aws

Amazon EKS

Eksctl 使用者指南

Copyright © 2026 Copyright informaiton pending.

Eksctl 使用者指南: Amazon EKS

Copyright © 2026 Copyright informaiton pending.

著作權資訊待定。

Table of Contents

什麼是 Eksctl？ 1
功能 1
Eksctl 常見問答集 2
一般 2
節點群組 2
Ingress 2
kubectl 3
乾執行 3
eksctl 中的一次性選項 5
教學課程 6
步驟 1：安裝 eksctl 6
步驟 2：建立叢集組態檔案 7
步驟 3：建立叢集 7
選用：刪除叢集 8
後續步驟 8
Eksctl 的安裝選項 9
先決條件 9
針對 Unix 9
適用於 Windows 10
使用 Git Bash： 11
Homebrew 11
Docker 12
Shell 完成 12
Bash 12
Zsh 12
魚 12
PowerShell 13
更新 13
叢集 14
主題： 14
建立和管理叢集 16
建立簡單的叢集 16
使用組態檔案建立叢集 16
更新新叢集的 kubeconfig 18

刪除叢集 18
乾執行 19
EKS 自動模式 19
建立已啟用自動模式的 EKS 叢集 20
更新 EKS 叢集以使用自動模式 21
停用自動模式 21
詳細資訊 22
EKS 存取項目 22
叢集身分驗證模式 22
存取項目資源 23
建立存取項目 25
取得存取項目 25
刪除存取項目 26
從 aws-auth ConfigMap 遷移 26
停用叢集建立器管理員許可 27
非 eksctl 建立的叢集 27
支援的命令 27
建立節點群組 29
EKS 連接器 30
註冊叢集 30
取消註冊叢集 31
詳細資訊 22
設定 kubelet 31
kubeReserved 計算 32
CloudWatch 記錄功能 33
啟用 CloudWatch 記錄 33
ClusterConfig 範例 34
EKS 完全私有叢集 36
建立完全私有的叢集 36
設定其他 AWS 服務的私有存取權 37
節點群組 38
叢集端點存取 39
使用者提供的 VPC 和子網路 39
管理完全私有的叢集 40
強制刪除完全私有的叢集 40
限制 40

透過 HTTP 代理伺服器傳出存取 40
詳細資訊 22
附加元件 41
建立附加元件 41
列出已啟用的附加元件 43
設定附加元件的版本 44
探索附加元件 44
探索附加元件的組態結構描述 44
使用組態值 45
使用自訂命名空間 46
更新附加元件 46
刪除附加元件 47
預設聯網附加元件的叢集建立彈性 48
Amazon EMR 49
EKS Fargate 支援 49
使用 Fargate 支援建立叢集 49
使用組態檔案建立具有 Fargate 支援的叢集 51
設計 Fargate 設定檔 53
管理 Fargate 設定檔 54
深入閱讀 57
叢集升級 57
更新控制平面版本 58
預設附加元件更新 58
更新預先安裝的附加元件 59
啟用區域轉移 60
建立已啟用區域轉移的叢集 60
在現有叢集上啟用區域轉移 60
詳細資訊 22
Karpenter 支援 61
自動安全群組標記 63
叢集組態結構描述 64
節點群組 65
主題： 14
使用節點群組 67
建立節點群組 67
組態檔案中的節點群組選擇 69

列出節點群組 70
節點群組不可變性 71
擴展節點群組 71
刪除和耗盡節點群組 72
其他功能 72
未受管節點群組 74
更新多個節點群組 75
更新預設附加元件 75
EKS 受管節點群組 76
建立受管節點群組 76
升級受管節點群組 80
處理節點的平行升級 81
更新受管節點群組 81
Nodegroup 運作狀態問題 82
管理標籤 82
擴展受管節點群組 82
詳細資訊 22
節點引導 83
AmazonLinux2023 83
啟動範本支援 84
使用提供的啟動範本建立受管節點群組 84
升級受管節點群組以使用不同的啟動範本版本 85
自訂 AMI 和啟動範本支援的注意事項 85
自訂子網路 86
為什麼 86
方法 86
刪除叢集 87
自訂 DNS 87
污點 88
執行個體選取器 89
建立叢集和節點群組 89
Spot 執行個體 92
受管節點群組 92
未受管節點群組 94
GPU 支援 96
ARM 支援 97

Auto Scaling 98
啟用 Auto Scaling 98
自訂 AMI 支援 101
設定節點 AMI ID 101
設定節點 AMI 系列 102
Windows 自訂 AMI 支援 104
Bottlerocket 自訂 AMI 支援 105
Windows 工作者節點 105
使用 Windows 支援建立新的叢集 106
將 Windows 支援新增至現有的 Linux 叢集 107
其他磁碟區映射 107
EKS 混合節點 108
簡介 108
聯網 109
憑證 110
附加元件支援 111
進一步參考 111
節點修復組態 111
基本節點修復組態 112
增強型節點修復組態 112
完成組態範例 114
CLI 參考 116
組態參考 117
詳細資訊 22
聯網 119
主題： 14
VPC 組態 119
叢集的專用 VPC 119
變更 VPC CIDR 120
使用現有的 VPC：與 kops 共用 120
使用現有的 VPC：其他自訂組態 121
自訂共用節點安全群組 124
NAT 閘道 125
子網路設定 125
將私有子網路用於初始節點群組 125
自訂子網路拓撲 125

叢集存取 128
管理對 Kubernetes API 伺服器端點的存取 128
限制對 EKS Kubernetes Public API 端點的存取 129
控制平面聯網 130
更新控制平面子網路 130
更新控制平面安全群組 131
IPv6 支援 132
定義 IP 系列 132
IAM 134
主題： 14
最低 IAM 政策 135
IAM 許可界限 138
設定 VPC CNI 許可界限 139
IAM 政策 139
支援的 IAM 附加元件政策 139
新增自訂執行個體角色 140
連接內嵌政策 141
依 ARN 連接政策 141
管理 IAM 使用者和角色 142
使用 CLI 命令編輯 ConfigMap 142
使用 ClusterConfig 檔案編輯 ConfigMap 143
服務帳戶的 IAM 角色 144
運作方式 144
來自 CLI 的用量 145
使用組態檔案 146
詳細資訊 22
EKS Pod 身分關聯 149
先決條件 149
建立 Pod 身分關聯 150
擷取 Pod 身分關聯 151
更新 Pod 身分關聯 151
刪除 Pod 身分關聯 152
EKS 附加元件支援 Pod 身分關聯 152
將現有的 iamserviceaccounts 和附加元件遷移至 Pod 身分關聯 158
跨帳戶 Pod 身分支援 159
進一步參考 111

部署選項 161
主題： 14
EKS Anywhere 161
AWS Outposts 支援 162
將現有叢集擴展至 AWS Outpost 162
在 AWS Outposts 上建立本機叢集 163
本機叢集上不支援的功能 166
詳細資訊 22
安全 168
withOIDC 168
disablePodIMDS 168
KMS 加密 168
建立已啟用 KMS 加密的叢集 169
在現有叢集上啟用 KMS 加密 169
疑難排解 171
失敗的堆疊建立 171
子網路 ID "subnet-11111111" 與 "subnet-22222222" 不同 171
刪除問題 172
kubectl 日誌和 kubectl 執行失敗並出現授權錯誤 172
公告 173
受管節點群組預設 173
自訂 AMIs節點群組引導覆寫 173
..... clxxv

什麼是 Eksctl？

eksctl 是一種命令列公用程式工具，可自動化和簡化建立、管理和操作 Amazon Elastic Kubernetes Service (Amazon EKS) 叢集的程序。eksctl 以 Go 編寫，透過 YAML 組態和 CLI 命令提供宣告式語法，以處理複雜的 EKS 叢集操作，否則需要跨不同 AWS 服務進行多個手動步驟。

對於需要持續大規模部署和管理 EKS 叢集的 DevOps 工程師、平台團隊和 Kubernetes 管理員來說，eksctl 特別重要。對於從自我管理的 Kubernetes 轉換到 EKS 或實作基礎設施即程式碼 (IaC) 實務的組織來說特別有用，因為它可以整合到現有的 CI/CD 管道和自動化工作流程中。此工具可抽象化 EKS 叢集設定所需的 AWS 服務之間的許多複雜互動，例如 VPC 組態、IAM 角色建立和安全群組管理。

eksctl 的主要功能包括能夠使用單一命令建立功能完整的 EKS 叢集、支援自訂聯網組態、自動化節點群組管理和 GitOps 工作流程整合。該工具透過宣告式方法管理叢集升級、擴展節點群組和處理附加元件管理。eksctl 還提供進階功能，例如 Fargate 設定檔組態、受管節點群組自訂和 Spot 執行個體整合，同時透過原生 AWS 開發套件整合維持與其他 AWS 工具和服務的相容性。

功能

目前實作的功能如下：

- 建立、取得、列出和刪除叢集
- 建立、耗盡和刪除節點群組
- 擴展節點群組
- 更新 叢集
- 使用自訂 AMIs
- 設定 VPC 網路
- 設定 API 端點的存取
- 支援 GPU 節點群組
- Spot 執行個體和混合執行個體
- IAM 管理和附加元件政策
- 列出叢集 Cloudformation 堆疊
- 安裝 coredns
- 為叢集寫入 kubeconfig 檔案

Eksctl 常見問答集

一般

我可以使用 **eksctl**來管理 未建立的叢集**eksctl**嗎？

是！從版本0.40.0中，您可以eksctl針對任何叢集執行 ，無論它是否由 建立eksctl。如需詳細資訊，請參閱the section called “非 eksctl 建立的叢集”。

節點群組

如何變更節點群組的執行個體類型？

從 的觀點來看eksctl，節點群組是不可變的。這表示一旦建立 ，唯一eksctl可以做的就是向上或向下擴展節點群組。

若要變更執行個體類型，請使用所需的執行個體類型建立新的節點群組，然後耗盡它，讓工作負載移至新的節點群組。在該步驟完成後，您可以刪除舊的節點群組。

如何查看節點群組產生的使用者資料？

首先，您需要管理節點群組的 Cloudformation 堆疊名稱：

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

您將看到類似 的名稱eksctl-CLUSTER_NAME-nodegroup-NODEGROUP_NAME。

您可以執行下列動作來取得使用者資料。請注意從 base64 解碼和解壓縮 gziped 資料的最後一行。

```
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

如何使用 設定輸入**eksctl**？

建議使用 AWS Load Balancer 控制器。有關如何將控制器部署到您的叢集，以及如何從舊的 ALB 傳入控制器遷移的文件，可在這裡找到。

對於 Nginx 傳入控制器，設定會與其他 Kubernetes 叢集上的設定相同。

kubectl

我使用的是 HTTPS 代理和叢集憑證驗證失敗，如何使用系統 CAs？

設定環境變數KUBECONFIG_USE_SYSTEM_CA以kubeconfig遵守系統憑證授權單位。

乾執行

試轉功能可讓您檢查和變更執行個體選取器相符的執行個體，再繼續建立節點群組。

當 使用執行個體選取器選項 和 `eksctl create cluster`呼叫 時`--dry-run`，eksctl 將輸出包含節點群組的 ClusterConfig 檔案，該節點群組代表 CLI 選項和執行個體類型設定為符合執行個體選取器資源條件的執行個體。

```
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
```

產生的 ClusterConfig 可接著傳遞至 `eksctl create cluster`：

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

使用 傳遞 ClusterConfig 檔案時`--dry-run`，eksctl 會輸出包含檔案中設定值的 ClusterConfig 檔案。

eksctl 中的一次性選項

有些一次性選項無法在 `ClusterConfig` 檔案中表示，例如 `--install-vpc-controllers`。

預期：

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

後面接著：

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

等同於在沒有 的情況下執行第一個命令`--dry-run`。

因此，eksctl 不允許`--dry-run`傳遞在傳遞時無法在組態檔案中表示的選項。

> **Important**
>
> 如果您需要傳遞 AWS 設定檔，請設定`AWS_PROFILE`環境變數，而不是傳遞 CLI `--profile` 選項。

教學課程

本主題會逐步引導您安裝和設定 eksctl，然後使用它來建立 Amazon EKS 叢集。

步驟 1：安裝 eksctl

請完成下列步驟，在您的 Linux 或 macOS 裝置上下載並安裝最新版本的 eksctl：

使用 Homebrew 安裝 eksctl

1. （先決條件）安裝 Homebrew。
2. 新增 AWS 輕點：

```
brew tap aws/tap
```

3. 安裝 eksctl

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

使用 eksctl 之前，請完成下列組態步驟：

1. 安裝先決條件：
   - 安裝 AWS CLI 2.x 版或更新版本。
   - 使用 Homebrew 安裝 kubectl：

```
brew install kubernetes-cli
```

2. 在您的環境中設定 AWS 登入資料：

```
aws configure
```

3. 驗證 AWS CLI 組態：

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

步驟 2：建立叢集組態檔案

使用下列步驟建立叢集組態檔案：

1. 建立名為 的新檔案cluster.yaml：

```
touch cluster.yaml
```

2. 新增下列基本叢集組態：

```
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. 自訂組態：
   - 更新 region以符合您想要的 AWS 區域。
   - instanceType 根據您的工作負載需求修改 。
   - maxSize 根據您的需求調整 minSize、 desiredCapacity和 。
4. 驗證組態檔案：

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

步驟 3：建立叢集

請依照下列步驟建立 EKS 叢集：

1. 使用組態檔案建立叢集：

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

2. 等待叢集建立（這通常需要 15-20 分鐘）。
3. 驗證叢集建立：

```
eksctl get cluster
```

4. 設定 kubectl 以使用您的新叢集：

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

5. 驗證叢集連線：

```
kubectl get nodes
```

您的叢集現在可以使用。

選用：刪除叢集

完成後請記得刪除叢集，以避免不必要的費用：

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

> **Note**
>
> 建立叢集可能會產生 AWS 費用。建立叢集之前，請務必檢閱 Amazon EKS 定價。

後續步驟

- 設定 Kubectl 以連線至叢集
- 部署範例應用程式

Eksctl 的安裝選項

eksctl 可從官方版本安裝 ，如下所述。我們建議您只eksctl從官方 GitHub 版本安裝 。您可以選擇使用第三方安裝程式，但請注意，AWS 不會維護或支援這些安裝方法。由您自行決定使用它們。

先決條件

您需要設定 AWS API 登入資料。適用於 AWS CLI 或任何其他工具 (kops、Terraform 等） 的 應該已足夠。您可以使用~/.aws/credentials檔案或環境變數。如需詳細資訊，請參閱 AWS CLI 參考。

您也需要在 中使用 AWS IAM Authenticator for Kubernetes 命令 (aws-iam-authenticator或 aws eks get-token (適用於 AWS CLI 1.16.156 版或更新版本) PATH。

用於建立 EKS 叢集的 IAM 帳戶應具有這些最低存取層級。

AWS 服務	存取層級
CloudFormation	完整存取
EC2	完整：標記限制：列出、讀取、寫入
EC2 Auto Scaling	有限：列出、寫入
EKS	完整存取
IAM	有限：列出、讀取、寫入、許可管理
Systems Manager	有限：清單、讀取

> **Note**
>
> eksctl v0.215.0 不再發佈 ARMv6 和 ARMv7 二進位檔。arm64 用於 ARM 系統。

針對 Unix

若要下載最新版本，請執行：

```
# for ARM systems, set ARCH to: `arm64`
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
```

適用於 Windows

直接下載（最新版本）：

- AMD64/x86_64
- ARM64

請務必將封存解壓縮至 PATH變數中的資料夾。

或者，驗證檢查總和：

1. 下載檢查總和檔案：最新
2. 使用命令提示來手動比較 CertUtil的輸出與下載的檢查總和檔案。

```
  REM Replace amd64 with arm64
  CertUtil -hashfile eksctl_Windows_amd64.zip SHA256
```

3. 使用 PowerShell 使用-eq運算子自動驗證以取得 True或 False結果：

```
# Replace amd64 with arm64
 (Get-FileHash -Algorithm SHA256 .\eksctl_Windows_amd64.zip).Hash -eq ((Get-Content .
\eksctl_checksums.txt) -match 'eksctl_Windows_amd64.zip' -split ' ')[0]
```

使用 Git Bash：

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

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

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

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

rm eksctl_$PLATFORM.zip
```

`eksctl` 可執行檔會放置在 中`$HOME/bin`，其位於 `$PATH` Git Bash 的 中。

Homebrew

您可以使用 Homebrew 在 MacOS 和 Linux 上安裝軟體。

AWS 會維護 Homebrew 點擊，包括 eksctl。

如需 Homebrew tap 的詳細資訊，請參閱 Github 上的專案和 eksctl 的 Homebrew 公式。

使用 Homebrew 安裝 eksctl

1. （先決條件）安裝 Homebrew
2. 新增 AWS 點擊

```
brew tap aws/tap
```

3. 安裝 eksctl

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

Docker

對於每個版本和 RC，容器映像都會推送到 ECR 儲存庫 `public.ecr.aws/eksctl/eksctl`。進一步了解 ECR Public Gallery - eksctl 上的用量。例如

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

Shell 完成

Bash

若要啟用 bash 完成，請執行以下操作，或將其放入 `~/.bashrc`或 `~/.profile`：

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

Zsh

若要完成 zsh，請執行：

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

並將以下內容放入 `~/.zshrc`：

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

請注意，如果您不是執行 oh-my-zsh 之類的分佈，則可能必須先啟用自動完成（並放入 `~/.zshrc` 使其持久）：

```
autoload -U compinit
compinit
```

魚

下列命令可用於魚自動完成：

```
mkdir -p ~/.config/fish/completions
```

```
eksctl completion fish > ~/.config/fish/completions/eksctl.fish
```

PowerShell

您可以參考下列命令來進行設定。請注意，視您的系統設定而定，路徑可能不同。

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

更新

> **Important**
>
> 如果您透過直接下載安裝 eksctl （不使用套件管理員），則需要手動更新它。

叢集

本章涵蓋使用 eksctl 建立和設定 EKS 叢集。它還包含附加元件和 EKS 自動模式。

主題：

- the section called “EKS 存取項目”
  - 將 aws-auth ConfigMap 取代為 EKS 存取項目，以簡化 Kubernetes RBAC 管理
  - 從 aws-auth ConfigMap 遷移現有的 IAM 身分映射以存取項目
  - 設定叢集身分驗證模式並控制叢集建立器管理員許可
- the section called “預設附加元件更新”
  - 透過更新較舊叢集上的預設 EKS 附加元件來保護叢集安全
- the section called “附加元件”
  - 自動化安裝、更新和移除附加元件的例行任務。
  - Amazon EKS 附加元件包括 AWS 附加元件、開放原始碼社群附加元件和市集附加元件。
- the section called “EKS 自動模式”
  - 讓 AWS 管理您的 EKS 基礎設施，以減少營運開銷
  - 設定自訂節點集區，而不是預設的一般用途和系統集區
  - 將現有的 EKS 叢集轉換為使用自動模式
- the section called “CloudWatch 記錄功能”
  - 透過啟用特定 EKS 控制平面元件的日誌來疑難排解叢集問題
  - 設定 EKS 叢集日誌的日誌保留期間
  - 使用 eksctl 命令修改現有的叢集記錄設定
- the section called “叢集升級”
  - 透過安全地升級 EKS 控制平面版本來維護安全性和穩定性
  - 透過將舊群組取代為新群組，跨節點群組推展升級
  - 更新預設叢集附加元件
- the section called “建立和管理叢集”
  - 使用預設受管節點群組快速開始使用基本 EKS 叢集
  - 使用具有特定組態的組態檔案建立自訂叢集
  - 使用私有聯網和自訂 IAM 政策在現有 VPCs 中部署叢集

- the section called “設定 kubelet”
  - 透過設定 kubelet 和系統協助程式保留來防止節點資源匱乏
  - 自訂記憶體和檔案系統可用性的移出閾值
  - 啟用或停用跨節點群組的特定 kubelet 功能閘道
- the section called “EKS 連接器”
  - 透過 EKS 主控台集中管理混合 Kubernetes 部署
  - 設定外部叢集存取的 IAM 角色和許可
  - 移除外部叢集並清除相關聯的 AWS 資源
- the section called “EKS 完全私有叢集”
  - 完全私有的 EKS 叢集沒有傳出網際網路存取，可滿足安全需求
  - 透過 VPC 端點設定 AWS 服務的私有存取權
  - 使用明確的聯網設定建立和管理私有節點群組
- the section called “Karpenter 支援”
  - 自動化節點佈建
  - 建立自訂 Karpenter 佈建器組態
  - 使用 Spot 執行個體中斷處理設定 Karpenter
- the section called “Amazon EMR”
  - 在 EMR 和 EKS 叢集之間建立 IAM 身分映射
- the section called “EKS Fargate 支援”
  - 定義 Pod 排程的自訂 Fargate 設定檔
  - 透過建立和組態更新來管理 Fargate 設定檔
- the section called “非 eksctl 建立的叢集”
  - 標準化在 eksctl 外部建立的叢集管理
  - 在現有的非 eksctl 叢集上使用 eksctl 命令
- the section called “啟用區域轉移”
  - 透過啟用快速區域容錯移轉功能來提高應用程式可用性
  - 在新的 EKS 叢集部署上設定區域轉移
  - 在現有 EKS 叢集上啟用區域轉移功能

建立和管理叢集

本主題說明如何使用 Eksctl 建立和刪除 EKS 叢集。您可以使用 CLI 命令或透過建立叢集組態 YAML 檔案來建立叢集。

建立簡單的叢集

使用下列命令建立簡單的叢集：

```
eksctl create cluster
```

這將在預設區域中建立 EKS 叢集 （ 如您的 AWS CLI 組態所指定 ），其中包含一個包含兩個 m5.large 節點的受管節點群組。

eksctl 現在預設會在不使用組態檔案時建立受管節點群組。若要建立自我管理節點群組，請傳遞`--managed=false`至 `eksctl create cluster`或 `eksctl create nodegroup`。

考量事項

- 在 中建立叢集時`us-east-1`，您可能會遇到 `UnsupportedAavailabilityZoneException`。如果發生這種情況，請複製建議的區域並傳遞 `--zones`旗標，例如：`eksctl create cluster --region=us-east-1 --zones=us-east-1a,us-east-1b,us-east-1d`。此問題可能發生在其他區域，但較不常見。在大多數情況下，您不需要使用 `--zone`旗標。

使用組態檔案建立叢集

您可以使用組態檔案來建立叢集，而不是旗標。

首先，建立`cluster.yaml`檔案：

```
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
```

接著，執行此命令：

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

這將如所述建立叢集。

如果您需要使用現有的 VPC，您可以使用如下的組態檔案：

```
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
```

叢集名稱或節點群組名稱只能包含英數字元（區分大小寫）和連字號。它必須以字母字元開頭，且不得超過 128 個字元，否則您會收到驗證錯誤。如需詳細資訊，請參閱 AWS CLoudFormation 使用者指南中的從 CloudFormation 主控台建立堆疊。 CLoudFormation

更新新叢集的 kubeconfig

建立叢集之後，適當的 kubernetes 組態會新增至您的 kubeconfig 檔案。這是您在環境變數中或KUBECONFIG~/.kube/config預設情況下設定的檔案。您可以使用 --kubeconfig旗標覆寫 kubeconfig 檔案的路徑。

可變更 kubeconfig 檔案寫入方式的其他旗標：

標記	type	use	預設值
--kubeconfig	string	寫入 kubeconfig 的路徑（與 --auto-kubeconfig 不相容）	$KUBECONFIG 或 ~/.kube/config
--set-kubeconfig-context	bool	如果為 true，則會在 kubeconfig 中設定目前內容；如果已設定內容，則會覆寫它	true
--auto-kubeconfig	bool	依叢集名稱儲存 kubeconfig 檔案	true
--write-kubeconfig	bool	kubeconfig 的切換寫入	true

刪除叢集

若要刪除此叢集，請執行：

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

> ⚠ Warning
>
> 使用 `--wait`旗標搭配刪除操作，以確保正確報告刪除錯誤。

如果沒有`--wait`旗標，eksctl 只會對叢集的 CloudFormation 堆疊發出刪除操作，而且不會等待刪除。在某些情況下，使用叢集或其 VPC 的 AWS 資源可能會導致叢集刪除失敗。如果您的刪除失敗或忘記等待標記，您可能必須前往 CloudFormation GUI 並從該處刪除 eks 堆疊。

> ⚠ Warning
>
> PDB 政策可能會在叢集刪除期間封鎖節點移除。

刪除具有節點群組的叢集時，Pod 中斷預算 (PDB) 政策可以防止節點成功移除。例如，`aws-ebs-csi-driver`已安裝 的叢集通常有兩個具有 PDB 政策的 Pod，一次只允許一個 Pod 無法使用，使得另一個 Pod 在刪除期間無法清除。若要在這些案例中成功刪除叢集，請使用 `disable-nodegroup-eviction`旗標略過 PDB 政策檢查：

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

如需更多範例組態檔案，請參閱 eksctl GitHub 儲存庫中的 `examples/`目錄。

乾執行

試執行功能可產生 ClusterConfig 檔案，略過叢集建立並輸出代表所提供 CLI 選項的 ClusterConfig 檔案，並包含 eksctl 設定的預設值。

如需詳細資訊，請參閱 Dry Run 頁面。

EKS 自動模式

eksctl 支援 EKS Auto Mode，這項功能可將 Kubernetes 叢集的 AWS 管理延伸到叢集本身之外，讓 AWS 也能設定和管理基礎設施，讓工作負載能夠順暢地運作。這可讓您委派關鍵基礎設施決策，並利用 AWS 的專業知識進行day-to-day操作。由 AWS 管理的叢集基礎設施包含許多 Kubernetes 功能

做為核心元件，而不是附加元件，例如運算自動擴展、Pod 和服務聯網、應用程式負載平衡、叢集 DNS、區塊儲存和 GPU 支援。

建立已啟用自動模式的 EKS 叢集

eksctl 已新增 autoModeConfig 欄位以啟用和設定自動模式。autoModeConfig 欄位的形狀為

```
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
```

如果 autoModeConfig.enabled 為 true，eksctl 會透過將 computeConfig.enabled: true、 kubernetesNetworkConfig.elasticLoadBalancing.enabled: true和 傳遞storageConfig.blockStorage.enabled: true至 EKS API 來建立 EKS 叢集，進而管理運算、儲存和聯網等資料平面元件。

若要在啟用自動模式的情況下建立 EKS 叢集，請如 所示設定 autoModeConfig.enabled: true。

```
# 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 會建立節點角色，以用於自動模式啟動的節點。eksctl 也會建立 general-purpose和 system 節點集區。若要停用預設節點集區的建立，例如，設定您自己的節點集區，使用不同的子網路集區，請設定 nodePools: []，如 中的

```
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.
```

更新 EKS 叢集以使用自動模式

若要更新現有的 EKS 叢集以使用自動模式，請執行

```
# 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**
>
> 如果叢集是由 eksctl 建立，並使用公有子網路做為叢集子網路，則自動模式會在公有子網路中啟動節點。若要針對自動模式啟動的工作者節點使用私有子網路，請更新叢集以使用私有子網路。

停用自動模式

若要停用自動模式，請設定 `autoModeConfig.enabled: false` 並執行

```
# 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
```

詳細資訊

- EKS 自動模式

EKS 存取項目

您可以使用 eksctl 來管理 EKS 存取項目。使用存取項目將 Kubernetes 許可授予 AWS IAM 身分。例如，您可以授予開發人員角色讀取叢集中 Kubernetes 資源的許可。

本主題說明如何使用 eksctl 來管理存取項目。如需存取項目的一般資訊，請參閱使用 EKS 存取項目授予 IAM 使用者對 Kubernetes 的存取權。

您可以連接 AWS 定義的 Kubernetes 存取政策，或與 Kubernetes 群組建立關聯。

如需可用預先定義政策的詳細資訊，請參閱將存取政策與存取項目建立關聯。

如果您需要定義客戶 Kubernetes 政策，請將 IAM Identity 與 Kubernetes 群組建立關聯，並將許可授予該群組。

叢集身分驗證模式

只有在叢集的身分驗證模式允許時，您才能使用存取項目。

如需詳細資訊，請參閱設定叢集身分驗證模式

使用 YAML 檔案設定身分驗證模式

`eksctl` 已在 ClusterConfig 下新增`accessConfig.authenticationMode`欄位，可設定為下列三個值之一：

- `CONFIG_MAP` - EKS API 中的預設值 - 只會使用 `aws-auth` ConfigMap

- API - 只會使用存取項目 API
- API_AND_CONFIG_MAP - 預設 - eksctl 可使用 aws-auth ConfigMap 和存取項目 API

在 ClusterConfig YAML 中設定身分驗證模式：

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

使用 命令更新身分驗證模式

如果您想要在已建立的非eksctl 現有叢集上使用存取項目，其中使用 CONFIG_MAP選項，使用者必須先authenticationMode將 設定為 API_AND_CONFIG_MAP。為此， eksctl 推出了更新叢集身分驗證模式的新命令，這兩者都適用於 CLI 旗標，例如

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

存取項目資源

存取項目具有類型，例如 STANDARD或 EC2_LINUX。類型取決於您使用存取項目的方式。

- standard 類型用於將 Kubernetes 許可授予 IAM 使用者和 IAM 角色。
  - 例如，您可以將存取政策連接至您用來存取主控台的角色或使用者，以在 AWS 主控台中檢視 Kubernetes 資源。
- EC2_LINUX 和 EC2_WINDOWS類型用於將 Kubernetes 許可授予 EC2 執行個體。執行個體使用這些許可來加入叢集。

如需存取項目類型的詳細資訊，請參閱建立存取項目

IAM 實體

您可以使用存取項目將 Kubernetes 許可授予 IAM 身分，例如 IAM 使用者和 IAM 角色。

使用 accessConfig.accessEntries 欄位將 IAM 資源的 ARN 與 Access Entries EKS API 建立關聯。例如：

```
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
```

除了關聯 EKS 政策之外，也可以指定 IAM 實體所屬的 Kubernetes 群組，進而透過 RBAC 授予許可。

受管節點群組和 Fargate

EKS API 會在幕後實現與這些資源存取項目的整合。新建立的受管節點群組和 Fargate Pod 將建立 API 存取項目，而不是使用預先載入的 RBAC 資源。現有的節點群組和 Fargate Pod 不會變更，並繼續依賴 aws-auth 組態映射中的項目。

自我管理節點群組

每個存取項目都有一個類型。對於授權自我管理節點群組， `eksctl`會為每個節點群組建立唯一的存取項目，並將主體 ARN 設定為節點角色 ARN，並將類型設定為 `EC2_LINUX`或`EC2_WINDOWS`取決於節點群組 amiFamily。

建立您自己的存取項目時，您也可以指定 `EC2_LINUX`（適用於與 Linux 或 Bottlerocket 自我管理節點搭配使用的 IAM 角色）、 `EC2_WINDOWS` （適用於與 Windows 自我管理節點搭配使用的 IAM 角

色）、 FARGATE_LINUX （適用於與 AWS Fargate (Fargate) 搭配使用的 IAM 角色） 或 STANDARD類型。如果您未指定類型，預設類型會設為 STANDARD。

> **Note**
>
> 刪除以預先存在的 建立的節點群組時instanceRoleARN，當沒有更多節點群組與其相關聯時，使用者有責任刪除對應的存取項目。這是因為 eksctl 不會嘗試查明非 eksctl 建立的自我管理節點群組是否仍在使用存取項目，因為它是一個複雜的程序。

建立存取項目

這可以在叢集建立期間以兩種不同的方式完成，指定所需的存取項目做為組態檔案的一部分並執行：

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

OR 建立叢集後，透過執行：

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

如需建立存取項目的範例組態檔案，請參閱 eksctl GitHub 儲存庫中的 40-access-entries.yaml。

取得存取項目

使用者可以執行下列其中一項，擷取與特定叢集相關聯的所有存取項目：

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

或

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

或者，若要僅擷取與特定 IAM 實體對應的存取項目，則應使用 --principal-arn旗標，例如

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

刪除存取項目

若要一次刪除單一存取項目，請使用：

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

若要刪除多個存取項目，請使用 `--config-file`旗標，並在最上層`accessEntry`欄位下指定與存取項目`principalARN's`對應的所有 ，例如

```
...
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
```

從 aws-auth ConfigMap 遷移

使用者可以執行下列動作，將現有的 IAM 身分從 configmap `aws-auth` 遷移至存取項目：

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

當`--target-authentication-mode`旗標設為 時`API`，身分驗證模式會切換為 API 模式 （ 如果已在 API 模式則略過 ）、IAM 身分映射會遷移至存取項目，並從叢集中刪除`aws-auth`組態對應。

當`--target-authentication-mode`旗標設定為 時`API_AND_CONFIG_MAP`，身分驗證模式會切換為 API_AND_CONFIG_MAP 模式 （ 如果已在`API_AND_CONFIG_MAP`模式下略過 ），IAM 身分映射會遷移至存取項目，但`aws-auth`組態對應會保留。

> **Note**
>
> `--target-authentication-mode` 旗標設定為 時`API`，如果 configmap `aws-auth` 具有下列其中一個限制條件，則此命令不會將身分驗證模式更新為 API 模式。

- 有一個帳戶層級身分映射。

- 一或多個角色/使用者會對應至開頭為字首 (EKS 特定群組`system:`除外，即 `system:masters`、`system:bootstrapperssystem:nodes`等 ） 的 kubernetes 群組。
- 一或多個 IAM 身分映射適用於 【服務連結角色】(link：IAM/latest/UserGuide/using-service-linked-roles.html)。

停用叢集建立器管理員許可

`eksctl` 已新增 欄位，`accessConfig.bootstrapClusterCreatorAdminPermissions: boolean`當 設定為 false 時， 會停用將叢集管理員許可授予建立叢集的 IAM 身分，即

將 選項新增至組態檔案：

```
accessConfig:
  bootstrapClusterCreatorAdminPermissions: false
```

和 執行：

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

非 eksctl 建立的叢集

您可以針對 未建立的叢集執行`eksctl`命令`eksctl`。

> **Note**
>
> Eksctl 只能支援名稱與 AWS CloudFormation 相容的未擁有叢集。任何不相符的叢集名稱都會失敗 CloudFormation API 驗證檢查。

支援的命令

下列命令可用於透過 以外的任何方式建立的叢集`eksctl`。命令、旗標和組態檔案選項的使用方式完全相同。

如果我們遺漏了一些功能，請讓我們知道。

✓ 建立：

- ✓ eksctl create nodegroup(請參閱以下備註)
- ✓ eksctl create fargateprofile
- ✓ eksctl create iamserviceaccount
- ✓ eksctl create iamidentitymapping

✓ 取得：

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

✓ 刪除：

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

✓ 升級：

- ✓ eksctl upgrade cluster
- ✓ eksctl upgrade nodegroup

✓ 設定/取消設定：

- ✓ eksctl set labels
- ✓ eksctl unset labels

✓ 擴展：

- ✓ eksctl scale nodegroup

✓ 耗盡：

- ✓ eksctl drain nodegroup

✓ 啟用：

- ✓ eksctl enable profile
- ✓ eksctl enable repo

✓ Utils：

- ✓ eksctl utils associate-iam-oidc-provider

- ✓ eksctl utils describe-stacks
- ✓ eksctl utils install-vpc-controllers
- ✓ eksctl utils nodegroup-health
- ✓ eksctl utils set-public-access-cidrs
- ✓ eksctl utils update-cluster-endpoints
- ✓ eksctl utils update-cluster-logging
- ✓ eksctl utils write-kubeconfig
- ✓ eksctl utils update-coredns
- ✓ eksctl utils update-aws-node
- ✓ eksctl utils update-kube-proxy

建立節點群組

eksctl create nodegroup 是唯一需要使用者特定輸入的命令。

由於使用者可以使用他們喜歡的任何聯網組態建立叢集，因此在時間範圍內， eksctl 不會嘗試擷取或猜測這些值。隨著我們進一步了解人們如何在非 eksctl 建立的叢集上使用此命令，這可能會在未來變更。

這表示為了在非 建立的叢集上建立節點群組或受管節點群組eksctl，必須提供包含 VPC 詳細資訊的組態檔案。至少：

```
---
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"

...
```

如需 VPC 組態選項的詳細資訊，請參閱聯網。

使用 EKS 連接器註冊非 EKS 叢集

您可以使用 EKS 連接器在 EKS 主控台中檢視 AWS 外部的叢集。此程序需要向 EKS 註冊叢集，並在外部 Kubernetes 叢集上執行 EKS Connector 代理程式。

`eksctl` 透過建立必要的 AWS 資源，並為 EKS Connector 產生 Kubernetes 資訊清單以套用至外部叢集，簡化註冊非 EKS 叢集。

註冊叢集

若要註冊或連接非 EKS Kubernetes 叢集，請執行

```
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
```

此命令會註冊叢集，並寫入三個檔案，其中包含 EKS Connector 的 Kubernetes 資訊清單，這些資訊清單必須在註冊到期之前套用至外部叢集。

> **Note**
>
> eks-connector-clusterrole.yaml 和 會將所有命名空間中 Kubernetes 資源的 get 和 list許可eks-connector-console-dashboard-full-access-clusterrole.yaml授予呼叫的 IAM 身分，且必須視需要進行相應編輯，才能將其套用至叢集。若要設定更多限制存取，請參閱授予使用者檢視叢集的存取權。

若要提供用於 EKS 連接器的現有 IAM 角色，請依照下列--role-arn方式透過 傳遞：

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

如果叢集已存在，eksctl 會傳回錯誤。

取消註冊叢集

若要取消註冊或中斷連接已註冊的叢集，請執行

```
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
```

此命令將取消註冊外部叢集並移除其相關聯的 AWS 資源，但您必須從叢集中移除 EKS 連接器 Kubernetes 資源。

詳細資訊

- EKS 連接器

自訂 kubelet 組態

系統資源可以透過 kubelet 的組態保留。建議這麼做，因為在資源匱乏的情況下，kubelet 可能無法移出 Pod，最終使節點變成 NotReady。若要這樣做，組態檔案可以包含 欄位，該kubeletExtraConfig欄位接受將內嵌到 中的自由格式 yamlkubelet.yaml。

中的某些欄位kubelet.yaml是由 eksctl 設定，因此不會過度寫入，例如 address、clusterDomain、authorization、 authentication或 serverTLSBootstrap。

下列範例組態檔案會建立節點群組，為 kubelet 保留 300m vCPU、300Mi記憶體和1Gi暫時性儲存；為 OS 系統常駐程式保留 300m vCPU、300Mi記憶體和1Gi暫時性儲存；當可用200Mi記憶體少於或根檔案系統少於 10% 時， 會開始移出 Pod。

```
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
```

在此範例中，假設 類型的執行個體m5a.xlarge有 4 個 vCPUs和 16GiB 的記憶體，CPUs Allocatable數量將為 3.4 和 15.4 GiB 的記憶體。請務必了解，在 中欄位的組態檔案中指定的值kubeletExtraconfig將完全覆寫 eksctl 指定的預設值。不過，省略一或多個kubeReserved參數會導致缺少的參數根據所使用的 aws 執行個體類型預設為 Sane 值。

kubeReserved 計算

雖然一般建議將混合執行個體 NodeGroup 設定為使用具有相同 CPU 和 RAM 組態的執行個體，但這不是嚴格的要求。因此，kubeReserved計算會使用 InstanceDistribution.InstanceTypes

欄位中最小的執行個體。如此一來，具有不同執行個體類型的 NodeGroups 就不會在最小執行個體上保留太多資源。不過，這可能會導致對最大執行個體類型而言太小的保留。

> **Warning**
>
> 根據預設， 會`eksctl`設定`featureGates.RotateKubeletServerCertificate=true`，但在`featureGates`提供自訂時，將會取消設定。除非您必須停用`featureGates.RotateKubeletServerCertificate=true`，否則您應該一律包含 。

CloudWatch 記錄功能

本主題說明如何為 EKS 叢集的控制平面元件設定 Amazon CloudWatch 記錄。CloudWatch 記錄可讓您了解叢集的控制平面操作，這對於疑難排解問題、稽核叢集活動和監控 Kubernetes 元件的運作狀態至關重要。

啟用 CloudWatch 記錄

由於資料擷取和儲存成本，EKS 控制平面的 CloudWatch 記錄預設不會啟用。

若要在建立叢集時啟用控制平面記錄，您需要在 中定義**`cloudWatch.clusterLogging.enableTypes`**設定 `ClusterConfig`（請參閱以下範例）。

因此，如果您的組態檔案具有正確的**`cloudWatch.clusterLogging.enableTypes`**設定，您可以使用 建立叢集`eksctl create cluster --config-file=<path>`。

如果您已建立叢集，您可以使用 `eksctl utils update-cluster-logging`。

> **Note**
>
> 根據預設，此命令會以計劃模式執行，您需要指定 `--approve` 旗標，才能將變更套用至叢集。

如果您使用的是組態檔案，請執行：

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

或者，您可以使用 CLI 旗標。

若要啟用所有類型的日誌，請執行：

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

若要啟用audit日誌，請執行：

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

若要啟用除controllerManager日誌以外的所有 日誌，請執行：

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

如果已啟用 api和 scheduler日誌類型，若要controllerManager同時停用scheduler和啟用 ，請執行：

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

這會將 api和 保留controllerManager為唯一啟用的日誌類型。

若要停用所有類型的日誌，請執行：

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

ClusterConfig 範例

在 EKS 叢集中， 下的 enableTypes 欄位clusterLogging可以取得可能值的清單，以啟用控制平面元件的不同類型日誌。

以下為可能值：

- api：啟用 Kubernetes API 伺服器日誌。
- audit：啟用 Kubernetes 稽核日誌。
- authenticator：啟用驗證器日誌。
- controllerManager：啟用 Kubernetes 控制器管理員日誌。

- `scheduler`：啟用 Kubernetes 排程器日誌。

若要進一步了解，請參閱 EKS 文件。

停用所有日誌

若要停用所有類型，請使用 `[]` 或完全移除 `cloudWatch`區段。

啟用所有日誌

您可以使用 `"*"`或 啟用所有類型`"all"`。例如：

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

啟用一或多個日誌

您可以透過列出您要啟用的類型來啟用類型子集。例如：

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

日誌保留期間

根據預設，日誌會無限期儲存在 CloudWatch Logs 中。您可以指定在 CloudWatch Logs 中保留控制平面日誌的天數。下列範例會保留日誌 7 天：

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

完成範例

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

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

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

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

EKS 完全私有叢集

eksctl 支援建立沒有傳出網際網路存取且只有私有子網路的完全私有叢集。VPC 端點用於啟用 AWS 服務的私有存取。

本指南說明如何在沒有傳出網際網路存取的情況下建立私有叢集。

建立完全私有的叢集

建立完全私有叢集的唯一必要欄位是 `privateCluster.enabled`：

```
privateCluster:
  enabled: true
```

建立叢集後，需要存取 Kubernetes API 伺服器的 eksctl 命令必須從叢集的 VPC 內執行、對等 VPC 或使用 AWS Direct Connect 等其他方式執行。如果 eksctl 命令是從叢集的 VPC 內執行，則需要存取 EKS APIs 的 eksctl 命令將無法運作。若要修正此問題，請建立 Amazon EKS 的介面端點，從 Amazon Virtual Private Cloud (VPC) 私下存取 Amazon Elastic Kubernetes Service (Amazon EKS) 管理 APIs。在未來版本中，eksctl 將新增建立此端點的支援，因此不需要手動建立。啟用適用於 Amazon EKS 的 AWS PrivateLink 後，需要存取 OpenID Connect 提供者 URL 的命令將需要從叢集 VPC 外部執行。

建立受管節點群組會繼續運作，而且如果命令是從叢集的 VPC、對等 VPC 內執行，或使用 AWS Direct Connect 等其他方式執行，則建立自我管理節點群組會繼續運作，因為它需要透過 EKS 介面端點存取 API 伺服器。

> **Note**
>
> VPC 端點會按小時計費，並根據用量計費。如需定價的詳細資訊，請參閱 AWS PrivateLink 定價

> **Warning**
>
> 不支援完全私有叢集`eu-south-1`。

設定其他 AWS 服務的私有存取權

為了讓工作者節點能夠私下存取 AWS 服務，eksctl 會為下列服務建立 VPC 端點：

- ECR 的界面端點 (`ecr.api`和 `ecr.dkr`) 可提取容器映像 (AWS CNI 外掛程式等)
- S3 用來提取實際映像層的閘道端點
- `aws-cloud-provider` 整合所需的 EC2 介面端點
- STS 的界面端點，可支援服務帳戶 (IRSA) 的 Fargate 和 IAM 角色
- 如果啟用 CloudWatch 記錄，則為 CloudWatch 記錄的介面端點 (`logs`)

這些 VPC 端點對於功能私有叢集至關重要，因此 eksctl 不支援設定或停用它們。不過，叢集可能需要其他 AWS 服務的私有存取權 （ 例如，Cluster Autoscaler 所需的 Autoscaling)。這些服務可在 中指定`privateCluster.additionalEndpointServices`，指示 eksctl 為每個服務建立 VPC 端點。

例如，若要允許私有存取 Autoscaling 和 CloudWatch 記錄：

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

中支援的端點`additionalEndpointServices`為 `autoscaling`、 `cloudformation`和 `logs`。

略過端點建立

如果已使用必要的 AWS 端點建立 VPC 並連結至 EKS 文件中所述的子網路， eksctl可以提供skipEndpointCreation如下的選項來略過建立：

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

此設定無法與 搭配使用additionalEndpointServices。它會略過所有端點建立。此外，只有在正確設定端點<#子網路拓撲時，才建議使用此設定。如果子網路 ID 正確，則會使用字首地址設定vpce路由，建立所有必要的 EKS 端點並將其連結至提供的 VPC。 eksctl 不會變更任何這些資源。

節點群組

全私有叢集僅支援私有節點群組 （受管和自我管理），因為叢集的 VPC 是在沒有任何公有子網路的情況下建立的。privateNetworking 欄位 (nodeGroup**[].privateNetworking** 和 managedNodeGroup[) 必須明確設定。在完全私有叢集中保持privateNetworking未設定是錯誤。

```
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
```

叢集端點存取

全私有叢集不支援在叢集建立clusterEndpointAccess期間修改。設定 clusterEndpoints.publicAccess或 時發生錯誤clusterEndpoints.privateAccess，因為完全私有叢集只能擁有私有存取權，而且允許修改這些欄位可能會中斷叢集。

使用者提供的 VPC 和子網路

eksctl 支援使用預先存在的 VPC 和子網路建立完全私有的叢集。只能指定私有子網路，在 下指定子網路時發生錯誤vpc.subnets.public。

eksctl 在提供的 VPC 中建立 VPC 端點，並修改所提供子網路的路由表。每個子網路都應有一個與其相關聯的明確路由表，因為 eksctl 不會修改主路由表。

```
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
```

管理完全私有的叢集

若要讓所有命令在建立叢集後運作，eksctl 將需要 EKS API 伺服器端點的私有存取權，以及傳出網際網路存取權 （適用於 `EKS:DescribeCluster`)。如果 eksctl 具有傳出網際網路存取，則支援不需要存取 API 伺服器的命令。

強制刪除完全私有的叢集

透過 eksctl 刪除完全私有叢集時可能會發生錯誤，因為 eksctl 不會自動存取叢集的所有資源。 `--force`存在以解決此問題：它會強制刪除叢集，並在發生錯誤時繼續。

限制

目前實作的限制是 eksctl 一開始會建立同時啟用公有和私有端點存取的叢集，並在所有操作完成後停用公有端點存取。這是必要的，因為 eksctl 需要存取 Kubernetes API 伺服器，以允許自我管理節點加入叢集，並支援 GitOps 和 Fargate。這些操作完成後，eksctl 會將叢集端點存取切換為僅限私有。此額外更新確實表示完全私有叢集的建立需要比標準叢集更長的時間。未來，eksctl 可能會切換至已啟用 VPC 的 Lambda 函數，以執行這些 API 操作。

透過 HTTP 代理伺服器傳出存取

eksctl 可以透過設定的 HTTP(S) 代理伺服器與 AWS APIs 通訊，但您需要確保正確設定代理排除清單。

一般而言，您需要設定適當的`no_proxy`環境變數，包括值 ，以確保叢集的 VPC 端點請求不會透過代理路由`.eks.amazonaws.com`。

如果您的代理伺服器執行「SSL 攔截」，而且您使用服務帳戶 (IRSA) 的 IAM 角色，則需要確保明確略過網域 的 SSL Man-in-the-Middle`oidc.<region>.amazonaws.com`。否則會導致 eksctl 取得不

正確的 OIDC 提供者根憑證指紋，而 AWS VPC CNI 外掛程式將無法啟動，因為無法取得 IAM 憑證，導致您的叢集無法運作。

詳細資訊

- EKS 私有叢集

附加元件

本主題說明如何使用 eksctl 管理 Amazon EKS 叢集的 Amazon EKS 附加元件。EKS 附加元件是一項功能，可讓您透過 EKS API 啟用和管理 Kubernetes 操作軟體，簡化安裝、設定和更新叢集附加元件的程序。

> **Warning**
>
> 從 eksctl v0.184.0（以及 metrics-server 的 v0.201.0) 開始，eksctl 會將預設附加元件 (vpc-cni、coredns、kube-proxy、metric-server) 安裝為 EKS 附加元件，而非自我管理附加元件。這表示您應該使用 `eksctl update addon`而非 `eksctl utils update-*` 命令。

當您想要使用 Cilium 和 Calico 等替代 CNI 外掛程式時，您可以建立不含任何預設聯網附加元件的叢集。

EKS 附加元件現在支援透過 EKS Pod Identity Associations 接收 IAM 許可，允許它們與叢集外部的 AWS 服務連線

建立附加元件

Eksctl 提供更多管理叢集附加元件的彈性：

在組態檔案中，您可以指定您想要的附加元件，以及（如果需要）要連接到這些附加元件的角色或政策：

```
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: '*'
```

您最多可以指定其中一個 `attachPolicy`、 `attachPolicyARNs`和 `serviceAccountRoleARN`。

如果未指定這些項目，則會使用已連接所有建議政策的角色來建立附加元件。

> **Note**
>
> 若要將政策連接至附加元件，您的叢集必須OIDC已啟用。如果未啟用，我們會忽略任何附加的政策。

然後，您可以在叢集建立程序期間建立這些附加元件：

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

或者，在建立叢集後，使用組態檔案或 CLI 旗標明確建立附加元件：

```
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**
>
> 使用 `--namespace-config`旗標將附加元件部署到自訂命名空間，而不是預設命名空間。

在建立附加元件期間，如果叢集上已存在自我管理版本的附加元件，您可以透過組態檔案設定 `resolveConflicts`選項來選擇解決潛在`configMap`衝突的方式，例如

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

對於附加元件建立， `resolveConflicts` 欄位支援三個不同的值：

- none - EKS 不會變更值。建立可能會失敗。
- `overwrite` - EKS 會將任何組態變更覆寫回 EKS 預設值。
- `preserve` - EKS 不會變更值。建立可能會失敗。（類似 none，但與[preserve更新附加元件](#)不同）。

列出已啟用的附加元件

您可以執行下列動作，查看叢集中已啟用哪些附加元件：

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

或

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

設定附加元件的版本

設定附加元件的版本是選用的。如果version欄位保持空白， eksctl會解析附加元件的預設版本。如需哪些版本是特定附加元件預設版本的詳細資訊，請參閱 EKS 的 AWS 文件。請注意，預設版本不一定是可用的最新版本。

附加元件版本可以設定為 latest。或者，您可以使用指定的 EKS 建置標籤來設定版本，例如 v1.7.5-eksbuild.1或 v1.7.5-eksbuild.2。它也可以設定為附加元件的發行版本，例如 v1.7.5或 1.7.5，並且會為您探索和設定eksbuild尾碼標籤。

請參閱以下章節，了解如何探索可用的附加元件及其版本。

探索附加元件

您可以執行下列動作，探索可在叢集上安裝哪些附加元件：

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

這將探索叢集的 kubernetes 版本並篩選該版本。或者，如果您想要查看特定 kubernetes 版本可用的附加元件，您可以執行：

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

您也可以透過篩選附加元件的 type、 owner和/或 來探索附加元件publisher。例如，若要查看特定擁有者的附加元件和您可以執行的類型：

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

types、 owners和 publishers旗標是選用的，可以一起或個別指定來篩選結果。

探索附加元件的組態結構描述

探索附加元件和版本後，您可以透過擷取其 JSON 組態結構描述來檢視自訂選項。

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

這會傳回此附加元件可用的各種選項的 JSON 結構描述。

使用組態值

ConfigurationValues 可以在建立或更新附加元件期間在組態檔案中提供。僅支援 JSON 和 YAML 格式。

例如，

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

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

> **Note**
>
> 請記住，修改附加元件組態值時，會出現組態衝突。

```
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`.
```

此外，get 命令現在也會ConfigurationValues擷取附加元件的 。例如

```
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
```

使用自訂命名空間

在建立附加元件期間，可以在組態檔案中提供自訂命名空間。建立附加元件後，命名空間就無法更新。

使用組態檔案

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

使用 CLI 旗標

或者，您可以使用 `--namespace-config`旗標指定自訂命名空間：

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

get 命令也會擷取附加元件的命名空間值

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

更新附加元件

您可以將附加元件更新為較新版本，並透過執行下列動作來變更附加的政策：

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

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

> **Note**
>
> 建立附加元件後，命名空間組態就無法更新。--namespace-config 旗標僅在附加元件建立期間可用。

與附加元件建立類似，更新附加元件時，您可以完全控制先前套用至該附加元件的 的設定變更configMap。具體而言，您可以保留或覆寫它們。此選用功能可透過相同的組態檔案欄位 取得resolveConflicts，例如

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

對於附加元件更新， resolveConflicts 欄位接受三個不同的值：

- none - EKS 不會變更值。更新可能會失敗。
- overwrite - EKS 會將任何組態變更覆寫回 EKS 預設值。
- preserve - EKS 會保留 值。如果您選擇此選項，建議您先在非生產叢集上測試任何欄位和值變更，再更新生產叢集上的附加元件。

刪除附加元件

您可以執行下列動作來刪除 附加元件：

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

這將刪除附加元件和與其相關聯的任何 IAM 角色。

當您刪除叢集時，也會刪除與附加元件相關聯的所有 IAM 角色。

預設聯網附加元件的叢集建立彈性

建立叢集時，EKS 會自動將 VPC CNI、CoreDNS 和 kube-proxy 安裝為自我管理附加元件。若要停用此行為以使用 Cilium 和 Calico 等其他 CNI 外掛程式，eksctl 現在支援建立不含任何預設聯網附加元件的叢集。若要建立這類叢集，請設定 `addonsConfig.disableDefaultAddons`，如 所示：

```
addonsConfig:
  disableDefaultAddons: true
```

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

若要建立僅包含 CoreDNS 和 kube-proxy 而非 VPC CNI 的叢集，請在 中明確指定附加元件，`addons`並設定 `addonsConfig.disableDefaultAddons`，如 所示：

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

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

作為此變更的一部分，如果 `addonsConfig.disableDefaultAddons` 未明確設定為 true，eksctl 現在會在叢集建立期間將預設附加元件安裝為 EKS 附加元件，而非自我管理附加元件。因此，`eksctl utils update-*`命令無法再用於更新使用 eksctl v0.184.0 及更高版本建立之叢集的附加元件：

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

相反地，`eksctl update addon` 應該立即使用。

若要進一步了解，請參閱 Amazon EKS 引入聯網附加元件的叢集建立彈性。

啟用 Amazon EMR 的存取

為了允許 EMR 在 Kubernetes API 上執行操作，需要授予其 SLR 所需的 RBAC 許可。eksctl 提供命令來建立 EMR 所需的 RBAC 資源，並更新 aws-auth ConfigMap 以使用 EMR 的 SLR 繫結角色。

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

EKS Fargate 支援

AWS Fargate 是 Amazon ECS 的受管運算引擎，可執行容器。在 Fargate 中，您不需要管理伺服器或叢集。

Amazon EKS 現在可以在 AWS Fargate 上啟動 Pod。如此一來，您就不必擔心如何佈建或管理 Pod 的基礎設施，也能更輕鬆地在 AWS 上建置和執行高效能的 Kubernetes 應用程式。

使用 Fargate 支援建立叢集

您可以使用下列方式新增具有 Fargate 支援的叢集：

```
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
```

此命令將已建立叢集和 Fargate 設定檔。此設定檔包含 AWS 在 Fargate 中執行個體化 Pod 所需的特定資訊。這些時間為：

- Pod 執行角色，以定義執行 Pod 所需的許可， 以及執行 Pod 的聯網位置 （子網路）。這可讓相同的聯網和安全性許可套用至多個 Fargate Pod，並更輕鬆地將叢集上的現有 Pod 遷移至 Fargate。
- 選取器以定義哪些 Pod 應在 Fargate 上執行。這是由 `namespace`和 組成`labels`。

未指定設定檔時，但使用`--fargate`預設 Fargate 設定檔啟用 Fargate 支援。此設定檔以 `default`和 `kube-system` 命名空間為目標，因此這些命名空間中的 Pod 將在 Fargate 上執行。

您可以使用下列命令來檢查已建立的 Fargate 設定檔：

```
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
```

若要進一步了解選取器，請參閱設計 Fargate 設定檔。

使用組態檔案建立具有 Fargate 支援的叢集

下列組態檔案會宣告 EKS 叢集，其中包含由一個 EC2 `m5.large`執行個體和兩個 Fargate 設定檔組成的節點群組。`default` 和 `kube-system` 命名空間中定義的所有 Pod 都會在 Fargate 上執行。`dev` 命名空間中也有標籤的所有 Pod `dev=passed`也會在 Fargate 上執行。任何其他 Pod 將在 的節點上排程`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
```

設計 Fargate 設定檔

每個選取器項目最多有兩個元件，命名空間和索引鍵/值對清單。建立選取器項目只需要 命名空間元件。所有規則（命名空間、索引鍵值對）必須套用至 Pod，以符合選取器項目。Pod 只需要符合要在設定檔上執行的一個選取器項目。符合選取器欄位中所有條件的任何 Pod 都會排程在 Fargate 上執行。任何不符合白名單 命名空間，但使用者手動設定 排程器的 Pod： 提交的 Fargate-scheduler 會卡在待定狀態，因為它們未獲授權可在 Fargate 上執行。

設定檔必須符合下列要求：

- 每個設定檔都必須選取一個選取器
- 每個選取器都必須包含命名空間；標籤為選用

範例：在 Fargate 中排程工作負載

若要針對上述範例在 Fargate 上排程 Pod，您可以建立名為 的命名空間，dev並在該處部署工作負載：

```
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
```

從最後一個`kubectl get pods`命令的輸出中，我們可以看到 Pod `nginx` 已部署在名為 的節點中`fargate-ip-192-168-183-140.ap-northeast-1.compute.internal`。

管理 Fargate 設定檔

若要在 Fargate 上部署 Kubernetes 工作負載，EKS 需要 Fargate 設定檔。如上述範例所示建立叢集時， 會透過建立預設設定檔`eksctl`來處理此問題。假設已有 叢集，也可以使用 `eksctl create fargateprofile`命令建立 Fargate 設定檔：

> **Note**
>
> 只有在 EKS 平台版本 `eks.5` 或更新版本上執行的叢集才支援此操作。

> **Note**
>
> 如果現有 是在 0.11.0 `eksctl`之前使用 版本建立的，您將需要在建立 Fargate 設定檔`eksctl upgrade cluster`之前執行 。

```
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"
```

您也可以指定要建立的 Fargate 設定檔名稱。此名稱開頭不得為字首 `eks-`。

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

使用此命令搭配 CLI 旗標 eksctl 只能建立具有簡單選擇器的單一 Fargate 設定檔。對於更複雜的選擇器，例如使用更多命名空間，eksctl 支援使用組態檔案：

```
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
```

若要查看叢集中現有的 Fargate 設定檔：

```
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
```

若要以 yaml 格式查看它們：

```
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
```

或 json 格式：

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

Fargate 設定檔的設計不可變。若要變更某些項目，請使用所需的變更建立新的 Fargate 設定檔，並使用 `eksctl delete fargateprofile`命令刪除舊的設定檔，如下列範例所示：

```
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"
}
```

請注意，設定檔刪除程序最多可能需要幾分鐘的時間。未指定`--wait`旗標時， `eksctl` 會樂觀地預期刪除設定檔，並在傳送 AWS API 請求後立即傳回 。若要`eksctl`等待設定檔成功刪除，請使用 ，`--wait`如上述範例所示。

深入閱讀

- AWS Fargate
- Amazon EKS 現在可以在 AWS Fargate 上啟動 Pod

叢集升級

`eksctl` 受管叢集可以透過 3 個簡單步驟進行升級：

1. 使用 的升級控制平面版本 `eksctl upgrade cluster`
2. 升級節點群組
3. 更新預設聯網附加元件 （ 如需詳細資訊，請參閱 the section called “預設附加元件更新”)：

仔細檢閱叢集升級相關資源：

- 《Amazon EKS 使用者指南》中的將現有叢集更新為新的 Kubernetes 版本
- 《EKS 最佳實務指南》中的叢集升級最佳實務

> **Note**
>
> 舊 `eksctl update cluster`將被取代。請改用 `eksctl upgrade cluster`。

更新控制平面版本

必須一次針對一個次要版本進行控制平面版本升級。

若要將控制平面升級至下一個可用的版本執行：

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

此命令不會立即套用任何變更，您需要使用 重新執行它--approve，才能套用變更。

可以使用 CLI 旗標指定叢集升級的目標版本：

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

或 搭配 組態檔案

```
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**
>
> --version 和 metadata.version引數允許的唯一值是目前版本的叢集或更高版本。不支援升級多個 Kubernetes 版本。

預設附加元件更新

本主題說明如何更新 EKS 叢集中包含的預設預先安裝附加元件。

> **Warning**
>
> eksctl 現在會將預設附加元件安裝為 EKS 附加元件，而非自我管理的附加元件。進一步了解其對預設聯網附加元件的叢集建立彈性的影響。
> 若要更新附加元件，`eksctl utils update-<addon>`無法用於使用 eksctl v0.184.0 及更高版本建立的叢集。本指南僅適用於在此變更之前建立的叢集。

每個 EKS 叢集包含 3 個預設附加元件：

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

更新預先安裝的附加元件

對於透過 `eksctl create addons`或在叢集建立時手動建立的官方 EKS 附加元件，管理它們的方式是透過 `eksctl create/get/update/delete addon`。在這種情況下，請參閱有關 EKS 附加元件的文件。

更新每個命令的程序都不同，因此您需要執行 3 個不同的命令。下列所有命令都接受 `--config-file`。根據預設，這些命令都會在計劃模式下執行，如果您對提議的變更感到滿意，請使用 重新執行`--approve`。

若要更新 `kube-proxy`，請執行：

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

若要更新 `aws-node`，請執行：

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

若要更新 `coredns`，請執行：

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

升級後，請務必執行 `kubectl get pods -n kube-system` 並檢查所有附加元件 Pod 是否都處於就緒狀態，您應該會看到如下內容：

```
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
```

在 EKS 叢集中支援區域轉移

EKS 現在支援 Amazon Application Recovery Controller (ARC) 區域轉移和區域自動轉移，可增強多可用區域叢集環境的彈性。使用 AWS Zonal Shift，客戶可以從受損的可用區域轉移叢集內流量，確保新的 Kubernetes Pod 和節點僅在運作狀態良好的可用區域中啟動。

建立已啟用區域轉移的叢集

```
# 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
```

在現有叢集上啟用區域轉移

若要啟用或停用現有叢集上的區域轉移，請執行

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

或沒有組態檔案：

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

詳細資訊

- EKS 區域轉移

Karpenter 支援

`eksctl` 支援將 Karpenter 新增至新建立的叢集。它會建立 Karpenter 入門區段中概述的所有必要先決條件，包括使用 Helm 安裝 Karpenter 本身。我們目前支援安裝版本 `0.28.0+`。如需更多詳細資訊，請參閱 Karpenter 相容性一節。

下列叢集組態概述典型的 Karpenter 安裝：

```
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
```

版本是 Karpenter 的版本，因為它可以在其 Helm 儲存庫中找到。也可以設定下列選項：

```
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
```

必須定義 OIDC 才能安裝 Karpenter。

成功安裝 Karpenter 後，請新增 NodePool(s) 和 NodeClass(s)，以允許 Karpenter 開始將節點新增至叢集。

NodePool 的 nodeClassRef區段必須符合 的名稱EC2NodeClass。例如：

```
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
```

請注意，您必須instanceProfile為 lauch 節點指定其中一個 role或 。如果您選擇使用 建立的設定檔instanceProfile名稱，eksctl請遵循 模式：eksctl-KarpenterNodeInstanceProfile-<cluster-name>。

自動安全群組標記

eksctl 當 Karpenter 啟用 (karpenter.version指定） 且karpenter.sh/discovery標籤存在於 中karpenter.sh/discovery時， 會自動使用 標記叢集的共用節點安全群組metadata.tags。這可啟用 AWS Load Balancer 控制器相容性。

請注意，使用 karpenter 0.32.0+，Provisioners 已被 NodePool 取代。

叢集組態結構描述

> **Note**
>
> 結構描述的位置目前正在遷移。

您可以使用 yaml 檔案來建立叢集。檢視結構描述參考。

例如：

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

此檔案的結構描述參考可在 GitHub 上取得。

如需使用 檔案的詳細資訊，請參閱 the section called “建立和管理叢集”。

節點群組

本章包含如何使用 Eksctl 建立和設定 Nodegroups 的相關資訊。節點群組是連接到 EKS 叢集的 EC2 執行個體群組。

主題：

- the section called “Spot 執行個體”
  - 使用受管節點群組建立和管理具有 Spot 執行個體的 EKS 叢集
  - 使用 MixedInstancesPolicy 設定未受管節點群組的 Spot 執行個體
  - 使用 Kubernetes 標籤區分 Spot `node-lifecycle` 和隨需執行個體
- the section called “Auto Scaling”
  - 使用允許使用叢集自動擴展器的 IAM 角色建立叢集或節點群組，以啟用 Kubernetes 叢集節點的自動擴展
  - 設定節點群組定義，以包含叢集自動擴展器擴展節點群組所需的標籤和註釋
  - 如果工作負載具有區域特定的需求，例如區域特定的儲存或親和性規則，請為每個可用區域建立個別節點群組
- the section called “EKS 受管節點群組”
  - 佈建和管理 Amazon EKS Kubernetes 叢集的 EC2 執行個體 （節點）
  - 輕鬆將錯誤修正、安全性修補程式和更新節點套用至最新的 Kubernetes 版本
- the section called “EKS 混合節點”
  - 使用與 AWS 雲端中使用的相同 AWS EKS 叢集、功能和工具，在客戶受管基礎設施上啟用執行內部部署和邊緣應用程式
  - 使用 AWS Site-to-Site VPN 或 AWS Direct Connect 等選項，設定聯網將內部部署網路連線至 AWS VPC
  - 設定遠端節點的登入資料，以使用 AWS Systems Manager (SSM) 或 AWS IAM Roles Anywhere 驗證 EKS 叢集
- the section called “節點修復組態”
  - 啟用 EKS 受管節點群組的節點修復，以自動監控和取代或重新啟動運作狀態不佳的工作者節點
- the section called “ARM 支援”
  - 使用 ARM 型 Graviton 執行個體建立 EKS 叢集，以提高效能和成本效益
- the section called “污點”

  - 將污點套用至 Kubernetes 叢集中的特定節點群組
  - 根據污點鍵、值和效果控制 Pod 的排程和移出
- the section called “啟動範本支援”
  - 使用提供的 EC2 啟動範本啟動受管節點群組
  - 升級受管節點群組以使用不同版本的啟動範本
  - 了解搭配受管節點群組使用自訂 AMIs 和啟動範本時的限制和考量
- the section called “使用節點群組”
  - 啟用節點群組中 EC2 執行個體的 SSH 存取
  - 向上或向下擴展節點群組中的節點數量
- the section called “自訂子網路”
  - 使用新的子網路擴充現有的 VPC，並將 Nodegroup 新增至該子網路
- the section called “節點引導”
  - 了解 AmazonLinux2023 中引入的新節點初始化程序（節點）
  - 了解 eksctl 為自我管理和 EKS 受管節點套用的預設 NodeConfig 設定
  - 透過提供具有自訂 NodeConfig 的 overrideBootstrapCommand 來自訂節點引導程序
- the section called “未受管節點群組”
  - 在 EKS 叢集中建立或更新未受管節點群組
  - 更新預設 Kubernetes 附加元件，例如 kube-proxy、aws-node 和 CoreDNS
- the section called “GPU 支援”
  - Eksctl 支援為節點群組選取 GPU 執行個體類型，以便在 EKS 叢集上使用 GPU 加速工作負載。
  - 選取已啟用 GPU 的執行個體類型時，Eksctl 會自動安裝 NVIDIA Kubernetes 裝置外掛程式，以利在叢集中使用 GPU 資源。
  - 使用者可以停用自動外掛程式安裝，並使用提供的命令手動安裝特定版本的 NVIDIA Kubernetes 裝置外掛程式。
- the section called “執行個體選取器”
  - 根據 vCPUs、記憶體、GPUs 和 CPU 架構等資源條件，自動產生適合的 EC2 執行個體類型清單
  - 使用符合指定執行個體選取器條件的執行個體類型建立叢集和節點群組
  - 在建立節點群組之前，執行試轉以檢查和修改執行個體選取器相符的執行個體類型
- the section called “其他磁碟區映射”
  - 在 EKS 叢集中設定受管節點群組的其他磁碟區映射
  - 自訂其他磁碟區的磁碟區屬性，例如大小、類型、加密、IOPS 和輸送量

  - 將現有的 EBS 快照附加為節點群組的額外磁碟區
- the section called “Windows 工作者節點”
  - 將 Windows 節點群組新增至現有的 Linux Kubernetes 叢集，以啟用執行中的 Windows 工作負載
  - 根據 `kubernetes.io/os`和 `kubernetes.io/arch`標籤，使用節點選擇器在適當的作業系統 (Windows 或 Linux) 上排程工作負載
- the section called “自訂 AMI 支援”
  - 使用 `--node-ami`旗標指定節點群組的自訂 AMI、查詢 AWS 以取得最新的 EKS 最佳化 AMI，或使用 AWS Systems Manager 參數存放區尋找 AMI。
  - 設定 `--node-ami-family`旗標以指定節點群組 AMI 的作業系統系列，例如 AmazonLinux2、Ubuntu2204 或 WindowsServer2022CoreContainer。
  - 對於 Windows 節點群組，請指定自訂 AMI，並透過 提供 PowerShell 引導指令碼`overrideBootstrapCommand`。
- the section called “自訂 DNS”
  - 覆寫用於內部和外部 DNS 查詢的 DNS 伺服器 IP 地址

使用節點群組

建立節點群組

除了與叢集一起建立的初始節點群組之外，您還可以新增一或多個節點群組。

若要建立其他節點群組，請使用：

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

> **Note**
>
> `--version` 受管節點群組不支援 flag。它一律會從控制平面繼承版本。

根據預設，新的未受管節點群組會從控制平面 (`--version=auto`) 繼承版本，但您可以指定不同的版本，您也可以使用 `--version=latest`強制使用最新版本。

此外，您可以使用與 相同的組態檔案`eksctl create cluster`：

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

從組態檔案建立節點群組

節點群組也可以透過叢集定義或組態檔案建立。根據下列範例組態檔案和名為 的現有叢集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
```

節點群組 ng-1-workers和 ng-2-builders可以使用此命令建立：

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

負載平衡

如果您已經準備好將現有的傳統負載平衡器或/和目標群組連接到節點群組，您可以在組態檔案中指定這些群組。建立節點群組時，傳統負載平衡器或/和目標群組會自動與 ASG 建立關聯。這僅支援透過 nodeGroups 欄位定義的自我管理節點群組。

```
# 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
```

組態檔案中的節點群組選擇

若要僅對組態檔案中指定的節點群組子集執行 `create`或 `delete`操作，有兩個接受 glob 清單的 CLI 旗標，`0`以及 `1`，例如：

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

使用上述範例組態檔案，即可使用下列命令建立工作者節點群組以外的所有工作者節點群組：

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

或者，可以刪除具有下列項目的建置器節點群組：

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

在這種情況下，我們也需要提供 `--approve`命令，才能實際刪除節點群組。

包含和排除規則

- 如果未指定 `--include`或 `--exclude` ，則會包含所有項目
- 如果只指定 `--include` ，則只會包含符合這些 glob 的節點群組
- 如果只指定 `--exclude` ，則會包含不符合這些 glob 的所有節點群組
- 如果指定兩者，則`--exclude`規則優先於 `--include`（即兩個群組中符合規則的節點群組將被排除）

列出節點群組

若要列出節點群組或所有節點群組的詳細資訊，請使用：

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

若要以 YAML 或 JSON 格式列出一或多個節點群組，其會輸出比預設日誌資料表更多的資訊，請使用：

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

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

節點群組不可變性

根據設計，節點群組是不可變的。這表示如果您需要變更某些項目 （擴展除外），例如 AMI 或節點群組的執行個體類型，則需要建立具有所需變更的新節點群組、移動負載並刪除舊節點群組。請參閱刪除和耗盡節點群組一節。

擴展節點群組

Nodegroup 擴展是一個可能需要幾分鐘的程序。未指定`--wait`旗標時，`eksctl`樂觀地預期節點群組會擴展，並在傳送 AWS API 請求後立即傳回。若要讓 `eksctl` 等待節點可用，請新增如下所示的`--wait`旗標。

> **Note**
>
> 向下/向內擴展節點群組 （即減少節點數量） 可能會導致錯誤，因為我們完全依賴 ASG 的變更。這表示移除/終止的節點 （未明確耗盡）。這可能是未來需要改進的領域。

直接呼叫更新受管節點群組組態的 EKS API 來擴展受管節點群組。

擴展單一節點群組

您可以使用 `eksctl scale nodegroup`命令來擴展節點群組：

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

例如，若要將 `ng-a345f4e1`中的節點群組擴展`cluster-1`到 5 個節點，請執行：

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

也可以使用傳遞至 的組態檔案來擴展節點群組，`--config-file`並指定應使用 擴展的節點群組名稱`--name`。Eksctl 將搜尋組態檔案，並探索該節點群組及其擴展組態值。

如果所需的節點數量NOT在目前最小和目前最大數量節點的範圍內，則會顯示一個特定錯誤。這些值也可以`--nodes-max`分別與旗標 `--nodes-min`和 一起傳遞。

擴展多個節點群組

Eksctl 可以探索和擴展使用 傳遞的組態檔案中找到的所有節點群組`--config-file`。

與擴展單一節點群組類似，相同的一組驗證會套用至每個節點群組。例如，所需的節點數量必須在節點數量下限和上限的範圍內。

刪除和耗盡節點群組

若要刪除節點群組，請執行：

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

包含和排除規則也可以與此命令搭配使用。

> **Note**
>
> 這將在刪除執行個體之前耗盡該節點群組中的所有 Pod。

若要在耗盡過程中略過移出規則，請執行：

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

所有節點都會進行封鎖，且所有 Pod 都會在刪除時從節點群組移出，但如果您需要耗盡節點群組而不將其刪除，請執行：

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

若要取消協調節點群組，請執行：

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

若要忽略 PodDisruptionBudget 設定等移出規則，請執行：

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

若要加速耗盡程序，您可以`--parallel <value>`為要平行耗盡的節點數量指定 。

其他功能

您也可以為節點群組啟用 SSH、ASG 存取和其他功能，例如：

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

更新標籤

中沒有特定命令eksctl可更新節點群組的標籤，但可以使用 輕鬆達成kubectl，例如：

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

SSH 存取

您可以在節點群組組態publicKeyPath中設定其中一個 publicKeyName和 publicKey，以啟用節點群組的 SSH 存取。或者，您可以使用 AWS Systems Manager (SSM) 在節點上 SSH，方法是使用設定節點群組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
```

未受管節點群組

在 中eksctl，設定--managed=false或使用 nodeGroups 欄位會建立未受管節點群組。請記住，未受管節點群組不會出現在 EKS 主控台中，一般規則只會知道 EKS 受管節點群組。

只有在執行 之後，您才應該升級節點群組eksctl upgrade cluster。（請參閱升級叢集。）

如果您的簡單叢集只有初始節點群組（即使用 建立eksctl create cluster），則程序非常簡單：

1. 取得舊節點群組的名稱：

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

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

2. 建立新的節點群組：

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

3. 刪除舊節點群組：

```
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.
> ```

更新多個節點群組

如果您有多個節點群組，您有責任追蹤每個節點群組的設定方式。您可以使用組態檔案來執行此操作，但如果您尚未使用，則需要檢查您的叢集，以了解每個節點群組的設定方式。

一般而言，您會希望：

- 檢閱您擁有哪些節點群組，以及哪些節點群組可以刪除或必須為新版本取代
- 記下每個節點群組的組態，下次考慮使用組態檔案來簡化升級

使用組態檔案更新

如果您使用的是組態檔案，則需要執行下列動作。

編輯組態檔案以新增節點群組，並移除舊的節點群組。如果您只想升級節點群組並保留相同的組態，您可以變更節點群組名稱，例如附加-v2到名稱。

若要建立組態檔案中定義的所有新節點群組，請執行：

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

一旦有新的節點群組，您就可以刪除舊的節點群組：

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

> **Note**
>
> 第一次執行處於計劃模式，如果您對提議的變更感到滿意，請使用 重新執行--approve。

更新預設附加元件

您可能需要更新叢集上安裝的聯網附加元件。如需詳細資訊，請參閱the section called “預設附加元件更新”。

EKS 受管節點群組

Amazon EKS 受管節點群組是一項功能，可自動化 Amazon EKS Kubernetes 叢集節點 (EC2 執行個體）的佈建和生命週期管理。客戶可以為其叢集佈建最佳化的節點群組，EKS 會將其節點保持在最新的 Kubernetes 和主機作業系統版本。

EKS 受管節點群組是自動擴展群組和相關聯的 EC2 執行個體，由 Amazon EKS 叢集的 AWS 管理。每個節點群組使用 Amazon EKS 最佳化的 Amazon Linux 2 AMI。Amazon EKS 可讓您輕鬆將錯誤修正和安全性修補程式套用至節點，並將它們更新至最新的 Kubernetes 版本。每個節點群組都會為您的叢集啟動自動擴展群組，可以跨越多個 AWS VPC 可用區域和子網路，以提供高可用性。

受管節點群組的新啟動範本支援 ???

> **Note**
>
> 「未受管節點群組」一詞已用於參考 eksctl 從一開始就支援的節點群組（透過 `nodeGroups` 欄位表示）。`ClusterConfig` 檔案會繼續使用 `nodeGroups` 欄位來定義未受管節點群組，並使用 `managedNodeGroups` 欄位定義受管節點群組。

建立受管節點群組

```
$ eksctl create nodegroup
```

新叢集

若要使用受管節點群組建立新的叢集，請執行

```
eksctl create cluster
```

若要建立多個受管節點群組並對組態有更多控制權，可以使用組態檔案。

> **Note**
>
> 受管節點群組與未受管節點群組沒有完整的功能同位。

```
# 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
```

您可以在這裡找到另一個用於建立受管節點群組的組態檔案範例。

您可以擁有同時具有受管和未受管節點群組的叢集。未受管節點群組不會在 AWS EKS 主控台中顯示，但`eksctl get nodegroup`會列出這兩種類型的節點群組。

```
# 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
```

全新支援自訂 AMI、安全群組、instancePrefix、instanceName、ebsOptimizedvolumeType、volumeName、volumeEncryp preBootstrapCommandsoverrideBootstrapCommand和 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'
```

如果您請求的執行個體類型僅適用於一個區域（且 eksctl 組態需要兩個 的規格），請務必將可用區域新增至節點群組請求：

```
# 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
```

對於僅在一個區域中可用的 Hpc6 系列等執行個體類型而言，這可能是如此。

現有叢集

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

秘訣：如果您使用 `ClusterConfig` 檔案來描述整個叢集，請在 `managedNodeGroups` 欄位中描述新的受管節點群組並執行：

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

升級受管節點群組

您可以隨時將節點群組更新為您正在使用的 AMI 類型的最新 EKS 最佳化 AMI 發行版本。

如果您的節點群組與叢集是相同的 Kubernetes 版本，您可以更新到您正在使用的 AMI 類型之 Kubernetes 版本的最新 AMI 發行版本。如果您的節點群組是叢集 Kubernetes 版本之前的 Kubernetes 版本，您可以將節點群組更新為符合節點群組 Kubernetes 版本的最新 AMI 發行版本，或更新為符合叢集 Kubernetes 版本的最新 AMI 發行版本。您無法將節點群組復原至較早的 Kubernetes 版本。

若要將受管節點群組升級至最新的 AMI 發行版本：

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

您可以使用下列方式，將節點群組升級至指定 Kubernetes 版本的最新 AMI 版本：

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

若要升級至特定 AMI 發行版本，而非最新版本，請傳遞 `--release-version`：

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

> **Note**
>
> 如果使用自訂 AMIs 部署受管節點，必須遵循下列工作流程，才能部署自訂 AMI 的新版本。

- 節點群組的初始部署必須使用啟動範本完成，例如

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

- 建立新的自訂 AMI 版本（使用 AWS EKS 主控台）。
- 使用新的 AMI ID 建立新的啟動範本版本（使用 AWS EKS 主控台）。
- 將節點升級至新版本的啟動範本，例如

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

處理節點的平行升級

多個受管節點可以同時升級。若要設定平行升級，請在建立節點群組時定義節點群組updateConfig的 。您可以在此處updateConfig找到範例。

若要避免因一次升級多個節點而導致工作負載停機，您可以在 的 maxUnavailable 欄位中指定此節點，以限制升級期間無法使用的節點數量updateConfig。或者，使用 maxUnavailablePercentage，將無法使用節點的最大數量定義為節點總數的百分比。

請注意， maxUnavailable不能高於 maxSize。此外， maxUnavailable和 maxUnavailablePercentage無法同時使用。

此功能僅適用於受管節點。

更新受管節點群組

eksctl 允許更新受管節點群組的 UpdateConfig 區段。本節定義兩個欄位： MaxUnavailable和 MaxUnavailablePercentage。您的節點群組在更新期間不會受到影響，因此不應預期停機。

命令update nodegroup應與使用 --config-file旗標的組態檔案搭配使用。節點群組應該包含 nodeGroup.updateConfig區段。如需詳細資訊，請參閱此處。

Nodegroup 運作狀態問題

EKS 受管節點群組會自動檢查節點群組和節點的組態是否有運作狀態問題，並透過 EKS API 和主控台進行報告。若要檢視節點群組的運作狀態問題：

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

管理標籤

EKS 受管節點群組支援連接套用至節點群組中 Kubernetes 節點的標籤。這是在叢集或節點群組建立期間，透過 eksctl 中的 labels 欄位指定。

若要在節點群組上設定新標籤或更新現有標籤：

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

若要從節點群組取消設定或移除標籤：

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

若要檢視節點群組上設定的所有標籤：

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

擴展受管節點群組

eksctl scale nodegroup 也支援受管節點群組。擴展受管或未受管節點群組的語法相同。

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

詳細資訊

- EKS 受管節點群組

節點引導

AmazonLinux2023

AL2023 推出新的節點初始化程序 nodeadm，使用 YAML 組態結構描述，捨棄指令碼的使用/etc/eks/bootstrap.sh。

> **Note**
>
> 使用 Kubernetes 1.30 版及更新版本時，Amazon Linux 2023 是預設作業系統。

AL2023 的預設設定

對於以自訂 AMIs 為基礎的自我管理節點和 EKS 受管節點， 會eksctl建立預設、最小，NodeConfig並自動將其插入節點群組的啟動範本使用者資料，即

```
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

--//--
```

對於以原生 AMIs 為基礎的 EKS 受管節點，預設值會由 EKS MNG NodeConfig新增至幕後，並直接附加到 EC2 的使用者資料。因此，在此案例中， eksctl 不需要將其包含在啟動範本中。

設定引導程序

若要設定 的進階屬性NodeConfig，或僅覆寫預設值，eksctl 可讓您NodeConfig透過 nodeGroup.overrideBootstrapCommand或 managedNodeGroup.overrideBootstrapCommand 指定自訂，例如

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

此自訂組態將由 eksctl 附加至使用者資料，並由 nodeadm與預設組態合併。在此處閱讀有關 合併多個組態物件nodeadm功能的詳細資訊。

受管節點群組的啟動範本支援

eksctl 支援使用提供的 EC2 啟動範本啟動受管節點群組。這可為節點群組啟用多個自訂選項，包括提供自訂 AMIs 和安全群組，以及傳遞使用者資料以進行節點引導。

使用提供的啟動範本建立受管節點群組

```
# 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
```

升級受管節點群組以使用不同的啟動範本版本

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

> **Note**
>
> 如果啟動範本使用自訂 AMI，則新版本也應使用自訂 AMI，否則升級操作將會失敗

如果啟動範本未使用自訂 AMI，也可以指定要升級至 的 Kubernetes 版本：

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

自訂 AMI 和啟動範本支援的注意事項

- 提供啟動範本時，不支援下列欄位：`instanceType`、`ami`、`ssh.allow`、`ssh.sourceSecurityGroupIds`、`securityGroups`、`ins` `maxPodsPerNode` `preBootstrapCommands``overrideBootstrapCommand`和 `disableIMDSv1`。

- 使用自訂 AMI (ami) 時，overrideBootstrapCommand也必須設定 來執行引導。
- overrideBootstrapCommand 只能在使用自訂 AMI 時設定。
- 提供啟動範本時，節點群組組態中指定的標籤僅適用於 EKS 節點群組資源，而不會傳播到 EC2 執行個體。

自訂子網路

您可以延伸具有新子網路的現有 VPC，並將 Nodegroup 新增至該子網路。

為什麼

如果叢集用完預先設定的 IPs，可以使用新的 CIDR 調整現有 VPC 的大小，以為其新增新的子網路。若要了解如何執行此操作，請閱讀 AWS Extending VPCs上的本指南。

TL；DR

前往 VPC 的組態，並按一下 Actions->Edit CIDRs並新增範圍。例如：

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

現在您需要新增子網路。視新的私有或公有子網路而定，您必須分別從私有或公有子網路複製路由資訊。

建立子網路後，請新增路由，並從 VPC 中的另一個子網路複製 NAT 閘道 ID 或網際網路閘道。如果是公有子網路，請小心啟用自動 IP 指派。Actions->Modify auto-assign IP settings->Enable auto-assign public IPv4 address。

也別忘了根據公有或私有子網路組態複製現有子網路的 TAGS。這很重要，否則子網路不會成為叢集的一部分，子網路中的執行個體將無法加入。

完成後，複製新子網路的 ID。視需要重複此動作。

方法

若要在建立的子網路 （ 子網路 ） 中建立節點群組，請執行下列命令：

```
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
```

或者，使用以下組態：

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

使用如下的組態：

```
# 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
```

等待節點群組建立，且新執行個體應具有子網路的新 IP 範圍 (IP)。

刪除叢集

由於新的新增透過在 CloudFormation 堆疊之外新增相依性來修改現有的 VPC，因此 CloudFormation 無法再移除叢集。

刪除叢集之前，請手動移除所有建立的額外子網路，然後呼叫 以繼續`eksctl`：

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

自訂 DNS

有兩種方式可以覆寫用於所有內部和外部 DNS 查詢的 DNS 伺服器 IP 地址。這相當於 的 `--cluster-dns`旗標`kubelet`。

首先， 透過 clusterDNS 欄位。Config 檔案接受名為 string的欄位，clusterDNS其中包含要使用的 DNS 伺服器的 IP 地址。這將傳遞至 kubelet，而 則會透過 /etc/resolv.conf 檔案將其傳遞至 Pod。如需詳細資訊，請參閱組態檔案的結構描述。

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

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

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

請注意，此組態只接受一個 IP 地址。若要指定多個地址，請使用 kubeletExtraConfig 參數：

```
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"]
```

污點

若要將污點套用至特定節點群組，請使用如下所示taints的組態區段：

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

您可以在此處找到完整的範例。

執行個體選取器

eksctl 支援為受管和自我管理節點群組指定多個執行個體類型，但使用超過 270 種 EC2 執行個體類型，使用者必須花時間找出最適合其節點群組的執行個體類型。使用 Spot 執行個體時更加困難，因為您需要選擇一組可與 Cluster Autoscaler 搭配使用的執行個體。

eksctl 現在與 EC2 執行個體選取器整合，可根據資源條件產生執行個體類型的清單來解決此問題：vCPUs、記憶體、GPUs 數量和 CPU 架構。當執行個體選取器條件通過時，eksctl 會建立節點群組，並將執行個體類型設定為符合所提供條件的執行個體類型。

建立叢集和節點群組

若要使用與傳遞給 eksctl 的執行個體選取器資源條件相符的執行個體類型，建立具有單一節點群組的叢集，請執行

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

這會建立叢集和 `instanceTypes` 欄位設定為 的受管節點群組 `[c5.large, c5a.large, c5ad.large, c5d.large, t2.medium, t3.medium, t3a.medium]`（傳回的一組執行個體類型可能會變更）。

對於未受管節點群組，將設定 `instancesDistribution.instanceTypes` 欄位：

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

執行個體選取器條件也可以在 ClusterConfig 中指定：

```
# 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
```

`eksctl create cluster` 和 支援下列執行個體選取器 CLI 選項`eksctl create nodegroup`：

`--instance-selector-vcpus`、`--instance-selector-memory`、`--instance-selector-gpus` 與 `instance-selector-cpu-architecture`

您可以在此處找到範例檔案。

乾執行

試轉功能可讓您檢查和變更執行個體選取器相符的執行個體，然後再繼續建立節點群組。

```
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
```

然後，產生的 ClusterConfig 可以傳遞給 `eksctl create cluster`：

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

代表 CLI 選項`instanceSelector`的欄位也會新增至 ClusterConfig 檔案，以用於可見性和文件用途。省略 `--dry-run` 時，會忽略此欄位，並使用 `instanceTypes` 欄位，否則 的任何變更`instanceTypes`都會被 eksctl 覆寫。

使用 傳遞 ClusterConfig 檔案時`--dry-run`，eksctl 會在擴展每個節點群組的執行個體選取器資源條件之後，輸出包含相同節點群組集的 ClusterConfig 檔案。

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

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

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

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

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

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

```
managedNodeGroups:
- amiFamily: AmazonLinux2
  # ...
  instanceSelector:
    cpuArchitecture: x86_64
    memory: 2GiB
    vCPUs: 2
  instanceTypes:
  - t3.small
  - t3a.small
nodeGroups:
- amiFamily: AmazonLinux2
  # ...
  instanceSelector:
    memory: "4"
    vCPUs: 2
  instanceType: mixed
  instancesDistribution:
    capacityRebalance: false
    instanceTypes:
    - c5.large
    - c5a.large
    - c5ad.large
    - c5d.large
    - t2.medium
    - t3.medium
    - t3a.medium
# ...
```

Spot 執行個體

受管節點群組

`eksctl` 支援使用 EKS 受管節點群組的 Spot 工作者節點，這項功能可讓具有容錯應用程式的 EKS 客戶輕鬆地為其 EKS 叢集佈建和管理 EC2 Spot 執行個體。EKS 受管節點群組將根據 Spot 最佳實務設定並啟動 Spot 執行個體的 EC2 Autoscaling 群組，並在執行個體被 AWS 中斷之前自動耗盡 Spot 工作者節點。使用此功能無需額外收費，客戶只需支付使用 AWS 資源的費用，例如 EC2 Spot 執行個體和 EBS 磁碟區。

若要使用 Spot 執行個體建立具有受管節點群組的叢集，請傳遞 `--spot`旗標和選用的執行個體類型清單：

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

若要在現有叢集上使用 Spot 執行個體建立受管節點群組：

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

若要透過組態檔案使用受管節點群組建立 Spot 執行個體：

```
# 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**
>
> 未受管節點群組不支援 spot和 instanceTypes 欄位，而是使用 instancesDistribution 欄位來設定 Spot 執行個體。請參閱以下內容

詳細資訊

- EKS Spot 節點群組
- EKS 受管節點群組容量類型

未受管節點群組

`eksctl` 透過 Auto Scaling 群組的 MixedInstancesPolicy 支援 Spot 執行個體。

以下是使用 50% Spot 執行個體和 50% 隨需執行個體的節點群組範例：

```
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
```

請注意，使用 `nodeGroups.X.instanceType` 欄位時不應設定 `instancesDistribution` 欄位。

此範例使用 GPU 執行個體：

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

此範例使用容量最佳化 Spot 配置策略：

```
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"
```

此範例使用capacity-optimized-prioritized配置策略：

```
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"
```

使用 capacity-optimized-prioritized 配置策略，然後在啟動範本覆寫清單中設定執行個體類型的順序，從最高到最低的優先順序（清單先到最後）。Amazon EC2 Auto Scaling 會盡力遵守執行個體類型的優先順序，但會先針對容量進行最佳化。對於必須將中斷可能性降至最低的工作負載而言，這是一個不錯的選項，但對於某些執行個體類型的偏好也很重要。如需詳細資訊，請參閱 ASG 購買選項。

請注意，使用 spotInstancePools 欄位時不應設定 spotAllocationStrategy 欄位。如果spotAllocationStrategy未指定 ，EC2 預設會使用 lowest-price策略。

以下是一個最少的範例：

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

若要區分 Spot 或隨需執行個體之間的節點，您可以使用具有 值`node-lifecycle`的 kubernetes 標籤`spot`，或`on-demand`取決於其類型。

instancesDistribution 中的參數

如需詳細資訊，請參閱叢集組態結構描述。

GPU 支援

Eksctl 支援為節點群組選取 GPU 執行個體類型。只要將相容的執行個體類型提供給 create 命令，或透過 組態檔案即可。

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

> **Note**
>
> 不再需要訂閱市集 AMI 以在 EKS 上支援 GPU。

AMI 解析程式 (`auto` 和 `auto-ssm`) 會看到您想要使用 GPU 執行個體類型，他們會選取正確的 EKS 最佳化加速 AMI。

Eksctl 會偵測是否已選取已啟用 GPU 執行個體類型的 AMI，並會自動安裝 NVIDIA Kubernetes 裝置外掛程式。

> **Note**
>
> Windows 和 Ubuntu AMIs不會隨附已安裝的 GPU 驅動程式，因此執行 GPU 加速工作負載將無法立即運作。

若要停用自動外掛程式安裝，並手動安裝特定版本，請使用 `--install-nvidia-plugin=false`搭配 create 命令。例如：

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

以及，對於 0.15.0 及更高版本，

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

或者，對於較舊的版本，

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

如果叢集只包含 Bottlerocket 節點群組，則會略過 NVIDIA Kubernetes 裝置外掛程式的安裝，因為 Bottlerocket 已處理裝置外掛程式的執行。如果您在叢集的組態中使用不同的 AMI 系列，您可能需要使用污點和容錯，以防止裝置外掛程式在 Bottlerocket 節點上執行。

ARM 支援

本主題說明如何使用 ARM 節點群組建立叢集，以及如何將 ARM 節點群組新增至現有叢集。

EKS 支援 64 位元 ARM 架構及其 Graviton 處理器。若要建立叢集，請選取其中一個 Graviton 型執行個體類型 (a1、t4g、m6g、m7g、m6gdc6g、c7g、c6gdr6g、、r7g、r6gdm8g、、r8g、、c8g) 並執行：

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

或使用組態檔案：

```
---
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
```

受管節點群組也支援 ARM：

```
---
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
```

AMI 解析程式 `auto-ssmauto`和 會根據 ARM 執行個體類型推斷正確的 AMI。只有 AmazonLinux2023、AmazonLinux2 和 Bottlerocket 系列具有 ARM 的 EKS 最佳化 AMIs。

> **Note**
>
> 1.15 版及更高版本的叢集支援 ARM。

Auto Scaling

啟用 Auto Scaling

您可以使用 IAM 角色建立叢集 （或現有叢集中的節點群組），以允許使用叢集自動擴展器：

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

此旗標也會設定 `k8s.io/cluster-autoscaler/enabled` 和 `k8s.io/cluster-autoscaler/<clusterName>` 標籤，因此節點群組探索應該可以運作。

叢集執行後，您將需要安裝 Cluster Autoscaler 本身。

您也應該將下列項目新增至受管或未受管節點群組定義 (Cluster Autoscaler)，以新增擴展節點群組所需的標籤：

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

從 0 向上擴展

如果您想要能夠從 0 擴展節點群組，並在節點群組上定義了標籤和/或污點，則需要將這些標籤傳播為 Auto Scaling 群組 (ASGs) 上的標籤。

其中一種方法是在節點群組定義的 `tags` 欄位中設定 ASG 標籤。例如，指定具有下列標籤和污點的節點群組：

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

您需要新增下列 ASG 標籤：

```
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"
```

對於受管和非受管節點群組，這可以透過將 `propagateASGTags`設定為 來自動完成`true`，這會將標籤和污點作為標籤新增至 Auto Scaling 群組：

```
nodeGroups:
```

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

區域感知 Auto Scaling

如果您的工作負載是區域特定的，您將需要為每個區域建立單獨的節點群組。這是因為 `cluster-autoscaler` 假設群組中的所有節點完全相同。因此，例如，如果縱向擴展事件是由需要區域特定 PVC （例如 EBS 磁碟區） 的 Pod 觸發，則新節點可能會排程在錯誤的 AZ 中，且 Pod 將無法啟動。

如果您的環境符合下列條件，則每個可用區域不需要個別的節點群組：

- 沒有區域特定的儲存需求。
- 主機以外的拓撲不需要 podAffinity。
- 區域標籤上不需要 nodeAffinity。
- 區域標籤上沒有 nodeSelector。

(在此處和此處閱讀更多資訊。)

如果您符合上述所有要求 （可能也符合其他要求），您應該使用跨越多個 AZs 的單一節點群組來確保安全。否則，您會想要建立個別的單一可用區域節點群組：

之前：

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

之後：

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

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

自訂 AMI 支援

設定節點 AMI ID

`--node-ami` 旗標會啟用多種進階使用案例，例如使用自訂 AMI 或即時查詢 AWS，以判斷要使用的 AMI。旗標可用於非 GPU 和 GPU 映像。

旗標可以取得影像的 AMI 影像 ID 以明確使用。它也可以使用下列「特殊」關鍵字：

關鍵字	說明
auto	指出應該透過查詢 AWS EC2 找到用於節點的 AMI。這與自動解析程式相關。
自動ssm	指出應該透過查詢 AWS SSM 參數存放區找到用於節點的 AMI。

> **Note**
>
> 目前，EKS 受管節點群組在使用自訂 AMI 時僅支援下列 AMIs 系列：`AmazonLinux2023`、`AmazonLinux2`、`Bottlerocket`、`Ubuntu2004`、`UbuntuPro2004``Ubuntu2204`和 `Ubuntu2404`

`--node-ami` 將 設定為 ID 字串時， `eksctl`會假設已請求自訂 AMI。對於 AmazonLinux2 和 Ubuntu 節點，EKS 受管和自我管理，這表示`overrideBootstrapCommand`這是必要的。對於 AmazonLinux2023，您可以在使用自訂 AMI `overrideBootstrapCommand`時省略 ，因為 eksctl 會自動產生所需的最小 NodeConfig。如需節點引導的詳細資訊，請參閱 GitHub 上的節點引導。

CLI 旗標範例：

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

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

Config 檔案範例：

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

`--node-ami` 旗標也可以與 搭配使用`eksctl create nodegroup`。

設定節點 AMI 系列

`--node-ami-family` 可以採用下列關鍵字：

關鍵字	說明
AmazonLinux2	指出應使用以 Amazon Linux 2 為基礎的 EKS AMI 映像（預設）。
AmazonLinux2023	表示應使用以 Amazon Linux 2023 為基礎的 EKS AMI 映像。
Ubuntu2004	指出應使用以 Ubuntu 20.04 LTS (Focal) 為基礎的 EKS AMI 映像（支援 EKS ⇐ 1.29)。
UbuntuPro2004	表示應使用以 Ubuntu Pro 20.04 LTS (Focal) 為基礎的 EKS AMI 映像（適用於 EKS >= 1.27、⇐ 1.29)。

關鍵字	說明
Ubuntu2204	表示應使用以 Ubuntu 22.04 LTS (Jammy) 為基礎的 EKS AMI 映像（適用於 EKS >= 1.29)。
UbuntuPro2204	表示應使用以 Ubuntu Pro 22.04 LTS (Jammy) 為基礎的 EKS AMI 映像（適用於 EKS >= 1.29)。
Ubuntu2404	表示應使用以 Ubuntu 24.04 LTS (Noble) 為基礎的 EKS AMI 映像（適用於 EKS >= 1.31)。
UbuntuPro2404	表示應使用以 Ubuntu Pro 24.04 LTS (Noble) 為基礎的 EKS AMI 映像（適用於 EKS >= 1.31)。
Bottlerocket	表示應使用以 Bottlerocket 為基礎的 EKS AMI 映像。
WindowsServer2019FullContainer	指出應使用以 Windows Server 2019 完整容器為基礎的 EKS AMI 映像。
WindowsServer2019CoreContainer	表示應使用以 Windows Server 2019 Core Container 為基礎的 EKS AMI 映像。
WindowsServer2022FullContainer	指出應使用以 Windows Server 2022 Full Container 為基礎的 EKS AMI 映像。
WindowsServer2022CoreContainer	指出應使用以 Windows Server 2022 Core Container 為基礎的 EKS AMI 映像。

CLI 旗標範例：

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

Config 檔案範例：

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

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

--node-ami-family 旗標也可以與 搭配使用eksctl create nodegroup。 eksctl 需要每當使用自訂 AMI 時，透過組態檔案或透過 --node-ami-family CLI 旗標明確設定 AMI 系列。

> **Note**
>
> 目前，EKS 受管節點群組在使用自訂 AMI 時僅支援下列 AMIs 系列：AmazonLinux2023、AmazonLinux2、Bottlerocket、Ubuntu2004、UbuntuPro2004Ubuntu2204和 Ubuntu2404

Windows 自訂 AMI 支援

只有自我管理的 Windows 節點群組可以指定自訂 AMI。 amiFamily 應該設定為有效的 Windows AMI 系列。

引導指令碼將可使用下列 PowerShell 變數：

```
$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
```

Config 檔案範例：

```
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
```

Bottlerocket 自訂 AMI 支援

對於 Bottlerocket 節點，`overrideBootstrapCommand`不支援 。相反地，若要指定自己的引導容器，應該使用 `bottlerocket` 欄位做為組態檔案的一部分。例如

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

Windows 工作者節點

從 1.14 版開始，Amazon EKS 支援允許執行 Windows 容器的 Windows 節點。除了具有 Windows 節點之外，叢集中的 Linux 節點還需要執行 CoreDNS，因為 Microsoft 尚不支援主機網路模式。因此，Windows EKS 叢集將混合 Windows 節點和至少一個 Linux 節點。Linux 節點對叢集的運作至關重要，因此，對於生產級叢集，建議至少有兩個適用於 HA 的 `t2.large` Linux 節點。

> **Note**
>
> 您不再需要在 Linux 工作者節點上安裝 VPC 資源控制器，即可在 2021 年 10 月 22 日之後建立的 EKS 叢集中執行 Windows 工作負載。您可以透過 ConfigMap 設定在 EKS 控制平面上啟用 Windows IP 地址管理 （詳細資訊請參閱 link：eks/latest/userguide/windows-support.html)。eksctl 會在建立 Windows 節點群組時自動修補 ConfigMap 以啟用 Windows IP 地址管理。

使用 Windows 支援建立新的叢集

組態檔案語法允許在單一命令中使用 Windows 支援建立功能完整的叢集：

```
# 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
```

若要使用 Windows 未受管節點群組建立新叢集而不使用組態檔案，請發出下列命令：

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

將 Windows 支援新增至現有的 Linux 叢集

若要在具有 Linux 節點 (AmazonLinux2 AMI 系列） 的現有叢集上啟用執行 Windows 工作負載，您需要新增 Windows 節點群組。

新增對 Windows 受管節點群組的支援 (--managed=true 或省略旗標）。

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

為了確保工作負載排程在正確的作業系統上，它們必須具有nodeSelector目標鎖定其必須執行的作業系統：

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

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

如果您使用的叢集早於 1.19 kubernetes.io/os和 kubernetes.io/arch標籤，則需要beta.kubernetes.io/arch分別以 beta.kubernetes.io/os和 取代。

詳細資訊

- EKS Windows 支援

其他磁碟區映射

作為額外的組態選項，在處理磁碟區映射時，可以在建立節點群組時設定額外的映射。

若要這樣做，請設定 欄位additionalVolumes，如下所示：

```
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'
```

如需選取 volumeNames 的詳細資訊，請參閱裝置命名文件。若要進一步了解 EBS 磁碟區、執行個體磁碟區限制或區塊型設備映射，請造訪此頁面。

EKS 混合節點

簡介

AWS EKS 推出混合節點，這項新功能可讓您在客戶管理的基礎設施上執行內部部署和邊緣應用程式，其具有您在 AWS 雲端中使用的相同 AWS EKS 叢集、功能和工具。AWS EKS Hybird Nodes 為內部部署環境提供 AWS 受管 Kubernetes 體驗，讓客戶能夠簡化和標準化您在內部部署、邊緣和雲端環境中執行應用程式的方式。在 EKS 混合節點中閱讀更多資訊。

為了促進對此功能的支援，eksctl 推出了名為 的新最上層欄位`remoteNetworkConfig`。任何與混合節點相關的組態都應透過此欄位設定，做為組態檔案的一部分；沒有 CLI 旗標對應項目。此外，在啟動時，任何遠端網路組態只能在叢集建立期間設定，之後無法更新。這表示您將無法更新現有的叢集以使用混合節點。

組態檔案的 remoteNetworkConfig區段可讓您在將遠端節點加入 EKS 叢集時設定兩個核心區域：聯網和登入資料。

聯網

EKS 混合節點可彈性使用您偏好的方法來將內部部署網路連線至 AWS VPC (AWS VPC)。有多種文件記錄的選項可供使用，包括 AWS Site-to-Site VPN 和 AWS Direct Connect，您可以選擇最適合您的使用案例的方法。在您可以選擇的大多數方法中，VPC 會連接到虛擬私有閘道 (VGW) 或傳輸閘道 (TGW)。如果您依賴 eksctl 為您建立 VPC，eksctl 也會在您的 VPC 範圍內設定任何與聯網相關的先決條件，以促進 EKS 控制平面與遠端節點之間的通訊，即

- 輸入/輸出 SG 規則
- 私有子網路路由表中的路由
- 連接至指定 TGW 或 VGW 的 VPC 閘道

範例組態檔案：

```
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"]
```

如果您選擇的連線方法不涉及使用 TGW 或 VGW，則不得依賴 eksctl 為您建立 VPC，而是提供預先存在的 VPC。相關注意事項是，如果您使用的是預先存在的 VPC，eksctl 不會對其進行任何修改，並確保所有聯網要求都由您負責。

> **Note**
>
> eksctl 不會在您的 AWS VPC 外部設定任何聯網基礎設施 （即從 VGW/TGW 到遠端網路的任何基礎設施）

憑證

EKS 混合節點使用 AWS IAM Authenticator 和由 AWS SSM 或 AWS IAM Roles Anywhere 佈建的臨時 IAM 登入資料來驗證 EKS 叢集。與自我管理節點群組類似，如果未另外提供，eksctl 會為您建立混合節點 IAM 角色，以供遠端節點擔任。此外，使用 IAM Roles Anywhere 做為登入資料提供者時，eksctl 會根據指定的憑證授權單位套件 (`iam.caBundleCert`) 設定設定檔和信任錨點，例如

```
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
```

在將遠端節點加入叢集、`NodeConfig`設定 和建立啟用 （ 如果使用 SSM) 的過程中`nodeadm`，稍後需要 eksctl 建立的混合節點角色 ARN。若要擷取它，請使用：

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

同樣地，如果使用 IAM Roles Anywhere，您可以擷取信任錨點的 ARN 和 eksctl 建立的任何設定檔，分別以 `RemoteNodesRoleARN`、`RemoteNodesTrustAnchorARN`或 取代來修改先前的命令`RemoteNodesAnywhereProfileARN`。

如果您有預先存在的 IAM Roles Anywhere 組態，或者您使用 SSM，您可以透過 為混合節點提供 IAM 角色`remoteNetworkConfig.iam.roleARN`。請記住，在此案例中，eksctl 不會為您建立信任錨點和任何設定檔，例如

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

若要將角色映射至 Kubernetes 身分，並授權遠端節點加入 EKS 叢集，eksctl 會使用混合節點 IAM 角色建立存取項目，做為主體 ARN 和類型 `HYBRID_LINUX`。即

```
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"
        ]
    }
]
```

附加元件支援

容器聯網界面 (CNI)：AWS VPC CNI 無法與混合節點搭配使用。Cilium 和 Calico 的核心功能支援與混合節點搭配使用。您可以使用您選擇的工具來管理 CNI，例如 Helm。如需詳細資訊，請參閱設定混合節點的 CNI。

> **Note**
>
> 如果您在叢集中為自我管理或 EKS 受管節點群組安裝 VPC CNI，則必須使用 `v1.19.0-eksbuild.1`或更新版本，因為這包括附加元件協助程式集的 udpate，以排除在混合節點上安裝 VPC CNI。

進一步參考

- EKS 混合節點 UserDocs
- 啟動公告

支援 EKS 受管節點群組的節點修復組態

EKS 受管節點群組支援節點修復，其中受管節點的運作狀態受到監控，而運作狀態不佳的工作者節點會被取代或重新啟動以回應。eksctl 現在提供對節點修復行為進行精細控制的完整組態選項。

基本節點修復組態

使用 CLI 旗標

若要使用基本節點修復建立具有受管節點群組的叢集，請傳遞 --enable-node-repair旗標：

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

若要在現有叢集上建立具有節點修復的受管節點群組：

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

使用組態檔案

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

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

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

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

增強型節點修復組態

閾值組態

您可以設定節點修復動作何時停止使用百分比或以計數為基礎的閾值。注意：您無法同時使用百分比和計數閾值。

閾值的 CLI 旗標

```
# Percentage-based threshold - repair stops when 20% of nodes are unhealthy
```

```
eksctl create cluster --enable-node-repair \
  --node-repair-max-unhealthy-percentage=20

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

閾值的組態檔案

```
managedNodeGroups:
- name: threshold-ng
  nodeRepairConfig:
    enabled: true
    # Stop repair actions when 20% of nodes are unhealthy
    maxUnhealthyNodeThresholdPercentage: 20
    # Alternative: stop repair actions when 3 nodes are unhealthy
    # maxUnhealthyNodeThresholdCount: 3
    # Note: Cannot use both percentage and count thresholds simultaneously
```

平行修復限制

控制可同時或平行修復的節點數量上限。這樣一來，您就能更精細地控制節點取代速度。注意：您無法同時使用百分比和計數限制。

平行限制的 CLI 旗標

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

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

平行限制的組態檔案

```
managedNodeGroups:
- name: parallel-ng
  nodeRepairConfig:
    enabled: true
    # Repair at most 15% of unhealthy nodes in parallel
```

```
    maxParallelNodesRepairedPercentage: 15
    # Alternative: repair at most 2 unhealthy nodes in parallel
    # maxParallelNodesRepairedCount: 2
    # Note: Cannot use both percentage and count limits simultaneously
```

自訂修復覆寫

指定特定修復動作的精細覆寫。透過這些覆寫，可控制認為節點符合修復資格之前的修復動作與修復延遲時間。如果您使用此值，則必須指定每個覆寫的所有值。

```
managedNodeGroups:
- name: custom-repair-ng
  instanceType: g4dn.xlarge  # GPU instances
  nodeRepairConfig:
    enabled: true
    maxUnhealthyNodeThresholdPercentage: 25
    maxParallelNodesRepairedCount: 1
    nodeRepairConfigOverrides:
      # Handle GPU-related failures with immediate termination
      - nodeMonitoringCondition: "AcceleratedInstanceNotReady"
        nodeUnhealthyReason: "NvidiaXID13Error"
        minRepairWaitTimeMins: 10
        repairAction: "Terminate"
      # Handle network issues with restart after waiting
      - nodeMonitoringCondition: "NetworkNotReady"
        nodeUnhealthyReason: "InterfaceNotUp"
        minRepairWaitTimeMins: 20
        repairAction: "Restart"
```

完成組態範例

如需所有組態選項的完整範例，請參閱 example/44-node-repair.yaml。

範例 1：具有百分比閾值的基本修復

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

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

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

範例 2：關鍵工作負載的保守修復

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

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

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

範例 3：具有專門修復的 GPU 工作負載

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

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

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

CLI 參考

節點修復旗標

旗標	說明	範例
`--enable-node-repair`	啟用自動節點修復	`--enable-node-repair`
`--node-repair-max-unhealthy-percentage`	修復前運作狀態不佳節點的百分比上限	`--node-repair-max-unhealthy-percentage=20`
`--node-repair-max-unhealthy-count`	修復前運作狀態不佳節點的最大計數	`--node-repair-max-unhealthy-count=5`
`--node-repair-max-parallel-percentage`	要平行修復的節點百分比上限	`--node-repair-max-parallel-percentage=15`
`--node-repair-max-parallel-count`	要平行修復的節點計數上限	`--node-repair-max-parallel-count=2`

注意：由於節點修復組態覆寫的複雜性，僅透過 YAML 組態檔案支援。

組態參考

nodeRepairConfig

欄位	Type	說明	Constraints	範例
`enabled`	boolean	啟用/停用節點修復	-	`true`
`maxUnhealthyNodeThresholdPercentage`	integer	運作狀態不佳節點的百分比閾值，超過此閾值後節點自動修復動作將停止	無法與 搭配使用 `maxUnhealthyNodeThresholdCount`	`20`
`maxUnhealthyNodeThresholdCount`	integer	不良節點的計數閾值，超過此閾值，節點自動修復動作將停止	無法與 搭配使用 `maxUnhealthyNodeThresholdPercentage`	`5`
`maxParallelNodesRepairedPercentage`	integer	可同時或平行修復的不良節點最大百分比	無法與 搭配使用 `maxParallelNodesRepairedCount`	`15`
`maxParallelNodesRepairedCount`	integer	可同時或平行修復的不良節點數量上限	無法與 搭配使用 `maxParallelNodesRepairedPercentage`	`2`

欄位	Type	說明	Constraints	範例
nodeRepairConfigOverrides	陣列	控制修復動作和延遲時間之特定修復動作的精細覆寫	必須為每個覆寫指定所有值	請參閱上述範例

nodeRepairConfigOverrides

欄位	Type	說明	有效值
nodeMonitoringCondition	string	節點監控代理程式回報此覆寫套用至的運作狀態不佳	"AcceleratedInstanceNotReady"，"NetworkNotReady"
nodeUnhealthyReason	string	節點監控代理程式回報此覆寫適用於的原因	"NvidiaXID13Error"，"InterfaceNotUp"
minRepairWaitTimeMins	integer	在嘗試修復具有指定條件和原因的節點之前，以分鐘為單位的最短等待時間	任何正整數
repairAction	string	符合所有指定條件時要對節點採取的修復動作	"Terminate"，"Restart"，"NoAction"

詳細資訊

- EKS 受管節點群組節點運作狀態

聯網

本章包含 Eksctl 如何為 EKS 叢集建立虛擬私有雲端 (VPC) 網路的相關資訊。

主題：

- the section called “VPC 組態”
  - 修改 VPC CIDR 範圍並設定 IPv6 定址
  - 使用現有的 VPC
  - 自訂新 EKS 叢集的 VPC、子網路、安全群組和 NAT 閘道
- the section called “子網路設定”
  - 為初始節點群組使用私有子網路，將其與公有網際網路隔離
  - 透過列出每個可用區域的多個子網路，並在節點群組組態中指定子網路，來自訂子網路拓撲
  - 將節點群組限制在 VPC 組態中的特定具名子網路
  - 將私有子網路用於節點群組時，請將 `privateNetworking`設定為 `true`
  - 使用 VPC 規格中的 `public`和 `private`組態提供完整的子網路規格
  - 節點群組組態中`availabilityZones`只能提供其中一個 `subnets`或
- the section called “叢集存取”
  - 管理 EKS 叢集中 Kubernetes API 伺服器端點的公有和私有存取
  - 透過指定允許的 CIDR 範圍，限制對 EKS Kubernetes 公有 API 端點的存取
  - 更新現有叢集的 API 伺服器端點存取組態和公有存取 CIDR 限制
- the section called “控制平面聯網”
  - 更新叢集的 EKS 控制平面所使用的子網路
- the section called “IPv6 支援”
  - 指定使用 EKS 叢集建立 VPC 時要使用的 IP 版本 (IPv4 或 IPv6)

VPC 組態

叢集的專用 VPC

根據預設，`eksctl create cluster`會為叢集建立專用 VPC。這是為了避免因各種原因干擾現有資源，包括安全性，但也因為偵測現有 VPC 中的所有設定具有挑戰性。

- 使用的預設 VPC CIDR `eksctl`為 `192.168.0.0/16`。
  - 它分為 8 個 (/19) 子網路 (3 個私有、3 個公有和 2 個預留 ）。
- 初始節點群組是在公有子網路中建立。
- 除非指定 ，否則會停用 SSH `--allow-ssh` 存取。
- 根據預設，節點群組允許來自連接埠 1025 - 65535 上控制平面安全群組的傳入流量。

> **Note**
>
> 在 `us-east-1` eksctl 中，預設只會建立 2 個公有和 2 個私有子網路。

變更 VPC CIDR

如果您需要設定與其他 VPC 的對等互連，或只是需要更大或更小範圍的 IPs，您可以使用 `--vpc-cidr` 旗標來變更它。如需選擇允許在 AWS VPC 中使用的 CIDR 區塊的指南，請參閱 AWS 文件。

如果您要建立 IPv6 叢集，您可以在 `VPC.IPv6Cidr` cluser 組態檔案中設定 。此設定僅在組態檔案中，而不是在 CLI 旗標中。

如果您擁有 IPv6 IP 地址區塊，您也可以攜帶自己的 IPv6 集區。請參閱將您自己的 IP 地址 (BYOIP) 帶到 Amazon EC2，了解如何匯入您自己的集區。然後使用 cluser config 檔案中`VPC.IPv6Cidr`的 來設定 Eksctl。

使用現有的 VPC：與 kops 共用

您可以使用由 kops 管理的現有 Kubernetes 叢集的 VPC。提供此功能是為了促進遷移和/或叢集對等互連。

如果您先前已建立具有 kops 的叢集，例如使用類似以下的命令：

```
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
```

您可以使用相同的 VPC 子網路在相同的 AZs 中建立 EKS 叢集 （請注意：至少需要 2 AZs/子網路 ）：

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

使用現有的 VPC：其他自訂組態

eksctl 為自訂 VPC 和子網路拓撲提供一些但不完整的彈性。

您可以使用 和 --vpc-private-subnets--vpc-public-subnets旗標提供私有和/或公有子網路，以使用現有的 VPC。由您決定是否確保您使用的子網路正確分類，因為沒有簡單的方法來驗證子網路是否實際為私有或公有，因為組態會有所不同。

鑑於這些旗標， eksctl create cluster 會自動判斷 VPC ID，但不會建立任何路由表或其他資源，例如網際網路/NAT 閘道。不過，它會為初始節點群組和控制平面建立專用安全群組。

您必須確保在不同AZs區域中提供至少 2 個子網路，EKS 會檢查此條件。如果您使用現有的 VPC，EKS 或 Eksctl 不會強制執行或檢查下列要求，而 EKS 會建立叢集。叢集的一些基本函數在沒有這些要求的情況下運作。（例如，標記並非絕對必要，測試已顯示可以在子網路上不設定任何標籤的情況下建立功能叢集，但無法保證這一律會保留，並建議標記。）

標準需求：

- 所有指定的子網路都必須位於相同 IPs 區塊內的相同 VPC 中
- 根據需求提供足夠數量的 IP 地址
- 足夠數量的子網路 （ 下限 2)，視需求而定
- 子網路至少會加上下列標籤：
  - kubernetes.io/cluster/<name> 標籤設定為 shared或 owned
  - kubernetes.io/role/internal-elb 私有子網路1的標籤設定為
  - kubernetes.io/role/elb 公有子網路1的標籤設定為
- 正確設定的網際網路和/或 NAT 閘道
- 路由表具有正確的項目，且網路正常運作
- 新：所有公有子網路都應MapPublicIpOnLaunch啟用 屬性 （即在 AWS 主控台Auto-assign public IPv4 address中）。受管節點群組和 Fargate 不會指派公有 IPv4 地址，必須在子網路上設定 屬性。

EKS 或 Kubernetes 可能會實施其他要求，而且完全由您隨時掌握任何要求和/或建議up-to-date，並視需要/可能實作這些要求。

套用的預設安全群組設定eksctl可能不足以與其他安全群組中的資源共用存取權。如果您想要修改安全群組的輸入/輸出規則，您可能需要使用其他工具來自動化變更，或透過 EC2 主控台執行。

如有疑問，請勿使用自訂 VPC。`eksctl create cluster` 不使用任何`--vpc-*`旗標的 一律會使用全功能專用 VPC 來設定叢集。

範例

使用具有 2 倍私有子網路和 2 倍公有子網路的自訂 VPC 建立叢集：

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

或使用下列同等組態檔案：

```
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
```

使用具有 3 倍私有子網路的自訂 VPC 建立叢集，並讓初始節點群組使用這些子網路：

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

```
  --node-private-networking
```

或使用下列同等組態檔案：

```
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
```

使用自訂 VPC 4x 公有子網路建立叢集：

```
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
```

您可以在儲存庫的 examples 資料夾中找到更多範例：

- 使用現有的 VPC
- 使用自訂 VPC CIDR

自訂共用節點安全群組

eksctl 將建立和管理共用節點安全群組，以允許未受管節點與叢集控制平面和受管節點之間的通訊。

如果您想要改為提供自己的自訂安全群組，您可以覆寫組態檔案中sharedNodeSecurityGroup的欄位：

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

根據預設，在建立叢集時， eksctl會將規則新增至此安全群組，以允許與 EKS 建立的預設叢集安全群組進行通訊。預設叢集安全群組由 EKS 控制平面和受管節點群組使用。

如果您想要自行管理安全群組規則，您可以透過在組態檔案中將 manageSharedNodeSecurityGroupRules設定為 false eksctl來防止 建立規則：

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

NAT 閘道

叢集的 NAT Gateway 可以設定為 `Disable`、 `Single`（預設）或 `HighlyAvailable`。`HighlyAvailable` 選項會在區域的每個可用區域中部署 NAT 閘道，因此如果 AZ 關閉，其他 AZs 中的節點仍然可以與網際網路通訊。

它可以透過 `--vpc-nat-mode` CLI 旗標或在叢集組態檔案中指定，如以下範例所示：

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

請參閱此處的完整範例。

> **Note**
>
> 只有在叢集建立期間才支援指定 NAT Gateway。在叢集升級期間不會碰觸到它。

子網路設定

將私有子網路用於初始節點群組

如果您想要隔離初始節點群組與公有網際網路，您可以使用 `--node-private-networking`旗標。搭配 `--ssh-access`旗標使用時，只能從 VPC 內部存取 SSH 連接埠。

> **Note**
>
> 使用 `--node-private-networking`旗標將導致傳出流量使用其彈性 IP 通過 NAT 閘道。另一方面，如果節點位於公有子網路中，傳出流量將不會通過 NAT 閘道，因此傳出流量具有每個個別節點的 IP。

自訂子網路拓撲

`eksctl` 版本`0.32.0`引進了進一步的子網路拓撲自訂，並能夠：

- 在 VPC 組態中列出每個 AZ 的多個子網路
- 在節點群組組態中指定子網路

在舊版中，自訂子網路必須由可用區域提供，這表示每個可用區域只能列出一個子網路。從識別金鑰可以是任意0.32.0的。

```
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

如果使用 AZ 做為識別金鑰，則可以省略該az值。

如果使用任意字串做為識別金鑰，如上所示，則：

- id 必須設定 (az 和cidr選用)
- az 必須設定 或 (cidr選用)

如果使用者在未指定 CIDR 和 ID 的情況下依 AZ 指定子網路，則該 AZ 中的子網路將從 VPC 中任意選擇，如果存在多個此類子網路。

Note

必須提供完整的子網路規格，即 VPC 規格中宣告的 public和 private組態。

節點群組可以透過 組態限制為具名子網路。在節點群組組態上指定子網路時，請使用 VPC 規格中指定的識別金鑰，而不是子網路 ID。例如：

```
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**
>
> 節點群組組態中availabilityZones只能提供其中一個 subnets或 。

在私有子網路中放置節點群組時，privateNetworking必須在節點群組true上將 設定為 ：

```
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
```

如需完整組態範例，請參閱 eksctl GitHub 儲存庫中的 24-nodegroup-subnets.yaml。

叢集存取

管理對 Kubernetes API 伺服器端點的存取

根據預設，EKS 叢集會公開 Kubernetes API 伺服器，但不會直接從 VPC 子網路 (public=true，private=false) 中公開。來自 VPC 內 API 伺服器的流量必須先退出 VPC 網路（而非 Amazon 的網路），然後重新輸入以到達 API 伺服器。

使用叢集組態檔案建立叢集時，叢集的 Kubernetes API 伺服器端點存取可以設定為公有和私有存取。範例如下：

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

設定 Kubernetes API 端點存取時，有一些額外的注意事項：

1. EKS 不允許叢集啟用私有或公有存取。
2. EKS 確實允許建立僅允許啟用私有存取的組態，但 eksctl 在叢集建立期間不支援它，因為它可防止 eksctl 將工作者節點加入叢集。
3. 根據預設，將叢集更新為只有私有的 Kubernetes API 端點存取表示 `eksctl utils update-kube-proxy` Kubernetes 命令（例如 `kubectl`) 以及 `eksctl delete cluster`、`eksctl utils write-kubeconfig`和 可能需要在叢集 VPC 中執行命令。
   - 這需要對各種 AWS 資源進行一些變更。如需詳細資訊，請參閱叢集 API 伺服器端點。
   - 您可以提供 `vpc.extraCIDRs` 將其他 CIDR 範圍附加到 ControlPlaneSecurityGroup，允許 VPC 外部的子網路到達 kubernetes API 端點。同樣地，您也可以提供 `vpc.extraIPv6CIDRs`來附加 IPv6 CIDR 範圍。

以下是如何使用 `utils`子命令設定 Kubernetes API 端點存取的範例：

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

若要使用 **`ClusterConfig`** 檔案更新設定，請使用：

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

請注意，如果您未傳遞旗標，它會保留目前的值。一旦您對提議的變更感到滿意，請新增 `approve`旗標以對執行中的叢集進行變更。

限制對 EKS Kubernetes Public API 端點的存取

EKS 叢集的預設建立會公開 Kubernetes API 伺服器。

此功能僅適用於公有端點。API 伺服器端點存取組態選項不會變更，而且您仍然可以選擇停用公有端點，讓叢集無法從網際網路存取。（來源：https：//https://github.com/aws/containers-roadmap/issues/108#issuecomment-552766489)

若要在建立叢集時限制存取一組 CIDRs的公有 API 端點，請設定 **`publicAccessCIDRs`** 欄位：

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

若要更新現有叢集的限制，請使用：

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

若要使用 **`ClusterConfig`** 檔案更新限制，請在 中設定新的 CIDRs**`vpc.publicAccessCIDRs`**並執行：

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

> **Important**
>
> 如果設定`publicAccessCIDRs`和建立節點群組，`privateAccess`則應將 設定為 ，`true`或將節點的 IPs新增至`publicAccessCIDRs`清單。

如果節點因為存取受限而無法存取叢集 API 端點，叢集建立將會失敗，`context deadline exceeded`因為節點無法存取公有端點且無法加入叢集。

若要在單一命令中更新叢集的 API 伺服器端點存取和公有存取 CIDRs，請執行：

```
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
```

若要使用組態檔案更新設定：

```
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
```

更新控制平面子網路和安全群組

本文件說明如何在初始建立後修改 EKS 叢集控制平面的聯網組態。這包括更新控制平面子網路和安全群組。

更新控制平面子網路

使用 eksctl 建立叢集時，會建立一組公有和私有子網路，並傳遞至 EKS API。EKS 會在這些子網路中建立 2 到 4 個跨帳戶彈性網路界面 (ENIs)，以啟用 EKS 受管 Kubernetes 控制平面與 VPC 之間的通訊。

若要更新 EKS 控制平面所使用的子網路，請執行：

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

若要使用組態檔案更新設定：

```
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
```

如果沒有 --approve旗標，eksctl 只會記錄提議的變更。一旦您對提議的變更感到滿意，請使用 --approve旗標重新執行 命令。

更新控制平面安全群組

為了管理控制平面和工作者節點之間的流量，EKS 支援傳遞其他安全群組，這些安全群組會套用至 EKS 佈建的跨帳戶網路介面。若要更新 EKS 控制平面的安全群組，請執行：

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

若要使用組態檔案更新設定：

```
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
```

若要更新叢集的控制平面子網路和安全群組，請執行：

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

若要使用組態檔案更新這兩個欄位：

```
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
```

如需完整範例，請參閱 cluster-subnets-sgs.yaml。

如果沒有 --approve旗標，eksctl 只會記錄提議的變更。一旦您對提議的變更感到滿意，請使用 --approve旗標重新執行 命令。

IPv6 支援

定義 IP 系列

當 eksctl建立 vpc 時，您可以定義將使用的 IP 版本。下列選項可供設定：

- IPv4
- IPv6

預設值為 IPv4。

若要定義它，請使用下列範例：

```
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**
>
> 此設定僅在組態檔案中，而不是在 CLI 旗標中。

如果您使用 IPv6，則必須設定下列需求：

- OIDC 已啟用
- 受管附加元件定義如上所示
- 叢集版本必須為 => 1.21
- vpc-cni 附加元件版本必須為 => 1.10.0
- IPv6 叢集不支援自我管理節點群組
- 未擁有的 IPv6 叢集不支援受管節點群組
- `vpc.nat` 和 `serviceIPv4CIDR` 欄位是由 eksctl 為 ipv6 叢集建立，不支援組態選項
- 不支援 AutoAllocateIPv6 搭配 IPv6
- 對於 IPv6 叢集，vpc-cni 的 IAM 角色必須具有與 IPv6 模式相關聯的必要 IAM 政策

也可以使用 IPv6 IP 系列來完成私有聯網。請遵循 EKS 私有叢集下概述的指示。

IAM

本章包含使用 AWS IAM 的相關資訊。

主題：

- the section called “管理 IAM 使用者和角色”
  - 管理 IAM 使用者和角色映射，以控制對 EKS 叢集的存取
  - 透過叢集組態檔案或 CLI 命令設定 IAM 身分映射
- the section called “服務帳戶的 IAM 角色”
  - 管理在 Amazon EKS 上執行且使用其他 AWS 服務之應用程式的精細許可
  - 使用 eksctl 建立和設定 IAM 角色和 Kubernetes 服務帳戶對
  - 啟用 EKS 叢集的 IAM OpenID Connect 提供者，以啟用服務帳戶的 IAM 角色
- the section called “IAM 許可界限”
  - 透過設定許可界限，控制授予 IAM 實體 （ 使用者或角色 ） 的最大許可
- the section called “EKS Pod 身分關聯”
  - 使用建議的 Pod 身分關聯設定 EKS 附加元件的 IAM 許可
  - 讓 Kubernetes 應用程式接收必要的 IAM 許可，以連線至叢集外的 AWS 服務
  - 簡化跨多個 EKS 叢集自動化 IAM 角色和服務帳戶的程序
- the section called “IAM 政策”
  - 管理 EKS 節點群組的 IAM 政策，包括支援各種附加元件政策，例如映像建置器、自動擴展器、外部 DNS、憑證管理員等。
  - 將自訂執行個體角色或內嵌政策連接至節點群組，以取得其他許可。
  - 將 ARN 的特定 AWS 受管政策連接至節點群組，確保包含必要的政策，例如 AmazonEKSWorkerNodePolicy 和 AmazonEKS_CNI_Policy。
- the section called “最低 IAM 政策”
  - 管理 AWS EC2 資源，包括負載平衡器、自動擴展群組和 CloudWatch 監控
  - 建立和管理 AWS CloudFormation 堆疊
  - 管理 Amazon Elastic Kubernetes Service (EKS) 叢集、節點群組和相關資源，例如 IAM 角色和政策

最低 IAM 政策

本文件說明執行 eksctl 主要使用案例所需的最低 IAM 政策。這些是用來執行整合測試的項目。

> **Note**
>
> 請記得將 取代<account_id>為您自己的 。

> **Note**
>
> AWS 管理政策是由 AWS 建立和管理。您無法變更 AWS 受管政策中定義的許可。

AmazonEC2FullAccess (AWS 受管政策)

檢視 AmazonEC2FullAccess 政策定義。

AWSCloudFormationFullAccess (AWS 受管政策)

檢視 AWSCloudFormationFullAccess 政策定義。

EksAllAccess

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        },
        {
            "Action": [
                "ssm:GetParameter",
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:*:123456789012:parameter/aws/*",
                "arn:aws:ssm:*::parameter/aws/*"
            ],
```

```
            "Effect": "Allow"
        },
        {
             "Action": [
               "kms:CreateGrant",
               "kms:DescribeKey"
             ],
             "Resource": "*",
             "Effect": "Allow"
        },
        {
             "Action": [
               "logs:PutRetentionPolicy"
             ],
             "Resource": "*",
             "Effect": "Allow"
        }
    ]
}
```

IamLimitedAccess

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:GetRole",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "iam:UpdateAssumeRolePolicy",
                "iam:AddRoleToInstanceProfile",
                "iam:ListInstanceProfilesForRole",
                "iam:PassRole",
                "iam:DetachRolePolicy",
                "iam:DeleteRolePolicy",
```

```
                "iam:GetRolePolicy",
                "iam:GetOpenIDConnectProvider",
                "iam:CreateOpenIDConnectProvider",
                "iam:DeleteOpenIDConnectProvider",
                "iam:TagOpenIDConnectProvider",
                "iam:ListAttachedRolePolicies",
                "iam:TagRole",
                "iam:UntagRole",
                "iam:GetPolicy",
                "iam:CreatePolicy",
                "iam:DeletePolicy",
                "iam:ListPolicyVersions"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:instance-profile/eksctl-*",
                "arn:aws:iam::123456789012:role/eksctl-*",
                "arn:aws:iam::123456789012:policy/eksctl-*",
                "arn:aws:iam::123456789012:oidc-provider/*",
                "arn:aws:iam::123456789012:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "arn:aws:iam::123456789012:role/eksctl-managed-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetUser"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/*",
                "arn:aws:iam::123456789012:user/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "eks.amazonaws.com",
```

```
                        "eks-nodegroup.amazonaws.com",
                        "eks-fargate.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

IAM 許可界限

許可界限是進階 AWS IAM 功能，其中已設定身分型政策可授予 IAM 實體的最大許可；其中這些實體是使用者或角色。為實體設定許可界限時，該實體只能執行其身分型政策及其許可界限允許的動作。

您可以提供許可界限，以便在該界限內建立 eksctl 建立的所有身分型實體。此範例示範如何將許可界限提供給 eksctl 建立的各種身分型實體：

```
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**
>
> 無法同時提供角色 ARN 和許可界限。

設定 VPC CNI 許可界限

請注意，建立啟用 OIDC 的 eksctl 叢集時，基於安全考量， 會自動`iamserviceaccount`為 VPC-CNI 建立 。如果您想要將許可界限新增至其中，則必須`iamserviceaccount`在組態檔案中手動指定：

```
iam:
  serviceAccounts:
    - metadata:
        name: aws-node
        namespace: kube-system
      attachPolicyARNs:
      - "arn:aws:iam::<arn>:policy/AmazonEKS_CNI_Policy"
      permissionsBoundary: "arn:aws:iam::11111:policy/entity/boundary"
```

IAM 政策

您可以將執行個體角色連接至節點群組。在節點上執行的工作負載會從節點接收 IAM 許可。如需 mroe 資訊，請參閱 Amazon EC2 的 IAM 角色。

此頁面列出 eksctl 中可用的預先定義 IAM 政策範本。這些範本可簡化授予 EKS 節點適當 AWS 服務許可的程序，而不必手動建立自訂 IAM 政策。

支援的 IAM 附加元件政策

所有支援的附加元件政策範例：

```
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
```

映像建置器政策

此imageBuilder政策允許完整 ECR （彈性容器登錄檔） 存取。這對於建置需要將映像推送至 ECR 的 CI 伺服器非常有用。

EBS 政策

此ebs政策會啟用新的 EBS CSI （彈性區塊存放區容器儲存介面） 驅動程式。

Cert Manager 政策

此certManager政策可讓 將記錄新增至 Route 53，以解決 DNS01 挑戰。如需詳細資訊，請參閱此處。

新增自訂執行個體角色

此範例會建立節點群組，重複使用來自另一個叢集的現有 IAM 執行個體角色：

```
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"
```

連接內嵌政策

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

依 ARN 連接政策

```
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**
>
> 如果節點群組包含 attachPolicyARNs，則AmazonEC2ContainerRegistryPullOnly在此範例中也必須包含預設節點政策，例如AmazonEKSWorkerNodePolicyAmazonEKS_CNI_Policy和 。

管理 IAM 使用者和角色

> **Note**
>
> AWS 建議the section called “EKS Pod 身分關聯”從 ConfigMap migraitng 至 `aws-auth` 。

EKS 叢集使用 IAM 使用者和角色來控制對叢集的存取。規則會在組態映射中實作

使用 CLI 命令編輯 ConfigMap

稱為 `aws-auth`。 `eksctl`提供命令來讀取和編輯此組態映射。

取得所有身分映射：

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

取得符合 arn 的所有身分映射：

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

建立身分映射：

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

刪除身分映射：

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

> **Note**
>
> 除非`--all`指定 ，否則上述命令會刪除單一映射 FIFO，在這種情況下會移除所有相符項目。如果找到符合此角色的更多映射，將會發出警告。

建立帳戶映射：

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

刪除帳戶映射：

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

使用 ClusterConfig 檔案編輯 ConfigMap

也可以在 ClusterConfig 中指定身分映射：

```
---
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
```

服務帳戶的 IAM 角色

> **Tip**
>
> `eksctl` 支援設定透過 EKS Pod Identity Associations 執行應用程式的 EKS 精細許可

Amazon EKS 支援此處的服務帳戶角色 (IRSA)】，可讓叢集運算子將 AWS IAM 角色映射至 Kubernetes 服務帳戶。

這可為在 EKS 上執行並使用其他 AWS 服務的應用程式提供精細的許可管理。這些可能是使用 S3、任何其他資料服務 (RDS、MQ、STS、DynamoDB) 或 Kubernetes 元件的應用程式，例如 AWS Load Balancer 控制器或 ExternalDNS。

您可以使用 輕鬆建立 IAM 角色和服務帳戶對`eksctl`。

> **Note**
>
> 如果您使用執行個體角色，並考慮改用 IRSA，則不應混合兩者。

運作方式

它透過 EKS 公開的 IAM OpenID Connect Provider (OIDC) 運作，且 IAM 角色必須參考 IAM OIDC Provider（特定於指定的 EKS 叢集）來建構，並參考其將繫結的 Kubernetes 服務帳戶。建立 IAM 角色後，服務帳戶應包含該角色的 ARN 做為註釋 (`eks.amazonaws.com/role-arn`)。根據預設，將建立或更新服務帳戶以包含角色註釋，這可以使用旗標 停用`--role-only`。

在 EKS 中，有一個許可控制器，根據 Pod 使用的服務帳戶上的註釋，分別將 AWS 工作階段登入資料注入 Pod 的角色。登入資料將由 `AWS_ROLE_ARN` & `AWS_WEB_IDENTITY_TOKEN_FILE`環境變數公開。由於使用了最新版本的 AWS 開發套件（如需確切版本的詳細資訊，請參閱此處），應用程式將使用這些登入資料。

資源`eksctl`的名稱為 iamserviceaccount，代表 IAM 角色和服務帳戶對。

來自 CLI 的用量

> **Note**
>
> 服務帳戶的 IAM 角色需要 Kubernetes 1.13 版或更新版本。

IAM OIDC 提供者預設為未啟用，您可以使用下列命令來啟用，或使用組態檔案（請參閱下文）：

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

一旦您有與叢集相關聯的 IAM OIDC 提供者，若要建立繫結至服務帳戶的 IAM 角色，請執行：

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

> **Note**
>
> 您可以指定`--attach-policy-arn`多次使用多個政策。

更具體地說，您可以透過執行以下操作來建立具有 S3 唯讀存取權的服務帳戶：

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

根據預設，它會在`default`命名空間中建立，但您可以指定任何其他命名空間，例如：

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

> **Note**
>
> 如果命名空間尚不存在，則會建立命名空間。

如果您已在叢集中建立服務帳戶（沒有 IAM 角色），則需要使用 `--override-existing-serviceaccounts` 旗標。

自訂標記也可以透過指定 套用至 IAM 角色--tags：

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

CloudFormation 會產生包含隨機字串的角色名稱。如果您偏好預先決定的角色名稱，您可以指定 --role-name：

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

當服務帳戶由一些其他工具建立和管理時，例如 helm，請使用 --role-only 來防止衝突。然後，另一個工具負責維護角色 ARN 註釋。請注意， --override-existing-serviceaccounts 對 roleOnly/--role-only 服務帳戶沒有影響，一律會建立角色。

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

當您擁有要與服務帳戶搭配使用的現有角色時，您可以提供 --attach-role-arn旗標，而不是提供政策。為了確保角色只能由指定的服務帳戶擔任，您應該在此處設定關係政策文件】。

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

若要更新服務帳戶角色許可，您可以執行 eksctl update iamserviceaccount。

> **Note**
>
> eksctl delete iamserviceaccount ServiceAccounts 即使 未建立 Kubernetes， 也會將其刪除eksctl。

使用組態檔案

若要iamserviceaccounts使用組態檔案管理 ，您會希望在 下設定iam.withOIDC: true和列出您想要的帳戶iam.serviceAccount。

所有命令都支援 --config-file，您可以像節點群組一樣管理 iamserviceaccounts。 eksctl create iamserviceaccount 命令支援 --include和 --exclude旗標 （ 如需這些運作方式的詳

細資訊，請參閱本節)。命令--only-missing也eksctl delete iamserviceaccount支援 ，因此您可以像節點群組一樣執行刪除。

> **Note**
>
> IAM 服務帳戶的範圍在命名空間中，即兩個同名的服務帳戶可能存在於不同的命名空間中。因此，若要將服務帳戶唯一定義為 --include、 --exclude旗標的一部分，您需要以 namespace/name 格式傳遞名稱字串。例如

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

使用 IRSA 搭配已知的使用案例，例如 cluster-autoscaler和 ，wellKnownPolicies包含啟用的選項cert-manager，作為 政策清單的速記。

支援的已知政策和其他屬性serviceAccounts會記錄在組態結構描述中。

您可以使用下列組態範例搭配 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
```

如果您在未設定這些欄位的情況下建立叢集，您可以使用下列命令來啟用您所需的一切：

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

詳細資訊

- 服務帳戶的精細 IAM 角色簡介
- EKS 使用者指南 - 服務帳戶的 IAM 角色
- 將 IAM 使用者和角色映射至 Kubernetes RBAC 角色

EKS Pod 身分關聯

AWS EKS 已為叢集管理員推出稱為 Pod Identity Association 的新增強型機制，以設定 Kubernetes 應用程式來接收與叢集外部 AWS 服務連線所需的 IAM 許可。不過，Pod Identity Association 會利用 IRSA，讓它可直接透過 EKS API 進行設定，因此完全不需要使用 IAM API。

因此，IAM 角色不再需要參考 OIDC 供應商，因此不會再繫結至單一叢集。這表示 IAM 角色現在可以跨多個 EKS 叢集使用，而不需要在每次建立新叢集時更新角色信任政策。這樣一來，就不需要重複角色，而且可簡化自動化 IRSA 的程序。

先決條件

在幕後，Pod 身分關聯的實作正在工作節點上執行代理程式做為協助程式集。若要在叢集上執行先決條件代理程式，EKS 提供稱為 EKS Pod Identity Agent 的新附加元件。因此，建立 Pod 身分關聯 （一般使用 和 `eksctl`) 需要預先安裝在叢集上的`eks-pod-identity-agent`附加元件。您可以使用與任何其他支援的附加元件`eksctl`相同的方式來建立此附加元件。

```
eksctl create addon --cluster my-cluster --name eks-pod-identity-agent
```

此外，如果在建立 Pod 身分關聯時使用預先存在的 IAM 角色，您必須將角色設定為信任新推出的 EKS 服務主體 (`pods.eks.amazonaws.com`)。IAM 信任政策範例如下所示：

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

如果您未將現有角色的 ARN 提供給建立命令， `eksctl`會在幕後建立一個角色，並設定上述信任政策。

建立 Pod 身分關聯

對於操縱 Pod 身分關聯， eksctl已在 下新增欄位iam.podIdentityAssociations，例如

```
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
```

如需完整範例，請參閱 pod-identity-associations.yaml。

> **Note**
>
> 除了permissionPolicy用作內嵌政策文件之外，所有其他欄位都有一個 CLI 旗標。

建立 Pod 身分關聯的方法如下。在叢集建立期間，透過指定所需的 Pod 身分關聯做為組態檔案的一部分並執行：

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

建立叢集後，使用組態檔案，例如

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

OR 使用 CLI 旗標，例如

```
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**
>
> 一次只能建立單一 IAM 角色與服務帳戶的關聯。因此，嘗試為相同的服務帳戶建立第二個 Pod 身分關聯將導致錯誤。

擷取 Pod 身分關聯

若要擷取特定叢集的所有 Pod 身分關聯，請執行下列其中一個命令：

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

或

```
eksctl get podidentityassociation --cluster my-cluster
```

此外，若要僅擷取指定命名空間內的 Pod 身分關聯，請使用 `--namespace`旗標，例如

```
eksctl get podidentityassociation --cluster my-cluster --namespace default
```

最後，若要擷取對應至特定 K8s 服務帳戶的單一關聯，請在上述命令中包含 `--service-account-name` ，即

```
eksctl get podidentityassociation --cluster my-cluster --namespace default --service-
account-name s3-reader
```

更新 Pod 身分關聯

若要更新一或多個 Pod 身分關聯的 IAM 角色，請將新的 傳遞`roleARN(s)`至組態檔案，例如

```
iam:
  podIdentityAssociations:
    - namespace: default
      serviceAccountName: s3-reader
```

```
      roleARN: new-role-arn-1
    - namespace: dev
      serviceAccountName: app-cache-access
      roleARN: new-role-arn-2
```

和 執行：

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

OR（更新單一關聯）`--role-arn` 透過 CLI 旗標傳遞新的 ：

```
eksctl update podidentityassociation --cluster my-cluster --namespace default --
service-account-name s3-reader --role-arn new-role-arn
```

刪除 Pod 身分關聯

若要刪除一或多個 Pod 身分關聯，請將 `namespace(s)`和 傳遞`serviceAccountName(s)`至組態檔案，例如

```
iam:
  podIdentityAssociations:
    - namespace: default
      serviceAccountName: s3-reader
    - namespace: dev
      serviceAccountName: app-cache-access
```

和 執行：

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

OR（若要刪除單一關聯）`--service-account-name` 透過 CLI 旗標傳遞 `--namespace`和 ：

```
eksctl delete podidentityassociation --cluster my-cluster --namespace default --
service-account-name s3-reader
```

EKS 附加元件支援 Pod 身分關聯

EKS 附加元件也支援透過 EKS Pod Identity Associations 接收 IAM 許可。組態檔案會公開三個允許設定的欄位：`addon.podIdentityAssociations`和`addonsConfig.autoApplyPodIdentityAssociations`和

addon.useDefaultPodIdentityAssociations。您可以使用 明確設定所需的 Pod 身分關聯，addon.podIdentityAssociations或使用 addonsConfig.autoApplyPodIdentityAssociations或 eksctl自動解析 （並套用） 建議的 Pod 身分組態addon.useDefaultPodIdentityAssociations。

> **Note**
>
> 並非所有 EKS 附加元件都會在啟動時支援 Pod 身分關聯。在此情況下，應使用 IRSA 設定繼續提供必要的 IAM 許可。

使用 IAM 許可建立附加元件

建立需要 IAM 許可的附加元件時， eksctl會先檢查 Pod 身分關聯或 IRSA 設定是否明確設定為組態檔案的一部分，如果是，請使用其中一個來設定附加元件的許可，例如

```
addons:
- name: vpc-cni
  podIdentityAssociations:
  - serviceAccountName: aws-node
    permissionPolicyARNs: ["arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"]
```

和 執行

```
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**
>
> 不允許同時設定 Pod 身分和 IRSA，這會導致驗證錯誤。

對於支援 Pod 身分的 EKS 附加元件， eksctl 提供在建立附加元件時自動設定任何建議 IAM 許可的選項。這可以透過在addonsConfig.autoApplyPodIdentityAssociations: true組態檔案中設定 來實現。例如，

```
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
```

和 執行

```
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
```

同樣，也可以透過 CLI 旗標完成，例如

```
eksctl create addon --cluster my-cluster --name vpc-cni --auto-apply-pod-identity-
associations
```

若要遷移現有的附加元件以搭配建議的 IAM 政策使用 Pod 身分，請使用

```
addons:
- name: vpc-cni
  useDefaultPodIdentityAssociations: true
```

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

使用 IAM 許可更新附加元件

更新附加元件時，指定 addon.PodIdentityAssociations 會在更新操作完成後，代表附加元件應擁有之狀態的單一事實來源。在幕後，會執行不同類型的操作，以達到所需的狀態，即

- 建立組態檔案中存在但叢集上缺少的 Pod 身分
- 刪除從組態檔案移除的現有 Pod 身分，以及任何相關聯的 IAM 資源
- 更新組態檔案中也存在且 IAM 許可集已變更的現有 Pod 身分

> **Note**
>
> EKS 附加元件擁有的 Pod 身分關聯生命週期由 EKS 附加元件 API 直接處理。

您無法針對與 Amazon EKS 附加元件搭配使用的關聯使用 `eksctl update podidentityassociation`（更新 IAM 許可）或 `eksctl delete podidentityassociations`（移除關聯）。反之，應使用 `eksctl update addon`或 `eksctl delete addon`。

讓我們看看上述的範例，從分析附加元件的初始 Pod 身分組態開始：

```
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"
    }
]
```

現在請使用下列組態：

```
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
```

和 執行

```
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
```

現在檢查 Pod 身分組態是否已正確更新

```
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"
    }
]
```

若要從附加元件移除所有 Pod 身分關聯，addon.PodIdentityAssociations必須明確設定為 []，例如

```
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: []
```

和 執行

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

刪除具有 IAM 許可的附加元件

刪除附加元件也會移除與附加元件相關聯的所有 Pod 身分。刪除叢集會為所有附加元件實現相同的效果。任何由 建立之 Pod 身分的 IAM 角色eksctl都會正常刪除。

將現有的 iamserviceaccounts 和附加元件遷移至 Pod 身分關聯

有一個 `eksctl` utils 命令可將服務帳戶的現有 IAM 角色遷移至 Pod 身分關聯，即

```
eksctl utils migrate-to-pod-identity --cluster my-cluster --approve
```

在幕後， 命令會套用下列步驟：

- 如果叢集上尚未處於作用中狀態，請安裝 `eks-pod-identity-agent` 附加元件
- 識別與 iamserviceaccounts 相關聯的所有 IAM 角色
- 識別與支援 Pod 身分關聯的 EKS 附加元件相關聯的所有 IAM 角色
- 使用額外的信任實體，更新所有已識別角色的 IAM 信任政策，指向新的 EKS 服務主體 （並選擇性地移除執行中的 OIDC 提供者信任關係）
- 為與 iamserviceaccounts 相關聯的篩選角色建立 Pod 身分關聯
- 使用 Pod 身分更新 EKS 附加元件 (EKS API 會在幕後建立 Pod 身分）

在沒有`--approve`旗標的情況下執行命令只會輸出由一組反映上述步驟的任務組成的計劃，例如

```
[#]  (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
```

現有的 OIDC 提供者信任關係一律會從與 EKS 附加元件相關聯的 IAM 角色中移除。此外，若要從與 iamserviceaccounts 相關聯的 IAM 角色中移除現有的 OIDC 提供者信任關係，請使用 `--remove-oidc-provider-trust-relationship`旗標執行 命令，例如

```
eksctl utils migrate-to-pod-identity --cluster my-cluster --approve --remove-oidc-
provider-trust-relationship
```

跨帳戶 Pod 身分支援

eksctl 支援 EKS Pod Identity 跨帳戶存取。此功能允許在 EKS 叢集中執行的 Pod 存取不同 AWS 帳戶中的 AWS 資源。

Usage

若要建立與跨帳戶存取的 Pod 身分關聯，請先設定 IAM 角色和政策，允許從來源 AWS 帳戶 （使用叢集） 存取目標 AWS 帳戶 （使用叢集可存取的資源）。如需此範例，請參閱「Amazon EKS Pod Identity 簡化跨帳戶存取。」

在每個帳戶中設定 IAM 角色後，請使用 eksctl 建立 Pod 身分關聯：

```
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
```

進一步參考

適用於 EKS 附加元件的官方 AWS Userdocs 支援 Pod 身分

Pod 身分關聯的官方 AWS 部落格文章

Pod Identity Associations 的官方 AWS userdocs

部署選項

本章涵蓋使用 eksctl 來管理部署到替代環境的 EKS 叢集。

如需 EKS 部署選項的最準確資訊，請參閱《EKS 使用者指南》中的跨雲端和內部部署環境部署 Amazon EKS 叢集。

主題：

- the section called “EKS Anywhere”
  - 搭配 Amazon EKS Anywhere 叢集使用 eksctl。
  - Amazon EKS Anywhere 是由 AWS 建置的容器管理軟體，可讓您更輕鬆地在內部部署和邊緣執行和管理 Kubernetes。
- the section called “AWS Outposts 支援”
  - 在 AWS Outposts 上使用 eksctl 搭配 EKS 叢集。
  - AWS Outposts 是全受管解決方案系列，可將 AWS 基礎設施和服務交付至幾乎任何內部部署或節點，以獲得真正一致的混合體驗。
  - eksctl 中的 AWS Outposts 支援可讓您使用整個 Kubernetes 叢集建立本機叢集，包括在 AWS Outposts 本機上執行的 EKS 控制平面和工作者節點。
- the section called “EKS 混合節點”
  - 使用您在 AWS 雲端中使用的相同 AWS EKS 叢集、功能和工具，在客戶受管基礎設施上執行內部部署和邊緣應用程式。

EKS Anywhere

`eksctl` 提供`EKS Anywhere`使用子命令 呼叫的 AWS 功能存取權`eksctl anywhere`。這需要 上存在的`eksctl-anywhere`二進位`PATH`。請依照此處概述的指示安裝 eksctl-anywhere 進行安裝。

完成後，請執行下列命令來執行任何位置的命令：

```
eksctl anywhere version
v0.5.0
```

如需 EKS Anywhere 的詳細資訊，請造訪 EKS Anywhere 網站。

AWS Outposts 支援

> **Warning**
>
> Outposts 不支援 EKS 受管節點群組。

將現有叢集擴展至 AWS Outpost

您可以將 AWS 區域中執行的現有 EKS 叢集擴展到 AWS Outposts，方法是設定 `nodeGroup.outpostARN`讓新節點群組在 Outposts 上建立節點群組，如下所示：

```
# 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
```

在此設定中，EKS 控制平面在 AWS 區域中執行，而具有 `outpostARN`設定的節點群組在指定的 Outpost 上執行。第一次在 Outposts 上建立節點群組時，eksctl 會透過在指定的 Outpost 上建立子網路來擴展 VPC。這些子網路用於建立已`outpostARN`設定的節點群組。

具有預先存在 VPC 的客戶需要在 Outposts 上建立子網路，並將其傳遞至 `nodeGroup.subnets`，如所示：

```
# 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"
```

在 AWS Outposts 上建立本機叢集

> **Note**
>
> 本機叢集僅支援 Outpost 機架。

> **Note**
>
> 當控制平面位於 Outposts 上時，節點群組僅支援 Amazon Linux 2。Outpost 上的節點群組僅支援 EBS gp2 磁碟區類型。

eksctl 中的 AWS Outposts 支援可讓您使用整個 Kubernetes 叢集建立本機叢集，包括在 AWS Outposts 本機上執行的 EKS 控制平面和工作者節點。客戶可以使用在 AWS Outposts 上本機執行的

EKS 控制平面和工作者節點建立本機叢集，也可以透過在 Outposts 上建立工作者節點，將 AWS 區域中執行的現有 EKS 叢集延伸到 AWS Outposts。

若要在 AWS Outposts 上建立 EKS 控制平面和節點群組，請將 `outpost.controlPlaneOutpostARN`設定為 Outpost ARN，如下所示：

```
# 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
```

這會指示 eksctl 在指定的 Outpost 上建立 EKS 控制平面和子網路。由於 Outposts 機架存在於單一可用區域中，eksctl 只會建立一個公有和私有子網路。eksctl 不會將建立的 VPC 與本機閘道建立關聯，因此 eksctl 將無法連線至 API 伺服器，而且將無法建立節點群組。因此，如果 在叢集建立期間`ClusterConfig`包含任何節點群組，則必須使用 執行 命令`--without-nodegroup`，如 所示：

```
eksctl create cluster -f outpost.yaml --without-nodegroup
```

建立叢集後，客戶有責任將 eksctl 建立的 VPC 與本機閘道建立關聯，以啟用與 API 伺服器的連線。在此步驟之後，可以使用 建立節點群組`eksctl create nodegroup`。

您可以選擇性地為 中的控制平面節點`outpost.controlPlaneInstanceType`或 中的節點群組指定執行個體類型`nodeGroup.instanceType`，但執行個體類型必須存在於 Outpost 上，否則 eksctl 會傳回錯誤。根據預設，eksctl 會嘗試在 Outpost 上為控制平面節點和節點群組選擇最小的可用執行個體類型。

當控制平面位於 Outpost 上時，會在該 Outpost 上建立節點群組。您可以選擇性地指定 中節點群組的 Outpost ARN，`nodeGroup.outpostARN`但必須符合控制平面的 Outpost ARN。

```
# 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
```

現有 VPC

具有現有 VPC 的客戶可以透過在 中指定子網路組態，在 AWS Outposts 上建立本機叢集`vpc.subnets`，如 所示：

```
# 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
```

子網路必須存在於 中指定的 Outpost 上，`outpost.controlPlaneOutpostARN`否則 eksctl 會傳回錯誤。如果您可以存取子網路的本機閘道，或可以連線至 VPC 資源，也可以在叢集建立期間指定節點群組。

本機叢集上不支援的功能

- 附加元件
- 服務帳戶的 IAM 角色
- IPv6
- 身分提供者
- Fargate
- KMS 加密

- 本地區域
- Karpenter
- 執行個體選取器
- 無法指定可用區域，因為它預設為 Outpost 可用區域。
- 不支援 `vpc.publicAccessCIDRs` 和 `vpc.autoAllocateIPv6`。
- 不支援對 API 伺服器的公有端點存取，因為本機叢集只能使用私有端點存取建立。

詳細資訊

- AWS Outposts 上的 Amazon EKS
- AWS Outposts 上 Amazon EKS 的本機叢集
- 建立本機叢集
- 在 Outpost 上啟動自我管理的 Amazon Linux 節點

安全

`eksctl` 提供一些選項，可改善 EKS 叢集的安全性。

`withOIDC`

啟用 `withOIDC`以自動建立 amazon CNI 外掛程式的 IRSA，並限制授予叢集中節點的許可，改為僅將必要的許可授予 CNI 服務帳戶。

此 AWS 文件會說明背景。

`disablePodIMDS`

對於受管和非受管節點群組，可使用 `disablePodIMDS`選項來防止在此節點群組中執行的所有非主機聯網 Pod 發出 IMDS 請求。

> **Note**
>
> 這無法與 搭配使用`withAddonPolicies`。

EKS 叢集的 KMS 信封加密

> **Note**
>
> Amazon Elastic Kubernetes Service (Amazon EKS) 會針對執行 Kubernetes 版本 1.28 或更新版本的 EKS 叢集中的所有 Kubernetes API 資料，提供預設封套加密。如需詳細資訊，請參閱《EKS 使用者指南》中所有 Kubernetes API 資料的預設信封加密。

EKS 支援使用 AWS KMS 金鑰來提供存放在 EKS 中 Kubernetes 秘密的信封加密。信封加密為存放在 Kubernetes 叢集中的應用程式秘密或使用者資料新增了額外的客戶受管加密層。

先前，Amazon EKS 支援僅在叢集建立期間使用 KMS 金鑰啟用信封加密。現在，您可以隨時為 Amazon EKS 叢集啟用信封加密。

閱讀 AWS 容器部落格上的使用 EKS 加密供應商支援defense-in-depth文章。

建立已啟用 KMS 加密的叢集

```
# 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
```

在現有叢集上啟用 KMS 加密

若要在尚未啟用的叢集上啟用 KMS 加密，請執行

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

或沒有組態檔案：

```
eksctl utils enable-secrets-encryption --cluster=kms-cluster --key-arn=arn:aws:kms:us-
west-2:<account>:key/<key> --region=<region>
```

除了在 EKS 叢集上啟用 KMS 加密之外，eksctl 還會使用新的 KMS 金鑰，透過註釋 更新它們來重新加密所有現有的 Kubernetes 秘密`eksctl.io/kms-encryption-timestamp`。透過傳遞 可以停用此行為`--encrypt-existing-secrets=false`，如 所示：

```
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>
```

如果叢集已啟用 KMS 加密，eksctl 將繼續重新加密所有現有的秘密。

> **Note**
>
> 啟用 KMS 加密後，就無法停用或更新它，以使用不同的 KMS 金鑰。

疑難排解

本主題包含如何使用 Eksctl 解決常見錯誤的指示。

失敗的堆疊建立

您可以使用 `--cfn-disable-rollback`旗標來停止 Cloudformation 復原失敗的堆疊，讓偵錯更容易。

子網路 ID "subnet-11111111" 與 "subnet-22222222" 不同

指定 VPC 子網路的組態檔案，如下所示：

```
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: []
```

錯誤`subnet ID "subnet-11111111" is not the same as "subnet-22222222"`表示指定的子網路未放置在正確的可用區域中。檢查 AWS 主控台中每個可用區域的正確子網路 ID。

在此範例中，VPC 的正確組態為：

```
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}
```

刪除問題

如果您的刪除無法運作，或者您忘記`--wait`新增刪除，您可能需要使用 amazon 的其他工具來刪除 cloudformation 堆疊。這可以透過 gui 或使用 aws cli 來完成。

kubectl 日誌和 kubectl 執行失敗並出現授權錯誤

如果您的節點部署在私有子網路中，或`kubectl logs`和`kubectl run`失敗並發生錯誤，如下所示：

```
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)
```

然後，您可能需要設定 enableDnsHostnames。您可以在此問題中找到更多詳細資訊。

公告

本主題涵蓋 Eksctl 新功能的過去公告。

受管節點群組預設

從 eksctl v0.58.0 開始，eksctl 預設會在未為 `eksctl create cluster`和 指定`ClusterConfig`檔案時建立受管節點群組`eksctl create nodegroup`。若要建立自我管理節點群組，請傳遞 `--managed=false`。如果使用受管節點群組中不支援的功能，例如 Windows 節點群組，這可能會中斷未使用組態檔案的指令碼。若要修正此問題，請傳遞 `--managed=false`，或使用建立自我管理節點群組的 `nodeGroups` 欄位，在`ClusterConfig`檔案中指定節點群組組態。

自訂 AMIs節點群組引導覆寫

此變更已在問題 Breaking： overrideBootstrapCommand 中宣布...。現在，它已經傳入此 PR。請仔細閱讀附加的問題，說明為什麼我們決定不使用引導指令碼或部分引導指令碼來支援自訂 AMIs。

我們仍提供協助程式！遷移希望不會那麼痛苦。 `eksctl`仍然提供指令碼，當來源時， 會匯出幾個有用的環境屬性和設定。此指令碼位於此處。

下列環境屬性將供您使用：

```
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
```

覆寫時需要使用的最小值，因此 `eksctl` 不會失敗， 是標籤！ `eksctl` 倚賴一組特定的標籤在節點上，以便找到它們。定義覆寫時，請提供此隱含最低覆寫命令：

```
    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}"
```

對於沒有傳出網際網路存取權的節點群組，您將需要提供 `--apiserver-endpoint`和 `--b64-cluster-ca`至引導指令碼，如下所示：

```
    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}
```

請注意 `--node-labels` 設定。如果未定義，節點將加入叢集，但在等待節點成為 時，最終`eksctl`會在最後一個步驟逾時`Ready`。它正在對具有標籤 的節點進行 Kubernetes 查詢`alpha.eksctl.io/nodegroup-name=<cluster-name>`。這僅適用於未受管節點群組。對於 受管，它使用不同的標籤。

如果完全可以切換到受管節點群組以避免此額外負荷，那麼現在就要這麼做了。讓所有覆寫變得更容易。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。