

Create Amazon EKS cluster by eksctl | How to create Amazon EKS cluster using EKSTCL | Create EKS Cluster in command line

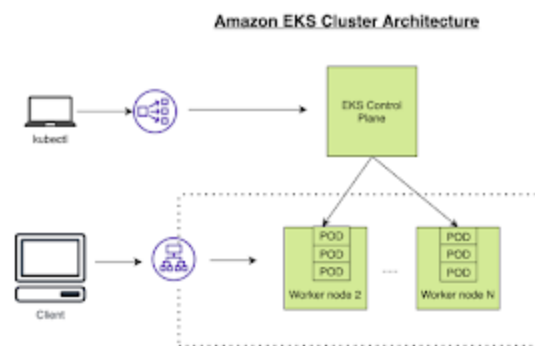
What is Amazon EKS

Amazon EKS is a fully managed container orchestration service. EKS allows you to quickly deploy a production ready Kubernetes cluster in AWS, deploy and manage containerized applications more easily with a fully managed Kubernetes service.

EKS takes care of Master node/control plane. We need to manage worker nodes.

Pre-requisites:

You can use **Windows, Macbook** or **any Ubuntu or Red Hat EC2 instance** to setup the below tools.



- **Install AWS CLI** – Command line tools for working with AWS services, including Amazon EKS.
- **Install eksctl** – A command line tool for working with EKS clusters that automates many individual tasks.
- **install kubectl** – A command line tool for working with Kubernetes clusters.
 - **Create AWS Access Keys**
 - Once you install all of the above, you need to have AWS credentials configured in your environment. The `aws configure` command is the fastest way to set up your AWS CLI installation for general use.
 - **Make sure you do not have any IAM role attached to the EC2 instance. You can remove IAM role by going into AWS console and Detach the IAM role.**
 - **aws configure**
 - the AWS CLI prompts you for four pieces of information: `access key`, `secret access key`, `AWS Region`, and `output format`.
 - **Create EKS Cluster using eksctl**
 - **eksctl create cluster --name demo-eks --region us-east-2 --nodegroup-name my-nodes --node-type t3.small --managed --nodes 2**

- the above command should create an EKS cluster in AWS, it might take 15 to 20 mins. The **eksctl** tool uses CloudFormation under the hood, creating one stack for the EKS master control plane and another stack for the worker nodes.

```

M:devops@aws:~$ eksctl create cluster --name demo-eks --region us-east-2 --nodegroup-name my-nodes
--node-type t3.small --managed
[+] eksctl version 0.20.1
[+] using region us-east-2
[+] setting availability zones to [us-east-2c us-east-2b us-east-2a]
[+] subnets for us-east-2c - public:[192.168.4.0/19 private:192.168.36.0/19]
[+] subnets for us-east-2b - public:[192.168.32.0/19 private:192.168.128.0/19]
[+] subnets for us-east-2a - public:[192.168.64.0/19 private:192.168.160.0/19]
[+] using Kubernetes version 1.17
[+] creating EKS cluster "demo-eks" in "us-east-2" region with managed nodes
[+] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
[+] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-2 --cluster=demo-eks'
[+] CloudWatch logging will not be enabled for cluster "demo-eks" in "us-east-2"
[+] you can enable it with 'eksctl utils update-cluster-logging --enable-types=[SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)] --region=us-east-2 --cluster=demo-eks'
[+] Kubernetes API endpoint access will use default of [publicAccess=true, privateAccess=false] for cluster "demo-eks" in "us-east-2"
[+] 2 sequential tasks: { create cluster control plane "demo-eks", 2 sequential sub-tasks: { no tasks, create managed nodegroup "my-nodes" } }
[+] building cluster stack "eksctl-demo-eks-cluster"
[+] deploying stack "eksctl-demo-eks-cluster"
[+] building managed nodegroup stack "eksctl-demo-eks-nodegroup-my-nodes"
[+] deploying stack "eksctl-demo-eks-nodegroup-my-nodes"
[+] waiting for the control plane availability...
[+] saved kubeconfig as "/Users/devops/.kube/config"
[+] no tasks
[+] all EKS cluster resources for "demo-eks" have been created
[+] nodegroup "my-nodes" has 2 node(s)
[+] node "ip-192-168-3-147.us-east-2.compute.internal" is ready
[+] node "ip-192-168-84-233.us-east-2.compute.internal" is ready
[+] nodegroup "my-nodes" is now ready
[+] waiting for at least 2 node(s) to become ready in "my-nodes"
[+] node "ip-192-168-3-147.us-east-2.compute.internal" is ready
[+] node "ip-192-168-84-233.us-east-2.compute.internal" is ready
[+] kubeconfig should work with "/Users/devops/.kube/config", try 'kubectl get nodes'
[+] EKS cluster "demo-eks" in "us-east-2" region is ready

```

eksctl get cluster --name demo-eks --region us-east-2

This should confirm that EKS cluster is up and running.

Connect to EKS cluster

kubectl get nodes

```

ubuntu@ip-172-31-32-245:~$ kubectl get nodes
NAME                                STATUS    ROLES    AGE    VERSION
ip-192-168-25-216.us-east-2.compute.internal Ready    <none>   43m    v1.17.11-eks-cf4c40
ip-192-168-51-213.us-east-2.compute.internal Ready    <none>   43m    v1.17.11-eks-cf4c40

```

kubectl get ns

```

ubuntu@ip-172-31-32-245:~$ kubectl get ns
NAME                STATUS    AGE
default             Active    49m
kube-node-lease     Active    49m
kube-public         Active    49m
kube-system         Active    49m

```

Deploy Nginx on a Kubernetes Cluster

Let us run some apps to make sure they are deployed to Kubernetes cluster. The below command will create deployment:

kubectl create deployment nginx --image=nginx

```

ubuntu@ip-172-31-2-128:~$ kubectl create deployment nginx --image=nginx
deployment.apps/nginx created

```

View Deployments

kubectl get deployments

```
ubuntu@ip-172-31-2-128:~$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx     1/1     1             1           10m
ubuntu@ip-172-31-2-128:~$
```

Delete EKS Cluster using eksctl

eksctl delete cluster --name demo-eks --region us-east-2

```
AK-DevOps-Couch:~ devopscouching$ eksctl delete cluster --name demo-eks --region us-east-2
[i] eksctl version 0.29.2
[i] using region us-east-2
[i] deleting EKS cluster "demo-eks"
[i] deleted 0 fargate profile(s)
[i] kubeconfig has been updated
[i] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
[i] 2 sequential tasks: { delete nodegroup "my-nodes", delete cluster control plane "demo-eks" [async] }
[i] will delete stack "eksctl-demo-eks-nodegroup-my-nodes"
[i] waiting for stack "eksctl-demo-eks-nodegroup-my-nodes" to get deleted
[i] will delete stack "eksctl-demo-eks-cluster"
[i] all cluster resources were deleted
AK-DevOps-Couch:~ devopscouching$
```

the above command should delete the EKS cluster in AWS, it might take a few mins to clean up the cluster.

Errors during Cluster creation

If you are having issues when creating a cluster, try to delete the cluster by executing the below command and re-create it.

eksctl delete cluster --name demo-eks --region us-east-2

or login to AWS console --> AWS Cloudformation --> delete the stack manually.

you can also delete the cluster under AWS console --> Elastic Kubernetes Service --> Clusters
Click on Delete cluster