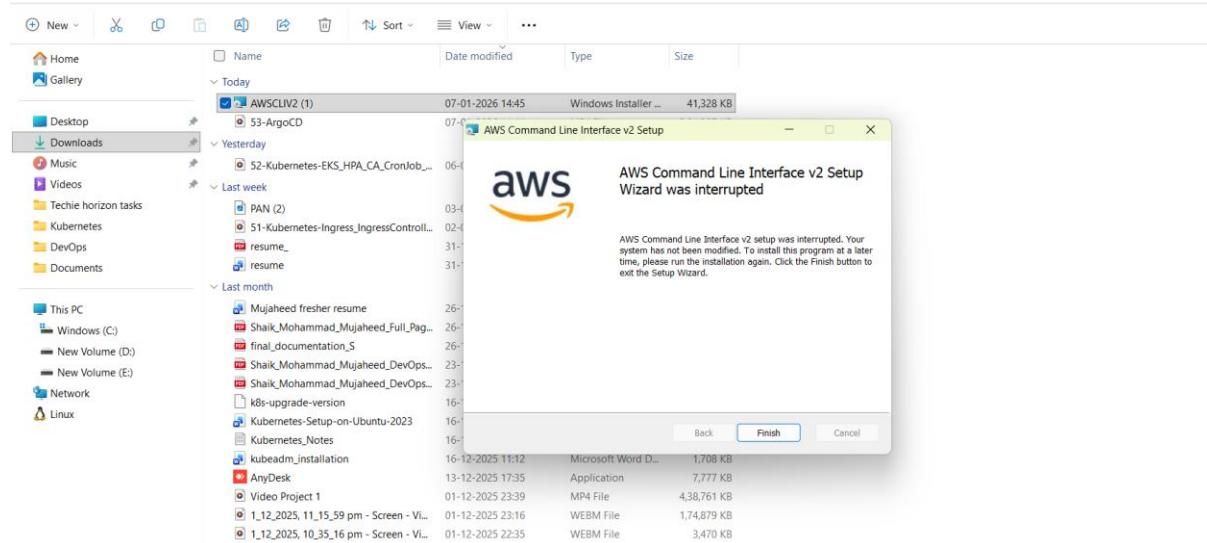


1. Setup eks cluster using eksctl

Download this from browser and install in windows machine.

- <https://awscli.amazonaws.com/AWSCLIV2.msi>



- aws --version
- aws configure

```
MUJJO SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ aws --version

aws-cli/2.30.4 Python/3.13.7 windows/11 exe/AMD64
```

```
MUJJO SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$

MUJJO SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ aws configure
AWS Access Key ID [*****QACQ]:
```

```
AWS Secret Access Key [*****+q1G]:
Default region name [us-east-1]:
Default output format [json]:
```

```
MUJJO SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$
```

[Download kubectl:](#)

- curl -LO
<https://dl.k8s.io/release/v1.29.0/bin/windows/amd64/kubectl.exe>
- mkdir -p ~/bin
- mv kubectl.exe ~/bin
- echo 'export PATH=\$PATH:\$HOME/bin' >> ~/.bashrc
- source ~/.bashrc

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ curl -LO https://dl.k8s.io/release/v1.29.0/bin/windows/amd64/kubectl.exe
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload   Total Spent  Left Speed
100  138  100  138    0     0   209      0 --:--:-- --:--:-- --:--:-- 209
100 48.6M 100 48.6M    0     0  14.4M      0  0:00:03  0:00:03 --:--:-- 20.2M

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ mkdir -p ~/bin

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ mv kubectl.exe ~/bin

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ source ~/.bashrc

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl version --client
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ |
```

Download eksctl:

- curl -LO
https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_Windows_amd64.zip
- unzip eksctl_Windows_amd64.zip
- mkdir -p ~/bin
- mv eksctl.exe ~/bin

```

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ curl -LO https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_windows_amd64.zip
% Total    % Received % Xferd  Average Speed   Time   Time     Current
          Dload  Upload Total   Spent   Left  Speed
0       0     0      0      0      0      0 --:--:-- --:--:-- 0
0       0     0      0      0      0      0 --:--:-- 0:00:01 --:--:-- 0
0       0     0      0      0      0      0 --:--:-- 0:00:01 --:--:-- 0
100 35.7M  100 35.7M    0      0  9.9M      0  0:00:03  0:00:03 --:--:-- 15.2M

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ unzip eksctl_windows_amd64.zip
Archive:  eksctl_windows_amd64.zip
  inflating: eksctl.exe

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ mkdir -p ~/bin

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ mv eksctl.exe ~/bin

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ eksctl version
0.221.0

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ |

```

- **eksctl create cluster **
- name my-eks-cluster **
- region us-east-1 **
- node-type c7i-flex.large **
- nodes 2**

It will take 20 min to create cluster.

```

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ eksctl create cluster \
> --name my-eks-cluster \
> --region us-east-1 \
> --node-type c7i-flex.large \
2026-01-07 15:04:56 [!] eksctl version 0.221.0
2026-01-07 15:04:56 [!] using region us-east-1
2026-01-07 15:04:58 [!] setting availability zones to [us-east-1d us-east-1f]
2026-01-07 15:04:58 [!] subnets for us-east-1d - public:192.168.0.0/19 private:192.168
2026-01-07 15:04:58 [!] subnets for us-east-1f - public:192.168.32.0/19 private:192.168
2026-01-07 15:04:58 [!] nodegroup "ng-aaecd7bf" will use "" [AmazonLinux2023/1.32]
2026-01-07 15:04:58 [!] Auto Mode will be enabled by default in an upcoming release of eksctl. Some add-ons will no longer be created by default. To maintain current behavior, explicitly enable them via eksctl config set-mode auto. Learn more: https://eksctl.io/usage/auto-mode/
2026-01-07 15:04:58 [!] using Kubernetes version 1.32
2026-01-07 15:04:58 [!] creating EKS cluster "my-eks-cluster" in "us-east-1" region with 2 nodes
2026-01-07 15:04:58 [!] will create 2 separate CloudFormation stacks for cluster and nodegroup
2026-01-07 15:04:58 [!] if you encounter any issues, check CloudFormation console or the CloudWatch logs for errors
2026-01-07 15:04:58 [!] Kubernetes API endpoint access will use default of {publicAccess=true} in "us-east-1"
2026-01-07 15:04:58 [!] CloudWatch logging will not be enabled for cluster "my-eks-cluster" in "us-east-1" region
2026-01-07 15:04:58 [!] you can enable it with 'eksctl utils update-cluster-logging --region=us-east-1 --cluster=my-eks-cluster'
2026-01-07 15:04:58 [!] default addons kube-proxy, coredns, metrics-server, vpc-cni will be used
2026-01-07 15:04:58 [!] 2 sequential tasks: { create cluster control plane "my-eks-cluster", 2 sequential sub-tasks: { 2 sequential sub-tasks: { 1 task: { create addons }, wait for control plane to become ready, }, create managed nodegroup "ng-aaecd7bf", }

```

```

2026-01-07 15:16:28 [v] building managed nodegroup stack "eksctl-my-eks-cluster-nodegroup-ng-aaecd7bf"
2026-01-07 15:16:30 [v] deploying stack "eksctl-my-eks-cluster-nodegroup-ng-aaecd7bf"
2026-01-07 15:16:30 [v] waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-ng-aaecd7bf"
2026-01-07 15:17:04 [v] waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-ng-aaecd7bf"
2026-01-07 15:17:58 [v] waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-ng-aaecd7bf"
2026-01-07 15:19:31 [v] waiting for CloudFormation stack "eksctl-my-eks-cluster-nodegroup-ng-aaecd7bf"
2026-01-07 15:19:32 [v] waiting for the control plane to become ready
2026-01-07 15:19:33 [v] saved kubeconfig as "C:\\\\Users\\\\Ashish\\\\.kube\\\\config"
2026-01-07 15:19:33 [v] no tasks
2026-01-07 15:19:33 [v] all EKS cluster resources for "my-eks-cluster" have been created
2026-01-07 15:19:35 [v] nodegroup "ng-aaecd7bf" has 2 node(s)
2026-01-07 15:19:35 [v] node "ip-192-168-28-220.ec2.internal" is ready
2026-01-07 15:19:35 [v] node "ip-192-168-41-142.ec2.internal" is ready
2026-01-07 15:19:35 [v] waiting for at least 2 node(s) to become ready in "ng-aaecd7bf"
2026-01-07 15:19:35 [v] nodegroup "ng-aaecd7bf" has 2 node(s)
2026-01-07 15:19:35 [v] node "ip-192-168-28-220.ec2.internal" is ready
2026-01-07 15:19:35 [v] node "ip-192-168-41-142.ec2.internal" is ready
2026-01-07 15:19:35 [v] created 1 managed nodegroup(s) in cluster "my-eks-cluster"
2026-01-07 15:19:36 [v] creating addon: metrics-server
2026-01-07 15:19:37 [v] successfully created addon: metrics-server
2026-01-07 15:19:40 [v] kubectl command should work with "C:\\\\Users\\\\Ashish\\\\.kube\\\\config", try 'kubectl get nodes'
2026-01-07 15:19:40 [v] EKS cluster "my-eks-cluster" in "us-east-1" region is ready

```

• kubectl get nodes

```

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get nodes

NAME           STATUS    ROLES      AGE     VERSION
ip-192-168-28-220.ec2.internal   Ready     <none>   4m32s   v1.32.9-eks-ecaa3a6
ip-192-168-41-142.ec2.internal   Ready     <none>   4m33s   v1.32.9-eks-ecaa3a6

```

• kubectl get pods -A

```

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get pods -A

NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE
kube-system   aws-node-7bvmr                      2/2    Running   0          4m45s
kube-system   aws-node-wv58z                      2/2    Running   0          4m45s
kube-system   coredns-6b9575c64c-2pz9g            1/1    Running   0          8m25s
kube-system   coredns-6b9575c64c-sg4w9            1/1    Running   0          8m25s
kube-system   kube-proxy-dmjst                   1/1    Running   0          4m45s
kube-system   kube-proxy-htq8                     1/1    Running   0          4m45s
kube-system   metrics-server-7cbd59cb7c-62lxf   1/1    Running   0          3m8s
kube-system   metrics-server-7cbd59cb7c-qt771   1/1    Running   0          3m8s

```

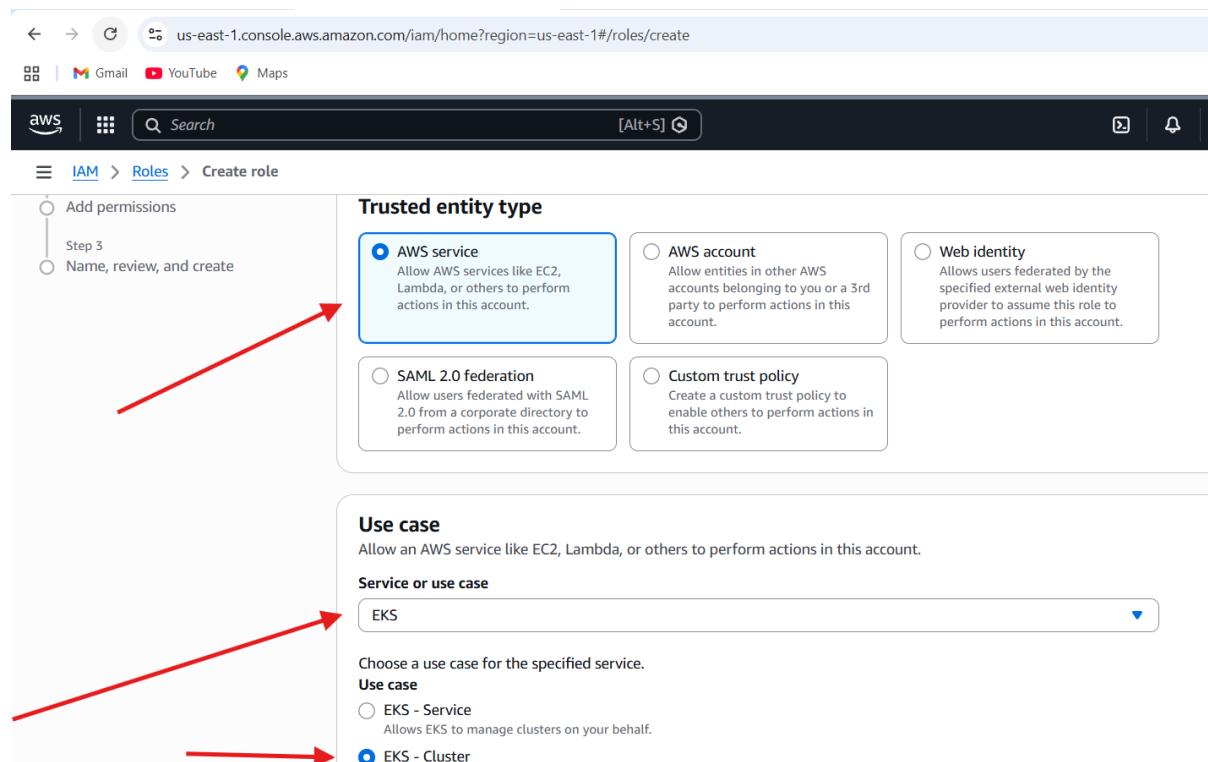
Go to eks service you find that a cluster has been created.

The screenshot shows the AWS Elastic Kubernetes Service (EKS) console. On the left, there is a navigation sidebar with options like 'Clusters', 'Settings', 'Amazon EKS Anywhere', and 'Related services'. A red arrow points from the 'Clusters' link in the sidebar to the main content area. The main content area displays a table titled 'Clusters (1)'. The table has columns for 'Cluster name', 'Status', 'Kubernetes version', 'Support period', 'Upgrade policy', 'Created', and 'Provider'. The single cluster listed is 'my-eks-cluster', which is 'Active', running '1.32', and has a support period 'Standard support until March 23, 2026'. The provider is listed as 'EKS'.

Clusters (1)						
Cluster name	Status	Kubernetes version	Support period	Upgrade policy	Created	Provider
my-eks-cluster	Active	1.32	Standard support until March 23, 2026	Extended support	20 minutes ago	EKS

2. setup eks cluster using console

Create an IAM Role for EKS Cluster



Trusted entity type

- AWS service**
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account**
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity**
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

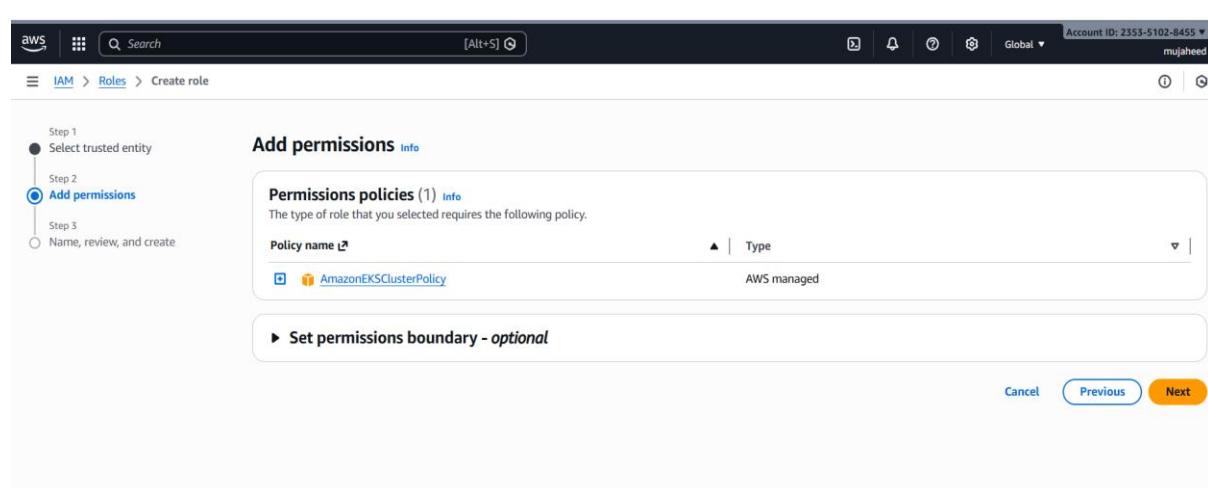
Service or use case

EKS

Choose a use case for the specified service.

Use case

- EKS - Service**
Allows EKS to manage clusters on your behalf.
- EKS - Cluster**
Allows the cluster Kubernetes control plane to manage AWS resources on your behalf.



Add permissions

Permissions policies (1)

The type of role that you selected requires the following policy.

Policy name

- AmazonEKSClusterPolicy**

Set permissions boundary - optional

Cancel Previous Next

The screenshot shows the final step of creating an IAM role. The navigation bar at the top includes the AWS logo, a search bar, and a 'Create role' button. On the left, a vertical navigation pane lists 'Step 1 Select trusted entity', 'Step 2 Add permissions', and 'Step 3 Name, review, and create', with the third step being the current active tab. The main content area is titled 'Name, review, and create'. It contains a 'Role details' section where the 'Role name' is set to 'eks-cluster-role' and the 'Description' is 'Allows the cluster Kubernetes control plane to manage AWS resources on your behalf.' Below this, a note specifies a maximum of 1000 characters for the description. A 'Step 1: Select trusted entities' section is shown, with a 'Trust policy' link.

Then click on create

Create an IAM Role for Worker Nodes:

The screenshot shows the first step of creating an IAM role, 'Trusted entity type'. The navigation bar and sidebar are identical to the previous screenshot. The main content area is titled 'Trusted entity type'. It displays four options: 'AWS service' (selected), 'AWS account', 'SAML 2.0 federation', and 'Custom trust policy'. Below this, a 'Use case' section allows selecting a service or use case, with 'EC2' selected. A note at the bottom states: 'Allow EC2 instances to call AWS services on your behalf.'

Attach these 3 policies

- **AmazonEKSTrustedNodePolicy**
- **AmazonEKS_CNI_Policy**
- **AmazonEC2ContainerRegistryReadOnly**

Screenshot of the AWS IAM 'Create role' wizard Step 2: Add permissions. The 'Permissions policies' section shows the 'AmazonEC2ContainerRegistryReadOnly' policy selected. A red arrow points to this section.

Screenshot of the AWS IAM 'Create role' wizard Step 3: Name, review, and create. The 'Role name' field contains 'eks-nodegroup-role'. A red arrow points to this field.

Go to eks and create cluster

Screenshot of the AWS EKS service page. The 'Get started' section features a prominent 'Create cluster' button. A red arrow points to this button.

The screenshot shows the 'Configure cluster' step of the EKS cluster creation wizard. The sidebar on the left lists steps 1 through 6. The main area contains sections for 'Configuration options - new' (with 'Quick configuration (with EKS Auto Mode)' selected), 'EKS Auto Mode - new' (with 'Use EKS Auto Mode' selected), and 'Cluster configuration'.

- Step 1 **Configure cluster**
- Step 2 Specify networking
- Step 3 Configure observability
- Step 4 Select add-ons
- Step 5 Configure selected add-ons settings
- Step 6 Review and create

Configure cluster

Configuration options - new Info

Choose how you would like to configure the cluster.

Quick configuration (with EKS Auto Mode) - new

Quickly create a cluster with production-grade default settings. The configuration uses EKS Auto Mode to automate infrastructure tasks like creating nodes and provisioning storage.

Custom configuration

To change default settings prior to creation, choose this option to use EKS Auto Mode and customize the cluster configuration.

EKS Auto Mode - new Info

Choose if you would like to use EKS's Auto Mode.

Use EKS Auto Mode

EKS automates routine cluster tasks for compute, storage, and networking. When a new pod can't fit onto existing nodes, EKS creates a new node. EKS combines cluster infrastructure managed by AWS with integrated Kubernetes capabilities to meet application compute needs. [View pricing ↗](#)

► Included capabilities

Cluster configuration Info



Search

[Alt+S] Ask Amazon Q



United States (N. Virginia) ▾

Amazon Elastic Kubernetes Service > Create EKS cluster

application compute needs. [View pricing ↗](#)

► Included capabilities

Cluster configuration Info

Name

Enter a unique name for this cluster. This property cannot be changed after the cluster is created.

my-eks-cluster1

The cluster name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of

Cluster IAM role Info

Select the Cluster IAM role to allow the Kubernetes control plane to manage AWS resources on your behalf. This cannot be changed after the cluster is created. To create a new role, follow the instructions in the [Amazon EKS User Guide ↗](#).

eks-cluster-role



Create recommendation

⚠ **Cluster role missing recommended managed policies**

The cluster role must have the following managed policies or equivalent permissions to use EKS Auto Mode:

- AmazonEKSServiceRoleForContainer
- AmazonEKSCognitoIdentityProviderRole
- AmazonEKSLoadBalancingPolicy
- AmazonEKSNetworkingPolicy



Search

[Alt+S] Ask Amazon Q



United States (N. Virginia) ▾

Amazon Elastic Kubernetes Service > Create EKS cluster

Kubernetes version settings

Kubernetes version Info

Select Kubernetes version for this cluster.

1.34



Upgrade policy Info

Choose one of the following options. You can switch the setting later while the standard support period is in effect.

Standard support

This option supports the Kubernetes version for 14 months after the release date. There is no additional cost. When standard support ends, your cluster will be auto-upgraded to the next version.

Extended support

This option supports the Kubernetes version for 26 months after the release date. The extended support period has an additional hourly cost that begins after the standard support period ends. When extended support ends, your cluster will be auto-upgraded to the next version.

Control plane scaling tier - new Info

Select a scaling tier for your control plane. While Standard mode scales dynamically, higher tiers pre-provision your environment with fixed, high-performance nodes to eliminate scaling latency and provides the sustained throughput required for demanding workloads. [View tier pricing ↗](#)

Use a scaling tier

For predictable and high performance from your cluster's control plane, choose a scaling tier.

aws | Search [Alt+S] Ask Amazon Q United States (N. Virginia)

Amazon Elastic Kubernetes Service > Create EKS cluster

For predictable and high performance from your cluster's control plane, choose a scaling tier.

Auto Mode Compute - new [Info](#)

Configure node management for your EKS cluster. EKS offers four compute options: EKS Auto Mode, EC2 Managed Node Groups, Fargate, and hybrid profiles, and hybrid nodes are configured after cluster creation. You can also create self-managed nodes.

Built-in node pools - optional [Info](#)

EKS Auto Mode uses node pools to create nodes for pods. The node IAM role will be associated with built-in node pools. Use the Kubernetes API after cluster creation to create and manage your own node pools.

Choose node pool(s) [general-purpose](#) [system](#)

Node IAM role [Info](#)

Nodes need an EC2 Instance IAM Role to launch and register with a cluster. To create a new custom role, follow the instructions in the [Amazon EKS User Guide](#).

eksctl-my-eks-cluster-nodegroup-[ng-NodeInstanceRole-rbOYt1e7kA48](#) [Create recommended](#)

Cluster access [Info](#)

Control how IAM principals can access this cluster.

Bootstrap cluster administrator access [Info](#)

Choose whether the IAM principal creating the cluster has Kubernetes cluster administrator access.

Allow cluster administrator access Allow cluster administrator access for your IAM principal.

Disallow cluster administrator access Disallow cluster administrator access for your IAM principal

aws | Search [Alt+S] Ask Amazon Q United States (N. Virginia)

Amazon Elastic Kubernetes Service > Create EKS cluster

Cluster access [Info](#)

Control how IAM principals can access this cluster.

Bootstrap cluster administrator access [Info](#)

Choose whether the IAM principal creating the cluster has Kubernetes cluster administrator access.

Allow cluster administrator access Allow cluster administrator access for your IAM principal.

Disallow cluster administrator access Disallow cluster administrator access for your IAM principal

Cluster authentication mode [Info](#)

Configure which source the cluster will use for authenticated IAM principals.

EKS API The cluster will source authenticated IAM principals only from EKS access entry APIs.

EKS API and ConfigMap The cluster will source authenticated IAM principals from both EKS access entry APIs and the aws-auth ConfigMap.

Envelope encryption [Info](#)

Envelope encryption is applied to all Kubernetes API data.

By default, AWS implements envelope encryption using an AWS owned key. Alternatively, you can setup your own customer managed key by providing the CMK ARN when configuring your EKS cluster.

Use your own AWS KMS key After a cluster is created, you can migrate from using an AWS owned key to a customer managed key (CMK), but not vice versa.

aws | Search [Alt+S] Ask Amazon Q United States (N. Virginia)

Amazon Elastic Kubernetes Service > Create EKS cluster

Use your own AWS KMS key After a cluster is created, you can migrate from using an AWS owned key to a customer managed key (CMK), but not vice versa.

ARC Zonal shift [Info](#)

Shift application traffic away from an impaired Availability Zone (AZ) in your EKS cluster. You can change this later.

Enabled EKS will register your cluster with ARC zonal shift to enable you to use zonal shift to shift application traffic away from an AZ.

Disabled EKS will not register your cluster with ARC zonal shift

Before you start a zonal shift, you need to setup your cluster environment to be resilient to an AZ failure beforehand.

Deletion protection

Deletion protection must be turned off to be able to delete a cluster. It can be turned on and off after the cluster is created.

Turn on deletion protection Deletion protection provides additional security against accidental cluster deletion.

Tags (0) [Info](#)

Click on next

The screenshot shows the 'Specify networking' step of the EKS cluster creation wizard. On the left, a vertical navigation bar lists six steps: Step 1 (Configure cluster), Step 2 (Specify networking, which is selected and highlighted in blue), Step 3 (Configure observability), Step 4 (Select add-ons), Step 5 (Configure selected add-ons settings), and Step 6 (Review and create). The main content area is titled 'Specify networking'. It contains a 'Networking' section with a note that IP address family and service IP address range cannot be changed after cluster creation. A dropdown menu shows 'vpc-06cf45eaab13624fe | Default'. Below it is a 'Subnets' section where two subnets are selected: 'subnet-04f12a817188fcadc | default-private-subnet' and 'subnet-0a192382de0e2bf6a | default-public-subnet'. At the bottom, there's an 'Additional security groups' section with a note about automatically created cluster security groups.

The screenshot shows the 'Cluster endpoint access' step of the EKS cluster creation wizard. The navigation bar on the left shows the user has moved to Step 3 ('Configure observability'). The main content area is titled 'Cluster endpoint access'. It contains a note about configuring access to the Kubernetes API server endpoint. There are three radio button options: 'Public' (selected), 'Public and private', and 'Private'. Each option has a detailed description below it. Below these options is a 'Advanced settings' section indicated by a right-pointing arrow.

Screenshot of the AWS EKS Create EKS cluster step 3: Select add-ons. The 'Control plane logs' section is shown, with several checkboxes for log types: API server, Audit, Scheduler, and others. The 'Scheduler' checkbox is selected.

Select addons that you are needed

Screenshot of the AWS EKS Create EKS cluster step 4: Select add-ons. The 'AWS add-ons (21)' section is displayed, showing three selected add-ons: Amazon VPC CNI, kube-proxy, and Node monitoring agent. Other options like CloudWatch Metrics and CloudWatch Metrics Insights are also listed.

Screenshot of the AWS EKS Create EKS cluster step 5: Review and create. The 'Step 1: Cluster' configuration is reviewed, including the cluster name (my-eks-cluster1), Kubernetes version (1.34), EKS Auto Mode (Enabled), Cluster IAM role (arn:awsiam::235351028455:role/eks-cluster-role), and other settings like Kubernetes cluster administrator access and node IAM role.

Click on create it will create a cluster in 10 min

The screenshot shows the AWS Lambda interface. At the top, there's a search bar and navigation links for 'Clusters' and 'my-eks-cluster'. On the left, a sidebar lists 'Amazon Elastic Kubernetes Service' settings, 'Amazon EKS Anywhere' enterprise subscriptions, and 'Related services' like Amazon ECR and AWS Batch. The main content area displays the 'my-eks-cluster' details. It includes a warning about the cluster's Kubernetes version reaching end-of-life support on March 23, 2026. Below this, the 'Cluster info' section provides status (Active), Kubernetes version (1.32), support period (Standard support until March 23, 2026), and provider (EKS). It also shows cluster health (0 issues), upgrade insights (5 pending), node health issues (0), and capability issues (0). A 'Details' section at the bottom shows the API server endpoint (<https://A2FAC1476F490B266BE92A2C25D636CD.g>), OpenID Connect provider URL (<https://oidc.eks.us-east-1.amazonaws.com/id/A2FA>), and creation time (an hour ago).

3. Setup HPA

- vi hpa-app.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: hpa-nginx

spec:

replicas: 1

selector:

matchLabels:

app: hpa-nginx

template:

metadata:

labels:

app: hpa-nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

resources:

requests:

cpu: "100m"

memory: "128Mi"

limits:

cpu: "200m"

memory: "256Mi"

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: hpa-nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hpa-nginx
  template:
    metadata:
      labels:
        app: hpa-nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
          resources:
            requests:
              cpu: "100m"
              memory: "128Mi"
            limits:
              cpu: "200m"
              memory: "256Mi"

```

- **kubectl apply -f hpa-app.yaml**
- **kubectl get pods**
- **kubectl top pods**

```

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi hpa-app.yaml

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f hpa-app.yaml
deployment.apps/hpa-nginx created

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
hpa-nginx-f86cfcc9d-71rp7   1/1     Running   0          12s

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl top pods
NAME                  CPU(cores)   MEMORY(bytes)
hpa-nginx-f86cfcc9d-71rp7   0m          3Mi

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ |

```

- **vi service.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: hpa-nginx-service
spec:
  selector:
    app: hpa-nginx
  ports:
    - port: 80
      targetPort: 80
```

```
apiVersion: v1
kind: Service
metadata:
  name: hpa-nginx-service
spec:
  selector:
    app: hpa-nginx
  ports:
    - port: 80
      targetPort: 80
```

- **kubectl apply -f service.yaml**

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi service.yaml

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f service.yaml
service/hpa-nginx-service created

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ |
```

- vi hpa.yaml

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-nginx
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: hpa-nginx
  minReplicas: 1
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
    target:
      type: Utilization
      averageUtilization: 50
```

```

apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-nginx
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: hpa-nginx
  minReplicas: 1
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
    target:
      type: Utilization
      averageUtilization: 50

```

- **kubectl apply -f hpa.yaml**
- **kubectl get hpa**

```

MUJUU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi hpa.yaml

MUJUU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f hpa.yaml
horizontalpodautoscaler.autoscaling/hpa-nginx created

MUJUU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get hpa
NAME      REFERENCE          TARGETS           MINPODS   MAXPODS   REPLICAS   AGE
hpa-nginx Deployment/hpa-nginx  cpu: <unknown>/50%  1          5          0          10s

MUJUU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get hpa
NAME      REFERENCE          TARGETS           MINPODS   MAXPODS   REPLICAS   AGE
hpa-nginx Deployment/hpa-nginx  cpu: 0%/50%     1          5          1          34s

MUJUU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ |

```

4. Setup cluster autoscale

- **kubectl get nodes**

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get nodes
NAME                      STATUS   ROLES      AGE     VERSION
ip-192-168-28-220.ec2.internal   Ready    <none>    120m   v1.32.9-eks-ecaa3a6
ip-192-168-41-142.ec2.internal   Ready    <none>    120m   v1.32.9-eks-ecaa3a6
```

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ |
```

- **vi cluster-autoscaler-policy.json**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "autoscaling:DescribeAutoScalingGroups",
        "autoscaling:DescribeAutoScalingInstances",
        "autoscaling:DescribeLaunchConfigurations",
        "autoscaling:DescribeScalingActivities",
        "autoscaling:SetDesiredCapacity",
        "autoscaling:TerminateInstanceInAutoScalingGroup",
        "ec2:DescribeLaunchTemplateVersions"
      ],
      "Resource": "*"
    }
  ]
}
```

- **aws iam create-policy \
--policy-name AmazonEKSClusterAutoscalerPolicy \
--policy-document file://cluster-autoscaler-policy.json**

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi cluster-autoscaler-policy.json

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ aws iam create-policy \
> --policy-name AmazonEKSClusterAutoscalerPolicy \
> --policy-document file://cluster-autoscaler-policy.json
{
  "Policy": {
    "PolicyName": "AmazonEKSClusterAutoscalerPolicy",
    "PolicyId": "ANPATNTADWL TWQDCMOSKO",
    "Arn": "arn:aws:iam::235351028455:policy/AmazonEKSClusterAutoscalerPolicy",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2026-01-07T11:51:11+00:00",
    "UpdateDate": "2026-01-07T11:51:11+00:00"
  }
}

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ |
```

Get OIDC provider:

```
eksctl utils associate-iam-oidc-provider \
```

```
--region us-east-1 \
```

```
--cluster my-eks-cluster \
```

```
--approve
```

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ eksctl utils associate-iam-oidc-provider \
>   --region us-east-1 \
>   --cluster my-eks-cluster \
>   --approve
2026-01-07 17:27:26 [i] will create IAM Open ID Connect provider for cluster "my-eks-cluster" in "us-east-1"
2026-01-07 17:27:27 [v] created IAM Open ID Connect provider for cluster "my-eks-cluster" in "us-east-1"
```

```
aws eks describe-cluster \
```

```
--name my-eks-cluster \
```

```
--query "cluster.identity.oidc.issuer" \
```

```
--output text
```

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ aws eks describe-cluster \
>   --name my-eks-cluster \
>   --query "cluster.identity.oidc.issuer" \
>   --output text
https://oidc.eks.us-east-1.amazonaws.com/id/A2FAC1476F490B266BE92A2C25D636CD
```

```
eksctl create iamserviceaccount \
```

```
--name cluster-autoscaler \
```

```
--namespace kube-system \
```

```
--cluster my-eks-cluster \
```

```
--attach-policy-arn
```

```
arn:aws:iam::235351028455:policy/AmazonEKSClusterAutos
calerPolicy \
```

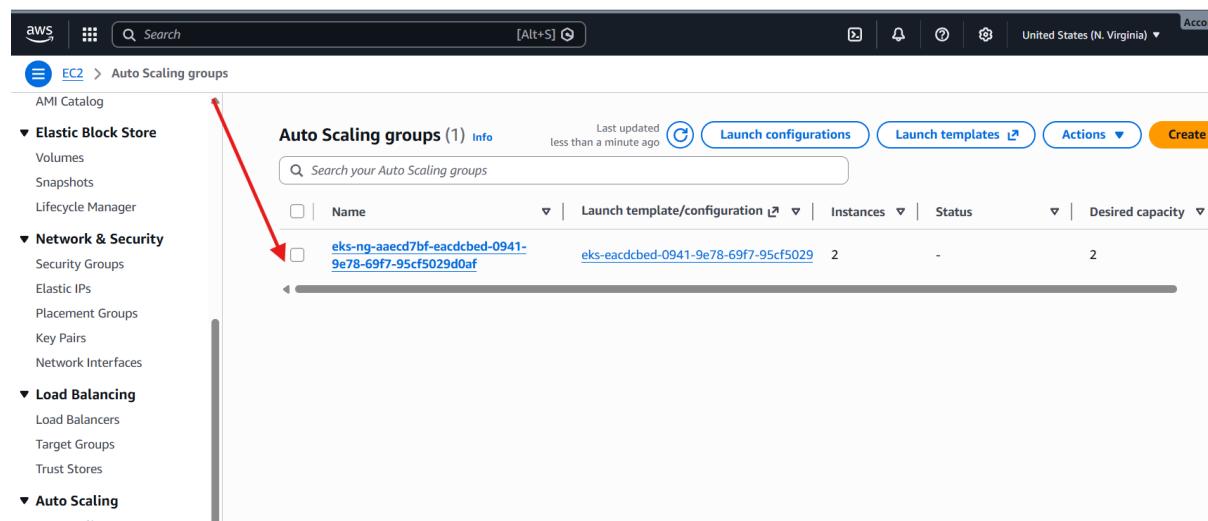
--approve \

--override-existing-serviceaccounts

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ eksctl create iamserviceaccount \
>   --name cluster-autoscaler \
>   --namespace kube-system \
>   --cluster my-eks-cluster \
>   --attach-policy-arn arn:aws:iam::235351028455:policy/AmazonEKSClusterAutoscalerPolicy \
>   --approve \
>   --override-existing-serviceaccounts
2026-01-07 17:29:23 [!] 1 iamserviceaccount (kube-system/cluster-autoscaler) was included (based on the
2026-01-07 17:29:23 [!] metadata of serviceaccounts that exist in Kubernetes will be updated, as --overr
2026-01-07 17:29:23 [!] 1 task: {
  2 sequential sub-tasks: {
    create IAM role for serviceaccount "kube-system/cluster-autoscaler",
    create serviceaccount "kube-system/cluster-autoscaler",
  }
} 2026-01-07 17:29:23 [!] building iamserviceaccount stack "eksctl-my-eks-cluster-addon-iamserviceac
2026-01-07 17:29:23 [!] deploying stack 'eksctl-my-eks-cluster-addon-iamserviceaccount-kube-system-clust
2026-01-07 17:29:24 [!] waiting for CloudFormation stack "eksctl-my-eks-cluster-addon-iamserviceaccount-
2026-01-07 17:29:55 [!] waiting for CloudFormation stack "eksctl-my-eks-cluster-addon-iamserviceaccount-
2026-01-07 17:29:57 [!] created serviceaccount "kube-system/cluster-autoscaler"

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
```

Replace your autoscaling group name in your script



- vi cluster-autoscaler.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: cluster-autoscaler

namespace: kube-system

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: cluster-autoscaler
  template:
    metadata:
      labels:
        app: cluster-autoscaler
      annotations:
        cluster-autoscaler.kubernetes.io/safe-to-evict: "false"
  spec:
    serviceAccountName: cluster-autoscaler
    containers:
      - name: cluster-autoscaler
        image: registry.k8s.io/autoscaling/cluster-
autoscaler:v1.29.0
    command:
      - ./cluster-autoscaler
      - --cloud-provider=aws
      - --balance-similar-node-groups
```

- **--skip-nodes-with-system-pods=false**
- **--skip-nodes-with-local-storage=false**
- **--nodes=1:5: eks-
ng-aaecd7bf-eacdcbed-0941-9e78-69f7-95cf5029d0af**

- **--stderrthreshold=info**

- **--v=4**

resources:

limits:

cpu: 100m

memory: 300Mi

requests:

cpu: 100m

memory: 300Mi

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cluster-autoscaler
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: cluster-autoscaler
  template:
    metadata:
      labels:
        app: cluster-autoscaler
      annotations:
        cluster-autoscaler.kubernetes.io/safe-to-evict: "false"
    spec:
      serviceAccountName: cluster-autoscaler
      containers:
        - name: cluster-autoscaler
          image: registry.k8s.io/autoscaling/cluster-autoscaler:v1.29.0
          command:
            - ./cluster-autoscaler
            - --cloud-provider=aws
            - --balance-similar-node-groups
            - --skip-nodes-with-system-pods=false
            - --skip-nodes-with-local-storage=false
            - --nodes=1:5:eks-  
ng-aaecd7bf-eacdcbed-0941-9e78-69f7-95cf5029d0af
            - --stderrthreshold=info
            - --v=4
          resources:
            limits:
              cpu: 100m
              memory: 300Mi
            requests:
              cpu: 100m
              memory: 300Mi
```

- **kubectl apply -f cluster-autoscaler.yaml**

```
MUJGU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi cluster-autoscaler.yaml

MUJGU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f cluster-autoscaler.yaml
deployment.apps/cluster-autoscaler created

MUJGU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
```

- **vi cluster-autoscaler-rbac.yaml**

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-autoscaler
rules:
- apiGroups: []
  resources: ["events", "endpoints"]
  verbs: ["create", "patch"]
- apiGroups: []
  resources: ["pods/eviction"]
  verbs: ["create"]
- apiGroups: []
  resources: ["pods/status"]
  verbs: ["update"]
- apiGroups: []
  resources: ["endpoints"]
  resourceNames: ["cluster-autoscaler"]
  verbs: ["get", "update"]
- apiGroups: []
  resources: ["nodes"]
  verbs: ["watch", "list", "get", "update"]
- apiGroups: []
  resources: ["pods", "services", "replicationcontrollers", "persistentvolumeclaims", "pods"]
  verbs: ["watch", "list", "get"]
- apiGroups: ["apps"]
  resources: ["statefulsets", "replicasets", "daemonsets", "deployments"]
  verbs: ["watch", "list", "get"]
- apiGroups: ["batch", "extensions"]
  resources: ["jobs"]
  verbs: ["watch", "list", "get"]
- apiGroups: ["policy"]
  resources: ["poddisruptionbudgets"]
  verbs: ["watch", "list"]
```

- **kubectl apply -f cluster-autoscaler-rbac.yaml**
- **kubectl rollout restart deployment cluster-autoscaler -n kube-system**
- **kubectl get pods -n kube-system | grep cluster-autoscaler**

- **kubectl logs -n kube-system deployment/cluster-autoscaler**

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi cluster-autoscaler-rbac.yaml

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f cluster-autoscaler-rbac.yaml
clusterrole.rbac.authorization.k8s.io/cluster-autoscaler created
clusterrolebinding.rbac.authorization.k8s.io/cluster-autoscaler created

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl rollout restart deployment cluster-autoscaler -n kube-system
deployment.apps/cluster-autoscaler restarted

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get pods -n kube-system | grep cluster-autoscaler
cluster-autoscaler-bf5bbcc5-kz48m 1/1 Running 0 14s
```

```
$ kubectl logs -n kube-system deployment/cluster-autoscaler
I0107 12:10:36.428462      1 main.go:595] Cluster Autoscaler 1.29.0
I0107 12:10:36.522544      1 leaderelection.go:250] attempting to acquire leader lease kube-system/cluster-autoscaler
E0107 12:10:36.527199      1 leaderelection.go:332] error retrieving resource lock kube-system/cluster-autoscaler" is forbidden: user "system:serviceaccount:kube-system:cluster-autoscaler" cannot get resource "leases" in the namespace "kube-system"
I0107 12:10:36.527218      1 leaderelection.go:255] failed to acquire lease kube-system/cluster-autoscaler
E0107 12:10:40.749574      1 leaderelection.go:332] error retrieving resource lock kube-system/cluster-autoscaler" is forbidden: user "system:serviceaccount:kube-system:cluster-autoscaler" cannot get resource "leases" in the namespace "kube-system"
I0107 12:10:40.749620      1 leaderelection.go:255] failed to acquire lease kube-system/cluster-autoscaler
E0107 12:10:42.809804      1 leaderelection.go:332] error retrieving resource lock kube-system/cluster-autoscaler" is forbidden: user "system:serviceaccount:kube-system:cluster-autoscaler" cannot get resource "leases" in the namespace "kube-system"
I0107 12:10:42.809917      1 leaderelection.go:255] failed to acquire lease kube-system/cluster-autoscaler
E0107 12:10:45.786321      1 leaderelection.go:332] error retrieving resource lock kube-system/cluster-autoscaler" is forbidden: User "system:serviceaccount:kube-system:cluster-autoscaler" cannot get resource "leases" in the namespace "kube-system"
I0107 12:10:45.786347      1 leaderelection.go:255] failed to acquire lease kube-system/cluster-autoscaler
E0107 12:10:48.808456      1 leaderelection.go:332] error retrieving resource lock kube-system/cluster-autoscaler" is forbidden: User "system:serviceaccount:kube-system:cluster-autoscaler" cannot get resource "leases" in the namespace "kube-system"
I0107 12:10:48.808476      1 leaderelection.go:255] failed to acquire lease kube-system/cluster-autoscaler
E0107 12:10:52.593628      1 leaderelection.go:332] error retrieving resource lock kube-system/cluster-autoscaler" is forbidden: User "system:serviceaccount:kube-system:cluster-autoscaler" cannot get resource "leases" in the namespace "kube-system"
```

Test the scaling:

- **vi test-pod.yaml**

apiVersion: v1

kind: Pod

metadata:

name: cpu-stress

spec:

containers:

- name: stress

image: busybox

command: ["sh", "-c", "sleep 3600"]

resources:

requests:

cpu: "2000m"

```
apiVersion: v1
kind: Pod
metadata:
  name: cpu-stress
spec:
  containers:
    - name: stress
      image: busybox
      command: ["sh", "-c", "sleep 3600"]
      resources:
        requests:
          cpu: "2000m"
```

- **kubectl apply -f test-pod.yaml**

if pods are pending state it will create new nodes

- **kubectl get pods -w**
- **kubectl get nodes -w**

```

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi test-pod.yaml

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f test-pod.yaml
pod/cpu-stress created

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get nodes -w
NAME           STATUS    ROLES
ip-192-168-28-220.ec2.internal   Ready
ip-192-168-41-142.ec2.internal   Ready

          NAME          READY   STATUS    RESTARTS   AGE
cpu-stress   0/1     Pending   0          54s
hpa-nginx-f86cfcc9d-7lrp7       1/1     Running   0          70m

```

5. Setup cronjob and job

- **vi job.yaml**

apiVersion: batch/v1

kind: Job

metadata:

name: sample-job

spec:

backoffLimit: 3

completions: 1

parallelism: 1

template:

spec:

restartPolicy: Never

containers:

```
- name: job-container
  image: busybox
  command:
    - sh
    - -c
    - echo "Job started"; sleep 10; echo "Job completed"
```

```
apiVersion: batch/v1
kind: Job
metadata:
  name: sample-job
spec:
  backoffLimit: 3
  completions: 1
  parallelism: 1
  template:
    spec:
      restartPolicy: Never
      containers:
        - name: job-container
          image: busybox
          command:
            - sh
            - -c
            - echo "Job started"; sleep 10; echo "Job completed"
```

- **kubectl apply -f job.yaml**

```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi job.yaml
```

```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f job.yaml
job.batch/sample-job created
```

```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$
```

- **kubectl get jobs**

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f job.yaml
job.batch/sample-job created

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get jobs
NAME      STATUS   COMPLETIONS DURATION   AGE
sample-job Complete  1/1          14s        2m16s
```

- **kubectl logs job/sample-job**

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl logs job/sample-job
Job started
Job completed
```

Cronjob:

- **vi cronjob.yaml**

apiVersion: batch/v1

kind: CronJob

metadata:

name: sample-cronjob

spec:

schedule: "*/2 * * * *" # Every 2 minutes

successfulJobsHistoryLimit: 3

failedJobsHistoryLimit: 1

jobTemplate:

spec:

template:

spec:

restartPolicy: Never

containers:

- name: cron-container

image: busybox

command:

- sh

- -c

- date; echo "CronJob executed"

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: sample-cronjob
spec:
  schedule: "*/2 * * * *"    # Every 2 minutes
  successfulJobsHistoryLimit: 3
  failedJobsHistoryLimit: 1
  jobTemplate:
    spec:
      template:
        spec:
          restartPolicy: Never
          containers:
            - name: cron-container
              image: busybox
              command:
                - sh
                - -c
                - date; echo "CronJob executed"
```

- **kubectl apply -f cronjob.yaml**

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi cronjob.yaml

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f cronjob.yaml
cronjob.batch/sample-cronjob created

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ |
```

- **kubectl get cronjobs**
- **kubectl get cronjobs**

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get cronjobs
NAME      SCHEDULE      TIMEZONE    SUSPEND   ACTIVE   LAST SCHEDULE   AGE
sample-cronjob  */2 * * * *  <none>    False     0        39s      105s

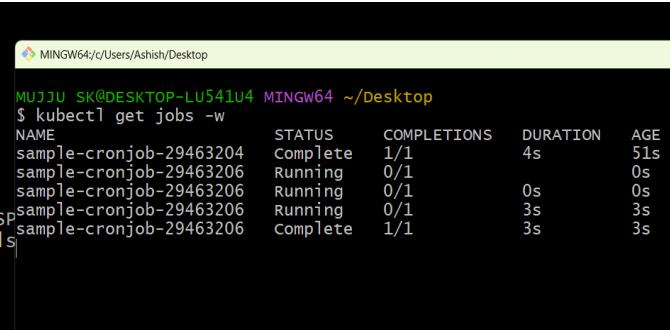
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ |
```

- **kubectl get jobs -w**

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi cronjob.yaml

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f cronjob.yaml
cronjob.batch/sample-cronjob created

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get cronjobs
NAME      SCHEDULE      TIMEZONE    SUSPEN
sample-cronjob  */2 * * * *  <none>    False


```

6. Create secret and inject inside pod

```
kubectl create secret generic app-secret \  
--from-literal=DB_USER=admin \  
--from-literal=DB_PASSWORD=admin123
```

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop  
$ kubectl create secret generic app-secret \  
> --from-literal=DB_USER=admin \  
> --from-literal=DB_PASSWORD=admin123  
secret/app-secret created
```

- **kubectl get secret app-secret**
- **kubectl describe secret app-secret**

```
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop  
$ kubectl get secret app-secret  
NAME      TYPE      DATA   AGE  
app-secret  Opaque    2      17s  
  
MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop  
$ kubectl describe secret app-secret  
Name:            app-secret  
Namespace:       default  
Labels:          <none>  
Annotations:     <none>  
  
Type:  Opaque  
  
Data  
====  
DB_USER:        5 bytes  
DB_PASSWORD:    8 bytes
```

- **vi secret-env-pod.yaml**

```
apiVersion: v1
```

kind: Pod

metadata:

name: secret-env-pod

spec:

containers:

- name: test-container

image: busybox

command: ["sh", "-c", "env | grep DB && sleep 3600"]

env:

- name: DB_USER

valueFrom:

secretKeyRef:

name: app-secret

key: DB_USER

- name: DB_PASSWORD

valueFrom:

secretKeyRef:

name: app-secret

key: DB_PASSWORD

```

apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: test-container
    image: busybox
    command: ["sh", "-c", "env | grep DB && sleep 3600"]
    env:
      - name: DB_USER
        valueFrom:
          secretKeyRef:
            name: app-secret
            key: DB_USER
      - name: DB_PASSWORD
        valueFrom:
          secretKeyRef:
            name: app-secret
            key: DB_PASSWORD

```

- **kubectl apply -f secret-env-pod.yaml**
- **kubectl logs secret-env-pod**

```

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ vi secret-env-pod.yaml

```

```

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl apply -f secret-env-pod.yaml
pod/secret-env-pod created

```

```

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl logs secret-env-pod
DB_PASSWORD=admin123
DB_USER=admin

```

```

MUJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ ...

```

7. Check different add-ons available on eks.

- Amazon VPC CNI
- CoreDNS
- kube-proxy
- Amazon EBS CSI Driver
- Metrics Server
- Amazon EFS CSI Driver
- AWS Load Balancer Controller
- Amazon GuardDuty Agent
- Amazon CloudWatch Observability
- Amazon VPC Lattice Controller

The screenshot shows the AWS EKS Add-ons page for the cluster 'my-eks-cluster'. It displays two add-ons: CoreDNS and Metrics Server, both in an active status. The CoreDNS add-on is for service discovery, and the Metrics Server add-on is for collecting cluster-wide resource usage data.

Add-on	Description	Category	Status	Version	EKS Pod Identity	IAM role for service account (IRSA)
CoreDNS	Enable service discovery within your cluster.	networking	Active	v1.11.4-eksbuild.2	Not required	Not required
Metrics Server	Install metrics-server to collect cluster-wide resource usage data for autoscaling and monitoring.	observability	Active	v0.8.0-eksbuild.6	Not required	Not required

The screenshot shows the AWS EKS Add-ons page for the cluster 'my-eks-cluster'. It displays two add-ons: kube-proxy and Amazon VPC CNI, both in an active status. The kube-proxy add-on is for service networking, and the Amazon VPC CNI add-on is for pod networking.

Add-on	Description	Category	Status	Version	EKS Pod Identity	IAM role for service account (IRSA)
kube-proxy	Enable service networking within your cluster.	networking	Active	v1.32.6-eksbuild.12	Not required	Not required
Amazon VPC CNI	Enable pod networking within your cluster.	networking	Active	v1.20.4-eksbuild.2	Not set	Not set

8. Upgrade eks cluster

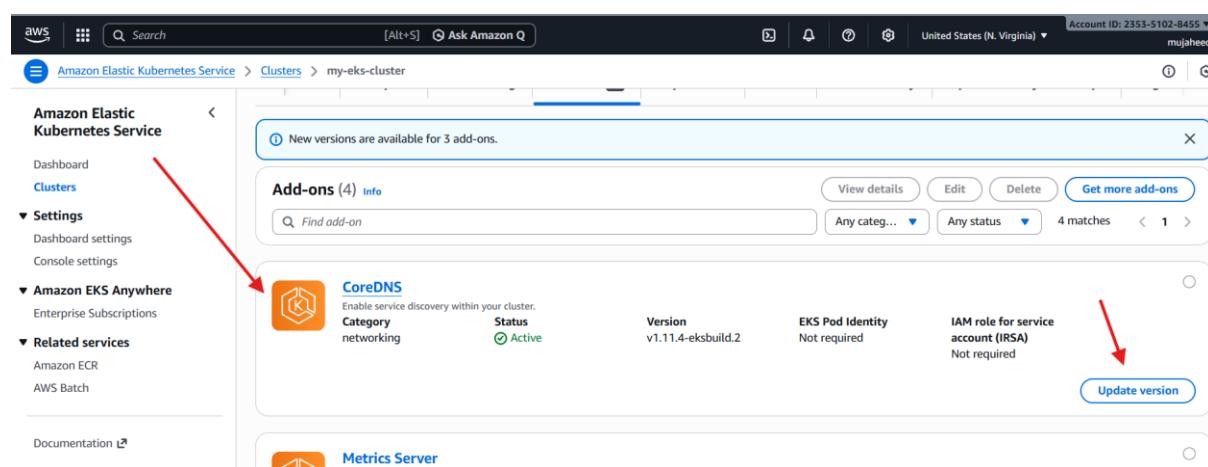
- **aws eks describe-cluster **
**--name my-eks-cluster **
--query cluster.version
- **kubectl get nodes**

```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl get nodes
NAME                               STATUS   ROLES      AGE      VERSION
ip-192-168-28-220.ec2.internal   Ready    <none>    3h58m   v1.32.9-eks-ecaa3a6
ip-192-168-41-142.ec2.internal   Ready    <none>    3h58m   v1.32.9-eks-ecaa3a6

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ aws eks describe-cluster \
>   --name my-eks-cluster \
>   --query cluster.version
"1.32"
```

Go to eks

- **select on your cluster**
- **select addons**
- **upgrade each add on with latest version**



aws Search [Alt+S] Ask Amazon Q Account ID: 2353-5102-8455 mujaheed

Amazon Elastic Kubernetes Service > Clusters > my-eks-cluster > coredns > Edit add-on

Configure CoreDNS

CoreDNS Info

Listed by	Category	Status
	networking	Active

Version
Select the version for this add-on.
v1.11.4-eksbuild.24

▶ Optional configuration settings

Cancel Save changes

aws Search [Alt+S] Ask Amazon Q United States (N. Virginia)

Amazon Elastic Kubernetes Service > Clusters > my-eks-cluster > coredns > e0e3d2af-5e36-3783-9ae0-0130881078a8

Amazon Elastic Kubernetes Service

- Dashboard
- Clusters**
- Settings
- Amazon EKS Anywhere
- Related services
- Amazon ECR
- AWS Batch
- Documentation ↗

Update ID: e0e3d2af-5e36-3783-9ae0-0130881078a8

General configuration

Update ID e0e3d2af-5e36-3783-9ae0-0130881078a8	AddonVersion v1.11.4-eksbuild.24
Status In progress	ResolveConflicts NONE
Type AddonUpdate	PodIdentityAssociations []

Errors (0)

Error code	Error message	Resource IDs
No errors		

The coredns add-on is now being updated. This process may take several minutes.

kube-proxy
Enable service networking within your cluster.

Category networking	Status Active	Version v1.32.6-eksbuild.12	EKS Pod Identity Not required	IAM role for service account (IRSA) Not required
------------------------	------------------	--------------------------------	----------------------------------	---

Update version

Amazon VPC CNI
Enable pod networking within your cluster.

Category networking	Status Active	Version v1.20.4-eksbuild.2	EKS Pod Identity Not set	IAM role for service account (IRSA) Not set
------------------------	------------------	-------------------------------	-----------------------------	--

Update version

Configure kube-proxy

kube-proxy Info Listed by  Category networking Status  Active	Version Select the version for this add-on. <input type="text" value="v1.32.6-eksbuild.12"/> ▼
► Optional configuration settings	
Cancel	

Configure kube-proxy

kube-proxy Info Listed by  Category networking Status  Active	Version Select the version for this add-on. <input type="text" value="v1.32.9-eksbuild.2"/> ▼
► Optional configuration settings	
Cancel Save changes	

Configure Amazon VPC CNI

Amazon VPC CNI Info Listed by  Category networking Status  Active	Version Select the version for this add-on. <input type="text" value="v1.20.4-eksbuild.2"/> ▼
▼ Add-on access	
<input checked="" type="radio"/> EKS Pod Identity <input type="radio"/> IAM roles for service accounts (IRSA)	
Pod Identity IAM role for service account: aws-node	

Configure Amazon VPC CNI

Amazon VPC CNI Info Listed by  Category networking Status  Active	Version Select the version for this add-on. <input type="text" value="v1.21.1-eksbuild.1"/> ▼
▼ Add-on access	
<input checked="" type="radio"/> EKS Pod Identity <input type="radio"/> IAM roles for service accounts (IRSA)	

Upgrade kubectl:

- curl -LO

<https://dl.k8s.io/release/v1.32.0/bin/windows/amd64/kubectl.exe>

```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ curl -LO https://dl.k8s.io/release/v1.32.0/bin/windows/amd64/kubectl.exe
% Total    % Received % Xferd  Average Speed   Time     Time      Current
                                         Dload  Upload   Total   Spent   Left  Speed
100  138  100  138    0     0   286      0 --:--:-- --:--:-- --:--:--  287
100 56.1M 100 56.1M    0     0  2458k      0  0:00:23  0:00:23 --:--:-- 3296k
```

- mv kubectl.exe /c/Windows/System32/kubectl.exe
- kubectl version

```
MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ mv kubectl.exe /c/Users/Ashish/bin/kubectl.exe

MUJJU SK@DESKTOP-LU541U4 MINGW64 ~/Desktop
$ kubectl version
Client Version: v1.32.0
Kustomize Version: v5.5.0
Server Version: v1.32.10-eks-b3126f4
```