

EKS Cluster

Introduction:

EKS cluster is provided by AWS as a PAAS service.

Objective:

1. Create an EKS cluster

1. Create an EKS Cluster

Step 1:

In AWS, search for **Elastic Kubernetes Service** and click it.

Go for create a cluster and fill the below information.

Above we are providing it a role Clusterrole which is having the below permission.

- AmazonEKSClusterPolicy : IAM role to allow the Kubernetes control plane to manage AWS resources on your behalf.

Click next and then follow the below steps.

EKS > Clusters > Create EKS cluster

Step 1
Configure cluster

Step 2
Specify networking

Step 3
Configure logging

Step 4
Select add-ons

Step 5
Configure selected add-ons settings

Step 6
Review and create

Specify networking

Networking Info
These properties cannot be changed after the cluster is created.

VPC Info
Select a VPC to use for your EKS cluster resources. To create a new VPC, go to the [VPC console](#).

vpc-06b2e914fbd36a5e0 | Default

Subnets Info
Choose the subnets in your VPC where the control plane may place elastic network interfaces (ENIs) to facilitate communication with your cluster. To create a new subnet, go to the corresponding page in the [VPC console](#).

Select subnets

subnet-07991b2e1c2afdf28 X

subnet-02e25bf73780684da X

subnet-05b3793fcac28734f X

Security groups Info
Choose the security groups to apply to the EKS-managed Elastic Network Interfaces that are created in your worker node subnets. To create a new security group, go to the corresponding page in the [VPC console](#).

Select security groups

sg-0fce79343921bee62 X

Above we have selected the default VPC and the subnets related to that VPC. We have also selected the default security group and make sure that necessary ports are open. In our example have opened all the ports.

Cluster endpoint access Info
Configure access to the Kubernetes API server endpoint.

☒ **Public**
The cluster endpoint is accessible from outside of your VPC. Worker node traffic will leave your VPC to connect to the endpoint.

☐ **Public and private**
The cluster endpoint is accessible from outside of your VPC. Worker node traffic to the endpoint will stay within your VPC.

☐ **Private**
The cluster endpoint is only accessible through your VPC. Worker node traffic to the endpoint will stay within your VPC.

► Advanced settings

Cancel

Previous

Next

Now select the Cluster endpoint access, we have a three options which are Public, Public and private , Private.

☒ **Public**

The cluster endpoint is accessible from outside of your VPC. Worker node traffic will leave your VPC to connect to the endpoint.

☐ **Public and private**

The cluster endpoint is accessible from outside of your VPC. Worker node traffic to the endpoint will stay within your VPC.

☐ **Private**

The cluster endpoint is only accessible through your VPC. Worker node traffic to the endpoint will stay within your VPC.

Choose as per your business requirement and click Next.

The screenshot shows the 'Configure logging' step in the AWS EKS console. On the left, a sidebar lists the steps: Step 1 (Configure cluster), Step 2 (Specify networking), Step 3 (Configure logging), Step 4 (Select add-ons), Step 5 (Configure selected add-ons settings), and Step 6 (Review and create). The main area is titled 'Configure logging' and contains a section for 'Control plane logging' with a description: 'Send audit and diagnostic logs from the Amazon EKS control plane to CloudWatch Logs.' Below this, there are five toggle switches, all of which are currently turned on: 'API server' (Logs pertaining to API requests to the cluster), 'Audit' (Logs pertaining to cluster access via the Kubernetes API), 'Authenticator' (Logs pertaining to authentication requests into the cluster), 'Controller manager' (Logs pertaining to state of cluster controllers), and 'Scheduler' (Logs pertaining to scheduling decisions). At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Next'.

Above we have the logging option, we are keeping everything default and click Next.

Now AWS will install EKS add-ons which will be used for the networking part. These add-ons are.

1. Kube-proxy
2. CoreDNS
3. Amazon VPC CNI

EKS > Clusters > Create EKS cluster

Step 1
Configure cluster

Step 2
Specify networking

Step 3
Configure logging

Step 4
Select add-ons

Step 5
Configure selected add-ons settings

Step 6
Review and create

Select add-ons

Review the add-ons from multiple categories, then select add-ons to enhance your cluster.

Amazon EKS add-ons (3) [Info](#)

kube-proxy [Info](#) ☒

Enable service networking within your cluster.

Category
networking

✔ Installed by default

CoreDNS [Info](#) ☒

Enable service discovery within your cluster.

Category
networking

✔ Installed by default

Amazon VPC CNI [Info](#) ☒

Enable pod networking within your cluster.

Category
networking

✔ Installed by default

Cancel Previous **Next**

Click Next and it will ask for the versions for add-ons. We are keeping it default and click Next.

EKS > Clusters > Create EKS cluster

Step 1
Configure cluster

Step 2
Specify networking

Step 3
Configure logging

Step 4
Select add-ons

Step 5
Configure selected add-ons settings

Step 6
Review and create

Configure selected add-ons settings

Configure the add-ons for your cluster by selecting settings.

kube-proxy [Info](#)

Category
networking

Status
✔ Installed by default

Version
Select the version for this add-on.
v1.21.2-eksbuild.2 ▼

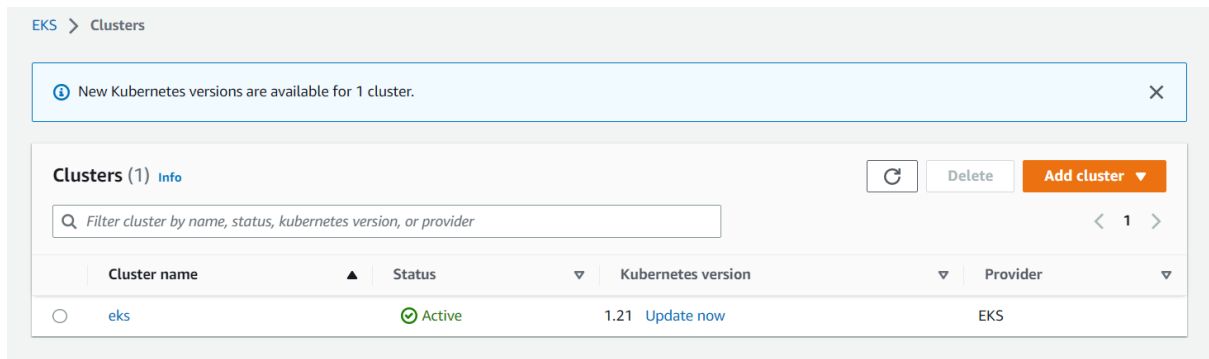
CoreDNS [Info](#)

Category
networking

Status
✔ Installed by default

Version
Select the version for this add-on.
v1.8.4-eksbuild.1 ▼

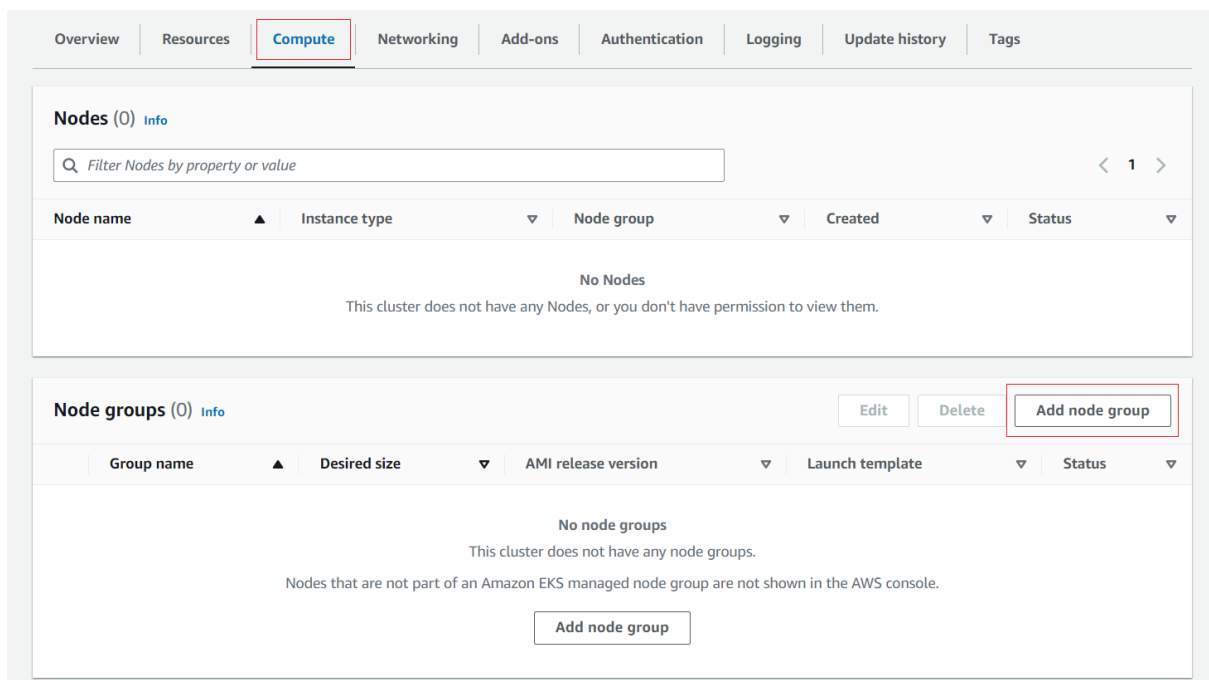
Next page will ask us to **review and create**. Go ahead and wait for our cluster.



Now our cluster is in active mode, we need to add nodes to it.

Step 2: Add nodes to the cluster

Now click our cluster and go to compute section. Further we need to add a node group which will be having the worker nodes.



Below we have given a name **workers** to our node group and also a role has been assigned to it.

EKS > Clusters > eks > Add node group

Step 1
Configure node group

Step 2
Set compute and scaling configuration

Step 3
Specify networking

Step 4
Review and create

Configure node group [Info](#)

A node group is a group of EC2 instances that supply compute capacity to your Amazon EKS cluster. You can add multiple node groups to your cluster.

Node group configuration

These properties cannot be changed after the node group is created.


Name
Assign a unique name for this node group.


workers


The node group 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 63.

Node IAM role [Info](#)
Select the IAM role that will be used by the nodes. To create a new role, go to the [IAM console](#).

nodegrouprole






 The selected role must not be used by a self-managed node group as this could lead to a service interruption upon managed node group deletion.

[Learn more](#) 

The role we have assigned is having three permission required cluster formation.

AmazonEKSWorkerNodePolicy	This policy allows Amazon EKS worker nodes to connect to Amazon EKS Clusters.
AmazonEC2ContainerRegistryReadOnly	Provides read-only access to Amazon EC2 Container Registry repositories.
AmazonEKS_CNI_Policy	This policy provides the Amazon VPC CNI Plugin (amazon-vpc-cni-k8s) the permissions it requires to modify the IP address configuration on your EKS worker nodes. This permission set allows the CNI to list, describe, and modify Elastic Network Interfaces on your behalf.

<input type="checkbox"/>	Policy name ↗	Type	Description
<input type="checkbox"/>	 AmazonEKSWorkerNodePolicy	AWS managed	This policy allows Amazon EKS worker nodes to connect to Amazon EKS Clusters.
<input type="checkbox"/>	 AmazonEC2ContainerRegistryReadOnly	AWS managed	Provides read-only access to Amazon EC2 Container Registry repositories.
<input type="checkbox"/>	 AmazonEKS_CNI_Policy	AWS managed	This policy provides the Amazon VPC CNI Plugin (amazon-vpc-cni-k8s) the permissi...

Then click next and set compute and scaling configuration.

We have used t2.micro instance type and 20GB in disk size.

EKS > Clusters > eks > Add node group

Step 1
Configure node group

Step 2
Set compute and scaling configuration

Step 3
Specify networking

Step 4
Review and create

Set compute and scaling configuration

Node group compute configuration

These properties cannot be changed after the node group is created.

AMI type [Info](#)

Select the EKS-optimized Amazon Machine Image for nodes.

Amazon Linux 2 (AL2_x86_64) ▼

Capacity type

Select the capacity purchase option for this node group.

On-Demand ▼

Instance types [Info](#)

Select instance types you prefer for this node group.

Select ▼

t2.micro

vCPU: 1 vCPU Memory: 1 GiB Network: Low to Moderate Max ENI: 2 Max IPs: 4

×

Disk size

Select the size of the attached EBS volume for each node.

20

GiB

Node group scaling configuration

Desired size

Set the desired number of nodes that the group should launch with initially.

2

nodes

Minimum size

Set the minimum number of nodes that the group can scale in to.

2

nodes

Maximum size

Set the maximum number of nodes that the group can scale out to.

2

nodes

Node group update configuration [Info](#)

Maximum unavailable

Set the maximum number or percentage of unavailable nodes to be tolerated during the node group version update.

☒ Number
Enter a number

☐ Percentage
Specify a percentage

Value

1

node

We have set the desired, minimum and maximum nodes to 2.

Maximum unavailable number has been set to 1 only.

Click next.


Specify networking

Node group network configuration

These properties cannot be changed after the node group is created.

Subnets [Info](#)

Specify the subnets in your VPC where your nodes will run. To create a new subnet, go to the corresponding page in the [VPC console](#).

Select subnets ▼ 

subnet-07991b2e1c2afdf28 ✕

subnet-02e25bf73780684da ✕

subnet-05b3793fcac28734f ✕

☐ Configure SSH access to nodes [Info](#)

Cancel Previous **Next**

We are using the same subnets which are a part of our default VPC.

Click **Next**, further click **Review and Create**.

Overview

Resources

Compute

Networking

Add-ons

Authentication

Logging

Update history

Tags

Nodes (2) Info

Q Filter Nodes by property or value

< 1 >

Node name	Instance type	Node group	Created	Status
ip-172-31-14-197.us-east-2.compute.internal	t2.micro	workers	Created 3 minutes ago	Ready
ip-172-31-18-184.us-east-2.compute.internal	t2.micro	workers	Created 3 minutes ago	Ready

Node groups (1) Info

EditDeleteAdd node group

Group name	Desired size	AMI release version	Launch template	Status
workers	2	1.21.14-20230105	-	Active

Now our cluster is completely ready and worker nodes are active too.

Step3: Access the cluster

Now we will be accessing the cluster from our machine. We need below packages as prerequisites.

1. AWS Installation
2. Kubectl Installation
3. Update kubeconfig file

1. **AWS Installation:**

Use the below command to install AWS in our machine.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
unzip awscliv2.zip
```

```
sudo ./aws/install
```

Once the AWS installation is down, we can check the status using below command.

```
aws --version
```

Now further configure aws using aws configure command. Also use **AWS access Key ID** and **AWS Secret Access Key** to configure it.

```
root@ip-172-31-33-180:~#  
root@ip-172-31-33-180:~# aws configure  
AWS Access Key ID [None]:  
AWS Secret Access Key [None]:  
Default region name [None]: us-east-2  
Default output format [None]:  
root@ip-172-31-33-180:~#  
root@ip-172-31-33-180:~# █
```

2. **Kubectl Installation:**

use the below command to install Kubectl package.

Download the kubectl binary for your cluster's Kubernetes version from Amazon S3

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.21.14/2022-10-31/bin/linux/amd64/kubectl
```

Apply execute permissions to the binary

```
chmod +x ./kubectl
```

Copy the binary to a folder in your PATH. If you have already installed a version of kubectl, then we recommend creating a \$HOME/bin/kubectl and ensuring that \$HOME/bin comes first in your \$PATH.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
```

Now our Kubectl package has been installed.

3. Update kubeconfig file:

Now our last step is to update the kubeconfig file in order to access the cluster. Use the below command.

```
aws eks --region us-east-2 update-kubeconfig --name eks
```

```
root@ip-172-31-33-180:~#  
root@ip-172-31-33-180:~# aws eks --region us-east-2 update-kubeconfig --name eks  
Added new context arn:aws:eks:us-east-2:685421549691:cluster/eks to /root/.kube/config  
root@ip-172-31-33-180:~#  
root@ip-172-31-33-180:~# kubectl get nodes  
NAME                                STATUS    ROLES    AGE   VERSION  
ip-172-31-14-197.us-east-2.compute.internal Ready    <none>   43m   v1.21.14-eks-fb459a0  
ip-172-31-18-184.us-east-2.compute.internal Ready    <none>   43m   v1.21.14-eks-fb459a0  
root@ip-172-31-33-180:~#
```

Now our cluster is completely ready and we can access it.

Let's run a pod using below command.

```
kubectl run pod --image nginx
```

```
root@ip-172-31-33-180:~# kubectl run pod --image nginx  
pod/pod created  
root@ip-172-31-33-180:~#  
root@ip-172-31-33-180:~# kubectl get pods  
NAME    READY   STATUS    RESTARTS   AGE  
pod     1/1     Running   0           9s  
root@ip-172-31-33-180:~#  
root@ip-172-31-33-180:~# kubectl get pods -o wide  
NAME    READY   STATUS    RESTARTS   AGE   IP            NODE                                NOMINATED NODE   READINESS GATES  
pod     1/1     Running   0           76s   172.31.2.150  ip-172-31-14-197.us-east-2.compute.internal  <none>           <none>  
root@ip-172-31-33-180:~#
```

