Amazon EKS

Guía del usuario de Eksctl

Copyright © 2025 Copyright informaiton pending.

Guía del usuario de Eksctl: Amazon EKS

Copyright © 2025 Copyright informaiton pending.

Información de derechos de autor pendiente.

Table of Contents

¿Qué es Eksctl?

eksctl es una herramienta de línea de comandos que automatiza y simplifica el proceso de creación, administración y operación de clústeres de Amazon Elastic Kubernetes Service (Amazon EKS). Escrito en Go, eksctl proporciona una sintaxis declarativa a través de configuraciones de YAML y comandos CLI para gestionar operaciones complejas de clústeres de EKS que, de otro modo, requerirían varios pasos manuales en diferentes servicios de AWS.

eksctl resulta especialmente útil para los DevOps ingenieros, los equipos de plataformas y los administradores de Kubernetes, que necesitan implementar y gestionar los clústeres de EKS de forma coherente y a escala. Resulta especialmente útil para las organizaciones que están pasando de un Kubernetes autogestionado a EKS, o para las que están implementando prácticas de infraestructura como código (IaC), ya que se puede integrar en los flujos de trabajo de automatización y los procesos de automatización existentes. CI/CD La herramienta elimina muchas de las interacciones complejas entre los servicios de AWS necesarios para la configuración del clúster de EKS, como la configuración de VPC, la creación de roles de IAM y la administración de grupos de seguridad.

Las características clave de eksctl incluyen la capacidad de crear clústeres EKS completamente funcionales con un solo comando, la compatibilidad con configuraciones de red personalizadas, la administración automatizada de grupos de nodos y la integración del flujo de trabajo. GitOps La herramienta gestiona las actualizaciones de clústeres, escala los grupos de nodos y gestiona la administración de complementos mediante un enfoque declarativo. eksctl también proporciona capacidades avanzadas como la configuración de perfiles de Fargate, la personalización de grupos de nodos gestionados y la integración de instancias puntuales, al tiempo que mantiene la compatibilidad con otras herramientas y servicios de AWS mediante la integración nativa del SDK de AWS.

Características

Las funciones que se implementan actualmente son:

- Crear, obtener, enumerar y eliminar clústeres
- Crea, vacía y elimina grupos de nodos
- Escala un grupo de nodos
- Actualice un clúster de

- Utilice la opción personalizada AMIs
- Configurar redes de VPC
- Configure el acceso a los puntos finales de la API
- Support para grupos de nodos de GPU
- Instancias puntuales e instancias mixtas
- Políticas de administración y complementos de IAM
- Enumere las pilas de Cloudformation de clústeres
- Instale coredns
- Escribe el archivo kubeconfig para un clúster

Preguntas frecuentes de Eksctl

General

¿Puedo utilizarlos **`eksctl`** para gestionar clústeres que no fueron creados por él? **`eksctl`**

Sí A partir de la versión, se `0.40.0` `eksctl` puede ejecutar en cualquier clúster, ya sea que lo haya creado `eksctl` o no. Para obtener más información, consulte the section called “Clústeres no creados por eksctl”.

Grupos de nodos

¿Cómo puedo cambiar el tipo de instancia de mi grupo de nodos?

Desde el punto de vista de, los grupos de nodos `eksctl` son inmutables. Esto significa que, una vez creado, lo único que se `eksctl` puede hacer es escalar el grupo de nodos hacia arriba o hacia abajo.

Para cambiar el tipo de instancia, crea un nuevo grupo de nodos con el tipo de instancia deseado y, a continuación, vacíalo para que las cargas de trabajo pasen al nuevo. Una vez completado ese paso, puedes eliminar el grupo de nodos anterior.

¿Cómo puedo ver los datos de usuario generados para un grupo de nodos?

Primero necesitarás el nombre de la pila de Cloudformation que administra el grupo de nodos:

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

Verás un nombre similar a. `eksctl-CLUSTER_NAME-nodegroup-NODEGROUP_NAME`

Puede ejecutar lo siguiente para obtener los datos de usuario. Observe la última línea que decodifica desde base64 y descomprime los datos comprimidos con gzip.

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

Ingress

¿Cómo configuro el ingreso con? **`eksctl`**

Recomendamos utilizar el controlador Load Balancer de AWS. Puede encontrar la documentación sobre cómo implementar el controlador en su clúster, así como sobre cómo migrar desde el antiguo controlador de ingreso ALB, aquí.

Para el controlador de entrada de Nginx, la configuración sería la misma que la de cualquier otro clúster de Kubernetes.

Kubectl

Utilizo un proxy HTTPS y no se puede validar el certificado del clúster. ¿Cómo puedo usar el sistema? CAs

Configure la variable de entorno `KUBECONFIG_USE_SYSTEM_CA` para `kubeconfig` respetar las autoridades de certificación del sistema.

Funcionamiento en seco

La función de ejecución en seco te permite inspeccionar y cambiar las instancias que coinciden con el selector de instancias antes de proceder a crear un grupo de nodos.

Cuando `eksctl create cluster` se llama con las opciones del selector de instancias y `--dry-run`, eksctl generará un ClusterConfig archivo que contiene un grupo de nodos que representa las opciones de CLI y los tipos de instancia configurados para las instancias que coinciden con los criterios de recursos del selector de instancias.

```
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
```

Luego, lo generado se ClusterConfig puede pasar a: `eksctl create cluster`

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

Cuando se pasa un ClusterConfig archivo`--dry-run`, eksctl generará un ClusterConfig archivo que contiene los valores establecidos en el archivo.

Opciones únicas en eksctl

Hay ciertas opciones únicas que no se pueden representar en el `ClusterConfig` archivo, por ejemplo, `--install-vpc-controllers`

Se espera que:

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

seguido de:

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

equivaldría a ejecutar el primer comando sin él`--dry-run`.

Por lo tanto, eksctl no permite pasar opciones que no se puedan representar en el archivo de configuración cuando se `--dry-run` pasa.

> **Important**
>
> Si necesita pasar un perfil de AWS, defina la variable de `AWS_PROFILE` entorno en lugar de pasar la opción `--profile` CLI.

Tutorial

En este tema se explica cómo instalar y configurar eksctl y, a continuación, cómo usarlo para crear un clúster de Amazon EKS.

Paso 1: Instalar eksctl

Complete los siguientes pasos para descargar e instalar la última versión de eksctl en su dispositivo Linux o macOS:

Para instalar eksctl con Homebrew

1. (Requisito previo) Instale Homebrew.
2. Añada el grifo de AWS:

```
brew tap aws/tap
```

3. Instalación de eksctl

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

Antes de usar eksctl, complete estos pasos de configuración:

1. Requisitos previos de instalación:
   - Instale la versión 2.x o posterior de AWS CLI.
   - Instale kubectl con Homebrew:

     ```
     brew install kubernetes-cli
     ```
2. Configure las credenciales de AWS en su entorno:

   ```
   aws configure
   ```
3. Compruebe la configuración de la CLI de AWS:

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

Paso 2: Crear un archivo de configuración del clúster

Cree un archivo de configuración de clúster siguiendo estos pasos:

1. Cree un archivo nuevo llamado`cluster.yaml`:

```
touch cluster.yaml
```

2. Agregue la siguiente configuración de clúster básica:

```
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. Personalice la configuración:
   - Actualice el `region` para que coincida con la región de AWS que desee.
   - Modifíquelo `instanceType` en función de sus requisitos de carga de trabajo.
   - Ajuste el `desiredCapacityminSize`, y `maxSize` según sus necesidades.
4. Valide el archivo de configuración:

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

Paso 3: Crear un clúster

Siga estos pasos para crear su clúster de EKS:

1. Cree el clúster mediante el archivo de configuración:

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

2. Espere a que se cree el clúster (esto suele tardar entre 15 y 20 minutos).
3. Compruebe la creación del clúster:

```
eksctl get cluster
```

4. Configura kubectl para usar tu nuevo clúster:

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

5. Verifica la conectividad del clúster:

```
kubectl get nodes
```

El clúster ya está listo para usarse.

Opcional: eliminar el clúster

Recuerde eliminar el clúster cuando haya terminado para evitar cargos innecesarios:

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

> **Note**
>
> La creación de clústeres puede conllevar gastos de AWS. Asegúrese de revisar los precios de Amazon EKS antes de crear un clúster.

Siguientes pasos

- Configure Kubectl para conectarse al clúster
- Implementación de una aplicación de muestra

Opciones de instalación para Eksctl

`eksctl`está disponible para su instalación desde las versiones oficiales, tal y como se describe a continuación. Le recomendamos que lo instale únicamente `eksctl` desde las GitHub versiones oficiales. Puede optar por utilizar un instalador de terceros, pero tenga en cuenta que AWS no mantiene ni admite estos métodos de instalación. Úselos según su propio criterio.

Requisito previo

Necesitará tener configuradas las credenciales de la API de AWS. Lo que funcione para AWS CLI o cualquier otra herramienta (kops, Terraform, etc.) debería ser suficiente. Puede utilizar variables de `~/.aws/credentials`archivo o de entorno. Para obtener más información, consulte la referencia de la CLI de AWS.

También necesitará el comando AWS IAM Authenticator for Kubernetes (`aws eks get-token`o `aws-iam-authenticator` bien (disponible en la versión 1.16.156 o superior de la CLI de AWS) en su. `PATH`

La cuenta de IAM utilizada para la creación del clúster de EKS debe tener estos niveles de acceso mínimos.

Servicio de AWS	Nivel de acceso
CloudFormation	Acceso completo
EC2	Completo: Tagging Limited: Listar, leer, escribir
EC2 Auto Scaling	Limitado: lista, escritura
EKS	Acceso completo
IAM	Limitado: gestión de listas, lectura, escritura y permisos
Systems Manager	Limited (Limitado): List, (Enumerar), Read (Lectura)

Para Unix

Para descargar la versión más reciente, ejecute:

```
# 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
```

Para Windows

Descarga directa (última versión):

- AMD64/x86_64
- ARMv6
- ARMv7
- ARM64

Asegúrese de descomprimir el archivo en una carpeta de la variable. `PATH`

Si lo desea, compruebe la suma de comprobación:

1. Descargue el archivo de suma de comprobación: el más reciente
2. Utilice la línea de comandos para comparar manualmente `CertUtil` el resultado con el archivo de suma de comprobación descargado.

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

3. Se utiliza PowerShell para automatizar la verificación mediante el `-eq` operador para obtener un `False` resultado `True` o:

```
# 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]
```

Uso de 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
```

El `eksctl` ejecutable se coloca en`$HOME/bin`, que proviene `$PATH` de Git Bash.

Homebrew

Puedes usar Homebrew para instalar software en macOS y Linux.

AWS mantiene una versión de Homebrew que incluye eksctl.

Para obtener más información sobre el grifo Homebrew, consulte el proyecto en Github y la fórmula Homebrew para eksctl.

Para instalar eksctl con Homebrew

1. (Requisito previo) Instale Homebrew
2. Añada el grifo de AWS

```
brew tap aws/tap
```

3. Instalación de eksctl

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

Docker

Para cada versión y RC, se envía una imagen de contenedor al repositorio `public.ecr.aws/eksctl/eksctl` de ECR. Obtenga más información sobre el uso en ECR Public Gallery: eksctl. Por ejemplo:

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

Finalización de Shell

Bash

Para habilitar la finalización de bash, ejecuta lo siguiente o ponlo en `~/.bashrc` o`~/.profile`:

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

Zash

Para completar zsh, ejecute:

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

e introduce lo siguiente: `~/.zshrc`

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

Ten en cuenta que si no estás ejecutando una distribución como esta, oh-my-zsh es posible que primero tengas que habilitar la función de autocompletado (e `~/.zshrc` insertarla para que sea persistente):

```
autoload -U compinit
compinit
```

Pescado

Los siguientes comandos se pueden usar para completar automáticamente los peces:

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

Powershell

Puede hacer referencia al siguiente comando para configurarlo. Tenga en cuenta que la ruta puede ser diferente en función de la configuración del sistema.

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

Actualizaciones

> **Important**
>
> Si instala eksctl descargándolo directamente (sin usar un administrador de paquetes), debe actualizarlo manualmente.

Clústeres

Este capítulo trata sobre la creación y configuración de clústeres de EKS mediante eksctl. También incluye complementos y el modo automático EKS.

Temas:

- the section called “Entradas de acceso a EKS”
  - Simplifique la administración del RBAC de Kubernetes sustituyendo aws-auth por entradas de acceso EKS ConfigMap
  - Migre las asignaciones de identidad de IAM existentes de aws-auth a las entradas de acceso ConfigMap
  - Configure los modos de autenticación del clúster y controle los permisos de administrador del creador del clúster
- the section called “Actualizaciones complementarias predeterminadas”
  - Mantenga los clústeres seguros actualizando los complementos de EKS predeterminados en los clústeres más antiguos
- the section called “Complementos”
  - Automatice las tareas rutinarias de instalación, actualización y eliminación de complementos.
  - Los complementos de Amazon EKS incluyen complementos de AWS, complementos de comunidad de código abierto y complementos de Marketplace.
- the section called “Modo automático de EKS”
  - Reduzca los gastos operativos al permitir que AWS administre su infraestructura de EKS
  - Configure grupos de nodos personalizados en lugar de los grupos de sistemas y de uso general predeterminados
  - Convierta los clústeres EKS existentes para usar el modo automático
- the section called “CloudWatch registro”
  - Solucione los problemas del clúster habilitando los registros para componentes específicos del plano de control EKS
  - Configure los períodos de retención de registros para los registros del clúster de EKS
  - Modifique la configuración de registro del clúster existente mediante los comandos eksctl
- the section called “Actualizaciones de clústeres”

  - Mantenga la seguridad y la estabilidad actualizando de forma segura las versiones del plano de control EKS
  - Implemente actualizaciones en todos los grupos de nodos sustituyendo los grupos antiguos por otros nuevos
  - Actualiza los complementos de clúster predeterminados
- the section called “Creación y administración de clústeres”
  - Comience rápidamente con los clústeres de EKS básicos utilizando los grupos de nodos gestionados de forma predeterminada
  - Cree clústeres personalizados mediante archivos de configuración con configuraciones específicas
  - Implemente los clústeres existentes VPCs con redes privadas y políticas de IAM personalizadas
- the section called “Configura kubelet”
  - Evite la escasez de recursos de los nodos configurando las reservas de kubelet y daemon del sistema
  - Personalice los umbrales de desalojo según la disponibilidad de la memoria y el sistema de archivos
  - Habilite o deshabilite las funciones específicas de kubelet en los grupos de nodos
- the section called “Conector EKS”
  - Centralice la administración de las implementaciones híbridas de Kubernetes a través de la consola EKS
  - Configure las funciones y los permisos de IAM para el acceso a clústeres externos
  - Elimine los clústeres externos y limpie los recursos de AWS asociados
- the section called “Clúster totalmente privado de EKS”
  - Cumpla los requisitos de seguridad con clústeres EKS totalmente privados que no tienen acceso saliente a Internet
  - Configure el acceso privado a los servicios de AWS a través de puntos de enlace de VPC
  - Cree y administre grupos de nodos privados con configuraciones de red explícitas
- the section called “Soporte de Karpenter”
  - Automatice el aprovisionamiento de nodos
  - Cree configuraciones personalizadas de aprovisionadores de Karpenter
  - Configure Karpenter con un sistema de gestión de interrupciones de instancias puntuales
- the section called “Amazon EMR”

  - Cree un mapeo de identidad de IAM entre el clúster EMR y EKS
- the section called “Soporte de EKS Fargate”
  - Defina perfiles Fargate personalizados para la programación de cápsulas
  - Administre los perfiles de Fargate mediante actualizaciones de creación y configuración
- the section called “Clústeres no creados por eksctl”
  - Estandarice la gestión de los clústeres creados fuera de eksctl
  - Utilice los comandos eksctl en clústeres existentes que no sean eksctl
- the section called “Habilitación del cambio de zona”
  - Mejore la disponibilidad de las aplicaciones al habilitar capacidades rápidas de conmutación por error de zona
  - Configure el cambio zonal en las nuevas implementaciones de clústeres de EKS
  - Habilite las funciones de cambio zonal en los clústeres de EKS existentes

Creación y administración de clústeres

En este tema se explica cómo crear y eliminar clústeres de EKS mediante Eksctl. Puede crear clústeres con un comando CLI o mediante la creación de un archivo YAML de configuración de clústeres.

Crear un clúster sencillo

Cree un clúster simple con el siguiente comando:

```
eksctl create cluster
```

Esto creará un clúster de EKS en su región predeterminada (según lo especificado en su configuración de AWS CLI) con un grupo de nodos administrado que contiene dos nodos m5.large.

eksctl ahora crea un grupo de nodos administrado de forma predeterminada cuando no se utiliza un archivo de configuración. Para crear un grupo de nodos autogestionado, pasa a o. `--managed=false` `eksctl create cluster` `eksctl create nodegroup`

Consideraciones

- Al crear clústeres en`us-east-1`, es posible que encuentres un. `UnsupportedAavailabilityZoneException` Si esto sucede, copia las zonas sugeridas y

pasa la `--zones` bandera, por ejemplo:`eksctl create cluster --region=us-east-1 --zones=us-east-1a,us-east-1b,us-east-1d`. Este problema puede ocurrir en otras regiones, pero es menos común. En la mayoría de los casos, no necesitarás usar la `--zone` bandera.

Cree un clúster mediante el archivo de configuración

Puede crear un clúster mediante un archivo de configuración en lugar de indicadores.

Primero, crea `cluster.yaml` el archivo:

```
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
```

A continuación, ejecuta este comando:

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

Esto creará un clúster tal y como se describe.

Si necesitabas usar una VPC existente, puedes usar un archivo de configuración como este:

```
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
```

El nombre del clúster o del grupo de nodos debe contener únicamente caracteres alfanuméricos (distingue mayúsculas de minúsculas) y guiones. Debe empezar con un carácter alfabético y no puede superar los 128 caracteres o recibirá un error de validación. Para obtener más información, consulte Crear una pila desde la CloudFormation consola en la guía del usuario de AWS CLoud Formation.

Actualice kubeconfig para un nuevo clúster

Una vez creado el clúster, se añadirá la configuración de kubernetes adecuada a tu archivo kubeconfig. Es el archivo que configuraste en la variable de entorno o de forma predeterminada. `KUBECONFIG` `~/.kube/config` La ruta al archivo kubeconfig se puede anular con la marca. `--kubeconfig`

Otros indicadores que pueden cambiar la forma en que se escribe el archivo kubeconfig:

marca	type	usar	valor predeterminado
--kubeconfig	cadena	ruta para escribir kubeconfig (incompatible con --auto-kubeconfig)	$KUBECONFIG o ~/.kube/config
--set-kubeconfig-context	bool	si es verdadero, el contexto actual se establecerá en kubeconfig; si un contexto ya está establecido, se sobrescribirá	true
--configuración automática de kube	bool	guarde el archivo kubeconfig por nombre de clúster	true
--write-kubeconfig	bool	alternar la escritura de kubeconfig	true

Eliminación de un clúster

Para eliminar este clúster, ejecuta:

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

> **Warning**
>
> Use la `--wait` marca con las operaciones de eliminación para asegurarse de que los errores de eliminación se notifiquen correctamente.

Sin la `--wait` marca, eksctl solo emitirá una operación de eliminación en la CloudFormation pila del clúster y no esperará a que se elimine. En algunos casos, los recursos de AWS que utilizan el clúster o su VPC pueden provocar un error al eliminar el clúster. Si se produce un error al eliminar o si se

olvida del indicador de espera, puede que tenga que ir a la CloudFormation GUI y eliminar las pilas eks desde allí.

> **Warning**
>
> Las políticas de PDB pueden bloquear la eliminación de nodos durante la eliminación del clúster.

Al eliminar un clúster con grupos de nodos, las políticas de Pod Disruption Budget (PDB) pueden impedir que los nodos se eliminen correctamente. Por ejemplo, los clústeres que están `aws-ebs-csi-driver` instalados suelen tener dos módulos con una política de PDB que permite que solo un módulo no esté disponible a la vez, por lo que el otro no se puede desalojar durante la eliminación. Para eliminar correctamente el clúster en estos escenarios, usa la `disable-nodegroup-eviction` marca para omitir las comprobaciones de las políticas de PDB:

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

Consulta el `examples/`directorio del GitHub repositorio de eksctl para ver más ejemplos de archivos de configuración.

Funcionamiento en seco

La función de ejecución en seco permite generar un ClusterConfig archivo que omite la creación del clúster y genera un ClusterConfig archivo que representa las opciones de CLI suministradas y contiene los valores predeterminados establecidos por eksctl.

Puede encontrar más información en la página de ensayo en seco.

Modo automático de EKS

eksctl es compatible con el modo automático de EKS, una función que extiende la administración de los clústeres de Kubernetes en AWS más allá del propio clúster, lo que permite a AWS configurar y gestionar también la infraestructura que permite el buen funcionamiento de sus cargas de trabajo. Esto le permite delegar decisiones clave de infraestructura y aprovechar la experiencia de AWS para day-to-day las operaciones. La infraestructura de clústeres administrada por AWS incluye muchas capacidades de Kubernetes como componentes principales, a diferencia de los complementos, como el escalado automático de cómputo, las redes de pods y servicios, el equilibrio de carga de aplicaciones, el DNS del clúster, el almacenamiento en bloques y la compatibilidad con GPU.

Creación de un clúster de EKS con el modo automático activado

`eksctl`ha añadido un nuevo `autoModeConfig` campo para activar y configurar el modo automático. La forma del `autoModeConfig` campo es

```
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
```

Si `autoModeConfig.enabled` es cierto, eksctl crea un clúster de EKS pasando y `storageConfig.blockStorage.enabled: true` a la API de EKS `computeConfig.enabled: truekubernetesNetworkConfig.elasticLoadBalancing.enabled: true`, lo que permite la gestión de los componentes del plano de datos, como la computación, el almacenamiento y las redes.

Para crear un clúster EKS con el modo automático activado, `autoModeConfig.enabled: true` defina, como en

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

autoModeConfig:
    enabled: true
```

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

eksctl crea un rol de nodo para usarlo en los nodos lanzados por el modo automático. eksctl también crea los grupos de nodos y. `general-purpose` `system` Para deshabilitar la creación de los grupos de nodos predeterminados, por ejemplo, para configurar sus propios grupos de nodos que usen un conjunto diferente de subredes, defina, como `nodePools: []`

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

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

Actualización de un clúster de EKS para usar el modo automático

Para actualizar un clúster EKS existente para que utilice el modo automático, ejecute

```
# 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**
>
> Si el clúster lo creó eksctl y utiliza subredes públicas como subredes de clúster, el modo automático lanzará nodos en las subredes públicas. Para usar subredes privadas para los nodos de trabajo lanzados por Auto Mode, actualice el clúster para que use subredes privadas.

Desactivar el modo automático

Para deshabilitar el modo automático, configúrelo `autoModeConfig.enabled: false` y ejecútelo

```
# 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
```

Más información

- Modo automático de EKS

Entradas de acceso a EKS

Puede utilizar eksctl para gestionar las entradas de acceso a EKS. Utilice las entradas de acceso para conceder permisos de Kubernetes a las identidades de IAM de AWS. Por ejemplo, puede conceder a un rol de desarrollador un permiso para leer los recursos de Kubernetes en un clúster.

En este tema se explica cómo usar eksctl para administrar las entradas de acceso. Para obtener información general sobre las entradas de acceso, consulte Otorgar a los usuarios de IAM acceso a Kubernetes con entradas de acceso EKS.

Puede adjuntar las políticas de acceso de Kubernetes definidas por AWS o asociar una identidad de IAM a un grupo de Kubernetes.

Para obtener más información sobre las políticas predefinidas disponibles, consulte Asociar políticas de acceso a entradas de acceso.

Si necesita definir las políticas de Kubernetes del cliente, asocie la identidad de IAM a un grupo de Kubernetes y conceda permisos a ese grupo.

Modo de autenticación de clústeres

Solo puede usar entradas de acceso si el modo de autenticación del clúster lo permite.

Para obtener más información, consulte Establecer el modo de autenticación de clúster

Configure el modo de autenticación con un archivo YAML

`eksctl`ha añadido un nuevo `accessConfig.authenticationMode` campo en ClusterConfig el que se puede establecer uno de los tres valores siguientes:

- `CONFIG_MAP`- predeterminado en la API EKS, solo se `aws-auth` ConfigMap usará
- `API`- solo se utilizará la API de entradas de acceso
- `API_AND_CONFIG_MAP`- entrada predeterminada`eksctl`: se pueden utilizar `aws-auth` ConfigMap tanto la API como la API de entradas de acceso

Configura el modo de autenticación en ClusterConfig YAML:

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

Actualice el modo de autenticación con un comando

Si desea utilizar entradas de acceso en un clúster ya existente y no creado por eksctl, en el que se utilice la `CONFIG_MAP` opción, el usuario tendrá que configurarla primero en. `authenticationMode` `API_AND_CONFIG_MAP` Para ello, `eksctl` ha introducido un nuevo comando para actualizar el modo de autenticación del clúster, que funciona tanto con los indicadores CLI, p. ej.

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

Acceda a los recursos de entrada

Las entradas de acceso tienen un tipo, como `STANDARD` o`EC2_LINUX`. El tipo depende de cómo se utilice la entrada de acceso.

- El `standard` tipo es para conceder permisos de Kubernetes a los usuarios y roles de IAM.
  - Por ejemplo, puede ver los recursos de Kubernetes en la consola de AWS adjuntando una política de acceso al rol o al usuario que utilice para acceder a la consola.
- Los `EC2_WINDOWS` tipos `EC2_LINUX` y sirven para conceder permisos de Kubernetes a las instancias. EC2 Las instancias utilizan estos permisos para unirse al clúster.

Para obtener más información sobre los tipos de entradas de acceso, consulte Crear entradas de acceso

Entidades de IAM

Puede utilizar las entradas de acceso para conceder permisos de Kubernetes a las identidades de IAM, como los usuarios y las funciones de IAM.

Utilice el `accessConfig.accessEntries` campo para asociar el ARN de un recurso de IAM a una API EKS de Access Entries. Por ejemplo:

```
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
```

Además de asociar las políticas de EKS, también se pueden especificar los grupos de Kubernetes a los que pertenece una entidad de IAM y, por lo tanto, conceder los permisos mediante RBAC.

Fargate y grupos de nodos gestionados

La integración con las entradas de acceso a estos recursos se logrará entre bastidores, mediante la API EKS. Los grupos de nodos gestionados y los pods de Fargate recién creados crearán entradas de acceso a la API, en lugar de utilizar recursos RBAC precargados. Los grupos de nodos y los pods de Fargate existentes no cambiarán y seguirán dependiendo de las entradas del mapa de configuración de aws-auth.

Grupos de nodos autogestionados

Cada entrada de acceso tiene un tipo. Para autorizar grupos de nodos autogestionados, `eksctl` creará una entrada de acceso única para cada grupo de nodos con el ARN principal establecido en el ARN del rol de nodo y el tipo establecido en AMIFamily o en función del grupo de nodos. `EC2_LINUX` `EC2_WINDOWS`

Al crear sus propias entradas de acceso, también puede especificar `EC2_LINUX` (para una función de IAM utilizada con nodos autogestionados de Linux o Bottlerocket), `EC2_WINDOWS` (para una función de IAM utilizada con nodos autogestionados de Windows), `FARGATE_LINUX` (para una función de IAM utilizada con AWS Fargate (Fargate)) o como un tipo. `STANDARD` Si no especifica un tipo, el tipo predeterminado se establece en. `STANDARD`

> **Note**
>
> Al eliminar un grupo de nodos creado con uno ya existente`instanceRoleARN`, es responsabilidad del usuario eliminar la entrada de acceso correspondiente cuando no haya más grupos de nodos asociados a ella. Esto se debe a que eksctl no intenta averiguar si grupos de nodos autogestionados no creados por eksctl siguen utilizando una entrada de acceso, ya que se trata de un proceso complicado.

Crea una entrada de acceso

Esto se puede hacer de dos maneras diferentes: durante la creación del clúster, especificando las entradas de acceso deseadas como parte del archivo de configuración y ejecutándolo:

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

O después de la creación del clúster, ejecuta:

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

Para ver un ejemplo de archivo de configuración para crear entradas de acceso, consulta 40-access-entries.yaml en el repositorio eksctl. GitHub

Obtenga la entrada de acceso

El usuario puede recuperar todas las entradas de acceso asociadas a un clúster determinado ejecutando una de las siguientes acciones:

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

OR

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

Como alternativa, para recuperar solo la entrada de acceso correspondiente a una determinada entidad de IAM, se debe utilizar la `--principal-arn` bandera, p. ej.

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

Eliminar la entrada de acceso

Para eliminar una sola entrada de acceso a la vez, utilice:

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

Para eliminar varias entradas de acceso, utilice la `--config-file` marca y especifique todas las entradas `principalARN's` correspondientes a las entradas de acceso en el `accessEntry` campo de nivel superior, p. ej.

```
...
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
```

Migre desde aws-auth ConfigMap

El usuario puede migrar sus identidades de IAM existentes de `aws-auth` configmap a las entradas de acceso ejecutando lo siguiente:

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

Cuando el `--target-authentication-mode` indicador está establecido en`API`, el modo de autenticación cambia a `API` modo (se omite si ya está en `API` modo), las asignaciones de identidad de IAM se migrarán a las entradas de acceso y `aws-auth` configmap se eliminará del clúster.

Cuando el `--target-authentication-mode` indicador está establecido en`API_AND_CONFIG_MAP`, el modo de autenticación cambia al modo (se omite si ya está en `API_AND_CONFIG_MAP` `API_AND_CONFIG_MAP` modo), las asignaciones de identidad de IAM se migrarán a las entradas de acceso, pero se conservará el mapa de configuración. `aws-auth`

> **Note**
>
> Si el `--target-authentication-mode` indicador está establecido en`API`, este comando no actualizará el modo de autenticación al `API` modo si `aws-auth` configmap tiene una de las siguientes restricciones.

- Hay un mapeo de identidad a nivel de cuenta.
- Uno o más Roles/Users están asignados a los grupos de kubernetes que comienzan con un prefijo `system:` (excepto los grupos específicos de EKS, es decir, `system:masters` etc.). `system:bootstrappers` `system:nodes`
- Uno o más mapas de identidad de IAM corresponden a un [rol vinculado al servicio] (enlace: -.html). IAM/latest/UserGuide/using service-linked-roles

Inhabilite los permisos de administrador del creador del clúster

`eksctl`ha agregado un nuevo campo `accessConfig.bootstrapClusterCreatorAdminPermissions: boolean` que, cuando se

establece en falso, deshabilita la concesión de permisos de administrador del clúster a la identidad de IAM que crea el clúster, es decir

añada la opción al archivo de configuración:

```
accessConfig:
  bootstrapClusterCreatorAdminPermissions: false
```

y ejecuta:

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

Clústeres no creados por eksctl

Puede ejecutar `eksctl` comandos en clústeres que no fueron creados por. `eksctl`

> **Note**
>
> Eksctl solo admite clústeres sin propietario con nombres que sean compatibles con AWS. CloudFormation Cualquier nombre de clúster que no coincida con este valor no pasará la comprobación de validación de la CloudFormation API.

Comandos admitidos

Los siguientes comandos se pueden usar en clústeres creados por cualquier medio que no sea`eksctl`. Los comandos, los indicadores y las opciones del archivo de configuración se pueden usar exactamente de la misma manera.

Si nos hemos perdido alguna funcionalidad, háznoslo saber.

- ✓ Crear:
  - ✓ `eksctl create nodegroup` (consulte la nota a continuación)
  - ✓ `eksctl create fargateprofile`
  - ✓ `eksctl create iamserviceaccount`
  - ✓ `eksctl create iamidentitymapping`
- ✓ Obtenga:

- ✓ `eksctl get clusters/cluster`
- ✓ `eksctl get fargateprofile`
- ✓ `eksctl get nodegroup`
- ✓ `eksctl get labels`

✓ Eliminar:

- ✓ `eksctl delete cluster`
- ✓ `eksctl delete nodegroup`
- ✓ `eksctl delete fargateprofile`
- ✓ `eksctl delete iamserviceaccount`
- ✓ `eksctl delete iamidentitymapping`

✓ Actualización:

- ✓ `eksctl upgrade cluster`
- ✓ `eksctl upgrade nodegroup`

✓ Configurar/desactivar:

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

✓ Escala:

- ✓ `eksctl scale nodegroup`

✓ Desagüe:

- ✓ `eksctl drain nodegroup`

✓ Habilitar:

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

✓ Utilidades:

- ✓ `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`

Creación de grupos de nodos

`eksctl create nodegroup`es el único comando que requiere una entrada específica por parte del usuario.

Como los usuarios pueden crear sus clústeres con cualquier configuración de red que deseen, por el momento no `eksctl` intentarán recuperar ni adivinar estos valores. Esto puede cambiar en el futuro a medida que aprendamos más sobre cómo las personas utilizan este comando en clústeres no creados por eksctl.

Esto significa que para crear grupos de nodos o grupos de nodos gestionados en un clúster que no fue creado por él`eksctl`, se debe proporcionar un archivo de configuración que contenga los detalles de la VPC. Como mínimo:

```
---
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"

...
```

Para obtener más información sobre las opciones de configuración de la VPC, consulte Redes.

Registrar clústeres que no son de EKS con el conector EKS

Puede usar el conector EKS para ver clústeres fuera de AWS en la consola EKS. Este proceso requiere registrar el clúster en EKS y ejecutar el agente EKS Connector en el clúster de Kubernetes externo.

`eksctl`simplifica el registro de clústeres que no son de EKS al crear los recursos de AWS necesarios y generar manifiestos de Kubernetes para que EKS Connector los aplique al clúster externo.

Registre el clúster

Para registrar o conectar un clúster de Kubernetes que no sea de EKS, ejecute

```
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
```

Este comando registrará el clúster y escribirá tres archivos que contienen los manifiestos de Kubernetes para EKS Connector y que deberán aplicarse al clúster externo antes de que caduque el registro.

> **Note**
>
> `eks-connector-clusterrole.yamlgety eks-connector-console-dashboard-full-access-clusterrole.yaml` otorga `list` permisos para los recursos de Kubernetes en todos los espacios de nombres a la identidad de IAM que realiza la llamada y, si es necesario, deben editarse en consecuencia antes de aplicarlos al clúster. Para configurar un acceso más restringido, consulta Cómo conceder acceso a un usuario para ver un clúster.

Para proporcionar una función de IAM existente para utilizarla en EKS Connector, pásela `--role-arn` como se indica en:

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

Si el clúster ya existe, eksctl devolverá un error.

Anule el registro del clúster

Para anular el registro de un clúster registrado o desconectarlo, ejecute

```
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
```

Este comando anulará el registro del clúster externo y eliminará sus recursos de AWS asociados, pero debe eliminar los recursos de Kubernetes del conector EKS del clúster.

Más información

- Conector EKS

Personalización de la configuración de kubelet

Los recursos del sistema se pueden reservar mediante la configuración del kubelet. Esto se recomienda porque, en caso de escasez de recursos, es posible que el kubelet no pueda desalojar

los pods y, finalmente, hacer que el nodo se convierta en uno. `NotReady` Para ello, los archivos de configuración pueden incluir un `kubeletExtraConfig` campo que acepte un yaml de formato libre que se incrustará en el. `kubelet.yaml`

Algunos campos de los `kubelet.yaml` están configurados por eksctl y, por lo tanto, no se pueden sobrescribir, como el,,`address`, `clusterDomain` o. `authentication` `authorization` `serverTLSBootstrap`

El siguiente archivo de configuración de ejemplo crea un grupo `300Mi` de nodos que reserva la `300m` vCPU, la memoria y `1Gi` el almacenamiento efímero para el kubelet; la `300m` vCPU`300Mi`, la memoria `1Gi` y el almacenamiento efímero para los demonios del sistema operativo; y activa el desalojo de los pods cuando hay menos de memoria disponible o menos del 10% del sistema de archivos raíz. `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
```

En este ejemplo, dadas las instancias de tipo `m5a.xlarge` que tienen 4 V CPUs y 16 GiB de memoria, la `Allocatable` cantidad de CPUs sería de 3,4 y 15,4 GiB de memoria. Es importante saber que los valores especificados en el archivo de configuración para los campos de entrada `kubeletExtraconfig` sobrescribirán por completo los valores predeterminados especificados por eksctl. Sin embargo, si se omite uno o más `kubeReserved` parámetros, los parámetros faltantes se establecerán de forma predeterminada con valores idénticos en función del tipo de instancia de AWS que se utilice.

`kubeReserved`cálculo

Si bien generalmente se recomienda configurar una instancia mixta NodeGroup para usar instancias con la misma configuración de CPU y RAM, no es un requisito estricto. Por lo tanto, el `kubeReserved` cálculo utiliza la instancia más pequeña del `InstanceDistribution.InstanceTypes` campo. De esta forma, NodeGroups con tipos de instancias dispares, no se reservarán demasiados recursos en la instancia más pequeña. Sin embargo, esto podría provocar una reserva demasiado pequeña para el tipo de instancia más grande.

> **Warning**
>
> De forma predeterminada`featureGates.RotateKubeletServerCertificate=true`, se `eksctl` establece, pero cuando `featureGates` se proporciona una configuración personalizada, no se establecerá. Siempre debes incluirlo`featureGates.RotateKubeletServerCertificate=true`, a menos que tengas que deshabilitarlo.

CloudWatch registro

En este tema se explica cómo configurar el CloudWatch registro de Amazon para los componentes del plano de control del clúster EKS. CloudWatch El registro proporciona visibilidad de las operaciones del plano de control del clúster, algo esencial para solucionar problemas, auditar las actividades del clúster y supervisar el estado de los componentes de Kubernetes.

Habilitar el registro CloudWatch

CloudWatch El registro del plano de control EKS no está activado de forma predeterminada debido a los costes de ingesta y almacenamiento de datos.

Para habilitar el registro en el plano de control cuando se crea el clúster, tendrá que definir una **`cloudWatch.clusterLogging.enableTypes`**configuración en el suyo `ClusterConfig` (consulte los ejemplos a continuación).

Por lo tanto, si tiene un archivo de configuración con la **`cloudWatch.clusterLogging.enableTypes`**configuración correcta, puede crear un clúster con él`eksctl create cluster --config-file=<path>`.

Si ya ha creado un clúster, puede usarlo`eksctl utils update-cluster-logging`.

> **Note**
>
> Si este comando se ejecuta en modo plan de forma predeterminada, tendrá que especificar un `--approve` indicador para aplicar los cambios a su clúster.

Si utilizas un archivo de configuración, ejecuta:

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

Como alternativa, puede usar indicadores CLI.

Para habilitar todos los tipos de registros, ejecute:

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

Para habilitar `audit` los registros, ejecuta:

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

Para habilitar todos los `controllerManager` registros excepto los registros, ejecuta:

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

Si los tipos de `scheduler` registro `api` y B ya estaban habilitados, para `scheduler` deshabilitarlos y habilitarlos `controllerManager` al mismo tiempo, ejecute:

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

Esto dejará `api` y `controllerManager` como los únicos tipos de registro activados.

Para deshabilitar todos los tipos de registros, ejecute:

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

Ejemplos de `ClusterConfig`

En un clúster de EKS, el `enableTypes` campo de abajo `clusterLogging` puede incluir una lista de valores posibles para habilitar los distintos tipos de registros para los componentes del plano de control.

A continuación se muestran los posibles valores:

- `api`: Habilita los registros del servidor de la API de Kubernetes.
- `audit`: Habilita los registros de auditoría de Kubernetes.
- `authenticator`: Habilita los registros del autenticador.
- `controllerManager`: Habilita los registros del administrador del controlador de Kubernetes.
- `scheduler`: Habilita los registros del programador de Kubernetes.

Para obtener más información, consulte la documentación de EKS.

Deshabilite todos los registros

Para deshabilitar todos los tipos, utilice `[]` o elimine la `cloudWatch` sección por completo.

Habilite todos los registros

Puede activar todos los tipos con `"*"` o`"all"`. Por ejemplo:

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

Habilite uno o más registros

Puede habilitar un subconjunto de tipos enumerando los tipos que desea habilitar. Por ejemplo:

```
cloudWatch:
```

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

Periodo de retención de registros

De forma predeterminada, los registros se almacenan en CloudWatch Registros de forma indefinida. Puede especificar el número de días durante los que se deben conservar los registros del plano de control en CloudWatch los registros. El siguiente ejemplo conserva los registros durante 7 días:

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

Ejemplo completo

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

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

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

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

Clúster totalmente privado de EKS

eksctl admite la creación de clústeres totalmente privados que no tienen acceso saliente a Internet y que solo tienen subredes privadas. Los puntos de enlace de VPC se utilizan para permitir el acceso privado a los servicios de AWS.

Esta guía describe cómo crear un clúster privado sin acceso saliente a Internet.

Cómo crear un clúster totalmente privado

El único campo obligatorio para crear un clúster totalmente privado es: `privateCluster.enabled`

```
privateCluster:
  enabled: true
```

Tras la creación del clúster, los comandos eksctl que necesiten acceso al servidor API de Kubernetes deberán ejecutarse desde la VPC del clúster, una VPC interconectada o mediante algún otro medio, como AWS Direct Connect. Los comandos eksctl que necesiten acceso al EKS no APIs funcionarán si se ejecutan desde la VPC del clúster. Para solucionar este problema, cree un punto de enlace de interfaz para que Amazon EKS pueda acceder de forma privada a la administración de Amazon Elastic Kubernetes Service (Amazon EKS APIs) desde su Amazon Virtual Private Cloud (VPC). En una versión futura, eksctl añadirá soporte para crear este punto final, por lo que no será necesario crearlo manualmente. Los comandos que necesiten acceder a la URL del proveedor de OpenID Connect deberán ejecutarse desde fuera de la VPC del clúster una vez que haya habilitado AWS para PrivateLink Amazon EKS.

La creación de grupos de nodos administrados seguirá funcionando y la creación de grupos de nodos autogestionados funcionará, ya que se necesita acceso al servidor de API a través del punto final de la interfaz EKS si el comando se ejecuta desde la VPC del clúster, una VPC interconectada o mediante algún otro medio, como AWS Direct Connect.

> **Note**
>
> Los puntos finales de VPC se cobran por hora y en función del uso. Puede encontrar más información sobre los precios en los PrivateLink precios de AWS

> **Warning**
>
> No se admiten clústeres totalmente privados. `eu-south-1`

Configuración del acceso privado a servicios de AWS adicionales

Para permitir que los nodos de trabajo accedan a los servicios de AWS de forma privada, eksctl crea puntos de enlace de VPC para los siguientes servicios:

- Puntos de enlace de interfaz para que ECR (ambos `ecr.apiecr.dkr`) extraiga imágenes de contenedores (complemento CNI de AWS, etc.)
- Un punto de enlace para que S3 extraiga las capas de imágenes reales
- Un punto final de interfaz para EC2 los requisitos de la `aws-cloud-provider` integración
- Un punto final de interfaz para que STS admita las funciones de Fargate e IAM para cuentas de servicios (IRSA)
- Un punto final de interfaz para el CloudWatch registro (`logs`) si CloudWatch el registro está activado

Estos puntos finales de VPC son esenciales para que un clúster privado funcione y, como tal, eksctl no admite su configuración ni desactivación. Sin embargo, es posible que un clúster necesite acceso privado a otros servicios de AWS (por ejemplo, el escalado automático requerido por el escalador automático del clúster). Estos servicios se pueden especificar en`privateCluster.additionalEndpointServices`, que indica a eksctl que cree un punto final de VPC para cada uno de ellos.

Por ejemplo, para permitir el acceso privado al escalado automático y al registro: CloudWatch

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

Los puntos finales compatibles `additionalEndpointServices` son`autoscaling`, `ycloudformation`. `logs`

Omitir la creación de puntos finales

Si ya se ha creado una VPC con los puntos de enlace de AWS necesarios configurados y vinculados a las subredes descritas en la documentación de EKS, `eksctl` puede omitir su creación proporcionando la siguiente opción: `skipEndpointCreation`

```
privateCluster:
  enabled: true
```

```
  skipEndpointCreation: true
```

Esta configuración no se puede usar junto con. `additionalEndpointServices` Se omitirá la creación de todos los puntos finales. Además, esta configuración solo se recomienda si la topología de <# subred del punto final está configurada correctamente. Si los identificadores de subred son correctos, el `vpce` enrutamiento se configura con direcciones de prefijo y se crean todos los puntos finales de EKS necesarios y se vinculan a la VPC proporcionada. `eksctl`no alterará ninguno de estos recursos.

Grupos de nodos

En un clúster totalmente privado, solo se admiten grupos de nodos privados (tanto gestionados como autogestionados), ya que la VPC del clúster se crea sin subredes públicas. El `privateNetworking` campo (y) se debe establecer de forma `nodeGroup[]`**`.privateNetworking`** explícita`managedNodeGroup[`. Es un error dejarlo `privateNetworking` sin configurar en un clúster totalmente privado.

```
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
```

Acceso al punto de enlace del clúster

Un clúster totalmente privado no admite la modificación `clusterEndpointAccess` durante la creación del clúster. Es un error configurar una `clusterEndpoints.publicAccess` u otra opción`clusterEndpoints.privateAccess`, ya que un clúster totalmente privado solo puede tener acceso privado y permitir la modificación de estos campos puede interrumpir el clúster.

VPC y subredes suministradas por el usuario

eksctl admite la creación de clústeres totalmente privados mediante una VPC y subredes preexistentes. Solo se pueden especificar subredes privadas y es un error especificar subredes en ellas. `vpc.subnets.public`

eksctl crea puntos de enlace de VPC en la VPC proporcionada y modifica las tablas de enrutamiento de las subredes proporcionadas. Cada subred debe tener asociada una tabla de rutas explícita, ya que eksctl no modifica la tabla de rutas principal.

```
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
```

Administrar un clúster totalmente privado

Para que todos los comandos funcionen después de la creación del clúster, eksctl necesitará acceso privado al punto final del servidor API de EKS y acceso saliente a Internet (para). `EKS:DescribeCluster` Se admitirán los comandos que no necesiten acceso al servidor API si eksctl tiene acceso saliente a Internet.

Eliminar por la fuerza un clúster totalmente privado

Es probable que se produzcan errores al eliminar un clúster totalmente privado a través de eksctl, ya que eksctl no tiene acceso automático a todos los recursos del clúster. `--force`existe para resolver esto: forzará la eliminación del clúster y continuará cuando se produzcan errores.

Limitaciones

Una limitación de la implementación actual es que eksctl crea inicialmente el clúster con el acceso a los puntos finales públicos y privados habilitados, y deshabilita el acceso a los puntos finales públicos una vez que se hayan completado todas las operaciones. Esto es necesario porque eksctl necesita acceso al servidor API de Kubernetes para permitir que los nodos autogestionados se unan al clúster y admitan Fargate. GitOps Una vez finalizadas estas operaciones, eksctl cambia el acceso al punto final del clúster a un acceso exclusivamente privado. Esta actualización adicional significa que la creación de un clúster totalmente privado llevará más tiempo que la de un clúster estándar. En el futuro, eksctl puede cambiar a una función Lambda habilitada para VPC para realizar estas operaciones de API.

Acceso saliente a través de servidores proxy HTTP

eksctl puede comunicarse con la AWS a APIs través de un servidor proxy HTTP (S) configurado, sin embargo, deberá asegurarse de configurar su lista de exclusión de proxy correctamente.

Por lo general, tendrá que asegurarse de que las solicitudes del punto final de la VPC de su clúster no se enruten a través de sus proxies. Para ello, debe configurar una variable de `no_proxy` entorno adecuada que incluya el valor. `.eks.amazonaws.com`

Si su servidor proxy realiza una «interceptación de SSL» y utiliza las funciones de IAM para las cuentas de servicio (IRSA), tendrá que asegurarse de omitir explícitamente el SSL para el dominio. Man-in-the-Middle `oidc.<region>.amazonaws.com` Si no lo hace, eksctl obtendrá una huella digital incorrecta del certificado raíz para el proveedor de OIDC y el complemento CNI para VPC de AWS no se iniciará debido a que no puede obtener las credenciales de IAM, lo que hará que su clúster no funcione.

Más información

- Clústeres privados de EKS

Complementos

En este tema se describe cómo administrar los complementos de Amazon EKS para sus clústeres de Amazon EKS mediante eksctl. Los complementos de EKS son una función que le permite habilitar y administrar el software operativo de Kubernetes a través de la API de EKS, lo que simplifica el proceso de instalación, configuración y actualización de los complementos de clúster.

> **Warning**
>
> eksctl ahora instala los complementos predeterminados (vpc-cni, coredns, kube-proxy) como complementos de EKS en lugar de como complementos autogestionables. Esto significa que debe utilizar comandos en lugar de comandos para los clústeres creados con eksctl v0.184.0 y versiones posteriores. `eksctl update addon` `eksctl utils update-*`

Puede crear clústeres sin ningún complemento de red predeterminado cuando desee utilizar complementos CNI alternativos como Cilium y Calico.

Los complementos de EKS ahora admiten la recepción de permisos de IAM a través de asociaciones de identidad de EKS Pod, lo que les permite conectarse con servicios de AWS fuera del clúster

Creación de complementos

Eksctl proporciona más flexibilidad para gestionar los complementos del clúster:

En su archivo de configuración, puede especificar los complementos que desea y (si es necesario) la función o las políticas que se les asignarán:

```
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: '*'
```

Puedes especificar como máximo uno de los siguientes: `attachPolicy` `attachPolicyARNs` y`serviceAccountRoleARN`.

Si no se especifica ninguna de estas opciones, el complemento se creará con un rol al que se adjunten todas las políticas recomendadas.

> **Note**
>
> Para poder adjuntar políticas a los complementos, el clúster debe tener `OIDC` habilitadas las opciones. Si no está activado, ignoramos las políticas adjuntas.

A continuación, puedes hacer que se creen estos complementos durante el proceso de creación del clúster:

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

O cree los complementos de forma explícita después de la creación del clúster mediante el archivo de configuración o los indicadores CLI:

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

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

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

> **Tip**
>
> Usa la `--namespace-config` marca para implementar complementos en un espacio de nombres personalizado en lugar del espacio de nombres predeterminado.

Durante la creación del complemento, si ya existe una versión autogestionada del complemento en el clúster, puede elegir cómo se resolverán los posibles `configMap` conflictos configurando la opción a través del archivo de configuración, p. ej. `resolveConflicts`

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

Para la creación de complementos, el `resolveConflicts` campo admite tres valores distintos:

- `none`- EKS no cambia el valor. La creación podría fallar.
- `overwrite`- EKS sobrescribe cualquier cambio de configuración con los valores predeterminados de EKS.
- `preserve`- EKS no cambia el valor. La creación podría fallar. (Similar a`none`, pero diferente, a `preserve`la actualización de complementos).

Listado de complementos habilitados

Para ver qué complementos están habilitados en su clúster, ejecute:

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

o

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

Configurar la versión del complemento

La configuración de la versión del complemento es opcional. Si el `version` campo se deja vacío, `eksctl` se resolverá la versión predeterminada del complemento. Puede encontrar más información sobre qué versión es la predeterminada para complementos específicos en la documentación de AWS sobre EKS. Tenga en cuenta que la versión predeterminada puede no ser necesariamente la última versión disponible.

La versión del complemento se puede configurar en. `latest` Como alternativa, la versión se puede configurar con la etiqueta de compilación EKS especificada, como `v1.7.5-eksbuild.1` o`v1.7.5-eksbuild.2`. También se puede configurar en la versión de lanzamiento del complemento, por ejemplo, `v1.7.5` o`1.7.5`, y la etiqueta de `eksbuild` sufijo se detectará y configurará automáticamente.

Consulta la siguiente sección sobre cómo descubrir los complementos disponibles y sus versiones.

Descubriendo complementos

Para descubrir qué complementos están disponibles para su instalación en el clúster, ejecute:

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

Esto descubrirá la versión de kubernetes de tu clúster y la filtrará. Como alternativa, si quieres ver qué complementos están disponibles para una versión concreta de Kubernetes, puedes ejecutar:

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

También puedes descubrir complementos filtrando por sus,. `type` `owner` and/or `publisher` Por ejemplo, para ver los complementos de un propietario y tipo en particular, puedes ejecutar:

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

Los `types` `publishers` indicadores `owners` y son opcionales y se pueden especificar juntos o individualmente para filtrar los resultados.

Descubriendo el esquema de configuración de los complementos

Tras descubrir el complemento y la versión, puede ver las opciones de personalización consultando su esquema de configuración JSON.

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

Esto devuelve un esquema JSON de las distintas opciones disponibles para este complemento.

Trabajando con valores de configuración

`ConfigurationValues`se puede proporcionar en el archivo de configuración durante la creación o actualización de los complementos. Solo se admiten los formatos JSON y YAML.

Por ejemplo ,

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

```
addons:
- name: coredns
```

```
  version: latest
  configurationValues: "{\"replicaCount\":3}"
  resolveConflicts: overwrite
```

> **Note**
>
> Tenga en cuenta que cuando se modifiquen los valores de configuración de los complementos, surgirán conflictos de configuración.

```
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`.
```

Además, el comando get ahora también se recuperará `ConfigurationValues` para el complemento. p.ej.

```
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
```

Uso de un espacio de nombres personalizado

Se puede proporcionar un espacio de nombres personalizado en el archivo de configuración durante la creación de los complementos. Un espacio de nombres no se puede actualizar una vez creado un complemento.

Uso del archivo de configuración

```
addons:
  - name: aws-ebs-csi-driver
    version: latest
```

```
    namespaceConfig:
      namespace: custom-namespace
```

Uso del indicador CLI

Como alternativa, puede especificar un espacio de nombres personalizado mediante el `--namespace-config` indicador:

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

El comando get también recuperará el valor del espacio de nombres del complemento

```
- ConfigurationValues: ""
  IAMRole: ""
  Issues: null
  Name: aws-ebs-csi-driver
  NamespaceConfig:
    namespace: custom-namespace
  NewerVersion: ""
  PodIdentityAssociations: null
  Status: ACTIVE
  Version: v1.47.0-eksbuild.1
```

Actualización de complementos

Puedes actualizar tus complementos a versiones más recientes y cambiar las políticas adjuntas ejecutando lo siguiente:

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

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

> **Note**
>
> La configuración del espacio de nombres no se puede actualizar una vez creado un complemento. La `--namespace-config` marca solo está disponible durante la creación del complemento.

Al igual que en la creación de complementos, al actualizar un complemento, tienes el control total sobre los cambios de configuración que hayas aplicado anteriormente en ese complemento. `configMap` En concreto, puedes conservarlos o sobrescribirlos. Esta funcionalidad opcional está disponible en el mismo campo del archivo de configuración, `resolveConflicts` p. ej.

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

Para actualizar el complemento, el `resolveConflicts` campo acepta tres valores distintos:

- `none`- EKS no cambia el valor. Es posible que se produzca un error en la actualización.
- `overwrite`- EKS sobrescribe cualquier cambio de configuración con los valores predeterminados de EKS.
- `preserve`- EKS conserva el valor. Si elige esta opción, le recomendamos que pruebe cualquier cambio de campo y valor en un clúster que no sea de producción antes de actualizar el complemento en su clúster de producción.

Eliminar complementos

Para eliminar un complemento, ejecute lo siguiente:

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

Esto eliminará el complemento y cualquier función de IAM asociada a él.

Al eliminar el clúster, también se eliminan todas las funciones de IAM asociadas a los complementos.

Flexibilidad de creación de clústeres para los complementos de red predeterminados

Cuando se crea un clúster, EKS instala automáticamente VPC CNI, CoreDNS y kube-proxy como complementos autogestionados. Para deshabilitar este comportamiento y poder usar otros complementos de CNI, como Cilium y Calico, eksctl ahora permite crear un clúster sin ningún complemento de red predeterminado. Para crear un clúster de este tipo, configúrelo como en: `addonsConfig.disableDefaultAddons`

```
addonsConfig:
  disableDefaultAddons: true
```

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

Para crear un clúster solo con CoredNS y kube-proxy y no con el CNI de VPC, especifique los complementos de forma explícita y configúrelos, como en: `addons` `addonsConfig.disableDefaultAddons`

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

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

Como parte de este cambio, eksctl ahora instala los complementos predeterminados como complementos de EKS en lugar de como complementos autogestionados durante la creación del clúster si no se establece explícitamente en true. `addonsConfig.disableDefaultAddons` Por lo tanto, `eksctl utils update-*` los comandos ya no se pueden usar para actualizar los complementos de los clústeres creados con eksctl v0.184.0 y versiones posteriores:

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

En su lugar, debería usarse ahora. `eksctl update addon`

Para obtener más información, consulte Amazon EKS introduce la flexibilidad de creación de clústeres para complementos de redes.

Habilitación del acceso a Amazon EMR

Para permitir que EMR realice operaciones en la API de Kubernetes, es necesario conceder a su SLR los permisos RBAC requeridos. eksctl proporciona un comando que crea los recursos RBAC

necesarios para EMR y los actualiza para vincular la función con la SLR para EMR. `aws-auth` ConfigMap

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

Soporte de EKS Fargate

AWS Fargate es un motor de procesamiento administrado para Amazon ECS que puede ejecutar contenedores. En Fargate, no necesita administrar servidores ni clústeres.

Amazon EKS ahora puede lanzar pods en AWS Fargate. Esto elimina la necesidad de preocuparse por la forma en que se aprovisiona o administra la infraestructura de los pods y facilita la creación y ejecución de aplicaciones de Kubernetes de alto rendimiento y alta disponibilidad en AWS.

Creación de un clúster con el soporte de Fargate

Puede añadir un clúster compatible con Fargate con:

```
eksctl create cluster --fargate
[#]  eksctl version 0.11.0
[#]  using region ap-northeast-1
[#]  setting availability zones to [ap-northeast-1a ap-northeast-1d ap-northeast-1c]
[#]  subnets for ap-northeast-1a - public:192.168.0.0/19 private:192.168.96.0/19
[#]  subnets for ap-northeast-1d - public:192.168.32.0/19 private:192.168.128.0/19
[#]  subnets for ap-northeast-1c - public:192.168.64.0/19 private:192.168.160.0/19
[#]  nodegroup "ng-dba9d731" will use "ami-02e124a380df41614" [AmazonLinux2/1.14]
[#]  using Kubernetes version 1.14
[#]  creating EKS cluster "ridiculous-painting-1574859263" in "ap-northeast-1" region
[#]  will create 2 separate CloudFormation stacks for cluster itself and the initial
 nodegroup
[#]  if you encounter any issues, check CloudFormation console or try 'eksctl utils
 describe-stacks --region=ap-northeast-1 --cluster=ridiculous-painting-1574859263'
[#]  CloudWatch logging will not be enabled for cluster "ridiculous-
painting-1574859263" in "ap-northeast-1"
[#]  you can enable it with 'eksctl utils update-cluster-logging --enable-
types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-northeast-1 --
cluster=ridiculous-painting-1574859263'
[#]  Kubernetes API endpoint access will use default of {publicAccess=true,
 privateAccess=false} for cluster "ridiculous-painting-1574859263" in "ap-northeast-1"
[#]  2 sequential tasks: { create cluster control plane "ridiculous-
painting-1574859263", create nodegroup "ng-dba9d731" }
```

```
[#]  building cluster stack "eksctl-ridiculous-painting-1574859263-cluster"
[#]  deploying stack "eksctl-ridiculous-painting-1574859263-cluster"
[#]  building nodegroup stack "eksctl-ridiculous-painting-1574859263-nodegroup-ng-
dba9d731"
[#]  --nodes-min=2 was set automatically for nodegroup ng-dba9d731
[#]  --nodes-max=2 was set automatically for nodegroup ng-dba9d731
[#]  deploying stack "eksctl-ridiculous-painting-1574859263-nodegroup-ng-dba9d731"
[#]  all EKS cluster resources for "ridiculous-painting-1574859263" have been created
[#]  saved kubeconfig as "/Users/marc/.kube/config"
[#]  adding identity "arn:aws:iam::123456789012:role/eksctl-ridiculous-painting-157485-
NodeInstanceRole-104DXUJOFDPO5" to auth ConfigMap
[#]  nodegroup "ng-dba9d731" has 0 node(s)
[#]  waiting for at least 2 node(s) to become ready in "ng-dba9d731"
[#]  nodegroup "ng-dba9d731" has 2 node(s)
[#]  node "ip-192-168-27-156.ap-northeast-1.compute.internal" is ready
[#]  node "ip-192-168-95-177.ap-northeast-1.compute.internal" is ready
[#]  creating Fargate profile "default" on EKS cluster "ridiculous-painting-1574859263"
[#]  created Fargate profile "default" on EKS cluster "ridiculous-painting-1574859263"
[#]  kubectl command should work with "/Users/marc/.kube/config", try 'kubectl get
 nodes'
[#]  EKS cluster "ridiculous-painting-1574859263" in "ap-northeast-1" region is ready
```

Este comando habrá creado un clúster y un perfil de Fargate. Este perfil contiene cierta información que AWS necesita para crear instancias de pods en Fargate. Estos son:

- Función de ejecución del pod para definir los permisos necesarios para ejecutar el pod y la ubicación de red (subred) para ejecutar el pod. Esto permite aplicar los mismos permisos de red y seguridad a varios pods de Fargate y facilita la migración de los pods existentes en un clúster a Fargate.
- Selector para definir qué cápsulas deben ejecutarse en Fargate. Está compuesto por una `namespace` y. `labels`

Cuando no se especifica el perfil pero la compatibilidad con Fargate está habilitada con `--fargate` un perfil de Fargate predeterminado, se crea un perfil de Fargate predeterminado. Este perfil se dirige a los espacios de nombres `default` y a los `kube-system` espacios de nombres para que los pods de esos espacios de nombres se ejecuten en Fargate.

El perfil de Fargate que se creó se puede comprobar con el siguiente comando:

```
eksctl get fargateprofile --cluster ridiculous-painting-1574859263 -o yaml
- name: fp-default
```

```
  podExecutionRoleARN: arn:aws:iam::123456789012:role/eksctl-ridiculous-
painting-1574859263-ServiceRole-EIFQOH0S1GE7
  selectors:
  - namespace: default
  - namespace: kube-system
  subnets:
  - subnet-0b3a5522f3b48a742
  - subnet-0c35f1497067363f3
  - subnet-0a29aa00b25082021
```

Para obtener más información sobre los selectores, consulte Diseño de perfiles de Fargate.

Crear un clúster compatible con Fargate mediante un archivo de configuración

El siguiente archivo de configuración declara un clúster EKS con un grupo de nodos compuesto por una EC2 `m5.large` instancia y dos perfiles de Fargate. Todos los pods definidos en los `kube-system` espacios de nombres `default` y se ejecutarán en Fargate. Todos los pods del espacio de `dev` nombres que también tengan la etiqueta también se `dev=passed` ejecutarán en Fargate. Los demás pods se programarán en el nodo de. `ng-1`

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

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

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

fargateProfiles:
  - name: fp-default
    selectors:
      # All workloads in the "default" Kubernetes namespace will be
      # scheduled onto Fargate:
      - namespace: default
      # All workloads in the "kube-system" Kubernetes namespace will be
```

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

```
eksctl create cluster -f cluster-fargate.yaml
[#]  eksctl version 0.11.0
[#]  using region ap-northeast-1
[#]  setting availability zones to [ap-northeast-1c ap-northeast-1a ap-northeast-1d]
[#]  subnets for ap-northeast-1c - public:192.168.0.0/19 private:192.168.96.0/19
[#]  subnets for ap-northeast-1a - public:192.168.32.0/19 private:192.168.128.0/19
[#]  subnets for ap-northeast-1d - public:192.168.64.0/19 private:192.168.160.0/19
[#]  nodegroup "ng-1" will use "ami-02e124a380df41614" [AmazonLinux2/1.14]
[#]  using Kubernetes version 1.14
[#]  creating EKS cluster "fargate-cluster" in "ap-northeast-1" region with Fargate
 profile and un-managed nodes
[#]  1 nodegroup (ng-1) was included (based on the include/exclude rules)
[#]  will create a CloudFormation stack for cluster itself and 1 nodegroup stack(s)
[#]  will create a CloudFormation stack for cluster itself and 0 managed nodegroup
 stack(s)
[#]  if you encounter any issues, check CloudFormation console or try 'eksctl utils
 describe-stacks --region=ap-northeast-1 --cluster=fargate-cluster'
[#]  CloudWatch logging will not be enabled for cluster "fargate-cluster" in "ap-
northeast-1"
[#]  you can enable it with 'eksctl utils update-cluster-logging --enable-
types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-northeast-1 --
cluster=fargate-cluster'
[#]  Kubernetes API endpoint access will use default of {publicAccess=true,
 privateAccess=false} for cluster "fargate-cluster" in "ap-northeast-1"
[#]  2 sequential tasks: { create cluster control plane "fargate-cluster", create
 nodegroup "ng-1" }
[#]  building cluster stack "eksctl-fargate-cluster-cluster"
[#]  deploying stack "eksctl-fargate-cluster-cluster"
[#]  building nodegroup stack "eksctl-fargate-cluster-nodegroup-ng-1"
[#]  --nodes-min=1 was set automatically for nodegroup ng-1
[#]  --nodes-max=1 was set automatically for nodegroup ng-1
[#]  deploying stack "eksctl-fargate-cluster-nodegroup-ng-1"
```

```
[#]  all EKS cluster resources for "fargate-cluster" have been created
[#]  saved kubeconfig as "/home/user1/.kube/config"
[#]  adding identity "arn:aws:iam::123456789012:role/eksctl-fargate-cluster-nod-
NodeInstanceRole-42Q80B2Z147I" to auth ConfigMap
[#]  nodegroup "ng-1" has 0 node(s)
[#]  waiting for at least 1 node(s) to become ready in "ng-1"
[#]  nodegroup "ng-1" has 1 node(s)
[#]  node "ip-192-168-71-83.ap-northeast-1.compute.internal" is ready
[#]  creating Fargate profile "fp-default" on EKS cluster "fargate-cluster"
[#]  created Fargate profile "fp-default" on EKS cluster "fargate-cluster"
[#]  creating Fargate profile "fp-dev" on EKS cluster "fargate-cluster"
[#]  created Fargate profile "fp-dev" on EKS cluster "fargate-cluster"
[#]  "coredns" is now schedulable onto Fargate
[#]  "coredns" is now scheduled onto Fargate
[#]  "coredns" is now scheduled onto Fargate
[#]  "coredns" pods are now scheduled onto Fargate
[#]  kubectl command should work with "/home/user1/.kube/config", try 'kubectl get
 nodes'
[#]  EKS cluster "fargate-cluster" in "ap-northeast-1" region is ready
```

Diseño de perfiles Fargate

Cada entrada del selector tiene hasta dos componentes, un espacio de nombres y una lista de pares clave-valor. Solo se requiere el componente de espacio de nombres para crear una entrada selectora. Todas las reglas (espacios de nombres, pares de valores clave) deben aplicarse a un pod para que coincidan con una entrada del selector. Un pod solo necesita coincidir con una entrada del selector para ejecutarse en el perfil. Cualquier módulo que cumpla todas las condiciones de un campo de selección estará programado para ejecutarse en Fargate. Cualquier pod que no coincida con ninguno de los espacios de nombres de la lista blanca pero en los que el usuario haya configurado manualmente el programador: el archivo fargate-scheduler quedaría atrapado en estado pendiente, ya que no estaba autorizado a ejecutarse en Fargate.

Los perfiles deben cumplir los siguientes requisitos:

- Es obligatorio un selector por perfil
- Cada selector debe incluir un espacio de nombres; las etiquetas son opcionales

Ejemplo: programar la carga de trabajo en Fargate

Para programar pods en Fargate para el ejemplo mencionado anteriormente, se podría, por ejemplo, crear un espacio de nombres llamado `dev` e implementar la carga de trabajo allí:

```
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
```

En el resultado del último `kubectl get pods` comando, podemos ver que el `nginx` pod está desplegado en un nodo llamado. `fargate-ip-192-168-183-140.ap-northeast-1.compute.internal`

Gestión de los perfiles de Fargate

Para implementar cargas de trabajo de Kubernetes en Fargate, EKS necesita un perfil de Fargate. Al crear un clúster, como en los ejemplos anteriores, `eksctl` se encarga de ello mediante la creación de un perfil predeterminado. Dado que el clúster ya existe, también es posible crear un perfil de Fargate con el `eksctl create fargateprofile` comando:

> **Note**
>
> Esta operación solo se admite en clústeres que se ejecutan en la versión de plataforma EKS `eks.5` o superior.

> **Note**
>
> Si el existente se creó con una versión `eksctl` anterior a la 0.11.0, tendrá que ejecutarlo `eksctl upgrade cluster` antes de crear el perfil de 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"
```

También puede especificar el nombre del perfil de Fargate que se va a crear. Este nombre no debe empezar por el prefijo`eks-`.

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

Al usar este comando con los indicadores CLI, eksctl solo puede crear un único perfil de Fargate con un selector simple. Para selectores más complejos, por ejemplo con más espacios de nombres, eksctl admite el uso de un archivo de configuración:

```
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
```

Para ver los perfiles de Fargate existentes en un clúster:

```
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
```

Y para verlos en `yaml` formato:

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

O en `json` formato:

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

Los perfiles Fargate son inmutables por diseño. Para cambiar algo, cree un nuevo perfil de Fargate con los cambios deseados y elimine el anterior con el `eksctl delete fargateprofile` comando como en el siguiente ejemplo:

```
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"
}
```

Tenga en cuenta que la eliminación del perfil es un proceso que puede tardar unos minutos. Si no se especifica el `--wait` indicador, de `eksctl` forma optimista espera que se elimine el perfil y vuelve a aparecer tan pronto como se envíe la solicitud a la API de AWS. Para `eksctl` esperar a que el perfil se haya eliminado correctamente, `--wait` utilice el ejemplo anterior.

Documentación adicional

- AWS Fargate
- Amazon EKS ahora puede lanzar pods en AWS Fargate

Actualizaciones de clústeres

Un clúster gestionado por `eksctl` se puede actualizar en tres sencillos pasos:

1. actualice la versión del plano de control con `eksctl upgrade cluster`
2. actualizar grupos de nodos
3. actualice los complementos de red predeterminados (para obtener más información, consultethe section called “Actualizaciones complementarias predeterminadas”):

Revise detenidamente los recursos relacionados con la actualización del clúster:

- Actualice el clúster existente a la nueva versión de Kubernetes en la Guía del usuario de Amazon EKS
- Prácticas recomendadas para las actualizaciones de clústeres en la guía de prácticas recomendadas de EKS

> **Note**
>
> El anterior `eksctl update cluster` quedará obsoleto. En su lugar, use `eksctl upgrade cluster`.

Actualización de la versión del plano de control

Las actualizaciones de las versiones del plano de control se deben realizar para las versiones secundarias de una en una.

Para actualizar el plano de control a la siguiente versión disponible, ejecute:

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

Este comando no aplicará ningún cambio de inmediato; tendrá que volver a ejecutarlo `--approve` para aplicar los cambios.

La versión de destino para la actualización del clúster se puede especificar con el indicador CLI:

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

o con el archivo de configuración

```
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**
>
> Los únicos valores permitidos para los `metadata.version` argumentos `--version` y son la versión actual del clúster o una versión superior. No se admiten las actualizaciones de más de una versión de Kubernetes.

Actualizaciones complementarias predeterminadas

En este tema se explica cómo actualizar los complementos preinstalados predeterminados que se incluyen en los clústeres de EKS.

> **Warning**
>
> eksctl ahora instala los complementos predeterminados como complementos de EKS en lugar de como complementos autogestionables. Obtenga más información sobre sus implicaciones en la flexibilidad de creación de clústeres para los complementos de red predeterminados.
> Para actualizar los complementos, `eksctl utils update-<addon>` no se puede usar en clústeres creados con eksctl v0.184.0 y versiones posteriores. Esta guía solo es válida para los clústeres creados antes de este cambio.

Hay 3 complementos predeterminados que se incluyen en cada clúster de EKS:

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

Actualice el complemento preinstalado

En el caso de los complementos oficiales de EKS que se crean manualmente a través de la creación del clúster `eksctl create addons` o al crearlos, la forma de gestionarlos es mediante: `eksctl create/get/update/delete addon` En esos casos, consulta la documentación sobre los complementos de EKS.

El proceso de actualización de cada uno de ellos es diferente, por lo que hay 3 comandos distintos que deberá ejecutar. Se aceptan todos los comandos siguientes`--config-file`. De forma predeterminada, cada uno de estos comandos se ejecuta en modo plan; si está satisfecho con los cambios propuestos, vuelva a ejecutarlo. `--approve`

Para actualizar`kube-proxy`, ejecuta:

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

Para actualizar`aws-node`, ejecuta:

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

Para actualizar`coredns`, ejecuta:

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

Una vez que lo hayas actualizado, asegúrate de ejecutarlo `kubectl get pods -n kube-system` y comprobar si todos los módulos de complementos están listos. Deberías ver algo como esto:

```
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 for Zonal Shift en clústeres de EKS

EKS ahora es compatible con el cambio zonal y el cambio automático zonal de Amazon Application Recovery Controller (ARC), lo que mejora la resiliencia de los entornos de clústeres Multi-AZ. Con AWS Zonal Shift, los clientes pueden alejar el tráfico del clúster de una zona de disponibilidad reducida, lo que garantiza que los nuevos pods y nodos de Kubernetes se lancen solo en zonas de disponibilidad en buen estado.

Crear un clúster con el cambio zonal activado

```
# 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
```

Habilitar el cambio zonal en un clúster existente

Para habilitar o deshabilitar el cambio zonal en un clúster existente, ejecute

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

o sin un archivo de configuración:

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

Más información

- Cambio zonal de EKS

Soporte de Karpenter

`eksctl`proporciona soporte para añadir Karpenter a un clúster recién creado. Creará todos los requisitos previos necesarios descritos en la sección Primeros pasos de Karpenter, incluida la instalación del propio Karpenter mediante Helm. Actualmente, admitimos la instalación de versiones. `0.28.0+` Consulte la sección de compatibilidad de Karpenter para obtener más detalles.

La siguiente configuración de clúster describe una instalación típica de Karpenter:

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

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

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

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

La versión es la versión de Karpenter, tal y como se encuentra en su repositorio Helm. También se pueden configurar las siguientes opciones:

```
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
```

Se debe definir el OIDC para poder instalar Karpenter.

Una vez que Karpenter se haya instalado correctamente, añada NodePool(s) y NodeClass(es) para que Karpenter pueda empezar a añadir nodos al clúster.

La NodePool `nodeClassRef` sección debe coincidir con el nombre de un. `EC2NodeClass` Por ejemplo:

```
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
```

Tenga en cuenta que debe especificar uno de `role` o `instanceProfile` cuatro nodos de lanzamiento. Si opta por utilizar `instanceProfile` el nombre del perfil creado, `eksctl` siga el patrón:`eksctl-KarpenterNodeInstanceProfile-<cluster-name>`.

Etiquetado automático de grupos de seguridad

`eksctl`etiqueta automáticamente el grupo de seguridad de nodos compartidos del clúster `karpenter.sh/discovery` cuando Karpenter está activado (`karpenter.version`especificado) y la `karpenter.sh/discovery` etiqueta existe en él. `metadata.tags` Esto permite la compatibilidad con el controlador Load Balancer de AWS.

Tenga en cuenta que con karpenter 0.32.0+, los aprovisionadores han quedado en desuso y se han sustituido por. NodePool

Esquema de configuración de clústeres

> **Note**
>
> La ubicación del esquema se está migrando actualmente.

Puedes usar un archivo yaml para crear un clúster. Consulta la referencia del esquema.

Por ejemplo:

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

La referencia del esquema de este archivo está disponible en GitHub.

Para obtener más información sobre el uso del archivo, consultethe section called “Creación y administración de clústeres”.

Grupos de nodos

Este capítulo incluye información sobre cómo crear y configurar grupos de nodos con Eksctl. Los grupos de nodos son grupos de EC2 instancias conectados a un clúster de EKS.

Temas:

  - Configure las credenciales para que los nodos remotos se autentiquen en el clúster de EKS mediante AWS Systems Manager (SSM) o AWS IAM Roles Anywhere
- the section called “Config de reparación de nodos”
  - Habilitar la reparación de nodos para que los grupos de nodos gestionados por EKS supervisen y reemplacen o reinicien automáticamente los nodos de trabajo en mal estado
- the section called “Soporte ARM”
  - Cree un clúster EKS con instancias Graviton basadas en ARM para mejorar el rendimiento y la rentabilidad
- the section called “Contaminaciones”
  - Aplica restricciones a grupos de nodos específicos de un clúster de Kubernetes
  - Controle la programación y el desalojo de los pods en función de las claves, los valores y los efectos de la contaminación
- the section called “Compatibilidad con las plantillas de lanzamiento”
  - Lanzamiento de grupos de nodos gestionados mediante una plantilla de lanzamiento proporcionada EC2
  - Actualizar un grupo de nodos gestionado para usar una versión diferente de una plantilla de lanzamiento
  - Conozca las limitaciones y consideraciones a la hora de utilizar plantillas de lanzamiento AMIs y personalizadas con grupos de nodos gestionados
- the section called “Trabaja con grupos de nodos”
  - Habilite el acceso SSH a EC2 las instancias del grupo de nodos
  - Aumente o reduzca el número de nodos de un grupo de nodos
- the section called “Subredes personalizadas”
  - Amplíe una VPC existente con una nueva subred y añada un grupo de nodos a esa subred
- the section called “Arranque de nodos”
  - Comprenda el nuevo proceso de inicialización de nodos (nodeadm) introducido en 2023 AmazonLinux
  - Conozca la NodeConfig configuración predeterminada que aplica eksctl a los nodos autogestionados y gestionados por EKS
  - Personalice el proceso de arranque de nodos proporcionando un módulo personalizado overrideBootstrapCommand NodeConfig
- the section called “Grupos de nodos no administrados”

  - Cree o actualice grupos de nodos no administrados en un clúster de EKS
  - Actualice los complementos predeterminados de Kubernetes, como kube-proxy, aws-node y CoredNS
- the section called “Soporte para GPU”
  - Eksctl admite la selección de tipos de instancias de GPU para los grupos de nodos, lo que permite el uso de cargas de trabajo aceleradas por la GPU en los clústeres de EKS.
  - Eksctl instala automáticamente el complemento para dispositivos NVIDIA Kubernetes cuando se selecciona un tipo de instancia compatible con la GPU, lo que facilita el uso de los recursos de la GPU del clúster.
  - Los usuarios pueden deshabilitar la instalación automática del complemento e instalar manualmente una versión específica del complemento para dispositivos NVIDIA Kubernetes mediante los comandos proporcionados.
- the section called “Selector de instancias”
  - Genera automáticamente una lista de los tipos de EC2 instancias adecuados en función de criterios de recursos como v CPUs GPUs, memoria y arquitectura de CPU
  - Crea clústeres y grupos de nodos con los tipos de instancias que coincidan con los criterios de selección de instancias especificados
  - Realiza una prueba para inspeccionar y modificar los tipos de instancias que coinciden con el selector de instancias antes de crear un grupo de nodos
- the section called “Mapeos de volumen adicionales”
  - Configura asignaciones de volumen adicionales para un grupo de nodos administrado en un clúster de EKS
  - Personalice las propiedades de los volúmenes, como el tamaño, el tipo, el cifrado, las IOPS y el rendimiento, para los volúmenes adicionales
  - Adjunte las instantáneas de EBS existentes como volúmenes adicionales al grupo de nodos
- the section called “Nodos Worker de Windows”
  - Agregue grupos de nodos de Windows a un clúster de Linux Kubernetes existente para permitir la ejecución de cargas de trabajo de Windows
  - Programe las cargas de trabajo en el sistema operativo correspondiente (Windows o Linux) mediante selectores de nodos basados en las etiquetas y `kubernetes.io/os` `kubernetes.io/arch`
- the section called “Soporte AMI personalizado”

- Utilice la `--node-ami` marca para especificar una AMI personalizada para los grupos de nodos, consulte a AWS para obtener la AMI más reciente optimizada para EKS o utilice AWS Systems Manager Parameter Store para buscar la AMI.
- Defina el `--node-ami-family` indicador para especificar la familia de sistemas operativos para la AMI del grupo de nodos, como AmazonLinux 2, Ubuntu2204 o 2022. WindowsServer CoreContainer
- Para los grupos de nodos de Windows, especifique una AMI personalizada y proporcione un script de PowerShell arranque mediante. `overrideBootstrapCommand`

- the section called “DNS personalizada”
  - Sobrescriba la dirección IP del servidor DNS utilizada para las búsquedas de DNS internas y externas

Trabaja con grupos de nodos

Crear grupos de nodos

Puede añadir uno o más grupos de nodos además del grupo de nodos inicial creado junto con el clúster.

Para crear un grupo de nodos adicional, usa:

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

> **Note**
>
> `--version`el indicador no es compatible con los grupos de nodos gestionados. Siempre hereda la versión del plano de control.

De forma predeterminada, los nuevos grupos de nodos no administrados heredan la versión del plano de control (`--version=auto`), pero puedes especificar una versión diferente y utilizarla también `--version=latest` para forzar el uso de la versión más reciente.

Además, puedes usar el mismo archivo de configuración que se usó para: `eksctl create cluster`

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

Crear un grupo de nodos a partir de un archivo de configuración

Los grupos de nodos también se pueden crear mediante una definición de clúster o un archivo de configuración. Dado el siguiente ejemplo de archivo de configuración y un clúster existente llamado: `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
```

Los nodos se agrupan `ng-1-workers` y se `ng-2-builders` pueden crear con este comando:

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

Equilibrio de carga

Si ya te has preparado para adjuntar los grupos or/and objetivo de los balanceadores de cargas clásicos existentes a los grupos de nodos, puedes especificarlos en el archivo de configuración. Los grupos or/and objetivo de los balanceadores de cargas clásicos se asocian automáticamente al ASG al crear grupos de nodos. Esto solo es compatible con los grupos de nodos autogestionados definidos mediante el campo. `nodeGroups`

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

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

nodeGroups:
  - name: ng-1-web
    labels: { role: web }
    instanceType: m5.xlarge
    desiredCapacity: 10
    privateNetworking: true
    classicLoadBalancerNames:
      - dev-clb-1
      - dev-clb-2
    asgMetricsCollection:
      - granularity: 1Minute
        metrics:
          - GroupMinSize
          - GroupMaxSize
          - GroupDesiredCapacity
          - GroupInServiceInstances
          - GroupPendingInstances
          - GroupStandbyInstances
          - GroupTerminatingInstances
          - GroupTotalInstances
  - name: ng-2-api
    labels: { role: api }
    instanceType: m5.2xlarge
    desiredCapacity: 2
    privateNetworking: true
    targetGroupARNs:
      - arn:aws:elasticloadbalancing:eu-north-1:01234567890:targetgroup/dev-target-
group-1/abcdef0123456789
```

Selección de grupos de nodos en los archivos de configuración

Para realizar una `delete` operación `create` o solo en un subconjunto de los grupos de nodos especificados en un archivo de configuración, hay dos indicadores de CLI que aceptan una lista de globos y, `0` por ejemplo: 1

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

Con el archivo de configuración del ejemplo anterior, se pueden crear todos los grupos de nodos de trabajadores excepto el de trabajadores con el siguiente comando:

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

O se puede eliminar el grupo de nodos de los constructores con:

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

En este caso, también necesitamos proporcionar el `--approve` comando para eliminar realmente el grupo de nodos.

Incluir y excluir reglas

- si no `--include` o `--exclude` se especifica, todo está incluido
- si solo `--include` se especifica, solo se incluirán los grupos de nodos que coincidan con esos globos
- si solo `--exclude` se especifica, se incluyen todos los grupos de nodos que no coincidan con esos globos
- si se especifican ambos, `--exclude` las reglas tienen prioridad sobre ellas `--include` (es decir, se excluirán los grupos de nodos que coincidan con las reglas de ambos grupos)

Listado de grupos de nodos

Para enumerar los detalles de un grupo de nodos o de todos los grupos de nodos, usa:

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

Para enumerar uno o más grupos de nodos en formato YAML o JSON, que generan más información que la tabla de registro predeterminada, usa:

```
# YAML format
```

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

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

Inmutabilidad del grupo de nodos

Por diseño, los grupos de nodos son inmutables. Esto significa que si necesitas cambiar algo (además de escalar), como la AMI o el tipo de instancia de un grupo de nodos, tendrás que crear un nuevo grupo de nodos con los cambios deseados, mover la carga y eliminar el anterior. Consulte la sección Eliminar y drenar grupos de nodos.

Escalar grupos de nodos

El escalado de grupos de nodos es un proceso que puede tardar unos minutos. Cuando no se especifica el `--wait` indicador, espera `eksctl` con optimismo que el grupo de nodos se escale y regresa tan pronto como se envíe la solicitud de la API de AWS. Para `eksctl` esperar a que los nodos estén disponibles, añada una `--wait` marca como la que se muestra a continuación.

> **Note**
>
> Escalar un grupo de nodos down/in (es decir, reducir el número de nodos) puede provocar errores, ya que dependemos únicamente de los cambios en el ASG. Esto significa que los nodos removed/terminated no se agotan de forma explícita. Esta puede ser un área de mejora en el futuro.

El escalado de un grupo de nodos gestionado se consigue llamando directamente a la API de EKS que actualiza la configuración de un grupo de nodos gestionado.

Escalar un único grupo de nodos

Se puede escalar un grupo de nodos mediante el comando: `eksctl scale nodegroup`

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

Por ejemplo, para escalar un grupo de nodos `cluster-1` a 5 nodos, `ng-a345f4e1` ejecuta:

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

También se puede escalar un grupo de nodos mediante un archivo de configuración que se pasa a `--config-file` y especificando el nombre del grupo de nodos con el que se debe escalar. `--name` Eksctl buscará en el archivo de configuración y descubrirá ese grupo de nodos, así como sus valores de configuración de escalado.

Si el número deseado de nodos está NOT dentro del rango del número mínimo actual y máximo actual, se mostrará un error específico. Estos valores también se pueden transmitir con indicadores `--nodes-min` y `--nodes-max` respectivamente.

Escalar varios grupos de nodos

Eksctl puede descubrir y escalar todos los grupos de nodos que se encuentran en un archivo de configuración que se transfiere. `--config-file`

De forma similar a escalar un solo grupo de nodos, se aplica el mismo conjunto de validaciones a cada grupo de nodos. Por ejemplo, el número deseado de nodos debe estar dentro del rango del número mínimo y máximo de nodos.

Eliminar y drenar grupos de nodos

Para eliminar un grupo de nodos, ejecuta:

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

Las reglas de inclusión y exclusión también se pueden usar con este comando.

> **Note**
>
> Esto agotará todos los pods de ese grupo de nodos antes de que se eliminen las instancias.

Para omitir las reglas de desalojo durante el proceso de vaciado, ejecuta:

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

Todos los nodos se acordonan y todos los pods se expulsan de un grupo de nodos al eliminarlos, pero si necesitas drenar un grupo de nodos sin eliminarlo, ejecuta:

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

Para desacordonar un grupo de nodos, ejecuta:

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

Para ignorar las reglas de desalojo, como la configuración, ejecuta: PodDisruptionBudget

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

Para acelerar el proceso de drenaje, puede especificar `--parallel <value>` el número de nodos que se van a drenar en paralelo.

Otras características

También puedes habilitar el acceso SSH y ASG y otras funciones para un grupo de nodos, por ejemplo:

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

Actualiza las etiquetas

No hay comandos específicos `eksctl` para actualizar las etiquetas de un grupo de nodos, pero se puede lograr fácilmente usando`kubectl`, por ejemplo:

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

Acceso SSH

Puede habilitar el acceso SSH para los grupos de nodos configurando uno de ellos `publicKeyName` y `publicKeyPath` en su configuración de `publicKey` grupo de nodos. Como alternativa, puede

usar AWS Systems Manager (SSM) para establecer SSH en los nodos, configurando el grupo de nodos con: `enableSsm`

```
managedNodeGroups:
  - name: ng-1
    instanceType: m5.large
    desiredCapacity: 1
    ssh: # import public key from file
      publicKeyPath: ~/.ssh/id_rsa_tests.pub
  - name: ng-2
    instanceType: m5.large
    desiredCapacity: 1
    ssh: # use existing EC2 key
      publicKeyName: ec2_dev_key
  - name: ng-3
    instanceType: m5.large
    desiredCapacity: 1
    ssh: # import inline public key
      publicKey: "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDqZEdzvHnK/GVP8nLngRHu/
GDi/3PeES7+Bx6l3koXn/Oi/UmM9/jcW5XGziZ/
oe1cPJ777eZV7muEvXg5ZMQBrYxUtYCdvd8Rt6DIoSqDLsIPqbuuNlQoBHq/PU2IjpWnp/
wrJQXMk94IIrGjY8QHfCnpuMENCucVaifgAhwyeyuO5KiqUmD8E0RmcsotHKBV9X8H5eqLXd8zMQaPl
+Ub7j5PG+9KftQu0F/QhdFvpSLsHaxvBzA5nhIltjkaFcwGQnD1rpCM3+UnQE7Izoa5Yt1xoUWRwnF
+L2TKovW7+bYQ1kxsuuiX149jXTCJDVjkYCqi7HkrXYqcC1sbsror someuser@hostname"
  - name: ng-4
    instanceType: m5.large
    desiredCapacity: 1
    ssh: # enable SSH using SSM
      enableSsm: true
```

Grupos de nodos no administrados

Al configurar `--managed=false` o usar el `nodeGroups` campo`eksctl`, se crea un grupo de nodos no administrado. Tenga en cuenta que los grupos de nodos no gestionados no aparecen en la consola de EKS, que, por regla general, solo conoce los grupos de nodos gestionados por EKS.

Deberías actualizar los grupos de nodos solo después de ejecutarlos. `eksctl upgrade cluster` (Consulte Actualización de clústeres).

Si tiene un clúster simple con solo un grupo de nodos inicial (es decir, creado con`eksctl create cluster`), el proceso es muy sencillo:

1. Obtén el nombre del grupo de nodos anterior:

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

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

2. Crea un nuevo grupo de nodos:

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

3. Elimine el grupo de nodos anterior:

```
 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.
> ```

Actualización de varios grupos de nodos

Si tienes varios grupos de nodos, es tu responsabilidad hacer un seguimiento de cómo se configuró cada uno de ellos. Puedes hacerlo mediante archivos de configuración, pero si aún no los has usado, tendrás que inspeccionar tu clúster para averiguar cómo se configuró cada grupo de nodos.

En términos generales, lo que buscas es:

- revise qué grupos de nodos tiene y cuáles se pueden eliminar o deben reemplazarse para la nueva versión

- anote la configuración de cada grupo de nodos, considere usar el archivo de configuración para facilitar las actualizaciones la próxima vez

Actualización con el archivo de configuración

Si está utilizando el archivo de configuración, deberá hacer lo siguiente.

Edite el archivo de configuración para añadir nuevos grupos de nodos y eliminar los antiguos. Si solo desea actualizar los grupos de nodos y mantener la misma configuración, puede simplemente cambiar los nombres de los grupos de nodos, por ejemplo, agregarlos al nombre. `-v2`

Para crear todos los nuevos grupos de nodos definidos en el archivo de configuración, ejecuta:

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

Una vez que tengas los nuevos grupos de nodos, puedes eliminar los antiguos:

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

> **Note**
>
> La primera ejecución es en modo plan, si estás satisfecho con los cambios propuestos, vuelve a ejecutarla con ellos. `--approve`

Actualización de los complementos predeterminados

Es posible que deba actualizar los complementos de red instalados en el clúster. Para obtener más información, consulte the section called “Actualizaciones complementarias predeterminadas”.

Grupos de nodos gestionados por EKS

Los grupos de nodos gestionados por Amazon EKS son una función que automatiza el aprovisionamiento y la administración del ciclo de vida de los nodos (instanciasEC2) para los clústeres de Kubernetes de Amazon EKS. Los clientes pueden aprovisionar grupos de nodos optimizados para sus clústeres y EKS los mantendrá actualizados con las versiones más recientes de Kubernetes y del sistema operativo host.

Un grupo de nodos gestionado por EKS es un grupo de escalado automático e EC2 instancias asociadas que AWS administra para un clúster de Amazon EKS. Cada grupo de nodos utiliza la AMI Amazon Linux 2 optimizada para Amazon EKS. Amazon EKS facilita la aplicación de correcciones de errores y parches de seguridad a los nodos, así como su actualización a las versiones más recientes de Kubernetes. Cada grupo de nodos lanza un grupo de escalado automático para el clúster, que puede abarcar varias subredes y zonas de disponibilidad de VPC de AWS para lograr una alta disponibilidad.

NUEVA plantilla de lanzamiento: compatibilidad con grupos de nodos administrados

> **Note**
>
> El término «grupos de nodos no gestionados» se ha utilizado para referirse a los grupos de nodos que eksctl ha admitido desde el principio (representados mediante el campo). `nodeGroups` El `ClusterConfig` archivo sigue utilizando el `nodeGroups` campo para definir los grupos de nodos no gestionados, y los grupos de nodos gestionados se definen con el campo. `managedNodeGroups`

Creación de grupos de nodos gestionados

```
$ eksctl create nodegroup
```

Nuevos clústeres

Para crear un clúster nuevo con un grupo de nodos gestionado, ejecute

```
eksctl create cluster
```

Para crear varios grupos de nodos gestionados y tener más control sobre la configuración, se puede utilizar un archivo de configuración.

> **Note**
>
> Los grupos de nodos administrados no tienen una paridad de funciones completa con los grupos de nodos no administrados.

```
# 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
```

Puedes encontrar otro ejemplo de un archivo de configuración para crear un grupo de nodos gestionado aquí.

Es posible tener un clúster con grupos de nodos gestionados y no gestionados. Los grupos de nodos no administrados no aparecen en la consola EKS de AWS, pero `eksctl get nodegroup` muestran ambos tipos de grupos de nodos.

```
# 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
```

NEW Support para AMI personalizadas, grupos de seguridad`instancePrefix`,`instanceName`,`ebsOptimized`,`volumeType`,`volumeName`,`volumeEncrypt` y `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'
```

Si solicitas un tipo de instancia que solo está disponible en una zona (y la configuración de eksctl requiere la especificación de dos), asegúrate de añadir la zona de disponibilidad a tu solicitud de grupo de nodos:

```
# 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
```

Esto puede ocurrir con tipos de instancias como la familia Hpc6, que solo están disponibles en una zona.

Clústeres existentes

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

Consejo: si utilizas un `ClusterConfig` archivo para describir todo el clúster, describe el nuevo grupo de nodos gestionado en el `managedNodeGroups` campo y ejecuta:

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

Actualización de los grupos de nodos gestionados

Puede actualizar un grupo de nodos a la última versión de la versión de AMI optimizada para EKS para el tipo de AMI que esté utilizando en cualquier momento.

Si su grupo de nodos es de la misma versión de Kubernetes que el clúster, puede actualizar a la última versión de AMI para esa versión de Kubernetes del tipo de AMI que esté utilizando. Si su grupo de nodos es la versión de Kubernetes anterior a la versión de Kubernetes del clúster, puede actualizar el grupo de nodos a la última versión de AMI que coincida con la versión de Kubernetes del grupo de nodos, o actualizar a la última versión de AMI que coincida con la versión de Kubernetes del clúster. No puedes revertir un grupo de nodos a una versión anterior de Kubernetes.

Para actualizar un grupo de nodos gestionado a la última versión de la AMI:

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

El grupo de nodos se puede actualizar a la última versión de AMI para una versión específica de Kubernetes mediante:

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

Para actualizar a una versión de lanzamiento de AMI específica en lugar de a la última versión, pase`--release-version`:

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

> **Note**
>
> Si los nodos gestionados se implementan de forma personalizada AMIs, se debe seguir el siguiente flujo de trabajo para implementar una nueva versión de la AMI personalizada.

- El despliegue inicial del grupo de nodos debe realizarse mediante una plantilla de lanzamiento. p. ej.

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

- cree una nueva versión de la AMI personalizada (mediante la consola AWS EKS).
- cree una nueva versión de la plantilla de lanzamiento con el nuevo ID de AMI (mediante la consola AWS EKS).
- actualice los nodos a la nueva versión de la plantilla de lanzamiento, p. ej.

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

Gestión de actualizaciones paralelas para nodos

Se pueden actualizar varios nodos gestionados simultáneamente. Para configurar las actualizaciones paralelas, defina el grupo `updateConfig` de nodos al crear el grupo de nodos. Puede encontrar un ejemplo `updateConfig` aquí.

Para evitar que sus cargas de trabajo se pierdan debido a la actualización de varios nodos a la vez, puede limitar el número de nodos que pueden dejar de estar disponibles durante una actualización especificándolo en el `maxUnavailable` campo de una`updateConfig`. Como alternativa, utilice`maxUnavailablePercentage`, que define el número máximo de nodos no disponibles como un porcentaje del número total de nodos.

Tenga en cuenta que `maxUnavailable` no puede ser superior a`maxSize`. Además, `maxUnavailable` `maxUnavailablePercentage` no se puede usar simultáneamente.

Esta función solo está disponible para los nodos gestionados.

Actualización de grupos de nodos gestionados

`eksctl`permite actualizar la UpdateConfigsección de un grupo de nodos gestionado. En esta sección se definen dos campos. `MaxUnavailable`y`MaxUnavailablePercentage`. Tus grupos de nodos no se ven afectados durante la actualización, por lo que no es de esperar un tiempo de inactividad.

El comando `update nodegroup` debe usarse con un archivo de configuración que use la bandera. `--config-file` El grupo de nodos debe contener una `nodeGroup.updateConfig` sección. Puede encontrar más información aquí.

Problemas de salud de Nodegroup

Los grupos de nodos gestionados por EKS comprueban automáticamente la configuración del grupo de nodos y los nodos para detectar problemas de estado y los notifican a través de la API y la consola de EKS. Para ver los problemas de estado de un grupo de nodos:

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

Administrar etiquetas

EKS Managed Nodegroups permite adjuntar etiquetas que se aplican a los nodos de Kubernetes del grupo de nodos. Esto se especifica mediante el `labels` campo de eksctl durante la creación del clúster o grupo de nodos.

Para establecer etiquetas nuevas o actualizar las etiquetas existentes en un grupo de nodos:

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

Para desconfigurar o eliminar etiquetas de un grupo de nodos:

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

Para ver todas las etiquetas configuradas en un grupo de nodos:

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

Escalar los grupos de nodos gestionados

`eksctl scale nodegroup`también es compatible con los grupos de nodos gestionados. La sintaxis para escalar un grupo de nodos gestionado o no gestionado es la misma.

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

Más información

- Grupos de nodos gestionados por EKS

Arranque de nodos

AmazonLinux2023

AL2023 introdujo un nuevo proceso de inicialización de nodos, nodeadm, que utiliza un esquema de configuración YAML, eliminando el uso de scripts. `/etc/eks/bootstrap.sh`

> **Note**
>
> Con las versiones 1.30 y superiores de Kubernetes, Amazon Linux 2023 es el sistema operativo predeterminado.

Configuración predeterminada para AL2

Para los nodos autogestionados y los nodos gestionados por EKS de forma personalizada AMIs, `eksctl` crea un valor predeterminado, mínimo `NodeConfig` y lo inyecta automáticamente en los datos de usuario de la plantilla de lanzamiento de los grupos de nodos, es decir

```
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

--//--
```

En el caso de los nodos gestionados por EKS y basados en nodos nativos AMIs, lo predeterminado `NodeConfig` es que EKS MNG los añada de forma clandestina y los añada directamente a los datos de usuario del nodo. EC2 Por lo tanto, en este escenario, `eksctl` no es necesario incluirlo en la plantilla de lanzamiento.

Configurar el proceso de arranque

Para establecer propiedades avanzadas o simplemente anular los valores predeterminadosNodeConfig, eksctl le permite especificar un valor personalizado mediante, p. ej. 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
```

eksctl añadirá esta configuración personalizada a los datos de usuario y la fusionará con la configuración predeterminada. nodeadm Obtenga más información sobre nodeadm la capacidad de fusionar varios objetos de configuración aquí.

Soporte de plantillas de lanzamiento para grupos de nodos gestionados

eksctl admite el lanzamiento de grupos de nodos gestionados mediante una plantilla de lanzamiento proporcionada. EC2 Esto permite múltiples opciones de personalización para los grupos de nodos, incluida la provisión de grupos personalizados AMIs y de seguridad y la transmisión de datos de usuario para el arranque de los nodos.

Creación de grupos de nodos gestionados mediante una plantilla de lanzamiento proporcionada

```
# 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
```

Actualizar un grupo de nodos gestionado para utilizar una versión de plantilla de lanzamiento diferente

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

> **Note**
>
> Si una plantilla de lanzamiento usa una AMI personalizada, la nueva versión también debería usar una AMI personalizada o la operación de actualización fallará

Si una plantilla de lanzamiento no utiliza una AMI personalizada, también se puede especificar la versión de Kubernetes a la que se va a actualizar:

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

Notas sobre el soporte de plantillas de lanzamiento y AMI personalizadas

- Cuando se proporciona una plantilla de lanzamiento, no se admiten los siguientes campos: `instanceTypeami`,`ssh.allow`,`ssh.sourceSecurityGroupIds`,`securityGroups`,`instancePrefi` `overrideBootstrapCommand` y`disableIMDSv1`.
- Cuando se utiliza una AMI personalizada (`ami`), también `overrideBootstrapCommand` debe configurarse para realizar el arranque.
- `overrideBootstrapCommand`solo se puede configurar cuando se utiliza una AMI personalizada.
- Cuando se proporciona una plantilla de lanzamiento, las etiquetas especificadas en la configuración del grupo de nodos se aplican únicamente al recurso del grupo de nodos de EKS y no se propagan a las instancias. EC2

Subredes personalizadas

Es posible ampliar una VPC existente con una nueva subred y añadir un grupo de nodos a esa subred.

¿Por qué

Si el clúster se queda sin la configuración preconfigurada IPs, es posible cambiar el tamaño de la VPC existente con un CIDR nuevo para agregarle una nueva subred. Para saber cómo hacerlo, lea esta guía sobre AWS Extending VPCs.

TL; DR

Vaya a la configuración de la VPC y añada, haga clic en Acciones->Editar CIDRs y añada una nueva gama. Por ejemplo:

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

Ahora necesita agregar una nueva subred. Dependiendo de si se trata de una nueva subred privada o pública, tendrá que copiar la información de enrutamiento de una subred privada o pública, respectivamente.

Una vez creada la subred, agregue el enrutamiento y copie el ID de la puerta de enlace NAT o la puerta de enlace de Internet de otra subred de la VPC. Tenga cuidado de activar la asignación

automática de IP si se trata de una subred pública. Acciones -> Modificar la configuración de IP de asignación automática -> Habilitar la asignación automática de direcciones públicas. IPv4

No olvides copiar también las etiquetas de las subredes existentes en función de la configuración de la subred pública o privada. Esto es importante; de lo contrario, la subred no formará parte del clúster y las instancias de la subred no podrán unirse.

Cuando termines, copia el ID de la nueva subred. Repita el procedimiento tantas veces como sea necesario.

Cómo

Para crear un grupo de nodos en las subredes creadas, ejecute el siguiente comando:

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

O bien, utilice la configuración de la siguiente manera:

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

Con una configuración como esta:

```
# 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
```

Espere a que se cree el grupo de nodos y las nuevas instancias deberían tener los nuevos rangos de IP de las subredes.

Eliminar el clúster

Dado que la nueva incorporación modificó la VPC existente al agregar una dependencia fuera de la CloudFormation pila, ya no CloudFormation se puede eliminar el clúster.

Antes de eliminar el clúster, elimine manualmente todas las subredes adicionales creadas y, a continuación, llame a: `eksctl`

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

DNS personalizada

Hay dos formas de sobrescribir la dirección IP del servidor DNS utilizada para todas las búsquedas de DNS internas y externas. Es el equivalente a la `--cluster-dns` bandera de. `kubelet`

La primera, es a través del `clusterDNS` campo. Config files acepta un `string` campo llamado `clusterDNS` con la dirección IP del servidor DNS que se va a utilizar. Esto se pasará al `kubelet` que, a su vez, lo pasará a los pods a través del `/etc/resolv.conf` archivo. Para obtener más información, consulta el esquema del archivo de configuración.

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

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

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

Tenga en cuenta que esta configuración solo acepta una dirección IP. Para especificar más de una dirección, utilice el `kubeletExtraConfig`parámetro:

```
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"]
```

Contaminaciones

Para aplicar manchas a un grupo de nodos específico, usa la sección de `taints` configuración de la siguiente manera:

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

Puedes encontrar un ejemplo completo aquí.

Selector de instancias

eksctl admite la especificación de varios tipos de instancias para grupos de nodos gestionados y autogestionados, pero con más de 270 tipos de EC2 instancias, los usuarios tienen que dedicar tiempo a determinar qué tipos de instancias serían adecuados para su grupo de nodos. Es aún más difícil cuando se utilizan instancias puntuales, ya que hay que elegir un conjunto de instancias que funcione bien junto con el escalador automático de clústeres.

eksctl ahora se integra con el selector de EC2 instancias, lo que soluciona este problema al generar una lista de tipos de instancias basada en criterios de recursos: vCPUs, memoria, # de y arquitectura de GPUs CPU. Cuando se cumplen los criterios del selector de instancias, eksctl crea un grupo de nodos con los tipos de instancias configurados según los tipos de instancias que coincidan con los criterios proporcionados.

Crea grupos de clústeres y nodos

Para crear un clúster con un único grupo de nodos que utilice tipos de instancias que coincidan con los criterios de recursos del selector de instancias pasados a eksctl, ejecuta

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

Esto creará un clúster y un grupo de nodos gestionado con el `instanceTypes` campo establecido en `[c5.large, c5a.large, c5ad.large, c5d.large, t2.medium, t3.medium, t3a.medium]` (el conjunto de tipos de instancias devuelto puede cambiar).

Para los grupos de nodos no administrados, el campo se configurará de la siguiente manera`instancesDistribution.instanceTypes`:

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

Los criterios del selector de instancias también se pueden especificar en: 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
```

Las siguientes opciones de CLI del selector de instancias son compatibles con `eksctl create cluster` y`eksctl create nodegroup`:

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

Puedes encontrar un archivo de ejemplo aquí.

Funcionamiento en seco

La función de ejecución en seco te permite inspeccionar y cambiar las instancias que coinciden con el selector de instancias antes de proceder a crear un grupo de nodos.

```
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
```

Luego, lo generado se ClusterConfig puede pasar a: `eksctl create cluster`

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

El `instanceSelector` campo que representa las opciones de CLI también se agregará al ClusterConfig archivo para fines de visibilidad y documentación. Si `--dry-run` se omite, este campo se ignorará y se utilizará; de lo contrario, eksctl `instanceTypes` anulará cualquier cambio realizado en él. `instanceTypes`

Cuando se pasa un ClusterConfig archivo`--dry-run`, eksctl generará un ClusterConfig archivo que contenga el mismo conjunto de grupos de nodos tras ampliar los criterios de recursos del selector de instancias de cada grupo de nodos.

```
# 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
# ...
```

Instancias de spot

Grupos de nodos gestionados

`eksctl`admite nodos de trabajo puntual mediante grupos de nodos gestionados por EKS, una función que permite a los clientes de EKS con aplicaciones tolerantes a errores aprovisionar y gestionar fácilmente instancias EC2 puntuales para sus clústeres de EKS. EKS Managed Nodegroup configurará y lanzará un grupo de instancias puntuales con EC2 escalado automático siguiendo las prácticas recomendadas de Spot y agotando automáticamente los nodos trabajadores de Spot antes de que AWS interrumpa las instancias. El uso de esta función no conlleva ningún cargo adicional y los clientes solo pagan por el uso de los recursos de AWS, como las instancias EC2 puntuales y los volúmenes de EBS.

Para crear un clúster con un grupo de nodos gestionado mediante instancias puntuales, pase la `--spot` marca y una lista opcional de tipos de instancias:

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

Para crear un grupo de nodos administrado con instancias de spot en un clúster existente, sigue estos pasos:

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

Para crear instancias de spot mediante grupos de nodos gestionados mediante un archivo de configuración:

```
# 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**
>
> Los grupos de nodos no administrados no admiten los `instanceTypes` campos `spot` y, en cambio, el `instancesDistribution` campo se usa para configurar las instancias de spot. Consulte a continuación

Más información

- Grupos de nodos puntuales de EKS
- Tipos de capacidad de grupos de nodos gestionados por EKS

Grupos de nodos no administrados

`eksctl`tiene soporte para instancias puntuales a través de los grupos MixedInstancesPolicy de Auto Scaling.

A continuación, se muestra un ejemplo de un grupo de nodos que utiliza un 50% de instancias puntuales y un 50% de instancias bajo demanda:

```
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
```

Ten en cuenta que el `nodeGroups.X.instanceType` campo no debe configurarse al `instancesDistribution` usarlo.

En este ejemplo se utilizan instancias de GPU:

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

En este ejemplo, se utiliza la estrategia de asignación de puntos con capacidad optimizada:

```
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"
```

En este ejemplo se utiliza la estrategia de asignación capacity-optimized-prioritized puntual:

```
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"
```

Usa la estrategia de `capacity-optimized-prioritized` asignación y, a continuación, establece el orden de los tipos de instancias en la lista de anulaciones de plantillas de lanzamiento, de mayor a menor prioridad (del primero al último de la lista). Amazon EC2 Auto Scaling respeta las prioridades de tipo de instancia haciendo todo lo posible, pero optimiza primero la capacidad. Esta es una buena opción para cargas de trabajo en las que se debe minimizar la posibilidad de interrupciones, pero también es importante preferir determinados tipos de instancias. Para obtener más información, consulte las opciones de compra de ASG.

Tenga en cuenta que el `spotInstancePools` campo no debe configurarse al usarlo. `spotAllocationStrategy` Si no `spotAllocationStrategy` se especifica, EC2 se utilizará la `lowest-price` estrategia de forma predeterminada.

He aquí un ejemplo mínimo:

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

Para distinguir los nodos entre instancias puntuales o bajo demanda, puedes usar la etiqueta kubernetes`node-lifecycle`, que tendrá el valor `spot` o `on-demand` según su tipo.

Parámetros en Instances/Distribution

Consulte el esquema de configuración del clúster para obtener más información.

Soporte para GPU

Eksctl admite la selección de tipos de instancias de GPU para grupos de nodos. Simplemente proporciona un tipo de instancia compatible al comando create o a través del archivo de configuración.

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

> **Note**
>
> Ya no es necesario suscribirse a la AMI del mercado para obtener compatibilidad con GPU en EKS.

Los solucionadores de AMI (`auto`y`auto-ssm`) verán que desea utilizar un tipo de instancia de GPU y seleccionarán la AMI acelerada optimizada para EKS correcta.

Eksctl detectará que se ha seleccionado una AMI con un tipo de instancia compatible con la GPU e instalará automáticamente el complemento para dispositivos NVIDIA Kubernetes.

> **Note**
>
> Windows y Ubuntu AMIs no vienen con los controladores de GPU instalados, por lo que ejecutar cargas de trabajo aceleradas por la GPU no funcionará de forma inmediata.

Para deshabilitar la instalación automática del complemento e instalar manualmente una versión específica, utilícelo `--install-nvidia-plugin=false` con el comando create. Por ejemplo:

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

y, para las versiones 0.15.0 y superiores,

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

o, para versiones anteriores,

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

Se omitirá la instalación del complemento para dispositivos NVIDIA Kubernetes si el clúster solo incluye grupos de nodos de Bottlerocket, ya que Bottlerocket ya se encarga de la ejecución del complemento del dispositivo. Si utilizas diferentes familias de AMI en las configuraciones de tu clúster, es posible que tengas que usar restricciones y toleraciones para evitar que el complemento del dispositivo se ejecute en los nodos de Bottlerocket.

Soporte ARM

En este tema se explica cómo crear un clúster con un grupo de nodos ARM y cómo agregar un grupo de nodos ARM a un clúster existente.

EKS admite la arquitectura ARM de 64 bits con sus procesadores Graviton. Para crear un clúster, selecciona uno de los tipos de instancias basados en Graviton (`a1t4g`,`m6g`,,`m7g`,`m6gd`,`c6g`,`c7g`,`c6gd`,`r6g`,`r7g`, `r6gd` `m8gr8g`,`c8g`) y ejecuta:

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

o usa un archivo de configuración:

```
---
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 también es compatible con los grupos de nodos gestionados:

```
---
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
```

Los resolutores de AMI `auto` y `auto-ssm` deducirán la AMI correcta en función del tipo de instancia de ARM. Solo las familias AmazonLinux 2023, AmazonLinux 2 y Bottlerocket tienen EKS optimizado para ARM. AMIs

> **Note**
>
> ARM es compatible con clústeres con la versión 1.15 y superior.

Auto Scaling

Activar Auto Scaling

Puede crear un clúster (o un grupo de nodos en un clúster existente) con la función de IAM que permita el uso del escalador automático de clústeres:

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

Este indicador también establece `k8s.io/cluster-autoscaler/enabled` y `k8s.io/cluster-autoscaler/<clusterName>` etiqueta, por lo que la detección de grupos de nodos debería funcionar.

Una vez que el clúster esté en ejecución, tendrá que instalar el propio Cluster Autoscaler.

También debes añadir lo siguiente a las definiciones de grupos de nodos gestionados o no gestionados para añadir las etiquetas necesarias para que el escalador automático de clústeres escale el grupo de nodos:

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

Ampliación desde 0

Si desea poder escalar su grupo de nodos desde 0 y tiene etiquetas and/or contaminadas definidas en sus grupos de nodos, necesitará propagarlas como etiquetas en sus grupos de Auto Scaling (). ASGs

Una forma de hacerlo es configurar las etiquetas ASG en el `tags` campo de las definiciones de los grupos de nodos. Por ejemplo, dado un grupo de nodos con las siguientes etiquetas y manchas:

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

```
      key: feaster
      value: "true"
      effect: NoSchedule
```

Tendrías que añadir las siguientes etiquetas ASG:

```
nodeGroups:
  - name: ng1-public
    ...
    labels:
      my-cool-label: pizza
    taints:
      feaster: "true:NoSchedule"
    tags:
      k8s.io/cluster-autoscaler/node-template/label/my-cool-label: pizza
      k8s.io/cluster-autoscaler/node-template/taint/feaster: "true:NoSchedule"
```

Tanto para los grupos de nodos gestionados como para los no gestionados, esto se puede hacer automáticamente `propagateASGTags` configurándolo en`true`, lo que añadirá las etiquetas y las manchas como etiquetas al grupo Auto Scaling:

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

Auto Scaling con reconocimiento de zonas

Si tus cargas de trabajo son específicas de una zona, tendrás que crear grupos de nodos independientes para cada zona. Esto se debe a que se `cluster-autoscaler` supone que todos los nodos de un grupo son exactamente equivalentes. Por ejemplo, si un pod desencadena un evento de ampliación que necesita un PVC específico para una zona (p. ej., un volumen de EBS), es posible que el nuevo nodo se programe en la zona de disponibilidad incorrecta y que el pod no pueda iniciarse.

No necesitará un grupo de nodos independiente para cada zona de disponibilidad si su entorno cumple los siguientes criterios:

- No hay requisitos de almacenamiento específicos por zona.
- No se requiere PodAffinity con una topología distinta de la del host.
- No se requiere NodeAffinity en la etiqueta de zona.
- No hay ningún NodeSelector en la etiqueta de zona.

(Lea más aquí y aquí).

Si cumples todos los requisitos anteriores (y posiblemente otros), deberías tener cuidado con un solo grupo de nodos que abarque varios. AZs De lo contrario, querrás crear grupos de nodos independientes en una zona de disponibilidad única:

ANTES DE:

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

DESPUÉS DE:

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

Soporte AMI personalizado

Configuración del ID de AMI del nodo

El `--node-ami` indicador permite varios casos de uso avanzados, como el uso de una AMI personalizada o la consulta de AWS en tiempo real para determinar qué AMI usar. El indicador se puede usar tanto para imágenes que no son de GPU como para imágenes de GPU.

El indicador puede tomar el identificador de imagen de la AMI de una imagen para su uso explícito. También puede incluir las siguientes palabras clave «especiales»:

Palabra clave	Descripción
auto	Indica que la AMI que se va a utilizar para los nodos se debe encontrar consultando a AWS EC2. Esto se relaciona con la resolución automática.
SSM automático	Indica que la AMI que se va a utilizar para los nodos debe encontrarse consultando el almacén de parámetros SSM de AWS.

> **Note**
>
> Por el momento, los grupos de nodos gestionados por EKS solo admiten las siguientes familias de AMI cuando se trabaja con elementos personalizados AMIs:`AmazonLinux2023`,`AmazonLinux2`,, `BottlerocketUbuntu2004`, y `UbuntuPro2004` `Ubuntu2204` `Ubuntu2404`

Al `--node-ami` configurar una cadena de ID, `eksctl` asumirá que se ha solicitado una AMI personalizada. Para los nodos AmazonLinux 2 y Ubuntu, tanto gestionados por EKS como autogestionados, esto significa que `overrideBootstrapCommand` es obligatorio. En AmazonLinux 2023, dado que deja de utilizar el `/etc/eks/bootstrap.sh` script para el arranque de nodos, se pasa a un proceso de inicialización de nodeadm (para obtener más información, consulta la documentación de arranque de nodos), no será compatible. `overrideBootstrapCommand`

Ejemplos de banderas CLI:

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

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

Ejemplo de archivo de configuración:

```
nodeGroups:
  - name: ng1
```

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

La `--node-ami` bandera también se puede usar con`eksctl create nodegroup`.

Configuración de la familia AMI del nodo

`--node-ami-family`Puede incluir las siguientes palabras clave:

Palabra clave	Descripción
AmazonLinux2.	Indica que se debe utilizar la imagen AMI de EKS basada en Amazon Linux 2 (predeterminada).
AmazonLinux2023	Indica que se debe utilizar la imagen AMI de EKS basada en Amazon Linux 2023.
Ubuntu 2004	Indica que se debe utilizar la imagen AMI de EKS basada en Ubuntu 20.04 LTS (Focal) (compatible con EKS 1.29).
UbuntuPro2004	Indica que se debe utilizar la imagen AMI de EKS basada en Ubuntu Pro 20.04 LTS (Focal) (disponible para EKS >= 1,27 y 1,29).

Palabra clave	Descripción
Ubuntu 2204	Indica que se debe utilizar la imagen AMI de EKS basada en Ubuntu 22.04 LTS (Jammy) (disponible para EKS >= 1.29).
UbuntuPro2204	Indica que se debe utilizar la imagen AMI de EKS basada en Ubuntu Pro 22.04 LTS (Jammy) (disponible para EKS >= 1.29).
Ubuntu 2404	Indica que se debe utilizar la imagen AMI de EKS basada en Ubuntu 24.04 LTS (Noble) (disponible para EKS >= 1.31).
UbuntuPro2404	Indica que se debe utilizar la imagen AMI de EKS basada en Ubuntu Pro 24.04 LTS (Noble) (disponible para EKS >= 1.31).
Bottlerocket	Indica que se debe utilizar la imagen AMI de EKS basada en Bottlerocket.
WindowsServer2019 FullContainer	Indica que se debe utilizar la imagen AMI de EKS basada en el contenedor completo de Windows Server 2019.
WindowsServer2019 CoreContainer	Indica que se debe utilizar la imagen AMI de EKS basada en Windows Server 2019 Core Container.
WindowsServer2022 FullContainer	Indica que se debe utilizar la imagen AMI de EKS basada en el contenedor completo de Windows Server 2022.
WindowsServer2022 CoreContainer	Indica que se debe utilizar la imagen AMI de EKS basada en Windows Server 2022 Core Container.

Ejemplo de indicador CLI:

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

Ejemplo de archivo de configuración:

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

La `--node-ami-family` bandera también se puede usar con`eksctl create nodegroup`. `eksctl`requiere que la familia AMI se establezca explícitamente mediante un archivo de configuración o mediante un indicador `--node-ami-family` CLI, siempre que se trabaje con una AMI personalizada.

> **Note**
>
> Por el momento, los grupos de nodos gestionados por EKS solo admiten las siguientes familias de AMI cuando se trabaja con elementos personalizados AMIs:`AmazonLinux2023`,`AmazonLinux2`,, `BottlerocketUbuntu2004`, y `UbuntuPro2004` `Ubuntu2204` `Ubuntu2404`

Compatibilidad con AMI personalizadas de Windows

Solo los grupos de nodos de Windows autoadministrados pueden especificar una AMI personalizada. `amiFamily`debe configurarse en una familia AMI de Windows válida.

Las siguientes PowerShell variables estarán disponibles para el script de arranque:

```
$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
```

Ejemplo de archivo de configuración:

```
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
```

Soporte AMI personalizado de Bottlerocket

Para los nodos de Bottlerocket, no se admite. `overrideBootstrapCommand` En su lugar, para designar su propio contenedor de arranque, se debe usar el `bottlerocket` campo como parte del archivo de configuración. Por ejemplo:

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

Nodos Worker de Windows

A partir de la versión 1.14, Amazon EKS admite nodos de Windows que permiten ejecutar contenedores de Windows. Además de tener nodos de Windows, se requiere un nodo de Linux en el clúster para ejecutar CoredNS, ya que Microsoft aún no admite el modo de red host. Por lo tanto, un clúster EKS de Windows será una mezcla de nodos de Windows y al menos un nodo de

Linux. Los nodos Linux son fundamentales para el funcionamiento del clúster y, por lo tanto, para un clúster de nivel de producción, se recomienda tener al menos dos nodos `t2.large` Linux para alta disponibilidad.

> **Note**
>
> Ya no es necesario instalar el controlador de recursos de VPC en los nodos de trabajo de Linux para ejecutar cargas de trabajo de Windows en los clústeres de EKS creados después del 22 de octubre de 2021. Puede habilitar la administración de direcciones IP de Windows en el plano de control de EKS mediante la configuración de ConfigMap (consulte el enlace: eks/latest/userguide/windows -support.html para obtener más información). eksctl la parcheará automáticamente para permitir la ConfigMap administración de direcciones IP de Windows cuando se cree un grupo de nodos de Windows.

Crear un nuevo clúster compatible con Windows

La sintaxis del archivo de configuración permite crear un clúster completamente funcional y compatible con Windows con un solo comando:

```
# cluster.yaml
# An example of ClusterConfig containing Windows and Linux node groups to support
 Windows workloads
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

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

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

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

```
    maxSize: 3

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

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

Para crear un clúster nuevo con un grupo de nodos no administrado por Windows sin usar un archivo de configuración, ejecute los siguientes comandos:

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

Añadir compatibilidad con Windows a un clúster de Linux existente

Para permitir la ejecución de cargas de trabajo de Windows en un clúster existente con nodos Linux (familia `AmazonLinux2` AMI), debe agregar un grupo de nodos de Windows.

Se ha agregado un NUEVO soporte para grupos de nodos administrados por Windows (--managed=true u omita la marca).

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

Para garantizar que las cargas de trabajo estén programadas en el sistema operativo correcto, deben estar orientadas al sistema operativo en el que deben ejecutarse: `nodeSelector`

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

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

```
    kubernetes.io/arch: amd64
```

Si utiliza un clúster más antiguo que `1.19` el `kubernetes.io/os` y, `kubernetes.io/arch` las etiquetas deben sustituirse por `beta.kubernetes.io/os` y`beta.kubernetes.io/arch`, respectivamente.

Más información

- Soporte de EKS para Windows

Mapeos de volumen adicionales

Como opción de configuración adicional, cuando se trata de mapeos de volumen, es posible configurar mapeos adicionales cuando se crea el grupo de nodos.

Para ello, defina el campo de la siguiente manera: `additionalVolumes`

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

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

managedNodeGroups:
  - name: ng-1-workers
    labels: { role: workers }
    instanceType: m5.xlarge
    desiredCapacity: 10
    volumeSize: 80
    additionalVolumes:
      - volumeName: '/tmp/mount-1' # required
        volumeSize: 80
        volumeType: 'gp3'
        volumeEncrypted: true
        volumeKmsKeyID: 'id'
        volumeIOPS: 3000
        volumeThroughput: 125
      - volumeName: '/tmp/mount-2'  # required
        volumeSize: 80
        volumeType: 'gp2'
```

```
        snapshotID: 'snapshot-id'
```

Para obtener más información sobre cómo seleccionar los nombres de los volúmenes, consulte la documentación sobre nombres de dispositivos. Para obtener más información sobre los volúmenes de EBS, los límites de volumen de las instancias o las asignaciones de dispositivos de bloques, visite esta página.

Nodos híbridos EKS

Introducción

AWS EKS presenta los nodos híbridos, una nueva característica que le permite ejecutar aplicaciones locales y periféricas en una infraestructura administrada por el cliente con los mismos clústeres, características y herramientas de AWS EKS que utiliza en la nube de AWS. AWS EKS Hybird Nodes ofrece una experiencia de Kubernetes administrada por AWS a los entornos locales para que los clientes simplifiquen y estandaricen la forma de ejecutar las aplicaciones en entornos locales, periféricos y en la nube. Obtenga más información en EKS Hybrid Nodes.

Para facilitar la compatibilidad con esta función, eksctl presenta un nuevo campo de nivel superior llamado. `remoteNetworkConfig` Cualquier configuración relacionada con los nodos híbridos se configurará a través de este campo, como parte del archivo de configuración; no hay indicadores de CLI equivalentes. Además, en el momento del lanzamiento, cualquier configuración de red remota solo se puede configurar durante la creación del clúster y no se puede actualizar posteriormente. Esto significa que no podrás actualizar los clústeres existentes para usar nodos híbridos.

La `remoteNetworkConfig` sección del archivo de configuración le permite configurar las dos áreas principales a la hora de unir nodos remotos a sus clústeres de EKS: redes y credenciales.

Redes

Los nodos híbridos de EKS se adaptan al método que prefiera para conectar sus redes locales a una VPC de AWS. Hay varias opciones documentadas disponibles, incluidas AWS Site-to-Site VPN y AWS Direct Connect, y puede elegir el método que mejor se adapte a su caso de uso. En la mayoría de los métodos que puede elegir, la VPC se conectará a una puerta de enlace privada virtual (VGW) o a una puerta de enlace de tránsito (TGW). Si confía en eksctl para crear una VPC para usted, eksctl también configurará, dentro del alcance de su VPC, cualquier requisito previo relacionado con la red para facilitar la comunicación entre su plano de control de EKS y los nodos remotos, es decir,

- reglas SG de entrada/salida

- rutas en las tablas de rutas de las subredes privadas
- el adjunto de la puerta de enlace de VPC a la TGW o VGW dada

Ejemplo de archivo de configuración:

```
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"]
```

Si el método de conectividad que ha elegido no implica el uso de un TGW o VGW, no debe confiar en eksctl para crear la VPC por usted, sino que debe proporcionar una ya existente. Por otro lado, si está utilizando una VPC preexistente, eksctl no la modificará y es su responsabilidad garantizar que se cumplan todos los requisitos de red.

> **Note**
>
> eksctl no configura ninguna infraestructura de red fuera de su VPC de AWS (es decir, ninguna infraestructura desde VGW/TGW las redes remotas)

Credenciales

Los nodos híbridos de EKS utilizan el AWS IAM Authenticator y las credenciales de IAM temporales proporcionadas por AWS SSM o AWS IAM Roles Anywhere para autenticarse en el clúster de EKS. De forma similar a los grupos de nodos autogestionados, si no se indica lo contrario, eksctl creará para usted un rol de IAM de nodos híbridos que asumirán los nodos remotos. Además, si utiliza IAM Roles Anywhere como proveedor de credenciales, eksctl configurará un perfil y un anclaje de confianza en función de un paquete de entidades de certificación determinado (p. ej. `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
```

El ARN del rol de nodos híbridos creado por eksctl se necesitará más adelante en el proceso de unir los nodos remotos al clústernodeadm, NodeConfig para configurar y crear activaciones (si se usa SSM). Para buscarlo, usa:

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

Del mismo modo, si utiliza IAM Roles Anywhere, puede obtener el ARN del trust anchor y del perfil anywhere creados por eksctl, modificando el comando anterior sustituyéndolo por o, respectivamente. RemoteNodesRoleARN RemoteNodesTrustAnchorARN RemoteNodesAnywhereProfileARN

Si ya dispone de una configuración de IAM Roles Anywhere o utiliza SSM, puede proporcionar una función de IAM para los nodos híbridos mediante. remoteNetworkConfig.iam.roleARN Ten en cuenta que, en este caso, eksctl no te creará el perfil Trust Anchor and Anywhere, p. ej.

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

Para asignar la función a una identidad de Kubernetes y autorizar a los nodos remotos a unirse al clúster de EKS, eksctl crea una entrada de acceso con la función de IAM de nodos híbridos como ARN principal y de tipo, p. ej. 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"
        ]
    }
]
```

Compatibilidad con complementos

Interfaz de redes de contenedores (CNI): la CNI de VPC de AWS no se puede usar con nodos híbridos. Las capacidades principales de Cilium y Calico se admiten para su uso con nodos híbridos. Puede administrar el CNI con las herramientas que prefiera, como Helm. Para obtener más información, consulte Configurar una CNI para nodos híbridos.

> **Note**
>
> Si instalas la VPC CNI en tu clúster para los grupos de nodos autogestionados o gestionados por EKS, tendrás que utilizar `v1.19.0-eksbuild.1` o una versión posterior, ya que incluye una actualización del daemonset del complemento para impedir que se instale en los nodos híbridos.

Más referencias

- Nodos híbridos EKS UserDocs
- Anuncio de lanzamiento

Soporte para Node Repair Config en grupos de nodos gestionados por EKS

Los grupos de nodos gestionados por EKS ahora son compatibles con Node Repair, mediante el cual se supervisa el estado de los nodos gestionados y, en consecuencia, se sustituyen o reinician los nodos de trabajo en mal estado.

Creación de un clúster: un grupo de nodos gestionado con la reparación de nodos habilitada

Para crear un clúster con un grupo de nodos gestionado mediante la reparación de nodos, pasa la siguiente marca: `--enable-node-repair`

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

Para crear un grupo de nodos administrado mediante la reparación de nodos en un clúster existente:

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

Para crear un clúster con un grupo de nodos gestionado mediante la reparación de nodos mediante un archivo de configuración:

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

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

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

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

Más información

- EKS Managed Nodegroup Nodegroup Node Health

Red

Este capítulo incluye información sobre cómo Eksctl crea redes de Nube Privada Virtual (VPC) para clústeres de EKS.

Temas:

- the section called “IPv6 Support”
  - Especifique la versión (IPv4 o IPv6) de IP que se utilizará al crear una VPC con un clúster EKS

Configuración de la VPC

VPC dedicada para clúster

De forma predeterminada, `eksctl create cluster` creará una VPC dedicada para el clúster. Esto se hace para evitar la interferencia con los recursos existentes por diversas razones, incluida la seguridad, pero también porque es difícil detectar todos los ajustes en una VPC existente.

- El CIDR de VPC predeterminado que utiliza es. `eksctl` `192.168.0.0/16`
  - Se divide en 8 (`/19`) subredes (3 privadas, 3 públicas y 2 reservadas).
- El grupo de nodos inicial se crea en subredes públicas.
- El acceso SSH está deshabilitado a menos que se especifique. `--allow-ssh`
- De forma predeterminada, el grupo de nodos permite el tráfico entrante desde el grupo de seguridad del plano de control en los puertos 1025 a 65535.

> **Note**
>
> En `us-east-1` eksctl solo crea 2 subredes públicas y 2 privadas de forma predeterminada.

Cambiar el CIDR de VPC

Si necesitas configurar el emparejamiento con otra VPC, o simplemente necesitas un rango mayor o menor IPs de, puedes `--vpc-cidr` usar flag para cambiarlo. Consulte los documentos de AWS para obtener guías sobre cómo elegir los bloques de CIDR cuyo uso está permitido en una VPC de AWS.

Si va a crear un IPv6 clúster, puede configurarlo `VPC.IPv6Cidr` en el archivo de configuración del clúster. Esta configuración solo se encuentra en el archivo de configuración, no en un indicador de CLI.

Si tienes un bloque de direcciones IPv6 IP, también puedes traer tu propio IPv6 grupo. Consulta Cómo importar tu propio grupo de direcciones IP (BYOIP) EC2 en Amazon. A continuación, utilice el archivo `VPC.IPv6Cidr` de configuración del clúster para configurar Eksctl.

Usa una VPC existente: compartida con kops

Puedes usar la VPC de un clúster de Kubernetes existente gestionado por kops. Esta función se proporciona para facilitar el emparejamiento de clústeres de migración. and/or

Si ya ha creado un clúster con kops, por ejemplo, utilizando comandos similares a estos:

```
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
```

Puede crear un clúster de EKS en el mismo AZs mediante las mismas subredes de VPC (NOTA: AZs/subnets se requieren al menos 2):

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

Utilizar la VPC existente: otra configuración personalizada

`eksctl`proporciona cierta flexibilidad, aunque no completa, para topologías de subred y VPC personalizadas.

Puede usar una VPC existente proporcionando subredes and/or públicas privadas mediante los `--vpc-private-subnets` indicadores y. `--vpc-public-subnets` Depende de usted asegurarse de que las subredes que utiliza estén clasificadas correctamente, ya que no existe una forma sencilla de comprobar si una subred es realmente privada o pública, ya que las configuraciones varían.

Con estos indicadores, `eksctl create cluster` determinará el ID de VPC automáticamente, pero no creará tablas de enrutamiento ni otros recursos, como internet/NAT puertas de enlace. Sin embargo, creará grupos de seguridad dedicados para el grupo de nodos inicial y el plano de control.

Debe asegurarse de proporcionar al menos 2 subredes diferentes AZs y EKS comprueba esta condición. Si utilizas una VPC existente, EKS o Eksctl no aplican ni comprueban los siguientes requisitos y EKS crea el clúster. Algunas funciones básicas del clúster funcionan sin estos requisitos. (Por ejemplo, el etiquetado no es estrictamente necesario; las pruebas han demostrado que es

posible crear un clúster funcional sin establecer etiquetas en las subredes; sin embargo, no hay garantía de que esto sea siempre válido y se recomienda etiquetar).

Requisitos estándar:

- todas las subredes dadas deben estar en la misma VPC, dentro del mismo bloque de IPs
- hay un número suficiente de direcciones IP disponibles, en función de las necesidades
- número suficiente de subredes (mínimo 2), en función de las necesidades
- las subredes se etiquetan con al menos lo siguiente:
  - `kubernetes.io/cluster/<name>`etiqueta establecida en o `shared` `owned`
  - `kubernetes.io/role/internal-el`betiqueta establecida en 1 para subredes privadas
  - `kubernetes.io/role/el`betiqueta establecida en 1 para subredes públicas
- puertas de enlace and/or NAT de Internet configuradas correctamente
- las tablas de enrutamiento tienen entradas correctas y la red funciona
- NUEVO: todas las subredes públicas deben tener la propiedad `MapPublicIpOnLaunch` habilitada (es decir, `Auto-assign public IPv4 address` en la consola de AWS). Los grupos de nodos gestionados y Fargate no asignan IPv4 direcciones públicas; la propiedad debe estar configurada en la subred.

Es posible que EKS o Kubernetes impongan otros requisitos, y es responsabilidad tuya cumplir con los requisitos. up-to-date and/or recommendations, and implement those as needed/possible

La configuración predeterminada de los grupos de seguridad aplicada `eksctl` puede o no ser suficiente para compartir el acceso con los recursos de otros grupos de seguridad. Si desea modificar ingress/egress las reglas de los grupos de seguridad, puede que necesite utilizar otra herramienta para automatizar los cambios o hacerlo mediante una EC2 consola.

En caso de duda, no utilices una VPC personalizada. Si se utiliza `eksctl create cluster` sin ningún `--vpc-*` indicador, siempre se configurará el clúster con una VPC dedicada totalmente funcional.

Ejemplos

Cree un clúster mediante una VPC personalizada con 2 subredes privadas y 2 subredes públicas:

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

```
  --vpc-public-subnets=subnet-0153e560b3129a696,subnet-009fa0199ec203c37
```

o usa el siguiente archivo de configuración equivalente:

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

metadata:
  name: my-test
  region: us-west-2

vpc:
  id: "vpc-11111"
  subnets:
    private:
      us-west-2a:
          id: "subnet-0ff156e0c4a6d300c"
      us-west-2c:
          id: "subnet-0426fb4a607393184"
    public:
      us-west-2a:
          id: "subnet-0153e560b3129a696"
      us-west-2c:
          id: "subnet-009fa0199ec203c37"

nodeGroups:
  - name: ng-1
```

Cree un clúster mediante una VPC personalizada con tres subredes privadas y haga que el grupo de nodos inicial utilice esas subredes:

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

o usa el siguiente archivo de configuración equivalente:

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

metadata:
```

```
  name: my-test
  region: us-west-2

vpc:
  id: "vpc-11111"
  subnets:
    private:
      us-west-2d:
          id: "subnet-0ff156e0c4a6d300c"
      us-west-2c:
          id: "subnet-0549cdab573695c03"
      us-west-2a:
          id: "subnet-0426fb4a607393184"

nodeGroups:
  - name: ng-1
    privateNetworking: true
```

Cree un clúster mediante subredes públicas 4x de una VPC personalizada:

```
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
```

Puedes encontrar más ejemplos en la carpeta del repositorio: `examples`

- usar una VPC existente
- uso de un CIDR de VPC personalizado

Grupo de seguridad de nodo compartido personalizado

`eksctl`creará y gestionará un grupo de seguridad de nodos compartidos que permita la comunicación entre los nodos no gestionados y el plano de control del clúster y los nodos gestionados.

Si, en su lugar, desea proporcionar su propio grupo de seguridad personalizado, puede anular el `sharedNodeSecurityGroup` campo del archivo de configuración:

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

De forma predeterminada, al crear el clúster, `eksctl` agregará reglas a este grupo de seguridad para permitir la comunicación hacia y desde el grupo de seguridad del clúster predeterminado que crea EKS. Tanto el plano de control de EKS como los grupos de nodos gestionados utilizan el grupo de seguridad de clúster predeterminado.

Si desea administrar las reglas del grupo de seguridad usted mismo, puede `eksctl` impedir la creación de las reglas `manageSharedNodeSecurityGroupRules` configurándolas `false` en el archivo de configuración:

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

Gateway NAT

La puerta de enlace NAT de un clúster se puede configurar para que sea`Disable`, `Single` (predeterminada) o`HighlyAvailable`. La `HighlyAvailable` opción implementará una puerta de

enlace NAT en cada zona de disponibilidad de la región, de modo que si una AZ está inactiva, los nodos de la otra zona AZs podrán seguir comunicándose con Internet.

Se puede especificar mediante el indicador `--vpc-nat-mode` CLI o en el archivo de configuración del clúster, como se muestra en el siguiente ejemplo:

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

Consulta el ejemplo completo aquí.

> **Note**
>
> La especificación de la puerta de enlace NAT solo se admite durante la creación del clúster. No se modifica durante la actualización de un clúster.

Configuración de subred

Utilice subredes privadas para el grupo de nodos inicial

Si prefieres aislar el grupo de nodos inicial de la Internet pública, puedes usar la bandera. `--node-private-networking` Cuando se usa junto con el `--ssh-access` indicador, solo se puede acceder al puerto SSH desde el interior de la VPC.

> **Note**
>
> Si se usa el `--node-private-networking` indicador, el tráfico saliente pasará por la puerta de enlace NAT utilizando su IP elástica. Por otro lado, si los nodos están en una subred pública, el tráfico saliente no pasará por la puerta de enlace NAT y, por lo tanto, el tráfico saliente tendrá la IP de cada nodo individual.

Topología de subred personalizada

`eksctl`La versión `0.32.0` introdujo una mayor personalización de la topología de subred con la capacidad de:

- Listar varias subredes por zona de disponibilidad en la configuración de VPC
- Especifique las subredes en la configuración del grupo de nodos

En versiones anteriores, las subredes personalizadas tenían que proporcionarse por zona de disponibilidad, lo que significaba que solo podía aparecer una subred por zona de disponibilidad. A partir de `0.32.0` la identificación, las claves pueden ser arbitrarias.

```
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

Si se utiliza la AZ como clave de identificación, se puede omitir el `az` valor.

Si utiliza una cadena arbitraria como clave de identificación, como la anterior, haga lo siguiente:

- `id`debe estar configurado (`az`y es `cidr` opcional)
- o `az` debe estar configurado (`cidr`opcional)

Si un usuario especifica una subred por AZ sin especificar el CIDR ni el ID, se elegirá una subred en esa AZ de la VPC, de forma arbitraria si existen varias subredes de este tipo.

> **Note**
>
> Se debe proporcionar una especificación de subred completa, es decir, ambas `public` y `private` las configuraciones declaradas en la especificación de VPC.

Los grupos de nodos se pueden restringir a subredes con nombre mediante la configuración. Al especificar subredes en la configuración del grupo de nodos, utilice la clave de identificación tal como se indica en la especificación de la VPC, no el identificador de subred. Por ejemplo:

```
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**
>
> Solo se `availabilityZones` puede proporcionar una de ellas en la configuración del `subnets` grupo de nodos.

Al colocar grupos de nodos dentro de una subred privada, `privateNetworking` debe configurarse en el grupo de nodos: `true`

```
vpc:
  id: "vpc-11111"
  subnets:
    public:
      private-one:
          id: "subnet-0153e560b3129a696"
```

```
    ... # subnet spec continued

nodeGroups:
  - name: ng-1
    instanceType: m5.xlarge
    desiredCapacity: 2
    privateNetworking: true
    subnets:
      - private-one
```

Consulta 24-nodegroup-subnets.yaml en el repositorio eksctl para ver un ejemplo de configuración completo. GitHub

Acceso al clúster

Administración del acceso a los puntos finales del servidor API de Kubernetes

De forma predeterminada, un clúster de EKS expone el servidor de API de Kubernetes de forma pública, pero no directamente desde las subredes de la VPC (public=true, private=false). El tráfico destinado al servidor de API desde la VPC debe salir primero de las redes de VPC (pero no de la red de Amazon) y, a continuación, volver a entrar para llegar al servidor de API.

El acceso al punto final del servidor API de Kubernetes para un clúster se puede configurar para el acceso público y privado al crear el clúster mediante el archivo de configuración del clúster. Ejemplo siguiente:

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

Hay algunas advertencias adicionales a la hora de configurar el acceso al punto final de la API de Kubernetes:

1. EKS no permite clústeres sin el acceso público o privado habilitado.
2. EKS permite crear una configuración que solo permita habilitar el acceso privado, pero eksctl no lo admite durante la creación del clúster, ya que impide que eksctl pueda unir los nodos de trabajo al clúster.

3. La actualización de un clúster para que solo tenga acceso privado al punto final de la API de Kubernetes significa que los comandos de Kubernetes, de forma predeterminada (por ejemplo`kubectl`), así como, y posiblemente el comando `eksctl delete clustereksctl utils write-kubeconfig`, `eksctl utils update-kube-proxy` deben ejecutarse dentro de la VPC del clúster.
   - Esto requiere algunos cambios en varios recursos de AWS. Para obtener más información, consulte el punto final del servidor de la API de clúster.
   - Puedes proporcionar `vpc.extraCIDRs` lo que añadirá rangos de CIDR adicionales a la ControlPlaneSecurityGroup, lo que permitirá que las subredes ajenas a la VPC lleguen al punto final de la API de kubernetes. Del mismo modo, también puedes añadir rangos de CIDR`vpc.extraIPv6CIDRs`. IPv6

El siguiente es un ejemplo de cómo se puede configurar el acceso al punto final de la API de Kubernetes mediante el subcomando: `utils`

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

Para actualizar la configuración mediante un **`ClusterConfig`** archivo, usa:

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

Ten en cuenta que si no pasas una bandera, esta mantendrá el valor actual. Una vez que esté satisfecho con los cambios propuestos, añada la `approve` marca para realizar el cambio en el clúster en ejecución.

Restringir el acceso al punto final de la API pública de EKS Kubernetes

La creación predeterminada de un clúster de EKS expone públicamente el servidor de la API de Kubernetes.

Esta función solo se aplica al punto final público. Las opciones de configuración de acceso al punto final del servidor API no cambiarán y, de todas formas, tendrás la opción de deshabilitar el punto final público para que no se pueda acceder al clúster desde Internet. (Fuente: https://github.com/aws/containers-roadmap/issues/108 #issuecomment -552766489)

Para restringir el acceso al punto final de la API pública a un conjunto de al crear un clúster, defina el campo: CIDRs **`publicAccessCIDRs`**

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

Para actualizar las restricciones de un clúster existente, usa:

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

Para actualizar las restricciones mediante un **`ClusterConfig`** archivo, defina la nueva CIDRs entrada **`vpc.publicAccessCIDRs`** y ejecute:

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

> **Important**
>
> Si la configuración `publicAccessCIDRs` y la creación de grupos de nodos se `privateAccess` debe configurar en `true` o se IPs deben añadir los nodos a la lista. `publicAccessCIDRs`

Si los nodos no pueden acceder al punto final de la API del clúster debido a un acceso restringido, la creación del clúster fallará `context deadline exceeded` debido a que los nodos no pueden acceder al punto final público y no pueden unirse al clúster.

Para actualizar el acceso al punto final del servidor API y el acceso CIDRs público de un clúster con un solo comando, ejecuta:

```
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
```

Para actualizar la configuración mediante un archivo de configuración:

```
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
```

Actualización de subredes y grupos de seguridad del plano de control

Esta documentación explica cómo modificar la configuración de red del plano de control del clúster EKS tras la creación inicial. Esto incluye la actualización de las subredes y los grupos de seguridad del plano de control.

Actualización de las subredes del plano de control

Cuando se crea un clúster con eksctl, se crea un conjunto de subredes públicas y privadas que se pasan a la API de EKS. EKS crea de 2 a 4 interfaces de red elásticas entre cuentas (ENIs) en esas subredes para permitir la comunicación entre el plano de control de Kubernetes gestionado por EKS y su VPC.

Para actualizar las subredes utilizadas por el plano de control de EKS, ejecute:

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

Para actualizar la configuración mediante un archivo de configuración:

```
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
```

Sin el `--approve` indicador, eksctl solo registra los cambios propuestos. Cuando esté satisfecho con los cambios propuestos, vuelva a ejecutar el comando con la marca. `--approve`

Actualización de los grupos de seguridad del plano de control

Para gestionar el tráfico entre el plano de control y los nodos de trabajo, EKS admite la transferencia de grupos de seguridad adicionales que se aplican a las interfaces de red entre cuentas

suministradas por EKS. Para actualizar los grupos de seguridad del plano de control de EKS, ejecute:

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

Para actualizar la configuración mediante un archivo de configuración:

```
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
```

Para actualizar las subredes del plano de control y los grupos de seguridad de un clúster, ejecute:

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

Para actualizar ambos campos mediante un archivo de configuración:

```
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
```

Para ver un ejemplo completo, consulta cluster-subnets-sgs.yaml.

Sin el `--approve` indicador, eksctl solo registra los cambios propuestos. Cuando esté satisfecho con los cambios propuestos, vuelva a ejecutar el comando con la marca. `--approve`

IPv6 Support

Defina la familia IP

Cuando `eksctl` crea una vpc, puede definir la versión de IP que se utilizará. Están disponibles las siguientes opciones para su configuración:

- IPv4
- IPv6

El valor predeterminado es `IPv4`.

Para definirlo, utilice el siguiente ejemplo:

```
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**
>
> Esta configuración solo se encuentra en el archivo de configuración, no en un indicador de CLI.

Si lo usa IPv6, debe configurar los siguientes requisitos:

- El OIDC está activado
- Los complementos gestionados se definen como se muestra arriba
- la versión del clúster debe ser => 1.21
- La versión del complemento vpc-cni debe ser => 1.10.0
- Los grupos de nodos autogestionados no son compatibles con los clústeres IPv6
- los grupos de nodos gestionados no son compatibles con los clústeres que no tienen propietario IPv6
- `vpc.naty` `serviceIPv4CIDR` los campos los crea eksctl para los clústeres de ipv6 y no son opciones de configuración compatibles
- AutoAllocateIPv6 no se admite junto con IPv6
- En el caso del IPv6 clúster, la función de IAM para vpc-cni debe tener las políticas de IAM requeridas para el modo asociado IPv6

Las redes privadas también se pueden realizar con IPv6 la familia IP. Siga las instrucciones descritas en el clúster privado de EKS.

IAM

En este capítulo se incluye información sobre cómo trabajar con AWS IAM.

Temas:

- the section called “Gestione los usuarios y las funciones de IAM”
  - Administre las asignaciones de usuarios y roles de IAM para controlar el acceso a un clúster de EKS
  - Configure las asignaciones de identidad de IAM mediante el archivo de configuración del clúster o los comandos CLI
- the section called “Funciones de IAM para cuentas de servicio”
  - Gestione los permisos detallados para las aplicaciones que se ejecutan en Amazon EKS y que utilizan otros servicios de AWS
  - Cree y configure los roles de IAM y los pares de cuentas de servicio de Kubernetes mediante eksctl
  - Habilite el proveedor IAM OpenID Connect para un clúster de EKS a fin de habilitar las funciones de IAM para las cuentas de servicio
- the section called “Límite de permisos de IAM”
  - Controle el número máximo de permisos que se conceden a las entidades de IAM (usuarios o roles) estableciendo un límite de permisos
- the section called “Asociaciones de identidad de EKS Pod”
  - Configure los permisos de IAM para los complementos de EKS mediante las asociaciones de identidad de pod recomendadas
  - Permita que las aplicaciones de Kubernetes reciban los permisos de IAM necesarios para conectarse con los servicios de AWS fuera del clúster
  - Simplifique el proceso de automatización de las funciones y las cuentas de servicio de IAM en varios clústeres de EKS
- the section called “Políticas de IAM”
  - Administre las políticas de IAM para los grupos de nodos de EKS, incluida la compatibilidad con varias políticas complementarias, como el generador de imágenes, el escalador automático, el DNS externo, el administrador de certificados y más.

- Adjunta funciones de instancia personalizadas o políticas integradas a los grupos de nodos para obtener permisos adicionales.
- Adjunte políticas gestionadas por AWS específicas por ARN a los grupos de nodos para garantizar que se incluyan las políticas obligatorias, como Amazon y EKSWorker NodePolicy Amazoneks_CNI_Policy.

- the section called “Políticas de IAM mínimas”
  - Administre EC2 los recursos de AWS, incluidos los balanceadores de carga, los grupos de autoscalamiento y la supervisión CloudWatch
  - Cree y gestione CloudFormation pilas de AWS
  - Administre los clústeres de Amazon Elastic Kubernetes Service (EKS), los grupos de nodos y los recursos relacionados, como las funciones y las políticas de IAM

Políticas de IAM mínimas

Este documento describe las políticas de IAM mínimas necesarias para ejecutar los principales casos de uso de eksctl. Estas son las que se utilizan para ejecutar las pruebas de integración.

> **Note**
>
> Recuerda reemplazarlos por `<account_id>` los tuyos propios.

> **Note**
>
> AWS crea y administra una política gestionada por AWS. No puede cambiar los permisos definidos en las políticas gestionadas por AWS.

Amazon EC2 FullAccess (política gestionada por AWS)

Consulta la definición EC2 FullAccess de la política de Amazon.

AWSCloudFormationFullAccess (Política gestionada por AWS)

Ver la definición AWSCloud FormationFullAccess de la política.

EksAllAccess

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

IamLimitedAccess

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

Límite de permisos de IAM

Un límite de permisos es una función avanzada de IAM de AWS en la que se han establecido los permisos máximos que una política basada en la identidad puede conceder a una entidad de IAM; esas entidades son usuarios o roles. Cuando se establece un límite de permisos para una entidad, esa entidad solo puede realizar las acciones que estén permitidas tanto por sus políticas basadas en la identidad como por sus límites de permisos.

Puede proporcionar su límite de permisos para que todas las entidades basadas en la identidad creadas por eksctl se creen dentro de ese límite. En este ejemplo se muestra cómo se puede proporcionar un límite de permisos a las distintas entidades basadas en la identidad que crea 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**
>
> No es posible proporcionar un ARN de rol y un límite de permisos.

Configuración del límite de permisos de la VPC CNI

Tenga en cuenta que cuando cree un clúster con OIDC habilitado, eksctl creará automáticamente un `iamserviceaccount` para el VPC-CNI por motivos de seguridad. Si quieres añadirle un límite de permisos, debes especificarlo manualmente en tu archivo de configuración: `iamserviceaccount`

```
iam:
  serviceAccounts:
    - metadata:
        name: aws-node
        namespace: kube-system
      attachPolicyARNs:
      - "arn:aws:iam::<arn>:policy/AmazonEKS_CNI_Policy"
      permissionsBoundary: "arn:aws:iam::11111:policy/entity/boundary"
```

Políticas de IAM

Puede adjuntar roles de instancia a grupos de nodos. Las cargas de trabajo que se ejecuten en el nodo recibirán permisos de IAM del nodo. Para obtener más información, consulte Funciones de IAM para Amazon EC2.

En esta página se enumeran las plantillas de políticas de IAM predefinidas disponibles en eksctl. Estas plantillas simplifican el proceso de conceder a sus nodos de EKS los permisos de servicio de AWS adecuados sin tener que crear manualmente políticas de IAM personalizadas.

Políticas complementarias de IAM compatibles

Ejemplo de todas las políticas complementarias compatibles:

```
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
```

Política de Image Builder

La `imageBuilder` política permite el acceso total al ECR (Elastic Container Registry). Esto resulta útil para crear, por ejemplo, un servidor de CI que necesite enviar imágenes al ECR.

Política de EBS

La `ebs` política habilita el nuevo controlador CSI (Elastic Block Store Container Storage Interface) de EBS.

Política de gestión de certificados

La `certManager` política permite agregar registros a Route 53 para resolver el desafío del DNS01. [Puede encontrar más información aquí.]

Añadir un rol de instancia personalizado

En este ejemplo, se crea un grupo de nodos que reutiliza un rol de instancia de IAM existente de otro clúster:

```
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"
```

Adjuntar políticas en línea

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

Adjuntar políticas por 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**
>
> Si un grupo de nodos incluye el`attachPolicyARNs`, también debe incluir las políticas de nodo predeterminadas`AmazonEKSWorkerNodePolicy`, `AmazonEKS_CNI_Policy` como en este ejemplo. `AmazonEC2ContainerRegistryPullOnly`

Gestione los usuarios y las funciones de IAM

> **Note**
>
> AWS sugiere migrar the section called “Asociaciones de identidad de EKS Pod” desde. `aws-auth` ConfigMap

Los clústeres de EKS utilizan usuarios y roles de IAM para controlar el acceso al clúster. Las reglas se implementan en un mapa de configuración

Edición ConfigMap con un comando CLI

llamado`aws-auth`. `eksctl`proporciona comandos para leer y editar este mapa de configuración.

Obtenga todos los mapeos de identidad:

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

Obtenga todos los mapeos de identidad que coincidan con un hilo:

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

Cree un mapeo de identidad:

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

Elimine un mapeo de identidad:

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

> **Note**
>
> El comando anterior elimina una sola FIFO de mapeo, a menos que `--all` se indique, en cuyo caso elimina todas las coincidencias. Avisará si se encuentran más mapeos que coincidan con este rol.

Crea un mapeo de cuentas:

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

Eliminar un mapeo de cuentas:

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

Edita ConfigMap usando un ClusterConfig archivo

Los mapeos de identidad también se pueden especificar en: 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
```

Funciones de IAM para cuentas de servicio

> **Tip**
>
> `eksctl`admite la configuración de permisos detallados para ejecutar aplicaciones de EKS mediante las asociaciones de identidad de EKS Pod

Amazon EKS admite aquí [Roles for Service Accounts (IRSA)], que permite a los operadores de clústeres asignar las funciones de IAM de AWS a las cuentas de servicio de Kubernetes.

Esto proporciona una administración de permisos detallada para las aplicaciones que se ejecutan en EKS y utilizan otros servicios de AWS. Pueden ser aplicaciones que usen S3, cualquier otro servicio de datos (RDS, MQ, STS, DynamoDB) o componentes de Kubernetes, como el controlador AWS Load Balancer o ExternalDNS.

Puede crear fácilmente pares de roles y cuentas de servicio de IAM con. `eksctl`

> **Note**
>
> Si ha utilizado roles de instancia y está pensando en usar IRSA en su lugar, no debe mezclar ambos.

Funcionamiento

Funciona a través del proveedor de IAM OpenID Connect (OIDC) que EKS expone, y las funciones de IAM deben construirse con referencia al proveedor de OIDC de IAM (específico de un clúster de EKS determinado) y una referencia a la cuenta de servicio de Kubernetes a la que estará vinculado. Una vez que se crea un rol de IAM, la cuenta de servicio debe incluir el ARN de ese rol como anotación `eks.amazonaws.com/role-arn` (). De forma predeterminada, la cuenta de servicio se crea o actualiza para incluir la anotación del rol, que se puede deshabilitar con la marca. `--role-only`

Dentro de EKS, hay un controlador de admisión que inyecta las credenciales de sesión de AWS en los pods de las funciones, respectivamente, en función de la anotación de la cuenta de servicio utilizada por el pod. Las credenciales se expondrán en función de las variables `AWS_ROLE_ARN` de `AWS_WEB_IDENTITY_TOKEN_FILE` entorno. Si se utiliza una versión reciente del SDK de AWS (consulte aquí los detalles de la versión exacta), la aplicación utilizará estas credenciales.

`eksctl`El nombre del recurso es iamserviceaccount, que representa un par de rol de IAM y cuenta de servicio.

Uso desde CLI

> **Note**
>
> Las funciones de IAM para las cuentas de servicio requieren la versión 1.13 o superior de Kubernetes.

El proveedor OIDC de IAM no está activado de forma predeterminada. Puede utilizar el siguiente comando para activarlo o utilizar el archivo de configuración (véase más abajo):

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

Una vez que tenga el proveedor OIDC de IAM asociado al clúster, para crear un rol de IAM vinculado a una cuenta de servicio, ejecute:

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

> **Note**
>
> Puede especificar `--attach-policy-arn` varias veces el uso de más de una política.

Más específicamente, puede crear una cuenta de servicio con acceso de solo lectura a S3 ejecutando:

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

De forma predeterminada, se creará en el espacio de `default` nombres, pero puede especificar cualquier otro espacio de nombres, por ejemplo:

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

> **Note**
>
> Si el espacio de nombres aún no existe, se creará.

Si ya tienes una cuenta de servicio creada en el clúster (sin un rol de IAM), tendrás que usar flag. `--override-existing-serviceaccounts`

También se puede aplicar un etiquetado personalizado al rol de IAM especificando: `--tags`

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

CloudFormation generará un nombre de rol que incluirá una cadena aleatoria. Si prefiere un nombre de rol predeterminado, puede especificar`--role-name`:

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

Cuando la cuenta de servicio la cree y administre otra herramienta, como Helm, úsela `--role-only` para evitar conflictos. La otra herramienta es entonces responsable de mantener la anotación ARN

del rol. Tenga en `--override-existing-serviceaccounts` cuenta que esto no afecta a las cuentas `--role-only` de servicio`roleOnly`/, ya que el rol siempre se creará.

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

Si ya tiene un rol que desea usar con una cuenta de servicio, puede proporcionar la `--attach-role-arn` marca en lugar de proporcionar las políticas. Para garantizar que el rol solo lo pueda asumir la cuenta de servicio especificada, debe establecer aquí [un documento de política de relaciones].

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

Para actualizar las funciones de una cuenta de servicio, puede ejecutar los permisos`eksctl update iamserviceaccount`.

> **Note**
>
> `eksctl delete iamserviceaccount`elimina Kubernetes `ServiceAccounts` aunque no los haya creado. `eksctl`

Uso con archivos de configuración

Para administrar `iamserviceaccounts` el uso del archivo de configuración, querrá configurar `iam.withOIDC: true` y enumerar la cuenta en la que desee`iam.serviceAccount`.

Todos los comandos son compatibles*`--config-file`*, por lo que puedes administrar las cuentas de iamservicede la misma manera que los grupos de nodos. El `eksctl create iamserviceaccount` comando admite `--include` y `--exclude` marca (consulte esta sección para obtener más información sobre su funcionamiento). Además, el `eksctl delete iamserviceaccount` comando también es compatible`--only-missing`, por lo que puede realizar eliminaciones de la misma manera que con los grupos de nodos.

> **Note**
>
> Las cuentas de servicio de IAM se clasifican dentro de un espacio de nombres, es decir, pueden existir dos cuentas de servicio con el mismo nombre en espacios de nombres

diferentes. Por lo tanto, para definir de forma exclusiva una cuenta de servicio como parte de los `--exclude` indicadores`--include`, tendrá que pasar la cadena del nombre en ese formato. `namespace/name` Por ejemplo:

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

La opción de habilitarla `wellKnownPolicies` se incluye para usar IRSA en casos de uso conocidos`cert-manager`, como las listas de políticas `cluster-autoscaler` y como forma abreviada de usarlas.

Las políticas conocidas compatibles y otras propiedades de `serviceAccounts` se documentan en el esquema de configuración.

Se usa el siguiente ejemplo de configuración con`eksctl create cluster`:

```
# An example of ClusterConfig with IAMServiceAccounts:
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: cluster-13
  region: us-west-2

iam:
  withOIDC: true
  serviceAccounts:
  - metadata:
      name: s3-reader
      # if no namespace is set, "default" will be used;
      # the namespace will be created if it doesn't exist already
      namespace: backend-apps
      labels: {aws-usage: "application"}
    attachPolicyARNs:
    - "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    tags:
      Owner: "John Doe"
      Team: "Some Team"
  - metadata:
      name: cache-access
      namespace: backend-apps
```

```
      labels: {aws-usage: "application"}
    attachPolicyARNs:
    - "arn:aws:iam::aws:policy/AmazonDynamoDBReadOnlyAccess"
    - "arn:aws:iam::aws:policy/AmazonElastiCacheFullAccess"
  - metadata:
      name: cluster-autoscaler
      namespace: kube-system
      labels: {aws-usage: "cluster-ops"}
    wellKnownPolicies:
      autoScaler: true
    roleName: eksctl-cluster-autoscaler-role
    roleOnly: true
  - metadata:
      name: some-app
      namespace: default
    attachRoleARN: arn:aws:iam::123:role/already-created-role-for-app
nodeGroups:
  - name: "ng-1"
    tags:
      # EC2 tags required for cluster-autoscaler auto-discovery
      k8s.io/cluster-autoscaler/enabled: "true"
      k8s.io/cluster-autoscaler/cluster-13: "owned"
    desiredCapacity: 1
```

Si creas un clúster sin estos campos configurados, puedes usar los siguientes comandos para habilitar todo lo que necesites:

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

Más información

- Presentamos funciones de IAM detalladas para las cuentas de servicio
- Guía del usuario de EKS: Funciones de IAM para cuentas de servicio
- Asignación de usuarios y roles de IAM a roles RBAC de Kubernetes

Asociaciones de identidad de EKS Pod

AWS EKS ha introducido un nuevo mecanismo mejorado denominado Pod Identity Association para que los administradores de clústeres puedan configurar las aplicaciones de Kubernetes para recibir

los permisos de IAM necesarios para conectarse con los servicios de AWS fuera del clúster. Pod Identity Association aprovecha el IRSA, pero lo hace configurable directamente a través de la API de EKS, lo que elimina por completo la necesidad de utilizar la API de IAM.

Como resultado, las funciones de IAM ya no necesitan hacer referencia a un proveedor de OIDC y, por lo tanto, ya no estarán vinculadas a un único clúster. Esto significa que las funciones de IAM ahora se pueden utilizar en varios clústeres de EKS sin necesidad de actualizar la política de confianza de los roles cada vez que se crea un clúster nuevo. Esto, a su vez, elimina la necesidad de duplicar las funciones y simplifica por completo el proceso de automatización del IRSA.

Requisitos previos

Entre bastidores, la implementación de las asociaciones de identidad de los módulos consiste en ejecutar un agente como un demonio en los nodos de trabajo. Para ejecutar el agente necesario en el clúster, EKS proporciona un nuevo complemento llamado EKS Pod Identity Agent. Por lo tanto, la creación de asociaciones de identidad de pods (en general y con`eksctl`) requiere que el `eks-pod-identity-agent` complemento esté preinstalado en el clúster. Este complemento se puede crear de la misma manera que cualquier otro complemento compatible. `eksctl`

```
eksctl create addon --cluster my-cluster --name eks-pod-identity-agent
```

Además, si utiliza un rol de IAM preexistente al crear una asociación de identidad de pod, debe configurar el rol para que confíe en el nuevo director de servicio de EKS (). `pods.eks.amazonaws.com` A continuación se muestra un ejemplo de política de confianza de IAM:

```
# Error: No files found with UUID: 44d1085a-03ca-431a-9774-b786a9774200
```

Si, por el contrario, no proporciona el ARN de un rol existente al comando create, `eksctl` creará uno entre bastidores y configurará la política de confianza anterior.

Creación de asociaciones de identidad de pods

Para manipular las asociaciones de identidad de los pods, `eksctl` ha añadido un nuevo campo en`iam.podIdentityAssociations`, p. ej.

```
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
```

Para ver un ejemplo completo, consulta pod-identity-associations.yaml.

> **Note**
>
> Además de `permissionPolicy` que se utiliza como documento de política en línea, todos los demás campos tienen una contraparte de marca CLI.

La creación de asociaciones de identidad de pods se puede lograr de las siguientes maneras. Durante la creación del clúster, especificando las asociaciones de identidad de los pods deseadas como parte del archivo de configuración y ejecutándolas:

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

Tras la creación del clúster, mediante un archivo de configuración, p. ej.

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

O utilizando indicadores CLI, p. ej.

```
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**
>
> Solo se puede asociar un único rol de IAM a una cuenta de servicio a la vez. Por lo tanto, si se intenta crear una segunda asociación de identidad de pod para la misma cuenta de servicio, se producirá un error.

Buscando asociaciones de identidad de pods

Para recuperar todas las asociaciones de identidad de los pods de un clúster determinado, ejecuta uno de los siguientes comandos:

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

OR

```
eksctl get podidentityassociation --cluster my-cluster
```

Además, para recuperar solo las asociaciones de identidad de los pods dentro de un espacio de nombres determinado, usa la `--namespace` marca, p. ej.

```
eksctl get podidentityassociation --cluster my-cluster --namespace default
```

Por último, para recuperar una sola asociación, correspondiente a una determinada cuenta de servicio de K8s, incluye también el comando anterior`--service-account-name`, es decir

```
eksctl get podidentityassociation --cluster my-cluster --namespace default --service-
account-name s3-reader
```

Actualización de las asociaciones de identidad de los pods

Para actualizar la función de IAM de una o más asociaciones de identidad de pods, pasa la nueva `roleARN(s)` al archivo de configuración, por ejemplo

```
iam:
```

```
  podIdentityAssociations:
    - namespace: default
      serviceAccountName: s3-reader
      roleARN: new-role-arn-1
    - namespace: dev
      serviceAccountName: app-cache-access
      roleARN: new-role-arn-2
```

y ejecuta:

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

O (para actualizar una sola asociación) pase la nueva `--role-arn` mediante indicadores CLI:

```
eksctl update podidentityassociation --cluster my-cluster --namespace default --
service-account-name s3-reader --role-arn new-role-arn
```

Eliminar las asociaciones de identidad de los pods

Para eliminar una o más asociaciones de identidad de un pod, pasa la `namespace(s)` dirección `serviceAccountName(s)` al archivo de configuración, p. ej.

```
iam:
  podIdentityAssociations:
    - namespace: default
      serviceAccountName: s3-reader
    - namespace: dev
      serviceAccountName: app-cache-access
```

y ejecuta:

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

O (para eliminar una sola asociación) pase los indicadores `--namespace` y `--service-account-name` mediante CLI:

```
eksctl delete podidentityassociation --cluster my-cluster --namespace default --
service-account-name s3-reader
```

Los complementos de EKS son compatibles con las asociaciones de identidad de los pods

Los complementos de EKS también permiten recibir permisos de IAM a través de las asociaciones de identidad de los pods de EKS. El archivo de configuración muestra tres campos que permiten configurarlos:`addon.podIdentityAssociations`, `yaddonsConfig.autoApplyPodIdentityAssociations`. `addon.useDefaultPodIdentityAssociations` Puede configurar de forma explícita las asociaciones de identidad del pod que deseeaddon`.podIdentityAssociations`, o bien hacer que `eksctl` se resuelva (y aplique) automáticamente la configuración de identidad del pod recomendada, utilizando una de las dos `addonsConfig.autoApplyPodIdentityAssociations` opciones. `addon.useDefaultPodIdentityAssociations`

> **Note**
>
> No todos los complementos de EKS admitirán las asociaciones de identidad de los pods en el momento del lanzamiento. En este caso, los permisos de IAM necesarios se seguirán proporcionando mediante la configuración de IRSA.

Crear complementos con permisos de IAM

Al crear un complemento que requiera permisos de IAM, `eksctl` comprobará primero si las asociaciones de identidad de los pods o los ajustes de IRSA se han configurado de forma explícita como parte del archivo de configuración y, de ser así, utilizará uno de ellos para configurar los permisos del complemento, p. ej.

```
addons:
- name: vpc-cni
  podIdentityAssociations:
  - serviceAccountName: aws-node
    permissionPolicyARNs: ["arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"]
```

y ejecuta

```
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**
>
> No se permite configurar las identidades de los módulos y el IRSA al mismo tiempo y se producirá un error de validación.

En el caso de los complementos de EKS que admiten identidades de pods, `eksctl` ofrece la opción de configurar automáticamente los permisos de IAM recomendados al crear el complemento. Esto se puede lograr simplemente configurándolo `addonsConfig.autoApplyPodIdentityAssociations: true` en el archivo de configuración, p. ej.

```
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
```

y ejecuta

```
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
```

De manera equivalente, lo mismo se puede hacer mediante indicadores CLI, p. ej.

```
eksctl create addon --cluster my-cluster --name vpc-cni --auto-apply-pod-identity-
associations
```

Para migrar un complemento existente y utilizar la identidad del pod con las políticas de IAM recomendadas, utilice

```
addons:
- name: vpc-cni
  useDefaultPodIdentityAssociations: true
```

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

Actualización de complementos con permisos de IAM

Al actualizar un complemento, especificarlo `addon.PodIdentityAssociations` representará la única fuente de información sobre el estado que tendrá el complemento una vez finalizada la operación de actualización. Entre bastidores, se realizan diferentes tipos de operaciones para alcanzar el estado deseado, es decir,

- cree las identidades de los pods que estén presentes en el archivo de configuración, pero que no estén en el clúster
- elimine las identidades de pod existentes que se eliminaron del archivo de configuración, junto con cualquier recurso de IAM asociado
- actualice las identidades de los pods existentes que también están presentes en el archivo de configuración y para las que se ha modificado el conjunto de permisos de IAM

> **Note**
>
> La API de complementos de EKS gestiona directamente el ciclo de vida de las asociaciones de identidad de los pods propiedad de EKS Add-ons.

No puede usar `eksctl update podidentityassociation` (para actualizar los permisos de IAM) ni `eksctl delete podidentityassociations` (para eliminar la asociación) para las asociaciones utilizadas con un complemento de Amazon EKS. En su lugar, `eksctl update addon` o se `eksctl delete addon` utilizará.

Veamos un ejemplo de lo anterior, empezando por analizar la configuración inicial de identidad del pod para el complemento:

```
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"
    }
]
```

Ahora usa la siguiente configuración:

```
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
```

y ejecuta

```
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
```

ahora compruebe que la configuración de identidad del pod se actualizó correctamente

```
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"
    }
]
```

Para eliminar todas las asociaciones de identidad de un pod de un complemento, `addon.PodIdentityAssociations` debe configurarse explícitamente en`[]`, p. ej.

```
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: []
```

y ejecuta

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

Eliminar complementos con permisos de IAM

Al eliminar un complemento, también se eliminarán todas las identidades de los pods asociadas al complemento. Al eliminar el clúster, se obtendrá el mismo efecto para todos los complementos. También se eliminarán todas las funciones de IAM creadas por `eksctl` las identidades de los pods.

Migración de las cuentas y complementos de iamservice existentes a las asociaciones de identidad de los pods

Hay un comando `eksctl` utils para migrar las funciones de IAM existentes para las cuentas de servicio a las asociaciones de identidad de los pods, es decir

```
eksctl utils migrate-to-pod-identity --cluster my-cluster --approve
```

Entre bastidores, el comando aplicará los siguientes pasos:

- instale el `eks-pod-identity-agent` complemento si aún no está activo en el clúster

- identifique todas las funciones de IAM asociadas a iamserviceaccounts
- identifique todas las funciones de IAM asociadas a los complementos de EKS que admiten asociaciones de identidad de pods
- actualice la política de confianza de IAM de todas las funciones identificadas y añada una entidad de confianza adicional que dirija al nuevo director del servicio de EKS (y, si lo desea, elimine la relación de confianza existente entre los proveedores de OIDC)
- cree asociaciones de identidad de pod para las funciones filtradas asociadas a iamserviceaccounts
- actualice los complementos de EKS con las identidades de los pods (la API de EKS creará las identidades de los pods entre bastidores)

Al ejecutar el comando sin la `--approve` marca, solo se generará un plan compuesto por un conjunto de tareas que reflejen los pasos anteriores, p. ej.

```
[#]  (plan) would migrate 2 iamserviceaccount(s) and 2 addon(s) to pod identity
 association(s) by executing the following tasks
[#]  (plan)

3 sequential tasks: { install eks-pod-identity-agent addon,
    ## tasks for migrating the addons
    2 parallel sub-tasks: {
        2 sequential sub-tasks: {
            update trust policy for owned role "eksctl-my-cluster--Role1-DDuMLoeZ8weD",
            migrate addon aws-ebs-csi-driver to pod identity,
        },
        2 sequential sub-tasks: {
            update trust policy for owned role "eksctl-my-cluster--Role1-xYiPFOVp1aeI",
            migrate addon vpc-cni to pod identity,
        },
    },
    ## tasks for migrating the iamserviceaccounts
    2 parallel sub-tasks: {
        2 sequential sub-tasks: {
            update trust policy for owned role "eksctl-my-cluster--Role1-QLXqHcq9O1AR",
            create pod identity association for service account "default/sa1",
        },
        2 sequential sub-tasks: {
            update trust policy for unowned role "Unowned-Role1",
            create pod identity association for service account "default/sa2",
        },
    }
```

```
}
[#]  all tasks were skipped
[!]  no changes were applied, run again with '--approve' to apply the changes
```

La relación de confianza existente entre los proveedores de OIDC siempre se elimina de las funciones de IAM asociadas a los complementos de EKS. Además, para eliminar la relación de confianza existente entre los proveedores de OIDC de las funciones de IAM asociadas a iamserviceaccounts, ejecute el comando con un indicador, p. ej. `--remove-oidc-provider-trust-relationship`

```
eksctl utils migrate-to-pod-identity --cluster my-cluster --approve --remove-oidc-
provider-trust-relationship
```

Soporte para Cross Account Pod Identity

eksctl admite el acceso multicuenta de EKS Pod Identity. Esta función permite que los pods que se ejecutan en su clúster de EKS accedan a los recursos de AWS en una cuenta de AWS diferente.

Uso

Para crear una asociación de identidad de pod con acceso entre cuentas, primero configure las funciones y políticas de IAM que permitan el acceso desde una cuenta de AWS de origen (con el clúster) a una cuenta de AWS de destino (con los recursos a los que puede acceder el clúster). Para ver un ejemplo de esto, consulte «Amazon EKS Pod Identity agiliza el acceso entre cuentas».

Una vez que se haya configurado un rol de IAM en cada cuenta, utilice eksctl para crear las asociaciones de identidad de los pods:

```
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
```

Más referencias

Compatibilidad con los complementos oficiales de AWS Userdocs for EKS para las identidades de los pods

Publicación oficial del blog de AWS sobre las asociaciones de identidad de los pods

Documentos de usuario oficiales de AWS para asociaciones de identidad de Pod

Opciones de implementación

Este capítulo trata sobre el uso de eksctl para gestionar los clústeres de EKS desplegados en entornos alternativos.

Para obtener la información más precisa sobre las opciones de implementación de EKS, consulte Implementación de clústeres de Amazon EKS en entornos locales y en la nube en la Guía del usuario de EKS.

Temas:

- the section called “EKS en cualquier lugar”
  - Utilice eksctl con los clústeres de Amazon EKS Anywhere.
  - Amazon EKS Anywhere es un software de administración de contenedores creado por AWS que facilita la ejecución y la administración de Kubernetes en las instalaciones y en la periferia.
- the section called “Soporte para AWS Outposts”
  - Use eksctl con clústeres de EKS en AWS Outposts.
  - AWS Outposts es una familia de soluciones totalmente gestionadas que ofrecen la infraestructura y los servicios de AWS prácticamente a cualquier ubicación local o perimetral para ofrecer una experiencia híbrida realmente coherente.
  - La compatibilidad con AWS Outposts en eksctl le permite crear clústeres locales con todo el clúster de Kubernetes, incluidos el plano de control de EKS y los nodos de trabajo, ejecutándose localmente en AWS Outposts.
- the section called “Nodos híbridos EKS”
  - Ejecute aplicaciones locales y periféricas en una infraestructura administrada por el cliente con los mismos clústeres, funciones y herramientas de AWS EKS que utiliza en la nube de AWS.

EKS en cualquier lugar

`eksctl`proporciona acceso a la función de AWS llamada `EKS Anywhere` con el subcomando`eksctl anywhere`. Esto requiere que el `eksctl-anywhere` binario esté presente. `PATH` Siga las instrucciones que se detallan aquí Instale eksctl-anywhere para instalarlo.

Una vez hecho esto, ejecute los comandos en cualquier lugar ejecutando:

```
eksctl anywhere version
v0.5.0
```

Para obtener más información sobre EKS Anywhere, visite el sitio web de EKS Anywhere.

Soporte para AWS Outposts

> **Warning**
>
> Los grupos de nodos gestionados por EKS no son compatibles con Outposts.

Ampliación de los clústeres existentes a AWS Outposts

Puede extender un clúster de EKS existente que se ejecute en una región de AWS a AWS Outposts configurando nuevos grupos de `nodeGroup.outpostARN` nodos para crear grupos de nodos en Outposts, como en:

```
# 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
```

En esta configuración, el plano de control de EKS se ejecuta en una región de AWS, mientras que los grupos de nodos con `outpostARN` set se ejecutan en el Outpost especificado. Cuando

se crea un grupo de nodos en Outposts por primera vez, eksctl amplía la VPC creando subredes en el Outpost especificado. Estas subredes se utilizan para crear grupos de nodos establecidos. `outpostARN`

Los clientes con una VPC preexistente deben crear las subredes en Outposts y `nodeGroup.subnets` pasarlas, como en:

```
# 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"
```

Creación de un clúster local en AWS Outposts

> **Note**
>
> Los clústeres locales solo admiten bastidores de Outpost.

> **Note**
>
> Solo Amazon Linux 2 es compatible con los grupos de nodos cuando el plano de control está en Outposts. Solo se admiten los tipos de volumen gp2 de EBS para los grupos de nodos de Outposts.

La compatibilidad con AWS Outposts en eksctl le permite crear clústeres locales con todo el clúster de Kubernetes, incluidos el plano de control de EKS y los nodos de trabajo, ejecutándose localmente en AWS Outposts. Los clientes pueden crear un clúster local con el plano de control de EKS y los nodos de trabajo ejecutándose localmente en AWS Outposts, o pueden extender un clúster de EKS existente que se ejecute en una región de AWS a AWS Outposts creando nodos de trabajo en Outposts.

Para crear el plano de control y los grupos de nodos de EKS en AWS Outposts, configúrelo en el ARN de Outpost, como `outpost.controlPlaneOutpostARN` se muestra en:

```
# 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
```

Esto indica a eksctl que cree el plano de control y las subredes de EKS en el Outpost especificado. Dado que existe un rack de Outposts en una única zona de disponibilidad, eksctl crea solo una subred pública y privada. eksctl no asocia la VPC creada a una puerta de enlace local y, por lo tanto, eksctl carecerá de conectividad con el servidor API y no podrá crear grupos de nodos. Por lo tanto,

si `ClusterConfig` contiene algún grupo de nodos durante la creación del clúster, el comando debe ejecutarse con, como en: `--without-nodegroup`

```
eksctl create cluster -f outpost.yaml --without-nodegroup
```

Es responsabilidad del cliente asociar la VPC creada por eksctl con la puerta de enlace local después de la creación del clúster para permitir la conectividad con el servidor API. Después de este paso, se pueden crear grupos de nodos utilizando. `eksctl create nodegroup`

Si lo desea, puede especificar el tipo de instancia para los nodos del plano de control `outpost.controlPlaneInstanceType` o para los grupos de nodos incluidos`nodeGroup.instanceType`, pero el tipo de instancia debe existir en Outpost o eksctl devolverá un error. De forma predeterminada, eksctl intenta elegir el tipo de instancia más pequeño disponible en Outpost para los nodos y grupos de nodos del plano de control.

Cuando el plano de control está en Outposts, se crean grupos de nodos en ese Outpost. Si lo desea, puede especificar el ARN de Outpost para el grupo de nodos `nodeGroup.outpost`ARN en, pero debe coincidir con el ARN de Outpost del plano de control.

```
# 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 existente

Los clientes con una VPC existente pueden crear clústeres locales en AWS Outposts especificando la configuración `vpc.subnets` de subred en, como en:

```
# 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
```

Las subredes deben existir en el Outpost especificado en `outpost.controlPlaneOutpostARN` o eksctl devolverá un error. También puedes especificar grupos de nodos durante la creación del clúster si tienes acceso a la puerta de enlace local de la subred o si tienes conectividad a los recursos de la VPC.

Funciones no compatibles con los clústeres locales

- Complementos
- Funciones de IAM para cuentas de servicio
- IPv6
- Proveedores de identidad
- Fargate
- Cifrado KMS
- Zonas locales
- Karpenter
- Selector de instancias
- No se pueden especificar las zonas de disponibilidad, ya que el valor predeterminado es la zona de disponibilidad de Outpost.
- `vpc.publicAccessCIDRs` y `vpc.autoAllocateIPv6` no son compatibles.
- No se admite el acceso de punto final público al servidor de API, ya que un clúster local solo se puede crear con un acceso de punto final solo privado.

Más información

- Amazon EKS en AWS Outposts
- Clústeres locales para Amazon EKS en AWS Outposts
- Creación de clústeres locales
- Lanzamiento de nodos de Amazon Linux autogestionados en un Outpost

Seguridad

`eksctl`proporciona algunas opciones que pueden mejorar la seguridad de su clúster EKS.

`withOIDC`

Habilite `withOIDC`la creación automática de una IRSA para el complemento Amazon CNI y limite los permisos concedidos a los nodos de su clúster; en lugar de ello, conceda los permisos necesarios únicamente a la cuenta de servicio del CNI.

Los antecedentes se describen en esta documentación de AWS.

`disablePodIMDS`

Para los grupos de nodos gestionados y no gestionados, la `disablePodIMDS`opción está disponible e impide que todos los pods de red no hospedados que se ejecutan en este grupo de nodos realicen solicitudes de IMDS.

> **Note**
>
> Esto no se puede usar junto con. `withAddonPolicies`

Cifrado de envolvente KMS para clústeres EKS

> **Note**
>
> Amazon Elastic Kubernetes Service (Amazon EKS) aplica cifrado de sobre de forma predeterminada a todos los datos de la API de Kubernetes en los clústeres que ejecutan la versión 1.28 o superior. Para obtener más información, consulte el cifrado de sobres predeterminado para todos los datos de la API de Kubernetes en la Guía del usuario de EKS.

EKS admite el uso de claves de AWS KMS para cifrar en sobres los secretos de Kubernetes almacenados en EKS. El cifrado de sobres añade una capa de cifrado adicional, administrada por el cliente, para los secretos de las aplicaciones o los datos de los usuarios que se almacenan en un clúster de Kubernetes.

Anteriormente, Amazon EKS permitía habilitar el cifrado de sobres mediante claves de KMS solo durante la creación del clúster. Ahora, puede habilitar el cifrado de sobres para los clústeres de Amazon EKS en cualquier momento.

Obtenga más información sobre el uso del soporte del proveedor de cifrado EKS en una defense-in-depth publicación en el blog sobre contenedores de AWS.

Crear un clúster con el cifrado KMS activado

```
# 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
```

Habilitar el cifrado KMS en un clúster existente

Para habilitar el cifrado KMS en un clúster que aún no lo tiene habilitado, ejecute

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

o sin un archivo de configuración:

```
eksctl utils enable-secrets-encryption --cluster=kms-cluster --key-arn=arn:aws:kms:us-
west-2:<account>:key/<key> --region=<region>
```

Además de habilitar el cifrado de KMS en el clúster de EKS, eksctl también vuelve a cifrar todos los secretos de Kubernetes existentes con la nueva clave de KMS actualizándolos con la anotación. `eksctl.io/kms-encryption-timestamp` Este comportamiento se puede desactivar de forma pasajera, como en: `--encrypt-existing-secrets=false`

```
eksctl utils enable-secrets-encryption --cluster=kms-cluster --key-arn=arn:aws:kms:us-
west-2:<account>:key/<key> --encrypt-existing-secrets=false --region=<region>
```

Si un clúster ya tiene activado el cifrado KMS, eksctl procederá a volver a cifrar todos los secretos existentes.

> **Note**
>
> Una vez que se habilita el cifrado KMS, no se puede deshabilitar ni actualizar para usar una clave KMS diferente.

Solución de problemas

Este tema incluye instrucciones sobre cómo resolver errores comunes con Eksctl.

Error en la creación de la pila

Puedes usar la `--cfn-disable-rollback` marca para impedir que Cloudformation revierta las pilas fallidas y así facilitar la depuración.

El identificador de subred «subnet-" no es lo mismo que «subnet-22222222"

Dado un archivo de configuración que especifica las subredes para una VPC como el siguiente:

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

metadata:
  name: test
  region: us-east-1

vpc:
  subnets:
    public:
      us-east-1a: {id: subnet-11111111}
      us-east-1b: {id: subnet-22222222}
    private:
      us-east-1a: {id: subnet-33333333}
      us-east-1b: {id: subnet-44444444}

nodeGroups: []
```

Un error `subnet ID "subnet-11111111" is not the same as "subnet-22222222"` significa que las subredes especificadas no están ubicadas en la zona de disponibilidad correcta. Compruebe en la consola de AWS cuál es el ID de subred correcto para cada zona de disponibilidad.

En este ejemplo, la configuración correcta para la VPC sería:

```
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}
```

Problemas de eliminación

Si la eliminación no funciona o si te olvidas de añadir `--wait` la eliminación, es posible que tengas que utilizar las demás herramientas de Amazon para eliminar las pilas de formación de nubes. Esto se puede lograr mediante la interfaz gráfica de usuario o con la AWS CLI.

Los registros de kubectl y la ejecución de kubectl fallan y se produce un error de autorización

Si los nodos están desplegados en una subred privada `kubectl logs` o `kubectl run` fallan y se produce un error como el siguiente:

```
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)
```

Entonces puede que tengas que enableDnsHostnamesconfigurarlo. Se pueden encontrar más detalles en este número.

Anuncios

Este tema cubre los anuncios anteriores de las nuevas funciones de Eksctl.

Grupos de nodos gestionados (predeterminados)

A partir de la versión 0.58.0 de eksctl, eksctl crea grupos de nodos gestionados de forma predeterminada cuando no se especifica un archivo para y. `ClusterConfig` `eksctl create cluster` `eksctl create nodegroup` Para crear un grupo de nodos `--managed=false` autogestionado, pase. Esto puede interrumpir los scripts que no utilizan un archivo de configuración si se utiliza una función que no es compatible con los grupos de nodos administrados, por ejemplo, los grupos de nodos de Windows. Para solucionar este problema`--managed=false`, transfiere o especifica la configuración del grupo de nodos en un `ClusterConfig` archivo utilizando el `nodeGroups` campo que crea un grupo de nodos autogestionado.

Anulación de Bootstrap de Nodegroup para personalizarla AMIs

Este cambio se anunció en el número Breaking: soon... overrideBootstrapCommand Ahora, ha sucedido en este PR. Lea detenidamente el número adjunto para saber por qué decidimos dejar de admitir scripts personalizados AMIs sin bootstrap o con scripts bootstrap parciales.

¡Todavía ofrecemos un ayudante! Esperemos que migrar no sea tan doloroso. `eksctl`sigue proporcionando un script que, una vez creado, exportará un par de propiedades y ajustes útiles del entorno. Este script se encuentra aquí.

Las siguientes propiedades del entorno estarán a su disposición:

```
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
```

¡Lo mínimo que hay que usar al anular, para que `eksctl` no falle, son las etiquetas! `eksctl`depende de que haya un conjunto específico de etiquetas en el nodo para que pueda encontrarlas. Al definir la anulación, proporcione este comando de anulación mínimo:

```
    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}"
```

En el caso de los grupos de nodos que no tienen acceso saliente a Internet, tendrás que introducir `--apiserver-endpoint` y en el script de arranque de la `--b64-cluster-ca` siguiente manera:

```
    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}
```

Ten en cuenta la configuración `--node-labels`. Si no está definido, el nodo se unirá al clúster, pero al final se `eksctl` agotará el tiempo de espera en el último paso cuando esté esperando a que lleguen los nodos. `Ready` Está realizando una búsqueda en Kubernetes de los nodos que tienen la etiqueta. `alpha.eksctl.io/nodegroup-name=<cluster-name>` Esto solo es válido para los grupos de nodos no administrados. En el caso de los gestionados, se utiliza una etiqueta diferente.

Si es posible cambiar a grupos de nodos gestionados para evitar esta sobrecarga, ha llegado el momento de hacerlo. Hace que todas las tareas de anulación sean mucho más fáciles.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.