang diperlukan untuk Cluster Autoscaler untuk menskalakan nodegroup:

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

## Meningkat dari 0

Jika Anda ingin dapat meningkatkan skala grup node Anda dari 0 dan Anda memiliki label and/or taint yang ditentukan pada nodegroup Anda, Anda harus menyebarkannya sebagai tag pada Grup Auto Scaling () Anda. ASGs

Salah satu cara untuk melakukannya adalah dengan mengatur tag ASG di `tags` bidang definisi nodegroup Anda. Misalnya, diberikan nodegroup dengan label dan taints berikut:

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

Anda perlu menambahkan tag ASG berikut:

```
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"
```

Untuk nodegroup terkelola dan tidak terkelola, ini dapat dilakukan secara otomatis dengan menyetel `propagateASGTags` ke`true`, yang akan menambahkan label dan taints sebagai tag ke grup Auto Scaling:

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

## Auto Scaling sadar zona

Jika beban kerja Anda khusus zona, Anda harus membuat nodegroup terpisah untuk setiap zona. Ini karena `cluster-autoscaler` mengasumsikan bahwa semua node dalam grup sama persis. Jadi, misalnya, jika peristiwa peningkatan skala dipicu oleh pod yang membutuhkan PVC khusus zona (misalnya volume EBS), node baru mungkin dijadwalkan di AZ yang salah dan pod akan gagal memulai.

Anda tidak memerlukan nodegroup terpisah untuk setiap AZ jika lingkungan Anda memenuhi kriteria berikut:

- Tidak ada persyaratan penyimpanan khusus zona.
- Tidak diperlukan PodAffinity dengan topologi selain host.
- Tidak diperlukan NodeAffinity pada label zona.
- Tidak ada NodeSelector pada label zona.

(Baca lebih lanjut di sini dan di sini.)

Jika Anda memenuhi semua persyaratan di atas (dan mungkin yang lain) maka Anda harus aman dengan satu nodegroup yang mencakup beberapa. AZs Jika tidak, Anda akan ingin membuat nodegroup single-AZ yang terpisah:

SEBELUM:

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

SETELAH:

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

# Dukungan AMI kustom

## Mengatur node AMI ID

`--node-ami`Bendera ini memungkinkan sejumlah kasus penggunaan lanjutan seperti menggunakan AMI khusus atau kueri AWS secara realtime untuk menentukan AMI mana yang akan digunakan. Bendera dapat digunakan untuk gambar non-GPU dan GPU.

Bendera dapat mengambil id gambar AMI untuk gambar untuk digunakan secara eksplisit. Ini juga dapat mengambil kata kunci 'khusus' berikut:

| Kata Kunci | Deskripsi |
| --- | --- |
| auto | Menunjukkan bahwa AMI yang akan digunakan untuk node harus ditemukan dengan menanyakan AWS EC2. Ini berkaitan dengan auto resolver. |
| ssm otomatis | Menunjukkan bahwa AMI yang akan digunakan untuk node harus ditemukan dengan menanyakan AWS SSM Parameter Store. |

> **Note**
>
> Saat ini, nodegroup terkelola EKS hanya mendukung Keluarga AMI berikut ketika bekerja dengan custom AMIs:`AmazonLinux2023`,,,`AmazonLinux2`, `BottlerocketUbuntu2004`, `UbuntuPro2004` dan `Ubuntu2204` `Ubuntu2404`

Saat menyetel `--node-ami` ke string ID, `eksctl` akan menganggap bahwa AMI kustom telah diminta. Untuk AmazonLinux 2 dan node Ubuntu, baik EKS dikelola dan dikelola sendiri, ini berarti itu `overrideBootstrapCommand` diperlukan. Untuk AmazonLinux tahun 2023, karena berhenti menggunakan `/etc/eks/bootstrap.sh` skrip untuk bootstrap node, yang mendukung proses inisialisasi nodeadm (untuk informasi lebih lanjut, silakan lihat dokumen bootstrap node), tidak didukung. `overrideBootstrapCommand`

Contoh bendera CLI:

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

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

Contoh file Config:

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

`--node-ami`Bendera juga dapat digunakan dengan`eksctl create nodegroup`.

## Mengatur simpul AMI Family

`--node-ami-family`Dapat mengambil kata kunci berikut:

| Kata Kunci | Deskripsi |
|---|---|
| AmazonLinux2 | Menunjukkan bahwa gambar EKS AMI berdasarkan Amazon Linux 2 harus digunakan (default). |
| AmazonLinux2023 | Menunjukkan bahwa gambar EKS AMI berdasarkan Amazon Linux 2023 harus digunakan. |
| Ubuntu2004 | Menunjukkan bahwa gambar EKS AMI berdasarkan Ubuntu 20.04 LTS (Focal) harus digunakan (didukung untuk EKS ⇐ 1.29). |
| UbuntuPro2004 | Menunjukkan bahwa gambar EKS AMI berdasarkan Ubuntu Pro 20.04 LTS (Focal) harus digunakan (tersedia untuk EKS >= 1.27, ⇐ 1.29). |
| Ubuntu2204 | Menunjukkan bahwa gambar EKS AMI berdasarkan Ubuntu 22.04 LTS (Jammy) harus digunakan (tersedia untuk EKS >= 1.29). |
| UbuntuPro2204 | Menunjukkan bahwa gambar EKS AMI berdasarkan Ubuntu Pro 22.04 LTS (Jammy) harus digunakan (tersedia untuk EKS >= 1.29). |
| Ubuntu2404 | Menunjukkan bahwa gambar EKS AMI berdasarkan Ubuntu 24.04 LTS (Noble) harus digunakan (tersedia untuk EKS >= 1.31). |

| Kata Kunci | Deskripsi |
| --- | --- |
| UbuntuPro2404 | Menunjukkan bahwa gambar EKS AMI berdasarkan Ubuntu Pro 24.04 LTS (Noble) harus digunakan (tersedia untuk EKS >= 1.31). |
| Bottlerocket | Menunjukkan bahwa gambar EKS AMI berdasarkan Bottlerocket harus digunakan. |
| WindowsServer2019 FullContainer | Menunjukkan bahwa gambar EKS AMI berdasarkan Windows Server 2019 Full Container harus digunakan. |
| WindowsServer2019 CoreContainer | Menunjukkan bahwa gambar EKS AMI berdasarkan Windows Server 2019 Core Container harus digunakan. |
| WindowsServer2022 FullContainer | Menunjukkan bahwa gambar EKS AMI berdasarkan Windows Server 2022 Full Container harus digunakan. |
| WindowsServer2022 CoreContainer | Menunjukkan bahwa gambar EKS AMI berdasarkan Windows Server 2022 Core Container harus digunakan. |

Contoh bendera CLI:

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

Contoh file Config:

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

`--node-ami-family`Bendera juga dapat digunakan dengan`eksctl create nodegroup`. `eksctl`mengharuskan AMI Family disetel secara eksplisit melalui file konfigurasi atau melalui flag `--node-ami-family` CLI, setiap kali bekerja dengan AMI khusus.

> **Note**
>
> Saat ini, nodegroup terkelola EKS hanya mendukung Keluarga AMI berikut ketika bekerja dengan custom AMIs:`AmazonLinux2023`,,,`AmazonLinux2`, `BottlerocketUbuntu2004`, `UbuntuPro2004` dan `Ubuntu2204` `Ubuntu2404`

## Dukungan AMI kustom Windows

Hanya nodegroup Windows yang dikelola sendiri yang dapat menentukan AMI kustom. `amiFamily`harus disetel ke keluarga Windows AMI yang valid.

PowerShell Variabel berikut akan tersedia untuk skrip bootstrap:

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

Contoh file Config:

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

## Dukungan AMI kustom bottlerocket

Untuk node Bottlerocket, tidak didukung. `overrideBootstrapCommand` Sebagai gantinya, untuk menunjuk wadah bootstrap mereka sendiri, seseorang harus menggunakan `bottlerocket` bidang sebagai bagian dari file konfigurasi. Misalnya

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

# Node Pekerja Windows

Dari versi 1.14, Amazon EKS mendukung Windows Nodes yang memungkinkan menjalankan wadah Windows. Selain memiliki node Windows, node Linux di cluster diperlukan untuk menjalankan CoreDNS, karena Microsoft belum mendukung mode jaringan host. Dengan demikian, cluster Windows EKS akan menjadi campuran node Windows dan setidaknya satu node Linux. Node Linux sangat penting untuk fungsi cluster, dan dengan demikian, untuk cluster tingkat produksi, disarankan untuk memiliki setidaknya dua node `t2.large` Linux untuk HA.

> **Note**
>
> Anda tidak perlu lagi menginstal pengontrol sumber daya VPC di node pekerja Linux untuk menjalankan beban kerja Windows di kluster EKS yang dibuat setelah 22 Oktober 2021. Anda dapat mengaktifkan manajemen alamat IP Windows pada bidang kontrol EKS melalui pengaturan ConfigMap (lihat tautan: eks/latest/userguide/windows -support.html untuk detailnya). eksctl akan secara otomatis menambal ConfigMap untuk mengaktifkan manajemen alamat IP Windows saat nodegroup Windows dibuat.

# Membuat cluster baru dengan dukungan Windows

Sintaks file konfigurasi memungkinkan pembuatan cluster yang berfungsi penuh dengan dukungan Windows dalam satu perintah:

```
# 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
```

Untuk membuat cluster baru dengan nodegroup Windows yang tidak dikelola tanpa menggunakan file konfigurasi, keluarkan perintah berikut:

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

## Menambahkan dukungan Windows ke cluster Linux yang ada

Untuk mengaktifkan menjalankan beban kerja Windows pada cluster yang ada dengan node Linux (keluarga `AmazonLinux2` AMI), Anda perlu menambahkan nodegroup Windows.

Dukungan BARU untuk nodegroup terkelola Windows telah ditambahkan (--managed=true atau hilangkan tanda).

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

Untuk memastikan beban kerja dijadwalkan pada OS yang tepat, mereka harus memiliki `nodeSelector` penargetan OS yang harus dijalankan:

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

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

Jika Anda menggunakan cluster yang lebih tua `1.19` dari `kubernetes.io/os` dan `kubernetes.io/arch` label perlu diganti dengan `beta.kubernetes.io/os` dan `beta.kubernetes.io/arch` masing-masing.

### Informasi lebih lanjut

- Dukungan EKS Windows

# Pemetaan Volume Tambahan

Sebagai opsi konfigurasi tambahan, saat menangani pemetaan volume, dimungkinkan untuk mengonfigurasi pemetaan tambahan saat nodegroup dibuat.

Untuk melakukan ini, atur bidang `additionalVolumes` sebagai berikut:

```
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'
```

Untuk detail selengkapnya tentang memilih Volumenames, lihat dokumentasi penamaan perangkat. Untuk mengetahui lebih lanjut tentang volume EBS, batas volume Instans, atau Blokir pemetaan perangkat, kunjungi halaman ini.

# Node Hibrida EKS

## Pengantar

Amazon EKS memperkenalkan Hybrid Nodes, fitur baru yang memungkinkan Anda menjalankan aplikasi lokal dan edge pada infrastruktur yang dikelola pelanggan dengan kluster, fitur, dan alat Amazon EKS yang sama yang Anda gunakan di AWS. Amazon EKS Hybrid Nodes menghadirkan pengalaman Kubernetes yang dikelola AWS ke lingkungan lokal bagi pelanggan untuk menyederhanakan dan menstandarisasi cara Anda menjalankan aplikasi di lingkungan lokal, edge, dan cloud. Baca lebih lanjut di EKS Hybrid Nodes.

Untuk memfasilitasi dukungan untuk fitur ini, eksctl memperkenalkan bidang tingkat atas baru yang disebut. `remoteNetworkConfig` Setiap konfigurasi terkait Node Hybrid harus diatur melalui bidang ini, sebagai bagian dari file konfigurasi; tidak ada rekan flag CLI. Selain itu, saat peluncuran, konfigurasi jaringan jarak jauh apa pun hanya dapat diatur selama pembuatan cluster dan tidak dapat diperbarui setelahnya. Ini berarti, Anda tidak akan dapat memperbarui cluster yang ada untuk menggunakan Hybrid Nodes.

`remoteNetworkConfig`Bagian dari file konfigurasi memungkinkan Anda untuk mengatur dua area inti ketika datang untuk menggabungkan node jarak jauh ke kluster EKS Anda: jaringan dan kredensional.

## Jaringan

EKS Hybrid Nodes fleksibel dengan metode pilihan Anda untuk menghubungkan jaringan lokal Anda ke VPC. Ada beberapa opsi terdokumentasi yang tersedia, termasuk AWS Site-to-Site VPN dan AWS Direct Connect, dan Anda dapat memilih metode yang paling sesuai dengan kasus penggunaan Anda. Di sebagian besar metode yang mungkin Anda pilih, VPC Anda akan dilampirkan ke gateway pribadi virtual (VGW) atau gateway transit (TGW). Jika Anda mengandalkan eksctl untuk membuat VPC untuk Anda, eksctl juga akan mengonfigurasi, dalam lingkup VPC Anda, prasyarat terkait jaringan apa pun untuk memfasilitasi komunikasi antara bidang kontrol EKS Anda dan node jarak jauh yaitu

- aturan SG masuk/keluar
- rute di tabel rute subnet pribadi
- lampiran gateway VPC ke TGW atau VGW yang diberikan

Contoh file konfigurasi:

```
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"]
```

Jika metode konektivitas pilihan Anda tidak melibatkan penggunaan TGW atau VGW, Anda tidak boleh mengandalkan eksctl untuk membuat VPC untuk Anda, dan sebagai gantinya menyediakan yang sudah ada sebelumnya. Pada catatan terkait, jika Anda menggunakan VPC yang sudah ada sebelumnya, eksctl tidak akan membuat perubahan apa pun terhadapnya, dan memastikan semua persyaratan jaringan berada di bawah tanggung jawab Anda.

> **Note**
>
> eksctl tidak menyiapkan infrastruktur jaringan apa pun di luar AWS VPC Anda (yaitu infrastruktur apa pun dari VGW/TGW jaringan jarak jauh)

## Kredensial

EKS Hybrid Nodes menggunakan AWS IAM Authenticator dan kredensi IAM sementara yang disediakan oleh AWS SSM atau AWS IAM Roles Anywhere untuk mengautentikasi dengan kluster EKS. Mirip dengan nodegroup yang dikelola sendiri, jika tidak disediakan lain, eksctl akan membuat untuk Anda Peran IAM Hybrid Nodes untuk diasumsikan oleh node jarak jauh. Selain itu, saat menggunakan IAM Roles Anywhere sebagai penyedia kredensional Anda, eksctl akan menyiapkan profil, dan mempercayai jangkar berdasarkan bundel otoritas sertifikat yang diberikan () mis. `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
```

ARN dari Peran Node Hybrid yang dibuat oleh eksctl diperlukan nanti dalam proses menggabungkan node jarak jauh Anda ke cluster, `NodeConfig` untuk mengaturnodeadm, dan membuat aktivasi (jika menggunakan SSM). Untuk mengambilnya, gunakan:

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

Demikian pula, jika menggunakan Peran IAM Di Mana Saja, Anda dapat mengambil ARN dari jangkar kepercayaan dan profil di mana saja yang dibuat oleh eksctl, mengubah perintah sebelumnya dengan mengganti dengan atau, masing-masing. `RemoteNodesRoleARN` `RemoteNodesTrustAnchorARN` `RemoteNodesAnywhereProfileARN`

Jika Anda memiliki konfigurasi IAM Roles Anywhere yang sudah ada sebelumnya, atau Anda menggunakan SSM, Anda dapat memberikan Peran IAM untuk node Hybrid melalui. `remoteNetworkConfig.iam.roleARN` Ingatlah bahwa dalam skenario ini, eksctl tidak akan membuat jangkar kepercayaan dan profil di mana pun untuk Anda. mis.

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

Untuk memetakan peran ke identitas Kubernetes dan mengotorisasi node jarak jauh untuk bergabung dengan kluster EKS, eksctl membuat entri akses dengan Peran IAM Node Hybrid sebagai ARN utama dan tipe. `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"
        ]
    }
]
```

## Dukungan add-on

Container Networking Interface (CNI): AWS VPC CNI tidak dapat digunakan dengan node hybrid. Kemampuan inti Cilium dan Calico didukung untuk digunakan dengan node hybrid. Anda dapat mengelola CNI Anda dengan alat pilihan Anda seperti Helm. Untuk informasi selengkapnya, lihat Mengkonfigurasi CNI untuk node hibrid.

> **Note**
>
> Jika Anda menginstal VPC CNI di cluster Anda untuk nodegroup yang dikelola sendiri atau yang dikelola EKS, Anda harus menggunakan `v1.19.0-eksbuild.1` atau yang lebih baru, karena ini termasuk pembaruan pada daemonset add-on untuk mengecualikannya agar tidak diinstal pada Hybrid Nodes.

### Referensi lebih lanjut

- Node Hibrida EKS UserDocs
- Pengumuman Peluncuran

## Support Node Repair Configuration untuk EKS Managed Nodegroups

EKS Managed Nodegroups mendukung Node Repair, di mana kesehatan node terkelola dipantau, dan node pekerja yang tidak sehat diganti atau di-boot ulang sebagai tanggapan. eksctl sekarang menyediakan opsi konfigurasi komprehensif untuk kontrol halus atas perilaku perbaikan node.

### Konfigurasi Perbaikan Node Dasar

#### Menggunakan bendera CLI

Untuk membuat cluster dengan nodegroup terkelola menggunakan perbaikan simpul dasar, berikan tanda: `--enable-node-repair`

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

Untuk membuat nodegroup terkelola dengan perbaikan node pada cluster yang ada:

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

#### Menggunakan file konfigurasi

```
# basic-node-repair.yaml
apiVersion: eksctl.io/v1alpha5
```

```
kind: ClusterConfig

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

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

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

# Konfigurasi Perbaikan Node yang Ditingkatkan

## Konfigurasi Ambang

Anda dapat mengonfigurasi kapan tindakan perbaikan node akan berhenti menggunakan ambang batas berbasis persentase atau hitungan. Catatan: Anda tidak dapat menggunakan ambang batas persentase dan hitungan secara bersamaan.

Bendera CLI untuk ambang batas

```
# Percentage-based threshold - repair stops when 20% of nodes are unhealthy
eksctl create cluster --enable-node-repair \
  --node-repair-max-unhealthy-percentage=20

# Count-based threshold - repair stops when 5 nodes are unhealthy
eksctl create cluster --enable-node-repair \
  --node-repair-max-unhealthy-count=5
```

File konfigurasi untuk ambang batas

```
managedNodeGroups:
- name: threshold-ng
  nodeRepairConfig:
    enabled: true
    # Stop repair actions when 20% of nodes are unhealthy
    maxUnhealthyNodeThresholdPercentage: 20
    # Alternative: stop repair actions when 3 nodes are unhealthy
    # maxUnhealthyNodeThresholdCount: 3
```

```
    # Note: Cannot use both percentage and count thresholds simultaneously
```

## Batas Perbaikan Paralel

Kontrol jumlah maksimum node yang dapat diperbaiki secara bersamaan atau paralel. Ini memberi Anda kontrol yang lebih halus atas kecepatan penggantian node. Catatan: Anda tidak dapat menggunakan batas persentase dan hitungan secara bersamaan.

Bendera CLI untuk batas paralel

```
# Percentage-based parallel limits - repair at most 15% of unhealthy nodes in parallel
eksctl create cluster --enable-node-repair \
  --node-repair-max-parallel-percentage=15

# Count-based parallel limits - repair at most 2 unhealthy nodes in parallel
eksctl create cluster --enable-node-repair \
  --node-repair-max-parallel-count=2
```

File konfigurasi untuk batas paralel

```
managedNodeGroups:
- name: parallel-ng
  nodeRepairConfig:
    enabled: true
    # Repair at most 15% of unhealthy nodes in parallel
    maxParallelNodesRepairedPercentage: 15
    # Alternative: repair at most 2 unhealthy nodes in parallel
    # maxParallelNodesRepairedCount: 2
    # Note: Cannot use both percentage and count limits simultaneously
```

## Penggantian Perbaikan Kustom

Tentukan penggantian granular untuk tindakan perbaikan tertentu. Penggantian ini mengontrol tindakan perbaikan dan waktu tunda perbaikan sebelum node dianggap memenuhi syarat untuk diperbaiki. Jika Anda menggunakan ini, Anda harus menentukan semua nilai untuk setiap penggantian.

```
managedNodeGroups:
- name: custom-repair-ng
  instanceType: g4dn.xlarge  # GPU instances
  nodeRepairConfig:
```

```
    enabled: true
    maxUnhealthyNodeThresholdPercentage: 25
    maxParallelNodesRepairedCount: 1
    nodeRepairConfigOverrides:
      # Handle GPU-related failures with immediate termination
      - nodeMonitoringCondition: "AcceleratedInstanceNotReady"
        nodeUnhealthyReason: "NvidiaXID13Error"
        minRepairWaitTimeMins: 10
        repairAction: "Terminate"
      # Handle network issues with restart after waiting
      - nodeMonitoringCondition: "NetworkNotReady"
        nodeUnhealthyReason: "InterfaceNotUp"
        minRepairWaitTimeMins: 20
        repairAction: "Restart"
```

## Contoh Konfigurasi Lengkap

Untuk contoh komprehensif dengan semua opsi konfigurasi, lihat examples/44-node-repair.yaml.

### Contoh 1: Perbaikan dasar dengan ambang persentase

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

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

managedNodeGroups:
- name: basic-ng
  instanceType: m5.large
  desiredCapacity: 3
  nodeRepairConfig:
    enabled: true
    maxUnhealthyNodeThresholdPercentage: 20
    maxParallelNodesRepairedPercentage: 15
```

### Contoh 2: Perbaikan konservatif untuk beban kerja kritis

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

```
metadata:
  name: critical-workload-cluster
  region: us-west-2

managedNodeGroups:
- name: critical-ng
  instanceType: c5.2xlarge
  desiredCapacity: 6
  nodeRepairConfig:
    enabled: true
    # Very conservative settings
    maxUnhealthyNodeThresholdPercentage: 10
    maxParallelNodesRepairedCount: 1
    nodeRepairConfigOverrides:
      # Wait longer before taking action on critical workloads
      - nodeMonitoringCondition: "NetworkNotReady"
        nodeUnhealthyReason: "InterfaceNotUp"
        minRepairWaitTimeMins: 45
        repairAction: "Restart"
```

## Contoh 3: Beban kerja GPU dengan perbaikan khusus

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

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

managedNodeGroups:
- name: gpu-ng
  instanceType: g4dn.xlarge
  desiredCapacity: 4
  nodeRepairConfig:
    enabled: true
    maxUnhealthyNodeThresholdPercentage: 25
    maxParallelNodesRepairedCount: 1
    nodeRepairConfigOverrides:
      # GPU failures require immediate termination
      - nodeMonitoringCondition: "AcceleratedInstanceNotReady"
        nodeUnhealthyReason: "NvidiaXID13Error"
        minRepairWaitTimeMins: 5
        repairAction: "Terminate"
```

# Referensi CLI

## Bendera Perbaikan Node

| Bendera | Deskripsi | Contoh |
|---|---|---|
| `--enable-node-repair` | Aktifkan perbaikan node otomatis | `--enable-node-repair` |
| `--node-repair-max-unhealthy-percentage` | Persentase maksimum node yang tidak sehat sebelum diperbaiki | `--node-repair-max-unhealthy-percentage=20` |
| `--node-repair-max-unhealthy-count` | Jumlah maksimum node yang tidak sehat sebelum diperbaiki | `--node-repair-max-unhealthy-count=5` |
| `--node-repair-max-parallel-percentage` | Persentase maksimum node untuk diperbaiki secara paralel | `--node-repair-max-parallel-percentage=15` |
| `--node-repair-max-parallel-count` | Jumlah maksimum node untuk diperbaiki secara paralel | `--node-repair-max-parallel-count=2` |

Catatan: Penggantian konfigurasi perbaikan node hanya didukung melalui file konfigurasi YAMB karena kompleksitasnya.

# Referensi Konfigurasi

## nodeRepairConfig

| Bidang | Tipe | Deskripsi | Batasan | Contoh |
|---|---|---|---|---|
| `enabled` | boolean | Aktifkan/nonaktifkan perbaikan node | - | `true` |

| Bidang | Tipe | Deskripsi | Batasan | Contoh |
| --- | --- | --- | --- | --- |
| `maxUnhealthyNodeThresholdPercentage` | integer | Ambang persentase node yang tidak sehat, di atasnya tindakan perbaikan otomatis node akan berhenti | Tidak dapat digunakan dengan `maxUnhealthyNodeThresholdCount` | `20` |
| `maxUnhealthyNodeThresholdCount` | integer | Hitung ambang node yang tidak sehat, di atasnya tindakan perbaikan otomatis node akan berhenti | Tidak dapat digunakan dengan `maxUnhealthyNodeThresholdPercentage` | `5` |
| `maxParallelNodesRepairedPercentage` | integer | Persentase maksimum node tidak sehat yang dapat diperbaiki secara bersamaan atau paralel | Tidak dapat digunakan dengan `maxParallelNodesRepairedCount` | `15` |
| `maxParallelNodesRepairedCount` | integer | Jumlah maksimum node tidak sehat yang dapat diperbaiki secara bersamaan atau paralel | Tidak dapat digunakan dengan `maxParallelNodesRepairedPercentage` | `2` |

| Bidang | Tipe | Deskripsi | Batasan | Contoh |
|---|---|---|---|---|
| `nodeRepairConfigOverrides` | array | Penggantian granular untuk tindakan perbaikan spesifik yang mengendalikan tindakan perbaikan dan waktu tunda | Semua nilai harus ditentukan untuk setiap penggantian | Lihat contoh di atas |

### nodeRepairConfigMengesampingkan

| Bidang | Tipe | Deskripsi | Nilai Valid |
|---|---|---|---|
| `nodeMonitoringCondition` | string | Kondisi tidak sehat yang dilaporkan oleh agen pemantau simpul yang berlaku untuk penggantian ini | `"AcceleratedInstanceNotReady"` , `"NetworkNotReady"` |
| `nodeUnhealthyReason` | string | Alasan yang dilaporkan oleh agen pemantau simpul bahwa penggantian ini berlaku | `"NvidiaXID13Error"` , `"InterfaceNotUp"` |
| `minRepairWaitTimeMins` | integer | Waktu minimum dalam hitungan menit untuk menunggu sebelum mencoba memperbaiki node dengan kondisi dan alasan yang ditentukan | Setiap bilangan bulat positif |
| `repairAction` | string | Tindakan perbaikan yang harus diambil untuk node ketika semua kondisi yang ditentukan terpenuhi | `"Terminate"` , `"Restart"` , `"NoAction"` |

## Informasi lebih lanjut

- EKS Managed Nodegroup Node Health

# Jaringan

Bab ini mencakup informasi tentang bagaimana Eksctl membuat jaringan Virtual Private Cloud (VPC) untuk kluster EKS.

## Topik:

- the section called “Konfigurasi VPC”
  - Ubah rentang CIDR VPC dan konfigurasikan pengalamatan IPv6
  - Gunakan VPC yang sudah ada
  - Sesuaikan VPC, subnet, grup keamanan, dan gateway NAT untuk cluster EKS baru
- the section called “Pengaturan Subnet”
  - Gunakan subnet pribadi untuk nodegroup awal untuk mengisolasinya dari internet publik
  - Kustomisasi topologi subnet dengan mencantumkan beberapa subnet per zona ketersediaan dan menentukan subnet dalam konfigurasi nodegroup
  - Batasi nodegroup ke subnet bernama tertentu dalam konfigurasi VPC
  - Saat menggunakan subnet pribadi untuk nodegroup, atur ke `privateNetworking` `true`
  - Berikan spesifikasi subnet lengkap dengan keduanya `public` dan `private` konfigurasi dalam spesifikasi VPC
  - Hanya satu dari `subnets` atau `availabilityZones` dapat disediakan dalam konfigurasi nodegroup
- the section called “Akses Cluster”
  - Mengelola akses publik dan pribadi ke endpoint server API Kubernetes di klaster EKS
  - Batasi akses ke titik akhir API publik EKS Kubernetes dengan menentukan rentang CIDR yang diizinkan
  - Memperbarui konfigurasi akses titik akhir server API dan pembatasan CIDR akses publik untuk klaster yang ada
- the section called “Kontrol jaringan pesawat”
  - Perbarui subnet yang digunakan oleh bidang kontrol EKS untuk sebuah cluster
- the section called “IPv6 Support”
  - Tentukan versi IP (IPv4 atau IPv6) yang akan digunakan saat membuat VPC dengan cluster EKS

# Konfigurasi VPC

## VPC Khusus untuk Cluster

Secara default `eksctl create cluster` akan membuat VPC khusus untuk cluster. Ini dilakukan untuk menghindari gangguan dengan sumber daya yang ada karena berbagai alasan, termasuk keamanan, tetapi juga karena sulit untuk mendeteksi semua pengaturan di VPC yang ada.

- Default VPC CIDR yang digunakan oleh adalah. `eksctl` `192.168.0.0/16`
  - Ini dibagi menjadi 8 (`/19`) subnet (3 pribadi, 3 publik & 2 cadangan).
- Nodegroup awal dibuat di subnet publik.
- Akses SSH dinonaktifkan kecuali `--allow-ssh` ditentukan.
- Nodegroup secara default memungkinkan lalu lintas masuk dari grup keamanan pesawat kontrol pada port 1025 - 65535.

> **Note**
>
> Di `us-east-1` eksctl hanya membuat 2 subnet publik dan 2 pribadi secara default.

## Ubah VPC CIDR

Jika Anda perlu mengatur peering dengan VPC lain, atau hanya membutuhkan rentang IPs yang lebih besar atau lebih kecil, Anda dapat `--vpc-cidr` menggunakan flag untuk mengubahnya. Silakan merujuk ke dokumen AWS untuk panduan tentang memilih blok CIDR yang diizinkan untuk digunakan dalam VPC.

Jika Anda membuat IPv6 cluster, Anda dapat mengonfigurasi `VPC.IPv6Cidr` dalam file konfigurasi cluster. Pengaturan ini hanya ada di file konfigurasi, bukan di flag CLI.

Jika Anda memiliki blok alamat IPv6 IP, Anda juga dapat membawa IPv6 kolam Anda sendiri. Lihat Membawa alamat IP Anda sendiri (BYOIP) ke Amazon EC2 tentang cara mengimpor kolam Anda sendiri. Kemudian gunakan file konfigurasi `VPC.IPv6Cidr` in the cluster untuk mengkonfigurasi Eksctl.

## Gunakan VPC yang ada: dibagikan dengan kops

Anda dapat menggunakan VPC dari klaster Kubernetes yang sudah ada yang dikelola oleh kops.
Fitur ini disediakan untuk memfasilitasi migrasi and/or cluster peering.

Jika sebelumnya Anda telah membuat cluster dengan kops, misalnya menggunakan perintah yang mirip dengan ini:

```
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
```

Anda dapat membuat cluster EKS dalam hal yang sama AZs menggunakan subnet VPC yang sama (CATATAN: setidaknya 2 AZs/subnets diperlukan):

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

## Gunakan VPC yang ada: konfigurasi khusus lainnya

`eksctl`menyediakan beberapa, tetapi tidak lengkap, fleksibilitas untuk VPC kustom dan topologi subnet.

Anda dapat menggunakan VPC yang ada dengan memasok subnet and/or publik pribadi menggunakan dan flag. `--vpc-private-subnets` `--vpc-public-subnets` Terserah Anda untuk memastikan subnet yang Anda gunakan dikategorikan dengan benar, karena tidak ada cara sederhana untuk memverifikasi apakah subnet benar-benar pribadi atau publik, karena konfigurasi bervariasi.

Mengingat flag ini, `eksctl create cluster` akan menentukan ID VPC secara otomatis, tetapi tidak akan membuat tabel routing atau sumber daya lainnya, seperti gateway. internet/NAT Namun, ini akan membuat grup keamanan khusus untuk nodegroup awal dan pesawat kontrol.

Anda harus memastikan untuk menyediakan setidaknya 2 subnet yang berbeda AZs dan kondisi ini diperiksa oleh EKS. Jika Anda menggunakan VPC yang ada, persyaratan berikut tidak diberlakukan atau diperiksa oleh EKS atau Eksctl dan EKS membuat cluster. Beberapa fungsi dasar cluster bekerja tanpa persyaratan ini. (Misalnya, penandaan tidak sepenuhnya diperlukan, pengujian telah menunjukkan bahwa dimungkinkan untuk membuat cluster fungsional tanpa tag yang ditetapkan pada subnet, namun tidak ada jaminan bahwa ini akan selalu berlaku dan penandaan direkomendasikan.)

Persyaratan standar:

- semua subnet yang diberikan harus dalam VPC yang sama, dalam blok yang sama IPs
- jumlah alamat IP yang cukup tersedia, berdasarkan kebutuhan
- jumlah subnet yang cukup (minimal 2), berdasarkan kebutuhan
- subnet ditandai dengan setidaknya yang berikut:
  - `kubernetes.io/cluster/<name>`tag disetel ke salah satu `shared` atau `owned`
  - `kubernetes.io/role/internal-elb`tag disetel ke 1 untuk subnet pribadi
  - `kubernetes.io/role/elb`tag disetel ke 1 subnet publik
- gateway and/or NAT internet yang dikonfigurasi dengan benar
- tabel routing memiliki entri yang benar dan jaringan berfungsi
- BARU: semua subnet publik harus `MapPublicIpOnLaunch` mengaktifkan properti (yaitu `Auto-assign public IPv4 address` di konsol AWS). Grup node terkelola dan Fargate tidak menetapkan IPv4 alamat publik, properti harus disetel di subnet.

Mungkin ada persyaratan lain yang diberlakukan oleh EKS atau Kubernetes, dan sepenuhnya terserah Anda untuk tetap up-to-date pada persyaratan apa pun. and/or recommendations, and implement those as needed/possible

Pengaturan grup keamanan default yang diterapkan oleh `eksctl` mungkin atau mungkin tidak cukup untuk berbagi akses dengan sumber daya di grup keamanan lain. Jika Anda ingin mengubah ingress/egress aturan grup keamanan, Anda mungkin perlu menggunakan alat lain untuk mengotomatiskan perubahan, atau melakukannya melalui konsol EC2.

Jika ragu, jangan gunakan VPC khusus. Menggunakan `eksctl create cluster` tanpa `--vpc-*` flag apa pun akan selalu mengonfigurasi cluster dengan VPC khusus yang berfungsi penuh.

Contoh

Buat cluster menggunakan VPC kustom dengan 2x subnet pribadi dan 2x publik:

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

atau gunakan file konfigurasi setara berikut:

```
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
```

Buat cluster menggunakan VPC kustom dengan 3x subnet pribadi dan buat nodegroup awal menggunakan subnet tersebut:

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

atau gunakan file konfigurasi setara berikut:

```
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
```

Buat cluster menggunakan subnet publik VPC 4x kustom:

```
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
```

Contoh lainnya dapat ditemukan di `examples` folder repo:

- menggunakan VPC yang ada
- menggunakan VPC CIDR khusus

## Grup Keamanan Node Bersama Kustom

`eksctl`akan membuat dan mengelola grup keamanan node bersama yang memungkinkan komunikasi antara node yang tidak dikelola dan bidang kontrol cluster dan node terkelola.

Jika Anda ingin menyediakan grup keamanan kustom Anda sendiri, Anda dapat mengganti `sharedNodeSecurityGroup` bidang di file konfigurasi:

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

Secara default, saat membuat cluster, `eksctl` akan menambahkan aturan ke grup keamanan ini untuk memungkinkan komunikasi ke dan dari grup keamanan cluster default yang dibuat EKS. Grup keamanan klaster default digunakan oleh bidang kontrol EKS dan grup node terkelola.

Jika Anda ingin mengelola sendiri aturan grup keamanan, Anda dapat `eksctl` mencegah pembuatan aturan dengan `manageSharedNodeSecurityGroupRules` menyetelnya `false` di file konfigurasi:

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

## Gerbang NAT

NAT Gateway untuk cluster dapat dikonfigurasi menjadi`Disable`, `Single` (default) atau`HighlyAvailable`. `HighlyAvailable`Opsi ini akan menggunakan Gateway NAT di setiap Availability Zone Wilayah, sehingga jika AZ sedang down, node di yang lain AZs masih dapat berkomunikasi ke Internet.

Itu dapat ditentukan melalui flag `--vpc-nat-mode` CLI atau dalam file konfigurasi cluster seperti contoh di bawah ini:

```
vpc:
  nat:
```

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

Lihat contoh lengkapnya di sini.

> **Note**
>
> Menentukan NAT Gateway hanya didukung selama pembuatan klaster. Itu tidak disentuh selama peningkatan cluster.

# Pengaturan Subnet

## Gunakan subnet pribadi untuk nodegroup awal

Jika Anda lebih suka mengisolasi nodegroup awal dari internet publik, Anda dapat menggunakan bendera. `--node-private-networking` Ketika digunakan bersama dengan `--ssh-access` bendera, port SSH hanya dapat diakses dari dalam VPC.

> **Note**
>
> Menggunakan `--node-private-networking` bendera akan menghasilkan lalu lintas keluar untuk melewati gateway NAT menggunakan IP Elastisnya. Di sisi lain, jika node berada di subnet publik, lalu lintas keluar tidak akan melalui gateway NAT dan karenanya lalu lintas keluar memiliki IP dari setiap node individu.

## Topologi subnet kustom

`eksctl`versi `0.32.0` memperkenalkan kustomisasi topologi subnet lebih lanjut dengan kemampuan untuk:

- Daftar beberapa subnet per AZ dalam konfigurasi VPC
- Tentukan subnet dalam konfigurasi nodegroup

Dalam versi sebelumnya, subnet khusus harus disediakan oleh zona ketersediaan, yang berarti hanya satu subnet per AZ yang dapat dicantumkan. Dari kunci `0.32.0` pengidentifikasi bisa sewenang-wenang.

```
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**

Jika menggunakan AZ sebagai kunci pengidentifikasi, `az` nilainya dapat dihilangkan.

Jika menggunakan string arbitrer sebagai kunci identifikasi, seperti di atas, baik:

- `id`harus diatur (`az`dan `cidr` opsional)
- atau `az` harus diatur (`cidr`opsional)

Jika pengguna menentukan subnet oleh AZ tanpa menentukan CIDR dan ID, subnet di AZ tersebut akan dipilih dari VPC, secara sewenang-wenang jika ada beberapa subnet tersebut.

**Note**

Spesifikasi subnet lengkap harus disediakan, yaitu keduanya `public` dan `private` konfigurasi yang dideklarasikan dalam spesifikasi VPC.

Nodegroups dapat dibatasi untuk subnet bernama melalui konfigurasi. Saat menentukan subnet pada konfigurasi nodegroup, gunakan kunci identifikasi seperti yang diberikan dalam spesifikasi VPC bukan id subnet. Contoh:

```
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**
>
> Hanya satu dari `subnets` atau `availabilityZones` dapat disediakan dalam konfigurasi nodegroup.

Saat menempatkan nodegroups di dalam subnet pribadi, `privateNetworking` harus disetel ke `true` pada nodegroup:

```
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
```

Lihat 24-nodegroup-subnets.yaml di repo eksctl untuk contoh konfigurasi lengkap. GitHub

# Akses Cluster

## Mengelola Akses ke Endpoint Kubernetes API Server

Secara default, kluster EKS mengekspos server API Kubernetes secara publik tetapi tidak langsung dari dalam subnet VPC (public=true, private=false). Lalu lintas yang ditujukan untuk server API dari dalam VPC harus terlebih dahulu keluar dari jaringan VPC (tetapi bukan jaringan Amazon) dan kemudian masuk kembali untuk mencapai server API.

Akses endpoint server API Kubernetes untuk klaster dapat dikonfigurasi untuk akses publik dan pribadi saat membuat klaster menggunakan file konfigurasi klaster. Contoh di bawah ini:

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

Ada beberapa peringatan tambahan saat mengonfigurasi akses endpoint API Kubernetes:

1. EKS tidak mengizinkan cluster tanpa akses pribadi atau publik diaktifkan.
2. EKS memang memungkinkan pembuatan konfigurasi yang hanya memungkinkan akses pribadi untuk diaktifkan, tetapi eksctl tidak mendukungnya selama pembuatan cluster karena mencegah eksctl untuk dapat bergabung dengan node pekerja ke cluster.
3. Memperbarui klaster agar hanya memiliki akses endpoint API Kubernetes pribadi berarti bahwa perintah Kubernetes, secara default, (misalnya`kubectl`) dan juga `eksctl delete clustereksctl utils write-kubeconfig`, dan mungkin perintah tersebut `eksctl utils update-kube-proxy` harus dijalankan di dalam VPC klaster.
   - Ini memerlukan beberapa perubahan pada berbagai sumber daya AWS. Untuk informasi selengkapnya, lihat Titik akhir server API Cluster.
   - Anda dapat menyediakan `vpc.extraCIDRs` yang akan menambahkan rentang CIDR tambahan ke ControlPlaneSecurityGroup, memungkinkan subnet di luar VPC mencapai titik akhir API kubernetes. Demikian pula Anda dapat memberikan `vpc.extraIPv6CIDRs` untuk menambahkan rentang IPv6 CIDR juga.

Berikut ini adalah contoh bagaimana seseorang dapat mengonfigurasi akses endpoint API Kubernetes menggunakan sub-perintah: `utils`

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

Untuk memperbarui pengaturan menggunakan **`ClusterConfig`** file, gunakan:

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

Perhatikan bahwa jika Anda tidak melewati bendera, itu akan mempertahankan nilai saat ini. Setelah Anda puas dengan perubahan yang diusulkan, tambahkan `approve` bendera untuk membuat perubahan ke cluster yang sedang berjalan.

## Membatasi Akses ke titik akhir API Publik EKS Kubernetes

Pembuatan default cluster EKS mengekspos server API Kubernetes secara publik.

Fitur ini hanya berlaku untuk titik akhir publik. Opsi konfigurasi akses titik akhir server API tidak akan berubah, dan Anda masih memiliki opsi untuk menonaktifkan titik akhir publik sehingga cluster Anda tidak dapat diakses dari internet. (Sumber: https://github.com/aws/ containers-roadmap/issues/108 #issuecomment -552766489)

Untuk membatasi akses ke titik akhir API publik ke satu set CIDRs saat membuat klaster, setel bidang: **`publicAccessCIDRs`**

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

Untuk memperbarui pembatasan pada klaster yang ada, gunakan:

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

Untuk memperbarui pembatasan menggunakan **`ClusterConfig`** file, atur yang baru CIDRs masuk **`vpc.publicAccessCIDRs`** dan jalankan:

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

> **Important**
>
> Jika pengaturan `publicAccessCIDRs` dan pembuatan grup simpul `privateAccess` harus disetel ke `true` atau node IPs 'harus ditambahkan ke daftar. `publicAccessCIDRs`

Jika node tidak dapat mengakses titik akhir API cluster karena akses terbatas, pembuatan cluster akan gagal `context deadline exceeded` karena node tidak dapat mengakses titik akhir publik dan gagal bergabung dengan cluster.

Untuk memperbarui akses endpoint server API dan akses publik CIDRs untuk klaster dalam satu perintah, jalankan:

```
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
```

Untuk memperbarui pengaturan menggunakan file konfigurasi:

```
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
```

# Memperbarui subnet pesawat kontrol dan kelompok keamanan

Dokumentasi ini menjelaskan cara memodifikasi konfigurasi jaringan bidang kontrol kluster EKS Anda setelah pembuatan awal. Ini termasuk memperbarui subnet pesawat kontrol dan kelompok keamanan.

## Memperbarui subnet bidang kontrol

Ketika sebuah cluster dibuat dengan eksctl, satu set subnet publik dan pribadi dibuat dan diteruskan ke EKS API. EKS membuat 2 hingga 4 cross-account elastic network interface (ENIs) di subnet tersebut untuk memungkinkan komunikasi antara bidang kontrol Kubernetes yang dikelola EKS dan VPC Anda.

Untuk memperbarui subnet yang digunakan oleh bidang kontrol EKS, jalankan:

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

Untuk memperbarui pengaturan menggunakan file konfigurasi:

```
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
```

Tanpa `--approve` bendera, eksctl hanya mencatat perubahan yang diusulkan. Setelah Anda puas dengan perubahan yang diusulkan, jalankan kembali perintah dengan `--approve` bendera.

## Memperbarui kelompok keamanan pesawat kontrol

Untuk mengelola lalu lintas antara bidang kontrol dan node pekerja, EKS mendukung melewati grup keamanan tambahan yang diterapkan ke antarmuka jaringan lintas akun yang disediakan oleh EKS. Untuk memperbarui grup keamanan untuk pesawat kontrol EKS, jalankan:

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

Untuk memperbarui pengaturan menggunakan file konfigurasi:

```
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
```

Untuk memperbarui subnet bidang kontrol dan grup keamanan untuk klaster, jalankan:

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

Untuk memperbarui kedua bidang menggunakan file konfigurasi:

```
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
```

Untuk contoh lengkap, lihat cluster-subnets-sgs.yaml.

Tanpa `--approve` bendera, eksctl hanya mencatat perubahan yang diusulkan. Setelah Anda puas dengan perubahan yang diusulkan, jalankan kembali perintah dengan `--approve` bendera.

# IPv6 Support

## Tentukan Keluarga IP

Saat `eksctl` membuat vpc, Anda dapat menentukan versi IP yang akan digunakan. Opsi berikut tersedia untuk dikonfigurasi:

- IPv4
- IPv6

Nilai default-nya adalah `IPv4`.

Untuk mendefinisikannya, gunakan contoh berikut:

```
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**
>
> Pengaturan ini hanya ada di file konfigurasi, bukan di flag CLI.

Jika Anda menggunakan IPv6, Anda harus mengkonfigurasi persyaratan berikut:

- OIDC diaktifkan
- addons terkelola didefinisikan sebagai pertunjukan di atas
- versi cluster harus => 1.21
- vpc-cni versi addon harus => 1.10.0
- nodegroup yang dikelola sendiri tidak didukung dengan cluster IPv6
- nodegroup terkelola tidak didukung dengan cluster yang tidak dimiliki IPv6
- `vpc.nat`dan `serviceIPv4CIDR` bidang dibuat oleh eksctl untuk cluster ipv6 dan tidak didukung opsi konfigurasi
- AutoAllocateIPv6 tidak didukung bersama dengan IPv6

- Untuk IPv6 cluster, peran IAM untuk vpc-cni harus mewajibkan kebijakan IAM untuk mode terkait IPv6

Jaringan pribadi dapat dilakukan dengan keluarga IPv6 IP juga. Silakan ikuti instruksi yang diuraikan di bawah EKS Private Cluster.

# IAM

Bab ini mencakup informasi tentang bekerja dengan AWS IAM.

## Topik:

- the section called “Kelola pengguna dan peran IAM”
  - Mengelola pengguna IAM dan pemetaan peran untuk mengontrol akses ke kluster EKS
  - Konfigurasikan pemetaan identitas IAM melalui file konfigurasi cluster atau perintah CLI
- the section called “Peran IAM untuk Akun Layanan”
  - Kelola izin berbutir halus untuk aplikasi yang berjalan di Amazon EKS yang menggunakan layanan AWS lainnya
  - Buat dan konfigurasikan pasangan IAM Roles dan Kubernetes Service Account menggunakan eksctl
  - Aktifkan Penyedia OpenID Connect IAM untuk klaster EKS untuk mengaktifkan Peran IAM untuk Akun Layanan
- the section called “Batas izin IAM”
  - Kontrol izin maksimum yang diberikan kepada entitas IAM (pengguna atau peran) dengan menetapkan batas izin
- the section called “Asosiasi Identitas EKS Pod”
  - Konfigurasikan izin IAM untuk add-on EKS menggunakan asosiasi identitas pod yang direkomendasikan
  - Aktifkan aplikasi Kubernetes untuk menerima izin IAM yang diperlukan untuk terhubung dengan layanan AWS di luar klaster
  - Sederhanakan proses otomatisasi peran IAM dan akun layanan di beberapa kluster EKS
- the section called “Kebijakan IAM”
  - Kelola kebijakan IAM untuk grup node EKS, termasuk dukungan untuk berbagai kebijakan add-on seperti pembuat gambar, auto scaler, DNS eksternal, manajer sertifikat, dan banyak lagi.
  - Lampirkan peran instance kustom atau kebijakan sebaris ke grup node untuk izin tambahan.
  - Lampirkan kebijakan AWS tertentu yang dikelola oleh ARN ke grup node, memastikan kebijakan yang diperlukan seperti Amazon dan EKSWorker NodePolicy Amazoneks_CNI_Policy disertakan.

- the section called “Kebijakan IAM minimum”
  - Mengelola sumber daya Amazon EC2, termasuk penyeimbang beban, grup auto-scaling, dan pemantauan CloudWatch
  - Membuat dan mengelola CloudFormation tumpukan AWS
  - Kelola klaster Amazon Elastic Kubernetes Service (EKS), grup node, dan sumber daya terkait seperti peran dan kebijakan IAM

# Kebijakan IAM minimum

Dokumen ini menjelaskan kebijakan IAM minimum yang diperlukan untuk menjalankan kasus penggunaan utama eksctl. Ini adalah yang digunakan untuk menjalankan tes integrasi.

> **Note**
>
> Ingatlah untuk mengganti `<account_id>` dengan milik Anda sendiri.

> **Note**
>
> Kebijakan AWS Managed dibuat dan dikelola oleh AWS. Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan terkelola AWS.

Amazon EC2 FullAccess (Kebijakan Terkelola AWS)

Lihat definisi EC2 FullAccess kebijakan Amazon.

AWSCloudFormationFullAccess (Kebijakan Terkelola AWS)

Lihat definisi AWSCloud FormationFullAccess kebijakan.

EksAllAccess

```
# Error: No files found with UUID: 27ad3ff9-60be-4128-8b83-f8833a6e39aa
```

IamLimitedAccess

```
# Error: No files found with UUID: 5500eeb9-bf3d-498d-999b-7f8036e705a5
```

# Batas izin IAM

Batas izin adalah fitur AWS IAM lanjutan di mana izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM telah ditetapkan; di mana entitas tersebut adalah pengguna atau peran. Ketika batas izin ditetapkan untuk entitas, entitas tersebut hanya dapat melakukan tindakan yang diizinkan oleh kebijakan berbasis identitas dan batas izinnya.

Anda dapat memberikan batas izin sehingga semua entitas berbasis identitas yang dibuat oleh eksctl dibuat dalam batas tersebut. Contoh ini menunjukkan bagaimana batas izin dapat diberikan ke berbagai entitas berbasis identitas yang dibuat oleh eksctl:

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

metadata:
  name: cluster-17
  region: us-west-2

iam:
  withOIDC: true
  serviceRolePermissionsBoundary: "arn:aws:iam::11111:policy/entity/boundary"
  fargatePodExecutionRolePermissionsBoundary: "arn:aws:iam::11111:policy/entity/
boundary"
  serviceAccounts:
    - metadata:
        name: s3-reader
      attachPolicyARNs:
      - "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
      permissionsBoundary: "arn:aws:iam::11111:policy/entity/boundary"

nodeGroups:
  - name: "ng-1"
    desiredCapacity: 1
    iam:
      instanceRolePermissionsBoundary: "arn:aws:iam::11111:policy/entity/boundary"
```

> **Warning**
>
> Hal ini tidak mungkin untuk memberikan kedua peran ARN dan batas izin.

## Mengatur Batas Izin VPC CNI

Harap dicatat bahwa ketika Anda membuat cluster dengan OIDC diaktifkan eksctl akan secara otomatis membuat untuk VPC-CNI `iamserviceaccount` untuk alasan keamanan. Jika Anda ingin menambahkan batas izin ke dalamnya maka Anda harus menentukan `iamserviceaccount` dalam file konfigurasi Anda secara manual:

```
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"
```

# Kebijakan IAM

Anda dapat melampirkan Peran Instance ke grup node. Beban kerja yang berjalan pada node akan menerima izin IAM dari node. Untuk informasi selengkapnya, lihat Peran IAM untuk Amazon EC2.

Halaman ini mencantumkan templat kebijakan IAM yang telah ditentukan sebelumnya yang tersedia di eksctl. Template ini menyederhanakan proses pemberian izin layanan AWS kepada node EKS Anda tanpa harus membuat kebijakan IAM khusus secara manual.

## Kebijakan add-on IAM yang didukung

Contoh semua kebijakan add-on yang didukung:

```
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
```

### Kebijakan Image Builder

`imageBuilder`Kebijakan ini memungkinkan akses penuh ECR (Elastic Container Registry). Ini berguna untuk membangun, misalnya, server CI yang perlu mendorong gambar ke ECR.

### Kebijakan EBS

`ebs`Kebijakan ini memungkinkan driver EBS CSI (Elastic Block Store Container Storage Interface) yang baru.

### Kebijakan Manajer Cert

`certManager`Kebijakan ini memungkinkan kemampuan untuk menambahkan catatan ke Route 53 untuk menyelesaikan DNS01 tantangan. Informasi lebih lanjut dapat ditemukan di sini.

## Menambahkan peran instance khusus

Contoh ini membuat nodegroup yang menggunakan kembali Peran Instans IAM yang ada dari cluster lain:

```
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"
```

## Melampirkan kebijakan inline

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

## Melampirkan kebijakan oleh ARN

```
nodeGroups:
  - name: my-special-nodegroup
    iam:
      attachPolicyARNs:
        - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
        - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
        - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
        - arn:aws:iam::aws:policy/ElasticLoadBalancingFullAccess
        - arn:aws:iam::1111111111:policy/kube2iam
      withAddonPolicies:
        autoScaler: true
        imageBuilder: true
```

> **Warning**
>
> Jika nodegroup menyertakan nodegroup `attachPolicyARNs` itu juga harus menyertakan kebijakan node default, seperti`AmazonEKSWorkerNodePolicy`, `AmazonEKS_CNI_Policy` dan `AmazonEC2ContainerRegistryPullOnly` dalam contoh ini.

# Kelola pengguna dan peran IAM

> **Note**
>
> AWS menyarankan untuk bermigrasi ke the section called “Asosiasi Identitas EKS Pod” dari. `aws-auth` ConfigMap

Kluster EKS menggunakan pengguna dan peran IAM untuk mengontrol akses ke cluster. Aturan diterapkan di peta konfigurasi

## Edit ConfigMap dengan Perintah CLI

disebut`aws-auth`. `eksctl`menyediakan perintah untuk membaca dan mengedit peta konfigurasi ini.

Dapatkan semua pemetaan identitas:

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

Dapatkan semua pemetaan identitas yang cocok dengan arn:

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

Buat pemetaan identitas:

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

Menghapus pemetaan identitas:

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

> **Note**
>
> Perintah di atas menghapus satu pemetaan FIFO kecuali `--all` diberikan dalam hal ini menghapus semua pencocokan. Akan memperingatkan jika lebih banyak pemetaan yang cocok dengan peran ini ditemukan.

Buat pemetaan akun:

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

Hapus pemetaan akun:

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

## Edit ConfigMap menggunakan ClusterConfig file

Pemetaan identitas juga dapat ditentukan dalam: 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
```

# Peran IAM untuk Akun Layanan

> **Tip**
>
> `eksctl`mendukung konfigurasi izin berbutir halus ke aplikasi yang menjalankan EKS melalui Asosiasi Identitas Pod EKS

Amazon EKS mendukung Roles for Service Accounts (IRSA)] yang memungkinkan operator klaster memetakan Peran AWS IAM ke Akun Layanan Kubernetes.

Ini memberikan manajemen izin berbutir halus untuk aplikasi yang berjalan di EKS dan menggunakan layanan AWS lainnya. Ini bisa berupa aplikasi yang menggunakan S3, layanan data lainnya (RDS, MQ, STS, DynamoDB), atau komponen Kubernetes seperti AWS Load Balancer controller atau ExternalDNS.

Anda dapat dengan mudah membuat pasangan Akun Peran dan Layanan IAM. `eksctl`

> **Note**
>
> Jika Anda menggunakan peran instance, dan mempertimbangkan untuk menggunakan IRSA sebagai gantinya, Anda tidak boleh mencampur keduanya.

## Cara kerjanya

Ia bekerja melalui IAM OpenID Connect Provider (OIDC) yang mengekspos EKS, dan Peran IAM harus dibangun dengan mengacu pada Penyedia IAM OIDC (khusus untuk klaster EKS tertentu), dan referensi ke Akun Layanan Kubernetes yang akan terikat padanya. Setelah Peran IAM dibuat, akun layanan harus menyertakan ARN dari peran tersebut sebagai anotasi `eks.amazonaws.com/role-arn` (). Secara default akun layanan akan dibuat atau diperbarui untuk menyertakan anotasi peran, ini dapat dinonaktifkan menggunakan bendera`--role-only`.

Di dalam EKS, terdapat pengontrol penerimaan yang menyuntikkan kredensial sesi AWS ke dalam pod masing-masing peran berdasarkan anotasi pada Akun Layanan yang digunakan oleh pod. Kredensi akan diekspos oleh variabel `AWS_ROLE_ARN` & `AWS_WEB_IDENTITY_TOKEN_FILE`

lingkungan. Mengingat versi AWS SDK terbaru digunakan (lihat di sini untuk detail versi yang tepat), aplikasi akan menggunakan kredensi ini.

Dalam `eksctl` nama sumber daya adalah iamserviceaccount, yang mewakili pasangan Akun Peran dan Layanan IAM.

## Penggunaan dari CLI

> **Note**
>
> Peran IAM untuk Akun Layanan memerlukan Kubernetes versi 1.13 atau lebih tinggi.

Penyedia IAM OIDC tidak diaktifkan secara default, Anda dapat menggunakan perintah berikut untuk mengaktifkannya, atau menggunakan file konfigurasi (lihat di bawah):

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

Setelah Anda memiliki Penyedia IAM OIDC yang terkait dengan cluster, untuk membuat peran IAM terikat ke akun layanan, jalankan:

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

> **Note**
>
> Anda dapat menentukan `--attach-policy-arn` beberapa kali untuk menggunakan lebih dari satu kebijakan.

Lebih khusus lagi, Anda dapat membuat akun layanan dengan akses hanya-baca ke S3 dengan menjalankan:

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

Secara default, itu akan dibuat di `default` namespace, tetapi Anda dapat menentukan namespace lain, misalnya:

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

> **Note**
>
> Jika namespace belum ada, itu akan dibuat.

Jika Anda memiliki akun layanan yang sudah dibuat di cluster (tanpa Peran IAM), Anda harus menggunakan `--override-existing-serviceaccounts` flag.

Penandaan khusus juga dapat diterapkan ke Peran IAM dengan menentukan: `--tags`

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

CloudFormation akan menghasilkan nama peran yang mencakup string acak. Jika Anda lebih suka nama peran yang telah ditentukan, Anda dapat menentukan`--role-name`:

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

Ketika akun layanan dibuat dan dikelola oleh beberapa alat lain, seperti helm, gunakan `--role-only` untuk mencegah konflik. Alat lain kemudian bertanggung jawab untuk mempertahankan peran anotasi ARN. Perhatikan bahwa tidak `--override-existing-serviceaccounts` berpengaruh pada akun`roleOnly`/`--role-only`layanan, peran akan selalu dibuat.

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

Bila Anda memiliki peran yang sudah ada yang ingin Anda gunakan dengan akun layanan, Anda dapat memberikan `--attach-role-arn` tanda alih-alih memberikan kebijakan. Untuk memastikan peran hanya dapat diasumsikan oleh akun layanan yang ditentukan, Anda harus menetapkan dokumen kebijakan hubungan di sini].

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

Untuk memperbarui izin peran akun layanan yang dapat Anda jalankan`eksctl update iamserviceaccount`.

> **Note**
>
> `eksctl delete iamserviceaccount`menghapus Kubernetes `ServiceAccounts` meskipun tidak dibuat oleh. `eksctl`

## Penggunaan dengan file konfigurasi

Untuk mengelola `iamserviceaccounts` menggunakan file konfigurasi, Anda akan mencari untuk mengatur `iam.withOIDC: true` dan daftar akun yang Anda inginkan di bawah`iam.serviceAccount`.

Semua perintah mendukung`--config-file`, Anda dapat mengelola iamserviceaccounts dengan cara yang sama seperti nodegroups. `eksctl create iamserviceaccount`Perintah mendukung `--include` dan `--exclude` menandai (lihat bagian ini untuk detail selengkapnya tentang cara kerjanya). Dan `eksctl delete iamserviceaccount` perintah mendukung `--only-missing` juga, sehingga Anda dapat melakukan penghapusan dengan cara yang sama seperti nodegroups.

> **Note**
>
> Akun layanan IAM dicakup dalam namespace, yaitu dua akun layanan dengan nama yang sama mungkin ada di ruang nama yang berbeda. Jadi, untuk secara unik mendefinisikan akun layanan sebagai bagian dari`--include`, `--exclude` bendera, Anda harus meneruskan string nama dalam format. `namespace/name` Misalnya

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

Opsi untuk mengaktifkan `wellKnownPolicies` disertakan untuk menggunakan IRSA dengan kasus penggunaan terkenal seperti `cluster-autoscaler` dan`cert-manager`, sebagai singkatan untuk daftar kebijakan.

Kebijakan terkenal yang didukung dan properti lainnya `serviceAccounts` didokumentasikan pada skema konfigurasi.

Anda menggunakan contoh konfigurasi berikut dengan`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
```

Jika Anda membuat cluster tanpa set bidang ini, Anda dapat menggunakan perintah berikut untuk mengaktifkan semua yang Anda butuhkan:

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

### Informasi lebih lanjut

- Memperkenalkan Peran IAM Berbutir Halus Untuk Akun Layanan
- Panduan Pengguna EKS - Peran IAM Untuk Akun Layanan
- Memetakan pengguna IAM dan peran ke peran RBAC Kubernetes

## Asosiasi Identitas EKS Pod

Amazon EKS telah memperkenalkan mekanisme baru yang disempurnakan yang disebut Pod Identity Association bagi administrator klaster untuk mengonfigurasi aplikasi Kubernetes agar menerima izin IAM yang diperlukan untuk terhubung dengan layanan AWS di luar klaster. Pod Identity Association memanfaatkan IRSA, namun membuatnya dapat dikonfigurasi secara langsung melalui EKS API, menghilangkan kebutuhan untuk menggunakan IAM API sama sekali.

Akibatnya, peran IAM tidak perlu lagi mereferensikan penyedia OIDC dan karenanya tidak akan terikat pada satu cluster lagi. Ini berarti, peran IAM sekarang dapat digunakan di beberapa kluster EKS tanpa perlu memperbarui kebijakan kepercayaan peran setiap kali cluster baru dibuat. Ini pada gilirannya, menghilangkan kebutuhan akan duplikasi peran dan menyederhanakan proses otomatisasi IRSA sama sekali.

### Prasyarat

Di balik layar, implementasi asosiasi identitas pod menjalankan agen sebagai daemonset pada node pekerja. Untuk menjalankan agen pra-syarat di cluster, EKS menyediakan add-on baru yang

disebut EKS Pod Identity Agent. Oleh karena itu, membuat asosiasi identitas pod (secara umum, dan dengan`eksctl`) memerlukan `eks-pod-identity-agent` addon yang sudah diinstal sebelumnya pada cluster. Addon ini dapat dibuat menggunakan dengan `eksctl` cara yang sama addon lain yang didukung.

```
eksctl create addon --cluster my-cluster --name eks-pod-identity-agent
```

Selain itu, jika menggunakan peran IAM yang sudah ada sebelumnya saat membuat asosiasi identitas pod, Anda harus mengonfigurasi peran tersebut untuk mempercayai prinsip layanan EKS () `pods.eks.amazonaws.com` yang baru diperkenalkan. Contoh kebijakan kepercayaan IAM dapat ditemukan di bawah ini:

```
# Error: No files found with UUID: 44d1085a-03ca-431a-9774-b786a9774200
```

Jika sebaliknya Anda tidak memberikan ARN dari peran yang ada ke perintah create, `eksctl` akan membuat satu di belakang layar dan mengkonfigurasi kebijakan kepercayaan di atas.

## Membuat Asosiasi Identitas Pod

Untuk memanipulasi asosiasi identitas pod, `eksctl` telah menambahkan bidang baru di bawah`iam.podIdentityAssociations`, mis.

```
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
```

Untuk contoh lengkap, lihat pod-identity-associations.yaml.

> **Note**
>
> Selain `permissionPolicy` yang digunakan sebagai dokumen kebijakan inline, semua bidang lainnya memiliki mitra bendera CLI.

Membuat asosiasi identitas pod dapat dicapai dengan cara-cara berikut. Selama pembuatan klaster, dengan menentukan asosiasi identitas pod yang diinginkan sebagai bagian dari file konfigurasi dan menjalankan:

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

Posting pembuatan cluster, menggunakan salah satu file konfigurasi mis.

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

ATAU menggunakan bendera CLI mis.

```
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**
>
> Hanya satu peran IAM yang dapat dikaitkan dengan akun layanan pada satu waktu. Oleh karena itu, mencoba membuat asosiasi identitas pod kedua untuk akun layanan yang sama akan menghasilkan kesalahan.

## Mengambil Asosiasi Identitas Pod

Untuk mengambil semua asosiasi identitas pod untuk klaster tertentu, jalankan salah satu perintah berikut:

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

ATAU

```
eksctl get podidentityassociation --cluster my-cluster
```

Selain itu, untuk mengambil hanya asosiasi identitas pod dalam namespace tertentu, gunakan flag, mis. `--namespace`

```
eksctl get podidentityassociation --cluster my-cluster --namespace default
```

Akhirnya, untuk mengambil satu asosiasi, sesuai dengan akun layanan K8s tertentu, juga termasuk `--service-account-name` ke perintah di atas, mis.

```
eksctl get podidentityassociation --cluster my-cluster --namespace default --service-
account-name s3-reader
```

## Memperbarui Asosiasi Identitas Pod

Untuk memperbarui peran IAM dari satu atau beberapa asosiasi identitas pod, teruskan yang baru `roleARN(s)` ke file konfigurasi misalnya

```
iam:
  podIdentityAssociations:
    - namespace: default
      serviceAccountName: s3-reader
      roleARN: new-role-arn-1
    - namespace: dev
      serviceAccountName: app-cache-access
      roleARN: new-role-arn-2
```

dan jalankan:

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

ATAU (untuk memperbarui satu asosiasi) meneruskan yang baru `--role-arn` melalui flag CLI:

```
eksctl update podidentityassociation --cluster my-cluster --namespace default --
service-account-name s3-reader --role-arn new-role-arn
```

## Menghapus Asosiasi Identitas Pod

Untuk menghapus satu atau beberapa asosiasi identitas pod, teruskan `namespace(s)` dan `serviceAccountName(s)` ke file konfigurasi misalnya

```
iam:
  podIdentityAssociations:
    - namespace: default
      serviceAccountName: s3-reader
    - namespace: dev
      serviceAccountName: app-cache-access
```

dan jalankan:

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

ATAU (untuk menghapus satu asosiasi) meneruskan `--namespace` dan `--service-account-name` melalui flag CLI:

```
eksctl delete podidentityassociation --cluster my-cluster --namespace default --
service-account-name s3-reader
```

## EKS Dukungan add-on untuk asosiasi identitas pod

Pengaya EKS juga mendukung penerimaan izin IAM melalui Asosiasi Identitas Pod EKS. File konfigurasi mengekspos tiga bidang yang memungkinkan konfigurasi ini:`addon.podIdentityAssociations`, dan. `addonsConfig.autoApplyPodIdentityAssociations` `addon.useDefaultPodIdentityAssociations` Anda dapat secara eksplisit mengonfigurasi asosiasi identitas pod yang diinginkan, menggunakan`addon.podIdentityAssociations`, atau `eksctl` secara otomatis menyelesaikan (dan menerapkan) konfigurasi identitas pod yang direkomendasikan, menggunakan salah satu atau`addonsConfig.autoApplyPodIdentityAssociations`. `addon.useDefaultPodIdentityAssociations`

> **Note**
>
> Tidak semua Pengaya EKS akan mendukung asosiasi identitas pod saat peluncuran. Untuk kasus ini, izin IAM yang diperlukan akan terus diberikan menggunakan pengaturan IRSA.

## Membuat addons dengan izin IAM

Saat membuat addon yang memerlukan izin IAM, pertama-tama `eksctl` akan memeriksa apakah asosiasi identitas pod atau pengaturan IRSA sedang dikonfigurasi secara eksplisit sebagai bagian dari file konfigurasi, dan jika demikian, gunakan salah satunya untuk mengonfigurasi izin untuk addon. misalnya

```
addons:
- name: vpc-cni
  podIdentityAssociations:
  - serviceAccountName: aws-node
    permissionPolicyARNs: ["arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"]
```

dan lari

```
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**
>
> Menyetel identitas pod dan IRSA secara bersamaan tidak diperbolehkan, dan akan menghasilkan kesalahan validasi.

Untuk Eks Add-on yang mendukung identitas pod, `eksctl` menawarkan opsi untuk secara otomatis mengonfigurasi izin IAM yang direkomendasikan, pada pembuatan addon. Ini dapat dicapai hanya dengan menyetel `addonsConfig.autoApplyPodIdentityAssociations: true` di file konfigurasi. misalnya

```
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
```

dan lari

```
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
```

Secara setara, hal yang sama dapat dilakukan melalui flag CLI misalnya

```
eksctl create addon --cluster my-cluster --name vpc-cni --auto-apply-pod-identity-
associations
```

Untuk memigrasikan addon yang ada untuk menggunakan identitas pod dengan kebijakan IAM yang direkomendasikan, gunakan

```
addons:
- name: vpc-cni
  useDefaultPodIdentityAssociations: true
```

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

## Memperbarui addons dengan izin IAM

Saat memperbarui addon, menentukan `addon.PodIdentityAssociations` akan mewakili satu sumber kebenaran untuk status yang akan dimiliki addon, setelah operasi pembaruan selesai. Di belakang layar, berbagai jenis operasi dilakukan untuk mencapai keadaan yang diinginkan yaitu

- buat identitas pod yang ada di file konfigurasi, tetapi tidak ada di cluster
- hapus identitas pod yang ada yang telah dihapus dari file konfigurasi, bersama dengan sumber daya IAM terkait
- perbarui identitas pod yang ada yang juga ada di file konfigurasi, dan set izin IAM telah berubah

> **Note**
>
> Siklus hidup asosiasi identitas pod yang dimiliki oleh Eks Add-on ditangani langsung oleh EKS Addons API.

Anda tidak dapat menggunakan `eksctl update podidentityassociation` (untuk memperbarui izin IAM) atau `eksctl delete podidentityassociations` (untuk menghapus asosiasi) untuk asosiasi yang digunakan dengan Add-on Amazon EKS. Sebaliknya, `eksctl update addon` atau `eksctl delete addon` akan digunakan.

Mari kita lihat contoh di atas, dimulai dengan menganalisis konfigurasi identitas pod awal untuk 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"
    }
]
```

Sekarang gunakan konfigurasi di bawah ini:

```
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
```

dan lari

```
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
```

sekarang periksa apakah konfigurasi identitas pod telah diperbarui dengan benar

```
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"
    }
]
```

Untuk menghapus semua asosiasi identitas pod dari addon, `addon.PodIdentityAssociations` harus disetel secara eksplisit ke, mis. `[]`

```
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: []
```

dan lari

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

### Menghapus addons dengan izin IAM

Menghapus addon juga akan menghapus semua identitas pod yang terkait dengan addon. Menghapus cluster akan mencapai efek yang sama, untuk semua addons. Setiap peran IAM untuk identitas pod, yang dibuat oleh`eksctl`, akan dihapus juga.

## Migrasi iamserviceaccounts dan addons yang ada ke asosiasi identitas pod

Ada perintah `eksctl` utils untuk memigrasi Peran IAM yang ada untuk akun layanan ke asosiasi identitas pod, yaitu

```
eksctl utils migrate-to-pod-identity --cluster my-cluster --approve
```

Di belakang layar, perintah akan menerapkan langkah-langkah berikut:

- instal `eks-pod-identity-agent` addon jika belum aktif di cluster
- mengidentifikasi semua Peran IAM yang terkait dengan iamserviceaccounts
- mengidentifikasi semua Peran IAM yang terkait dengan addon EKS yang mendukung asosiasi identitas pod
- memperbarui kebijakan kepercayaan IAM dari semua peran yang diidentifikasi, dengan entitas tepercaya tambahan, menunjuk ke prinsip Layanan EKS yang baru (dan, secara opsional, menghapus hubungan kepercayaan penyedia OIDC yang keluar)
- buat asosiasi identitas pod untuk peran yang difilter yang terkait dengan iamserviceaccounts
- perbarui addons EKS dengan identitas pod (EKS API akan membuat identitas pod di belakang layar)

Menjalankan perintah tanpa `--approve` bendera hanya akan menampilkan rencana yang terdiri dari serangkaian tugas yang mencerminkan langkah-langkah di atas, mis.

```
[#]  (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
```

Hubungan kepercayaan penyedia OIDC yang ada selalu dihapus dari Peran IAM yang terkait dengan Pengaya EKS. Selain itu, untuk menghapus hubungan kepercayaan penyedia OIDC yang ada dari Peran IAM yang terkait dengan iamserviceaccounts, jalankan perintah dengan flag, mis. `--remove-oidc-provider-trust-relationship`

```
eksctl utils migrate-to-pod-identity --cluster my-cluster --approve --remove-oidc-
provider-trust-relationship
```

## Dukungan Identitas Pod Lintas Akun

eksctl mendukung akses lintas akun EKS Pod Identity. Fitur ini memungkinkan pod yang berjalan di kluster EKS Anda untuk mengakses sumber daya AWS di akun AWS yang berbeda.

### Penggunaan

Untuk membuat asosiasi identitas pod dengan akses lintas akun, pertama-tama siapkan Peran dan Kebijakan IAM yang memungkinkan akses dari akun AWS sumber (dengan klaster) ke akun AWS

target (dengan sumber daya yang dapat diakses klaster). Untuk contoh ini, lihat "Amazon EKS Pod Identity merampingkan akses lintas akun."

Setelah Peran IAM dikonfigurasi di setiap akun, gunakan eksctl untuk membuat asosiasi identitas pod:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  # The cluster name and service account name should match the target
  # account policy's trust relationship.
  name: my-cluster
  region: us-west-2
  version: "1.32"

addons:
  - name: vpc-cni
  - name: coredns
  - name: kube-proxy
  - name: eks-pod-identity-agent

iam:
  podIdentityAssociations:
  - namespace: default
    serviceAccountName: demo-app-sa
    createServiceAccount: true
    # The source role in the same account as the cluster
    roleARN: arn:aws:iam::1111111111:role/account-a-role
    # The target role in a different account
    targetRoleARN: arn:aws:iam::2222222222:role/account-b-role
    # Optional: Disable session tags
    disableSessionTags: false

managedNodeGroups:
  - name: my-cluster
    instanceType: m6a.large
    privateNetworking: true
    minSize: 2
    desiredCapacity: 2
    maxSize: 3
```

# Referensi lebih lanjut

AWS Userdocs resmi untuk EKS Dukungan Add-on untuk identitas pod

Posting Blog AWS Resmi di Asosiasi Identitas Pod

Dokumen pengguna AWS resmi untuk Asosiasi Identitas Pod

# Opsi deployment

Bab ini mencakup penggunaan eksctl untuk mengelola kluster EKS yang diterapkan ke lingkungan alternatif.

Untuk informasi paling akurat tentang opsi penerapan EKS, lihat Menerapkan kluster Amazon EKS di seluruh lingkungan cloud dan lokal di Panduan Pengguna EKS.

## Topik:

- the section called “EKS Di Mana Saja”
  - Gunakan eksctl dengan cluster Amazon EKS Anywhere.
  - Amazon EKS Anywhere adalah perangkat lunak manajemen kontainer yang dibuat oleh AWS yang membuatnya lebih mudah untuk menjalankan dan mengelola Kubernetes di lokasi maupun di edge.
- the section called “AWS Outposts Support”
  - Gunakan eksctl dengan kluster EKS di AWS Outposts.
  - AWS Outposts adalah rangkaian solusi terkelola penuh yang menghadirkan infrastruktur dan layanan AWS ke hampir semua lokasi lokal atau edge untuk pengalaman hybrid yang benar-benar konsisten.
  - Dukungan AWS Outposts di eksctl memungkinkan Anda membuat cluster lokal dengan seluruh klaster Kubernetes, termasuk bidang kontrol EKS dan node pekerja, yang berjalan secara lokal di AWS Outposts.
- the section called “Node Hibrida EKS”
  - Jalankan aplikasi lokal dan edge pada infrastruktur yang dikelola pelanggan dengan kluster, fitur, dan alat Amazon EKS yang sama yang Anda gunakan di AWS.

# EKS Di Mana Saja

`eksctl`menyediakan akses ke fitur AWS yang disebut `EKS Anywhere` dengan sub perintah`eksctl anywhere`. Ini membutuhkan `eksctl-anywhere` biner yang ada di`PATH`. Silakan ikuti instruksi yang diuraikan di sini Instal eksctl-anywhere untuk menginstalnya.

Setelah selesai, jalankan perintah di mana saja dengan menjalankan:

```
eksctl anywhere version
v0.5.0
```

Untuk informasi lebih lanjut tentang EKS Anywhere, silakan kunjungi Situs Web EKS Anywhere.

# AWS Outposts Support

> **Warning**
>
> EKS Managed Nodegroups tidak didukung di Outposts.

## Memperluas cluster yang ada ke AWS Outposts

Anda dapat memperluas kluster EKS yang ada yang berjalan di wilayah AWS ke AWS Outposts dengan menyetel nodegroup baru `nodeGroup.outpostARN` untuk membuat nodegroup di Outposts, seperti pada:

```
# 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
```

Dalam penyiapan ini, bidang kontrol EKS berjalan di wilayah AWS sementara nodegroup dengan `outpostARN` set berjalan di Outpost yang ditentukan. Ketika nodegroup sedang dibuat di Outposts

untuk pertama kalinya, eksctl memperluas VPC dengan membuat subnet pada Outpost yang ditentukan. Subnet ini digunakan untuk membuat nodegroup yang telah ditetapkan. `outpostARN`

Pelanggan dengan VPC yang sudah ada sebelumnya diharuskan untuk membuat subnet di Outposts dan `nodeGroup.subnets` meneruskannya, seperti pada:

```
# 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"
```

## Membuat klaster lokal di AWS Outposts

> **Note**
>
> Cluster lokal hanya mendukung rak Outpost.

> **Note**
>
> Hanya Amazon Linux 2 yang didukung untuk nodegroups saat control plane berada di Outposts. Hanya tipe volume EBS gp2 yang didukung untuk nodegroup di Outposts.

Dukungan AWS Outposts di eksctl memungkinkan Anda membuat cluster lokal dengan seluruh klaster Kubernetes, termasuk bidang kontrol EKS dan node pekerja, yang berjalan secara lokal di AWS Outposts. Pelanggan dapat membuat klaster lokal dengan bidang kontrol EKS dan node pekerja yang berjalan secara lokal di AWS Outposts, atau mereka dapat memperluas kluster EKS yang ada yang berjalan di wilayah AWS ke AWS Outposts dengan membuat node pekerja di Outposts.

Untuk membuat bidang kontrol EKS dan nodegroup di AWS Outposts, `outpost.controlPlaneOutpostARN` setel ke Outpost ARN, seperti pada:

```
# 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
```

Ini menginstruksikan eksctl untuk membuat bidang kontrol EKS dan subnet pada Outpost yang ditentukan. Karena rak Outposts ada di zona ketersediaan tunggal, eksctl hanya membuat satu subnet publik dan pribadi. eksctl tidak mengaitkan VPC yang dibuat dengan gateway lokal dan, dengan demikian, eksctl akan kekurangan konektivitas ke server API dan tidak akan dapat membuat nodegroup. Oleh karena itu, jika `ClusterConfig` berisi nodegroup selama pembuatan cluster, perintah harus dijalankan dengan`--without-nodegroup`, seperti pada:

```
eksctl create cluster -f outpost.yaml --without-nodegroup
```

Merupakan tanggung jawab pelanggan untuk mengaitkan VPC yang dibuat eksctl dengan gateway lokal setelah pembuatan cluster untuk mengaktifkan konektivitas ke server API. Setelah langkah ini, nodegroups dapat dibuat menggunakan. `eksctl create nodegroup`

Anda dapat secara opsional menentukan jenis instance untuk node bidang kontrol di `outpost.controlPlaneInstanceType` atau untuk nodegroups di`nodeGroup.instanceType`, tetapi tipe instance harus ada di Outpost atau eksctl akan mengembalikan kesalahan. Secara default, eksctl mencoba memilih jenis instance terkecil yang tersedia di Outpost untuk node bidang kontrol dan nodegroup.

Ketika bidang kontrol berada di Outposts, nodegroups dibuat di Outpost itu. Anda dapat secara opsional menentukan ARN Outpost untuk nodegroup `nodeGroup.outpost`ARN tetapi harus cocok dengan Outpost ARN pesawat kontrol.

```
# outpost-fully-private.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: outpost-fully-private
  region: us-west-2

privateCluster:
  enabled: true

outpost:
  # Required.
  controlPlaneOutpostARN: "arn:aws:outposts:us-west-2:1234:outpost/op-1234"
  # Optional, defaults to the smallest available instance type on the Outpost.
  controlPlaneInstanceType: m5d.large
```

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

metadata:
```

```
  name: outpost
  region: us-west-2

outpost:
  # Required.
  controlPlaneOutpostARN: "arn:aws:outposts:us-west-2:1234:outpost/op-1234"
  # Optional, defaults to the smallest available instance type on the Outpost.
  controlPlaneInstanceType: m5d.large

  controlPlanePlacement:
    groupName: placement-group-name
```

## VPC yang ada

Pelanggan dengan VPC yang ada dapat membuat cluster lokal di AWS Outposts dengan menentukan konfigurasi subnet di, seperti pada: `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
```

Subnet harus ada di Outpost yang ditentukan dalam `outpost.controlPlaneOutpostARN` atau eksctl akan mengembalikan kesalahan. Anda juga dapat menentukan nodegroup selama pembuatan klaster jika Anda memiliki akses ke gateway lokal untuk subnet, atau memiliki konektivitas ke sumber daya VPC.

## Fitur yang tidak didukung pada cluster lokal

- Addons
- Peran IAM untuk Akun Layanan
- IPv6
- Penyedia Identitas
- Fargate
- Enkripsi KMS
- Local Zones
- Karpenter
- Pemilih Instance
- Availability Zone tidak dapat ditentukan karena default ke zona ketersediaan Outpost.
- `vpc.publicAccessCIDRs`dan `vpc.autoAllocateIPv6` tidak didukung.
- Akses titik akhir publik ke server API tidak didukung karena klaster lokal hanya dapat dibuat dengan akses titik akhir khusus pribadi.

## Informasi lebih lanjut

- Amazon EKS di AWS Outposts
- Cluster lokal untuk Amazon EKS di AWS Outposts
- Membuat cluster lokal
- Meluncurkan node Amazon Linux yang dikelola sendiri di Outpost

# Keamanan

`eksctl`menyediakan beberapa opsi yang dapat meningkatkan keamanan cluster EKS Anda.

## `withOIDC`

Aktifkan `withOIDC`untuk secara otomatis membuat IRSA untuk plugin amazon CNI dan membatasi izin yang diberikan ke node di cluster Anda, alih-alih memberikan izin yang diperlukan hanya ke akun layanan CNI.

Latar belakang dijelaskan dalam dokumentasi AWS ini.

## `disablePodIMDS`

Untuk nodegroup terkelola dan tidak terkelola, `disablePodIMDS`opsi tersedia mencegah semua pod jaringan non host yang berjalan di nodegroup ini membuat permintaan IMDS.

> **Note**
>
> Ini tidak bisa digunakan bersamaan dengan `withAddonPolicies`.

# Enkripsi Amplop KMS untuk kluster EKS

> **Note**
>
> Amazon Elastic Kubernetes Service (Amazon EKS) menyediakan enkripsi amplop default untuk semua data API Kubernetes di kluster EKS yang menjalankan Kubernetes versi 1.28 atau lebih tinggi. Untuk informasi selengkapnya, lihat Enkripsi amplop default untuk semua Data API Kubernetes di Panduan Pengguna EKS.

EKS mendukung penggunaan kunci AWS KMS untuk menyediakan enkripsi amplop rahasia Kubernetes yang disimpan di EKS. Enkripsi amplop menambahkan lapisan enkripsi tambahan yang dikelola pelanggan untuk rahasia aplikasi atau data pengguna yang disimpan dalam klaster Kubernetes.

Sebelumnya, Amazon EKS mendukung pengaktifan enkripsi amplop menggunakan kunci KMS hanya selama pembuatan klaster. Sekarang, Anda dapat mengaktifkan enkripsi amplop untuk kluster Amazon EKS kapan saja.

Baca selengkapnya tentang Menggunakan dukungan penyedia enkripsi EKS untuk defense-in-depth posting di blog AWS container.

## Membuat cluster dengan enkripsi KMS diaktifkan

```
# 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
```

## Mengaktifkan enkripsi KMS pada cluster yang ada

Untuk mengaktifkan enkripsi KMS pada cluster yang belum mengaktifkannya, jalankan

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

atau tanpa file konfigurasi:

```
eksctl utils enable-secrets-encryption --cluster=kms-cluster --key-arn=arn:aws:kms:us-
west-2:<account>:key/<key> --region=<region>
```

Selain mengaktifkan enkripsi KMS pada kluster EKS, eksctl juga mengenkripsi ulang semua rahasia Kubernetes yang ada menggunakan kunci KMS baru dengan memperbaruinya dengan anotasi. `eksctl.io/kms-encryption-timestamp` Perilaku ini dapat dinonaktifkan dengan lewat`--encrypt-existing-secrets=false`, seperti pada:

```
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>
```

Jika kluster sudah mengaktifkan enkripsi KMS, eksctl akan melanjutkan untuk mengenkripsi ulang semua rahasia yang ada.

> **Note**
>
> Setelah enkripsi KMS diaktifkan, enkripsi KMS tidak dapat dinonaktifkan atau diperbarui untuk menggunakan kunci KMS yang berbeda.

# Pemecahan Masalah

Topik ini mencakup petunjuk tentang cara mengatasi kesalahan umum dengan Eksctl.

## Pembuatan tumpukan yang gagal

Anda dapat menggunakan `--cfn-disable-rollback` flag untuk menghentikan Cloudformation agar tidak mengembalikan tumpukan yang gagal untuk mempermudah proses debug.

## subnet ID “subnet-11111111" tidak sama dengan “subnet-22222222"

Diberikan file konfigurasi yang menentukan subnet untuk VPC seperti berikut ini:

```
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: []
```

Kesalahan `subnet ID "subnet-11111111" is not the same as "subnet-22222222"` berarti bahwa subnet yang ditentukan tidak ditempatkan di zona Ketersediaan yang tepat. Periksa di konsol AWS yang merupakan ID subnet yang tepat untuk setiap Availability Zone.

Dalam contoh ini, konfigurasi yang benar untuk VPC adalah:

```
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}
```

## Masalah penghapusan

Jika penghapusan Anda tidak berfungsi, atau Anda lupa menambahkan `--wait` pada penghapusan, Anda mungkin perlu menggunakan alat amazon lainnya untuk menghapus tumpukan cloudformation. Ini dapat dicapai melalui gui atau dengan aws cli.

## kubectl logs dan kubectl run gagal dengan Kesalahan Otorisasi

Jika node Anda di-deploy di subnet pribadi dan `kubectl logs` atau `kubectl run` gagal dengan kesalahan seperti berikut:

```
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)
```

Maka Anda mungkin perlu mengatur enableDnsHostnames. Detail lebih lanjut dapat ditemukan dalam masalah ini.

# Pengumuman

Topik ini mencakup pengumuman sebelumnya tentang fitur Eksctl baru.

## Nodegroups Terkelola Default

Pada eksctl v0.58.0, eksctl membuat nodegroup terkelola secara default ketika file tidak ditentukan untuk dan. `ClusterConfig` `eksctl create cluster` `eksctl create nodegroup` Untuk membuat nodegroup yang dikelola sendiri, teruskan. `--managed=false` Ini dapat merusak skrip yang tidak menggunakan file konfigurasi jika fitur yang tidak didukung dalam nodegroup terkelola, misalnya, nodegroups Windows, sedang digunakan. Untuk memperbaikinya, teruskan`--managed=false`, atau tentukan konfigurasi nodegroup Anda dalam `ClusterConfig` file menggunakan `nodeGroups` bidang yang membuat nodegroup yang dikelola sendiri.

## Nodegroup Bootstrap Override Untuk Kustom AMIs

Perubahan ini diumumkan dalam edisi Breaking: overrideBootstrapCommand soon.... Sekarang, telah terjadi dalam PR ini. Harap baca masalah terlampir dengan cermat tentang mengapa kami memutuskan untuk menjauh dari mendukung kustom AMIs tanpa skrip bootstrap atau dengan skrip bootstrap sebagian.

Kami masih menyediakan pembantu! Bermigrasi semoga tidak terlalu menyakitkan. `eksctl`masih menyediakan skrip, yang bila bersumber, akan mengekspor beberapa properti dan pengaturan lingkungan yang bermanfaat. Script ini terletak di sini.

Properti lingkungan berikut akan siap membantu Anda:

```
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
```

Minimum yang perlu digunakan saat mengganti sehingga `eksctl` tidak gagal, adalah label! `eksctl`bergantung pada satu set label tertentu untuk berada di node, sehingga dapat menemukannya. Saat mendefinisikan override, berikan perintah penggantian minimum ini:

```
    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}"
```

Untuk nodegroup yang tidak memiliki akses internet keluar, Anda harus menyediakan `--apiserver-endpoint` dan `--b64-cluster-ca` ke skrip bootstrap sebagai berikut:

```
    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}
```

Perhatikan pengaturan `--node-labels`. Jika ini tidak ditentukan, node akan bergabung dengan cluster, tetapi pada akhirnya `eksctl` akan habis pada langkah terakhir ketika menunggu node menjadi`Ready`. Ini melakukan pencarian Kubernetes untuk node yang memiliki label. `alpha.eksctl.io/nodegroup-name=<cluster-name>` Ini hanya berlaku untuk nodegroup yang tidak dikelola. Untuk dikelola itu menggunakan label yang berbeda.

Jika, sama sekali, dimungkinkan untuk beralih ke nodegroup terkelola untuk menghindari overhead ini, saatnya telah tiba sekarang untuk melakukan itu. Membuat semua overriding jauh lebih mudah.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.