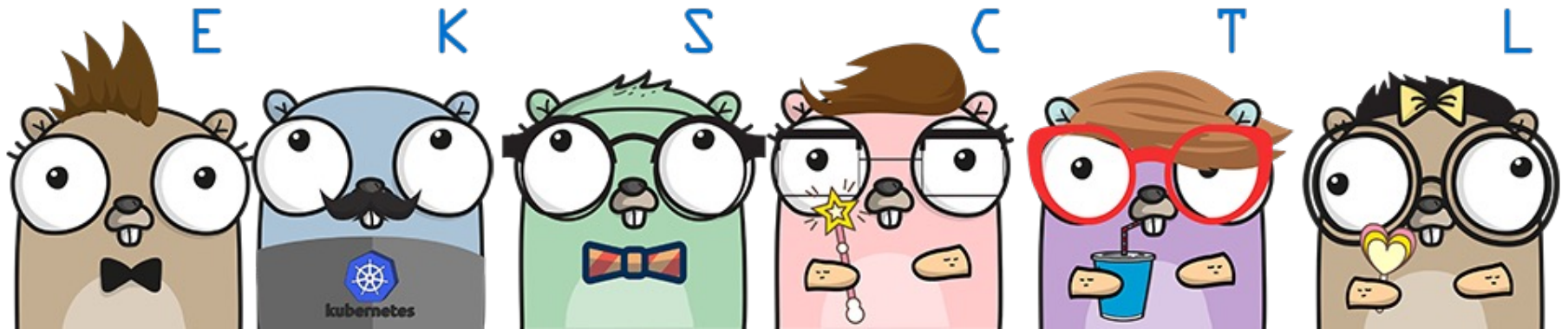
GS Neotek

DIY Amazon EKS with eksctl

요리조리 eksctl

SA 임지훈

CONTENTS



1

Intro

eksctl

What is eksctl?

eksctl - The official CLI for Amazon EKS

sponsored by weaveworks and built by on

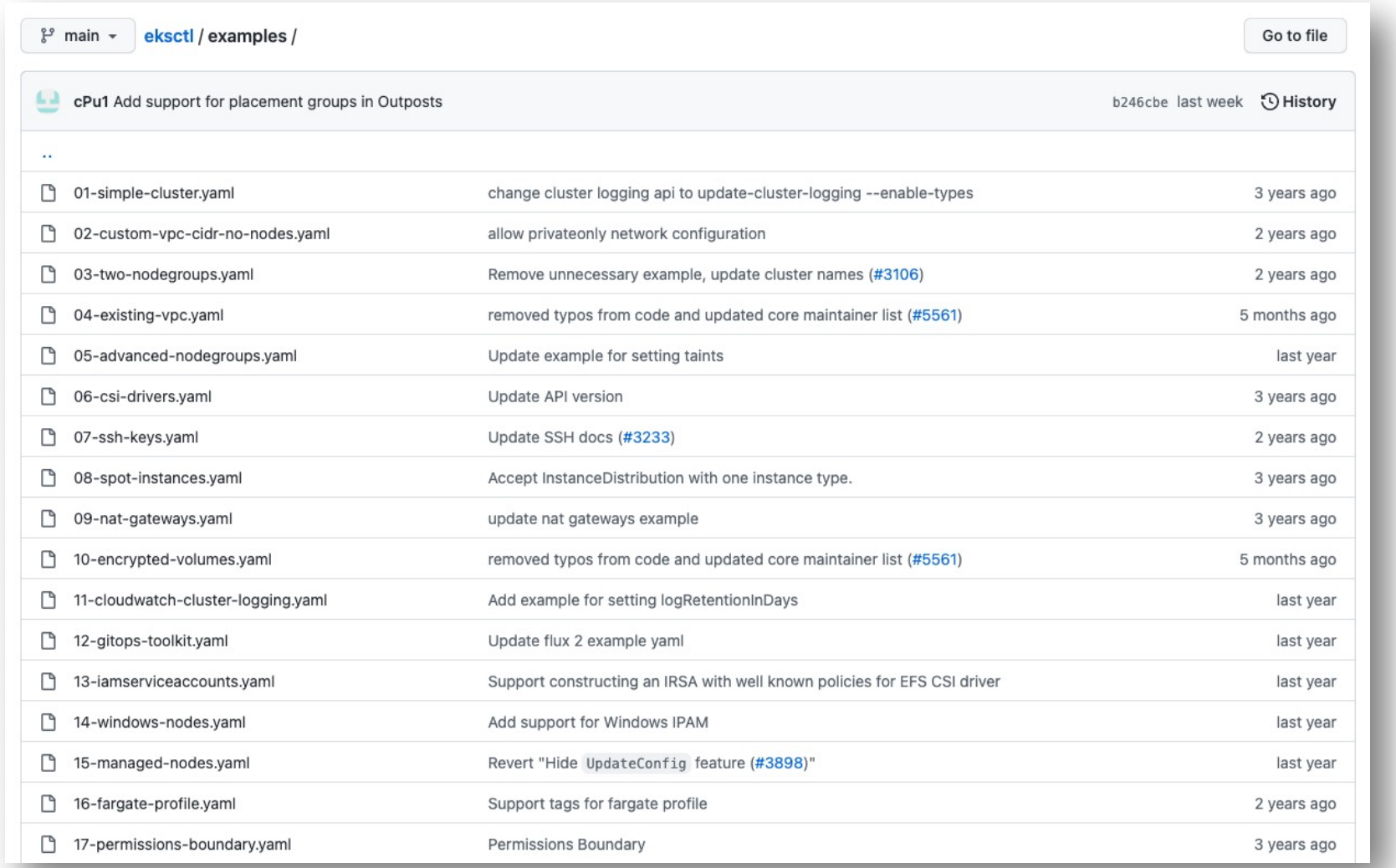
New for 2022

`eksctl` now supports creating clusters and nodegroups on AWS Outposts.

`eksctl` is a simple CLI tool for creating and managing clusters on EKS - Amazon's managed Kubernetes service for EC2. It is written in Go, uses CloudFormation, was created by Weaveworks and it welcomes contributions from the community. Create a basic cluster in minutes with just one command:

weaveworks/eksctl

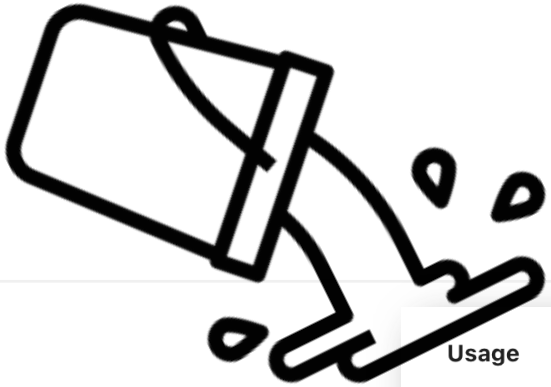
https://github.com/weaveworks/eksctl/tree/main/examples



main ▾ eksctl / examples / Go to file

cPu1 Add support for placement groups in Outposts b246cbe last week History

..		
01-simple-cluster.yaml	change cluster logging api to update-cluster-logging --enable-types	3 years ago
02-custom-vpc-cidr-no-nodes.yaml	allow privateonly network configuration	2 years ago
03-two-nodegroups.yaml	Remove unnecessary example, update cluster names (#3106)	2 years ago
04-existing-vpc.yaml	removed typos from code and updated core maintainer list (#5561)	5 months ago
05-advanced-nodegroups.yaml	Update example for setting taints	last year
06-csi-drivers.yaml	Update API version	3 years ago
07-ssh-keys.yaml	Update SSH docs (#3233)	2 years ago
08-spot-instances.yaml	Accept InstanceDistribution with one instance type.	3 years ago
09-nat-gateways.yaml	update nat gateways example	3 years ago
10-encrypted-volumes.yaml	removed typos from code and updated core maintainer list (#5561)	5 months ago
11-cloudwatch-cluster-logging.yaml	Add example for setting logRetentionInDays	last year
12-gitops-toolkit.yaml	Update flux 2 example yaml	last year
13-iamserviceaccounts.yaml	Support constructing an IRSA with well known policies for EFS CSI driver	last year
14-windows-nodes.yaml	Add support for Windows IPAM	last year
15-managed-nodes.yaml	Revert "Hide `UpdateConfig` feature (#3898)"	last year
16-fargate-profile.yaml	Support tags for fargate profile	2 years ago
17-permissions-boundary.yaml	Permissions Boundary	3 years ago



Record of intent

A Kubernetes object is a "record of intent"

Usage

- Clusters
  - Creating and managing clusters
  - AWS Outposts Support
  - Non eksctl-created clusters
  - Registering non-EKS clusters with EKS Connector
  - Customizing kubelet configuration
  - CloudWatch logging
  - EKS Fully-Private Cluster
  - Addons
  - Enabling Access for Amazon EMR
  - EKS Fargate Support
  - Cluster upgrades
  - Default add-on updates
- Nodegroups
- GitOps
- Security
- Networking
- IAM
- Dry Run

Using Config Files

You can create a cluster using a config file instead of flags.

First, create `cluster.yaml` file:

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

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

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

Next, run this command:

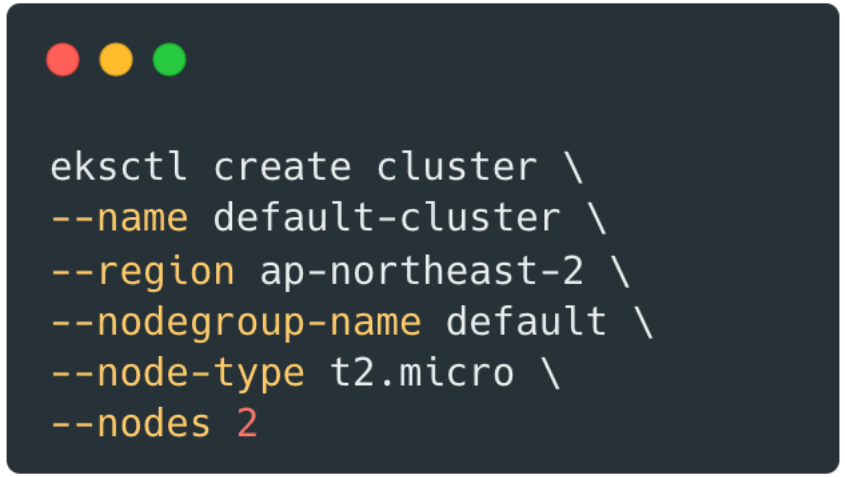
```
eksctl create cluster -f cluster.yaml
```

2

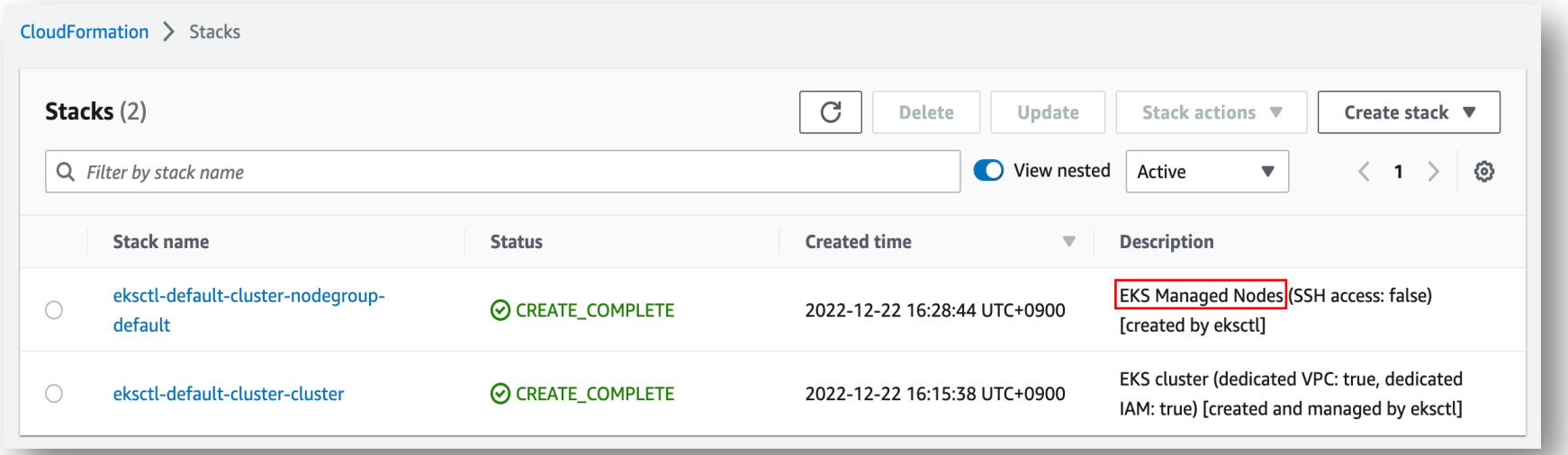
Basic

Default command

eksctl: Amazon EKS Cluster with One Command



```
eksctl create cluster \
--name default-cluster \
--region ap-northeast-2 \
--nodegroup-name default \
--node-type t2.micro \
--nodes 2
```



CloudFormation > Stacks

Stacks (2)

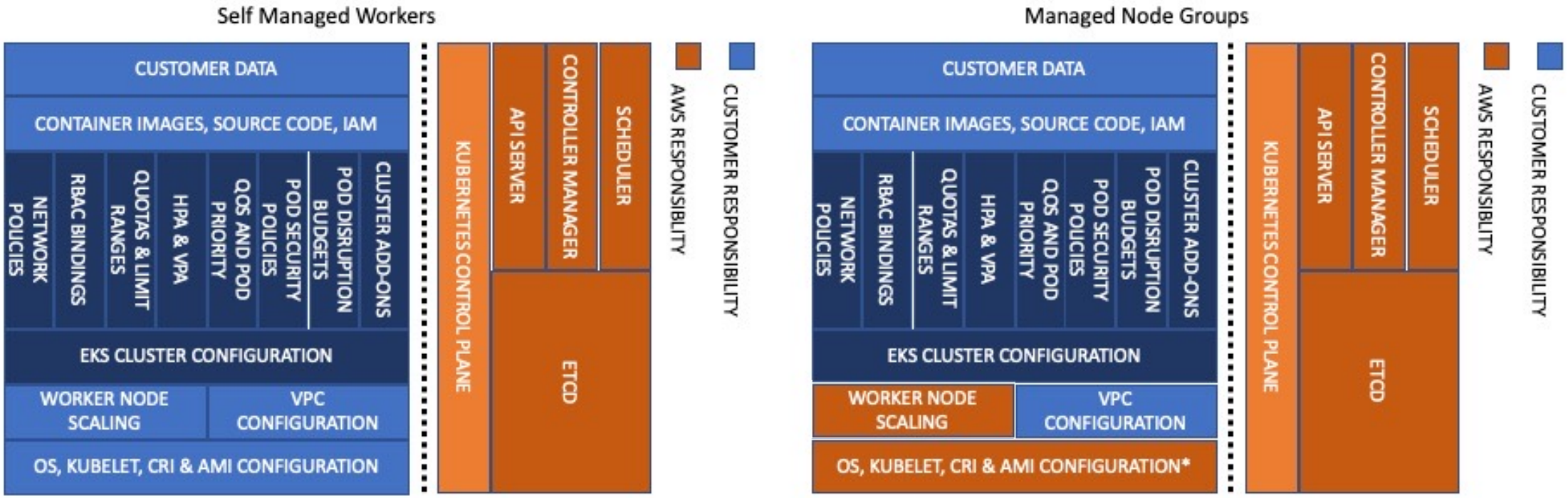
Delete | Update | Stack actions | Create stack

Filter by stack name | View nested | Active | 1

Stack name	Status	Created time	Description
eksctl-default-cluster-nodegroup-default	CREATE_COMPLETE	2022-12-22 16:28:44 UTC+0900	EKS Managed Nodes (SSH access: false) [created by eksctl]
eksctl-default-cluster-cluster	CREATE_COMPLETE	2022-12-22 16:15:38 UTC+0900	EKS cluster (dedicated VPC: true, dedicated IAM: true) [created and managed by eksctl]

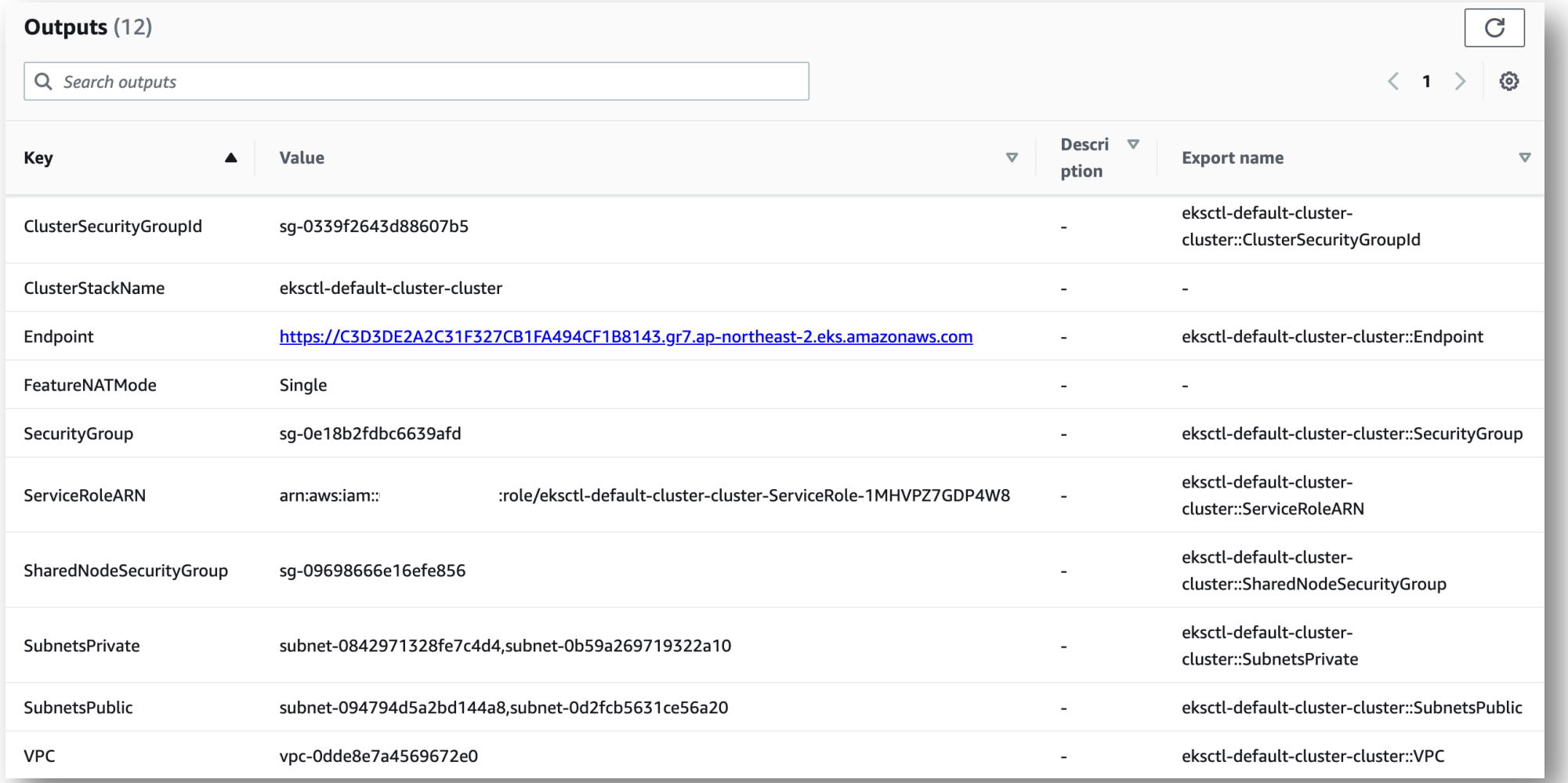
Understanding the Shared Responsibility Model

https://aws.github.io/aws-eks-best-practices/security/docs/



Default command

CloudFormation – Outputs (Cluster)



Outputs (12)

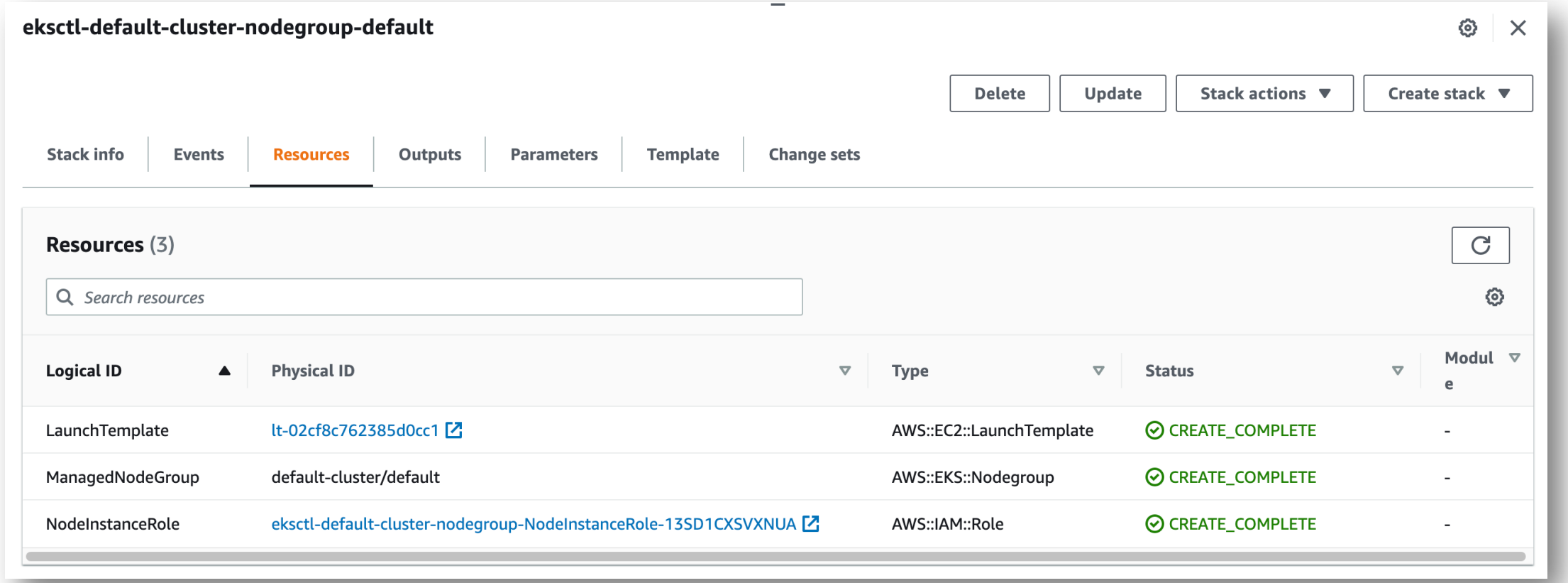
Search outputs

1

Key	Value	Description	Export name
ClusterSecurityGroupId	sg-0339f2643d88607b5	-	eksctl-default-cluster-cluster::ClusterSecurityGroupId
ClusterStackName	eksctl-default-cluster-cluster	-	-
Endpoint	https://C3D3DE2A2C31F327CB1FA494CF1B8143.gr7.ap-northeast-2.eks.amazonaws.com	-	eksctl-default-cluster-cluster::Endpoint
FeatureNATMode	Single	-	-
SecurityGroup	sg-0e18b2fdbc6639afd	-	eksctl-default-cluster-cluster::SecurityGroup
ServiceRoleARN	arn:aws:iam:: :role/eksctl-default-cluster-cluster-ServiceRole-1MHVPZ7GDP4W8	-	eksctl-default-cluster-cluster::ServiceRoleARN
SharedNodeSecurityGroup	sg-09698666e16efe856	-	eksctl-default-cluster-cluster::SharedNodeSecurityGroup
SubnetsPrivate	subnet-0842971328fe7c4d4,subnet-0b59a269719322a10	-	eksctl-default-cluster-cluster::SubnetsPrivate
SubnetsPublic	subnet-094794d5a2bd144a8,subnet-0d2fcb5631ce56a20	-	eksctl-default-cluster-cluster::SubnetsPublic
VPC	vpc-0dde8e7a4569672e0	-	eksctl-default-cluster-cluster::VPC

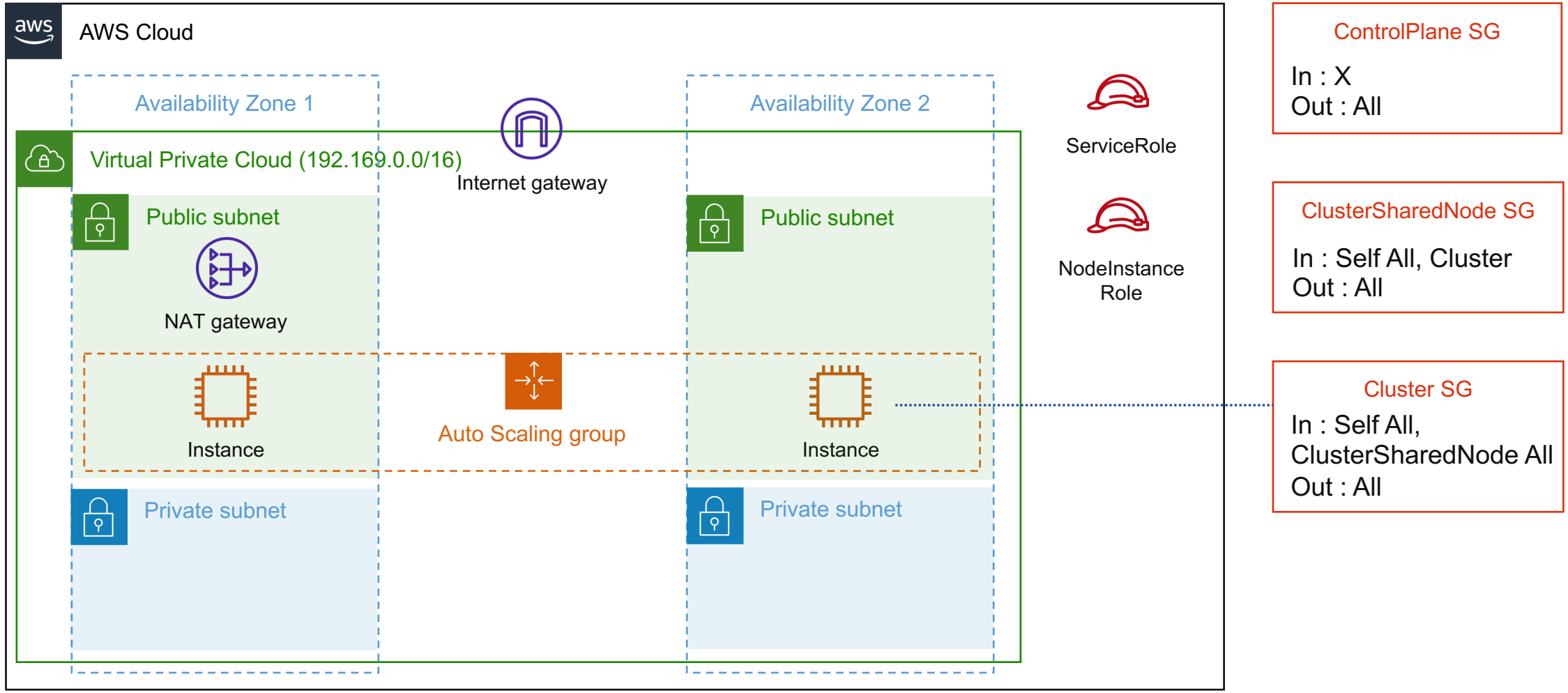
Default command

CloudFormation – Outputs (Nodegroup)



Architecture (default)

AWSKRUG



Security Groups (default)

https://docs.aws.amazon.com/ko_kr/eks/latest/userguide/sec-group-reqs.html

Security Groups (3) Info

Filter security groups

search: eks | Clear filters

Name	Security group ID	Description
eksctl-default-cluster-cluster/ControlPlaneSecurityGroup	sg-0e18b2fdbc6639afd	Communication between the control plane and worker nodegroups
eksctl-default-cluster-cluster/ClusterSharedNodeSecurityGroup	sg-09698666e16efe856	Communication between all nodes in the cluster
eks-cluster-sg-default-cluster-372376978	sg-0339f2643d88607b5	EKS created security group applied to ENI that is attached to EKS Control Plane master nodes, as well as any managed workloads.

Cluster security group Info

sg-0339f2643d88607b5

Additional security groups

sg-0e18b2fdbc6639afd

API server endpoint access

Info

Public

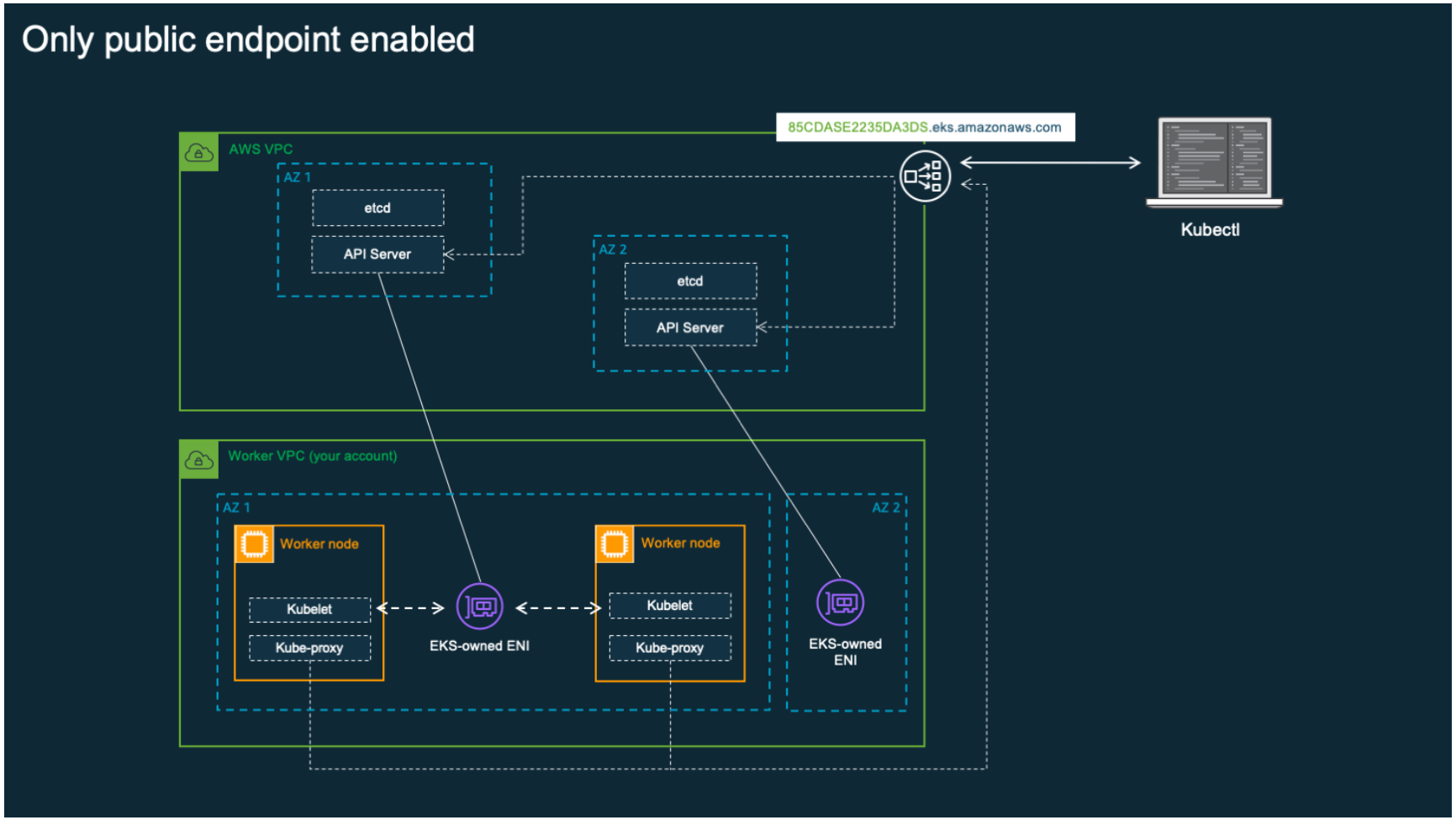
Public access source allowlist

0.0.0.0/0 (open to all traffic)

```
"sourceIPAddress": "eks.amazonaws.com",
"userAgent": "eks.amazonaws.com",
"requestParameters": {
    "groupName": "eks-cluster-sg-default-cluster
    "groupDescription": "EKS created security gr
    "vpcId": "vpc-0dde8e7a4569672e0"
},
```

EKS Cluster Architecture

https://aws.amazon.com/ko/blogs/containers/de-mystifying-cluster-networking-for-amazon-eks-worker-nodes/

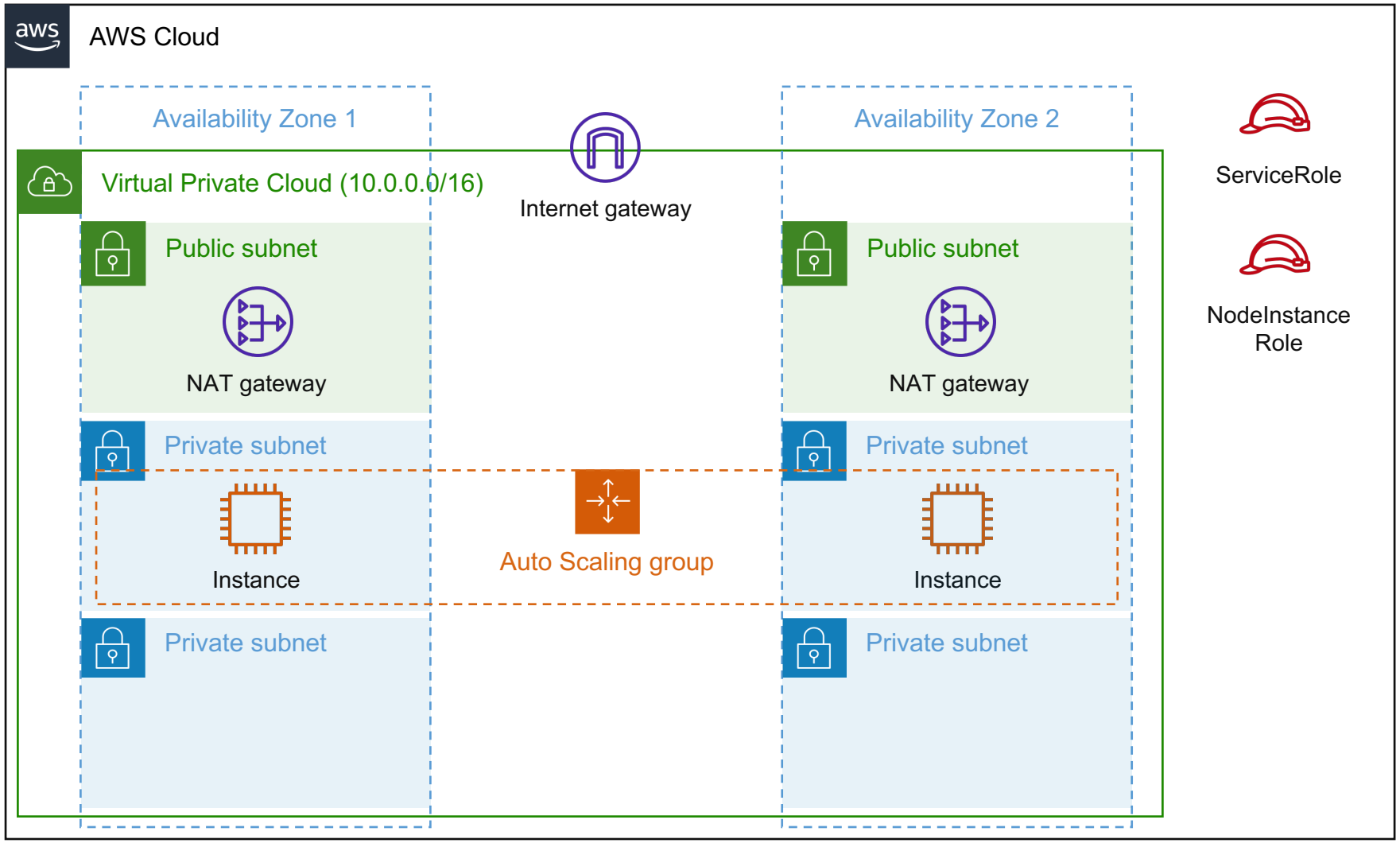


3

Advanced

Architecture (3-Tier)

AWSKRUG



Unmanaged

AWSKRUG

```
# eksctl create cluster -f 01-unmanaged.yaml --dry-run
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: demo
  region: ap-northeast-2
vpc:
  id: "<VPC ID>"
  clusterEndpoints:
    privateAccess: true
  subnets:
    private:
      private1:
        id: "<Subnet ID>"
      private2:
        id: "<Subnet ID>"
nodeGroups:
  - name: my-nodegroup
    instanceType: t3.micro
    desiredCapacity: 1
    subnets:
      - <Subnet ID>
      - <Subnet ID>
```

Tags

Key	Value
aws:cloudformation:logical-id	NodeGroup
aws:ec2launchtemplate:id	lt-0ed78c8fad0316099
eksctl.io/v1alpha2/nodegroup-name	my-nodegroup
alpha.eksctl.io/eksctl-version	0.113.0
aws:cloudformation:stack-name	eksctl-demo-nodegroup-my-nodegroup
aws:cloudformation:stack-id	arn:aws:cloudformation:ap-northeast-2:0
alpha.eksctl.io/nodegroup-name	my-nodegroup
alpha.eksctl.io/nodegroup-type	unmanaged
aws:ec2launchtemplate:version	1
kubernetes.io/cluster/demo	owned

💡 *When placing nodegroups inside a private subnet, privateNetworking must be set to true on the nodegroup*

Unmanaged – Customizing kubelet config

AWSKRUG


```yaml
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
```

```yaml
nodeGroups:
  - name: my-nodegroup
    kubeletExtraConfig:
      imageGCHighThresholdPercent: 70
      imageGCLowThresholdPercent: 50
```

Dry Run (Cheat Sheet)

https://eksctl.io/usage/schema/

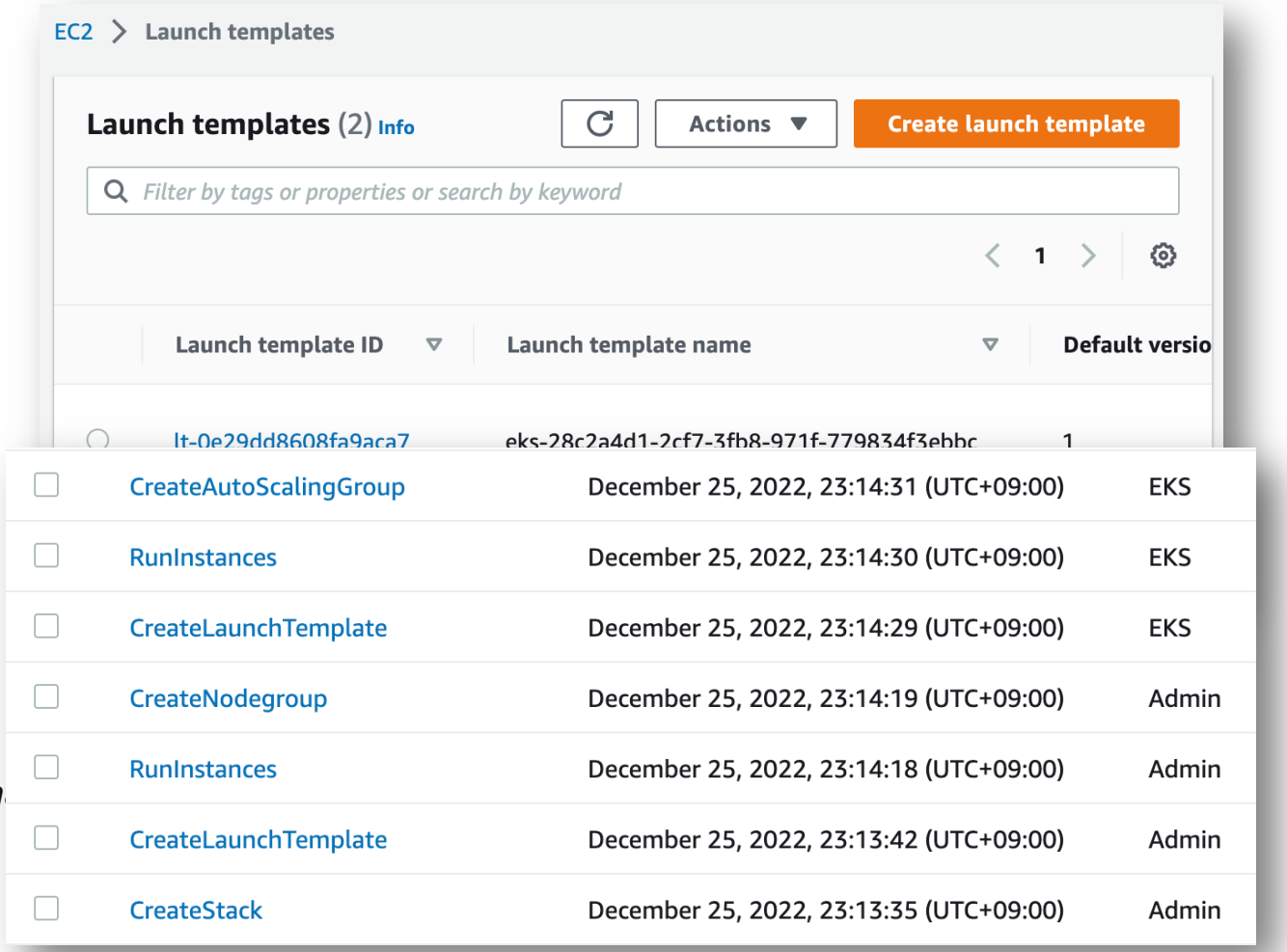

```
apiVersion: eksctl.io/v1alpha5
iam:
  vpcResourceControllerPolicy: true
  withOIDC: false
kind: ClusterConfig
metadata:
  name: demo
  region: ap-northeast-2
  version: "1.22"
nodeGroups:
- amiFamily: AmazonLinux2
  containerRuntime: dockerd
  desiredCapacity: 1
  disableIMDSv1: false
  disablePodIMDS: false
  iam:
    withAddonPolicies:
      albIngress: false
      appMesh: null
      appMeshPreview: null
      autoScaler: false
      awsLoadBalancerController: false
      certManager: false
      cloudWatch: false
      ebs: false
      efs: false
      externalDNS: false
      fsx: false
      imageBuilder: false
      xRay: false
  instanceSelector: {}
  instanceType: t3.micro
  labels:
    alpha.eksctl.io/cluster-name: demo
    alpha.eksctl.io/nodegroup-name: my-nodegroup
  name: my-nodegroup
  privateNetworking: false
  securityGroups:
    withLocal: true
    withShared: true
  ssh:
    allow: false
  subnets:
  - subnet-0659aa123b16fe5dd
  - subnet-0dd2698a32d1014ea
  volumeIOPS: 3000
  volumeSize: 80
  volumeThroughput: 125
  volumeType: gp3
privateCluster:
  enabled: false
  skipEndpointCreation: false
vpc:
  clusterEndpoints:
    privateAccess: true
    publicAccess: true
  id: vpc-01439c15c75bd816c
  manageSharedNodeSecurityGroupRules: true
  nat:
    gateway: Disable
  subnets:
    private:
      private1:
        az: private1
        id: subnet-0659aa123b16fe5dd
      private2:
        az: private2
        id: subnet-0dd2698a32d1014ea
```

Managed

https://docs.aws.amazon.com/eks/latest/APIReference/API_CreateNodegroup.html


```yaml
managedNodeGroups:
  - name: managed-ng
    privateNetworking: true
    instanceType: t3.small
    desiredCapacity: 1
    volumeSize: 10
    subnets:
      - <Subnet ID>
      - <Subnet ID>
```

$ eksctl create nodegroup -f <filename>.yaml
$ eksctl delete nodegroup --cluster=demo --name=man



Managed – Spot, On-Demand

AWSKRUG

```
# `instanceTypes` defaults to [`m5.large`]
- name: spot
  spot: true
  privateNetworking: true
  desiredCapacity: 1
  volumeSize: 10
  subnets:
    - <Subnet ID>
    - <Subnet ID>

# On-Demand instances
- name: on-demand
  instanceTypes: ["t2.small", "t3.small"]
  privateNetworking: true
  desiredCapacity: 1
  volumeSize: 10
  subnets:
    - <Subnet ID>
    - <Subnet ID>
```

Unmanaged Nodegroups

`eksctl` has support for spot instances through the MixedInstancesPolicy for Auto Scaling Groups.

Here is an example of a nodegroup that uses 50% spot instances and 50% on demand instances:

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

Feature parity with unmanaged nodegroups

EKS Managed Nodegroups are managed by AWS EKS and do not offer the same level of configuration as unmanaged nodegroups. The unsupported options are noted below.

- `iam.instanceProfileARN` is not supported for managed nodegroups.
- `instancesDistribution` field is not supported

Managed – Custom AMI

AWSKRUG

NEW Support for custom AMI, security groups, `instancePrefix`, `ebsOptimized`, `volumeType`, `volumeName`, `volumeEncrypted`, `maxPodsPerNode`, `preBootstrapCommands`, `overrideBootstrapC`

```yaml
# cluster.yaml
# A cluster with an unmanaged nodegroup and two manage
---
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
    disableIMDSv1: true
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh managed-cluster --kubelet-
```

```yaml
managedNodeGroups:
  - name: custom-ng
    privateNetworking: true
    instanceType: t3.medium
    desiredCapacity: 1
    ami: ami-0d35aabb05c2cedb4
    maxPodsPerNode: 4 # 17
    volumeSize: 10
    volumeName: /dev/xvda
    volumeEncrypted: true
    subnets:
      - subnet-0659aa123b16fe5dd
      - subnet-0dd2698a32d1014ea
    overrideBootstrapCommand: |
      #!/bin/bash
```

```
demo --use-max-pods false --kubelet-extra-args '--max-pods=4 --node-labels
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL zsh

```
❯ eksctl create nodegroup -f 04-managed-custom.yaml --dry-run
Error: could not create cluster provider from options: managedNodeGroups[0].maxPo
dsPerNode is not supported when using a custom AMI (managedNodeGroups[0].ami)
```

Managed – Custom AMI

https://eksctl.io/announcements/nodegroup-override-announcement/


```
❯ k get node ip-10-0-129-122.ap-northeast-2.compute.internal -ojsonpath='{.status.capacity.pods}{"\n"}'
4
❯ k get po -A
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE
default       nginx                      0/1     Pending   0          13s
kube-system   aws-node-jw4ls             1/1     Running   0          9m20s
kube-system   coredns-556f6dffc4-6hg2s   1/1     Running   0          15m
kube-system   coredns-556f6dffc4-9lbxc   1/1     Running   0          15m
kube-system   kube-proxy-4lzlw           1/1     Running   0          9m20s
```

For nodegroups that have no outbound internet access, you'll need to supply `--apiserver-endpoint` and `--b64-cluster-ca` to the bootstrap script as follows:

```
    overrideBootstrapCommand: |
      #!/bin/bash

      source /var/lib/cloud/scripts/eksctl/bootstrap.helper.sh

      # Note "--node-labels=${NODE_LABELS}" needs the above helper sourced to work
      /etc/eks/bootstrap.sh ${CLUSTER_NAME} --container-runtime containerd --kubel
        --apiserver-endpoint ${API_SERVER_URL} --b64-cluster-ca ${B64_CLUSTER_CA}
```

💡 **Bare minimum override command**

Managed – Custom Launch Template

AWSKRUG

```
set -o xtrace

# Inject imageGCHighThresholdPer
if ! grep -q imageGCHighThresho
then
    sed -i '/"apiVersion*/a \ \
config.json
fi

# Inject imageGCLowThresholdPer
if ! grep -q imageGCLowThreshol
then
    sed -i '/"imageGCHigh*/a \
config.json
fi
```

Launch template Info

Switch to launch configuration

Launch template

Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

eks-f8c2a63f-c484-dad8-37a8-d666c4f3115c

Create a launch template

Version

1

- Latest (3)
- Default (3)
- 3
- 2
- 1 ✓

ersion

AMI ID

ami-0171653eda904c279

Launch template

eks-f8c2a63f-c484-dad8-37a8-d666c4f3115c

lt-0d9dae14748cbadc7

Security groups

-

Instance type

-

Request Spot Instances

No

Launch Template support for Managed

AWSKRUG

Creating managed nodegroups using a provided launch template

```yaml
# 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 unspeci

  - 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
```

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

metadata:
  name: demo
  region: ap-northeast-2

managedNodeGroups:
  - name: <LT Name>
    privateNetworking: true
    subnets:
      - <Subnet ID>
      - <Subnet ID>
    launchTemplate:
```

```
2023-01-02 11:44:24 [✖]  instance type must be set in the launch
template if managedNodeGroup.instanceTypes is not specified
Error: failed to create nodegroups for cluster "demo"
```

Launch Template support for Managed

https://eksctl.io/usage/launch-template-support/

Notes on custom AMI and launch template support

- When a launch template is provided, the following fields are not supported: `ami`, `ssh.allow`, `ssh.sourceSecurityGroupIds`, `securityGroups`, `insta` `instanceName`, `ebsOptimized`, `volumeEncrypted`, `volumeKmsKeyID`, `volu` `maxPodsPerNode`, `preBootstrapCommands`, `overrideBootstrapCommand` an `disableIMDSv1`.
- When using a custom AMI (`ami`), `overrideBootstrapCommand` must also be set to perform the bootstrapping.
- `overrideBootstrapCommand` can only be set when using a custom AMI.

eksctl-demo-nodegroup-managed-ng (lt-093231ae4b90c9f98) **Custom LT**

Launch template version details

Version: 2 | Description: -

Instance details | Storage | Resource tags | Network interfaces | Advanced details

AMI ID: - | Instance type: t2.micro

eks-e4c2b830-705f-eae9-01d4-d77b8950588f (lt-094734336f6818115)

Made by EKS

Launch template version details

Version: 1 (Default) | Description: -

Instance details | Storage | Resource tags | Network interfaces | Advanced details

AMI ID: ami-0ed39f52c0b66ac73 | Instance type: t2.micro

Fargate

https://eksctl.io/usage/fargate-support/


```
$ eksctl create fargateprofile -f fargate.yaml
$ eksctl delete fargateprofile --cluster <Cluster Name> --name <Fargate Name>
```

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

metadata:
  name: demo
  region: ap-northeast-2

fargateProfiles:
  - name: fp-default
    selectors:
      - namespace: default
      - namespace: kube-system
```

Monitoring

https://eksctl.io/usage/cloudwatch-cluster-logging/


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

$ eksctl utils update-cluster-logging -f example.yaml --approve
```

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

IAM

https://eksctl.io/usage/iam-policies/

https://eksctl.io/usage/iamserviceaccounts/


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

`--dry-run` 대신 `--approve` 옵션 제공

```
iam:
  withOIDC: true
  serviceAccounts:
    - metadata:
        name: aws-load-balancer
        namespace: kube-system
      wellKnownPolicies:
        awsLoadBalancerController: true
      tags:
        environment: demo
```

```
$ eksctl delete iamserivceaccount ₩
   --cluster <Cluster Name> ₩
   --name <ServiceAccount Name>
```

OIDC는 삭제 명령어 없음

Add-ons

https://eksctl.io/usage/addons/


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

```
❯ eksctl utils describe-addon-versions --kubernetes-version 1.22 | grep AddonName
                    "AddonName": "aws-ebs-csi-driver",
                    "AddonName": "vpc-cni",
                    "AddonName": "adot",
                    "AddonName": "coredns",
                    "AddonName": "kube-proxy",
                    "AddonName": "kubecost_kubecost",
                    "AddonName": "dynatrace_dynatrace-operator",
                    "AddonName": "upbound_universal-crossplane",
                    "AddonName": "teleport_teleport",
                    "AddonName": "factorhouse_kpow",
                    "AddonName": "tetrate-io_istio-distro",
```

$ eksctl delete addon --cluster <Cluster Name> --name <Addon Name>

Upgrade / Update

https://eksctl.io/usage/cluster-upgrade/

https://eksctl.io/usage/addon-upgrade/


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

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

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

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

```
$ eksctl utils update-kube-proxy --cluster=<clusterName>
$ eksctl utils update-aws-node --cluster=<clusterName>
$ eksctl utils update-coredns --cluster=<clusterName>
```

Q & A

More is more

cloudplayer@gsneotek.com