How to Create EKS Cluster Using AWS CLI

Creating an Amazon Elastic Container Service for Kubernetes (EKS) cluster using the AWS Command Line Interface (AWS CLI) offers a hands-on and customizable approach to managing your cluster. This article will guide you through the process of installing and configuring the AWS CLI and then walk you through the steps to create an EKS cluster using the AWS CLI and setting up kubectl client tool.

Overview

1. [Install AWS CLI and configure](https://sujitpatel.in/article/how-to-create-eks-cluster-using-aws-cli/#Install-AWS-CLI-and-Configure)
2. [Create VPC, subnets, and security groups](https://sujitpatel.in/article/how-to-create-eks-cluster-using-aws-cli/#Create-VPC,-subnets-and-security-groups)
3. [Create EKS Cluster](https://sujitpatel.in/article/how-to-create-eks-cluster-using-aws-cli/#Create-EKS-Cluster)
4. [Create Node Group](https://sujitpatel.in/article/how-to-create-eks-cluster-using-aws-cli/#Create-Node-Group)
5. [Install and Configure Kubectl](https://sujitpatel.in/article/how-to-create-eks-cluster-using-aws-cli/#Install-and-Configure-Kubectl)

Steps:

**Step 1: Install AWS CLI and Configure**

Installing AWS CLI is very simple, ……...

**Step 2: Create VPC, subnets and security groups**

You can skip this step if you have already created the **VPC, Subnets, and security group**.

EKS requires a VPC to create and manage your cluster. You can create a VPC using the AWS CLI by running the following command:

**Create VPC**

**aws ec2 create-vpc --cidr-block 10.0.0.0/16**

This will create a new VPC with a CIDR block of 10.0.0.0/16. Replace this with your desired CIDR block if needed and note the**vpc-id**from the output.

**Create subnet**

After creating the VPC, you need to create at least two subnets in different availability zones. You can create a subnet using the following command:

aws ec2 create-subnet --vpc-id <vpc-id> --cidr-block 10.0.1.0/24 --availability-zone ap-south-1a

aws ec2 create-subnet --vpc-id <vpc-id> --cidr-block 10.0.2.0/24 --availability-zone ap-south-1b

Replace <vpc-id> with the ID of the VPC that you created in the previous step, and replace the CIDR block with your desired subnet CIDR block if needed. Also note subnet-ids from the output.

**Create security group**

Next, you need to create a security group for your cluster nodes. You can create a security group using the following command:

aws ec2 create-security-group --group-name eks-node-group --description "EKS Node Group" --vpc-id <vpc-id>

Replace <vpc-id> with the ID of the VPC that you created earlier. Take note of **group-id** from the output.

**Authorize inbound traffic**

To allow communication between your nodes and the EKS control plane, you need to authorize inbound traffic to the security group. You can do this using the following command:

aws ec2 authorize-security-group-ingress --group-id <group-id> --protocol tcp --port 22 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-id <group-id> --protocol tcp --port 80 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-id <group-id> --protocol tcp --port 443 --cidr 0.0.0.0/0

Replace <group-id> with the ID of the security group that you created in the previous step. You can change the port as per your requirement.

**Step 3: Create EKS Cluster**

Now that you have the required network infrastructure in place, you can create the EKS cluster. You can create the cluster using the following command:

aws eks create-cluster --name <cluster-name> --role-arn <role-arn> --resources-vpc-config subnetIds=<subnet-ids>,securityGroupIds=<security-group-ids>

Replace <cluster-name> with a name for your cluster, <role-arn> with the ARN of an IAM role that has the necessary permissions to create an EKS cluster, <subnet-ids> with a comma-separated list of the subnet IDs that we created earlier, and <security-group-ids> with the ID of the security group that we created.

**After you have run the create-cluster command, you need to wait for the cluster to be created. This can take several minutes. You can check the status of the cluster using the following command:**

aws eks describe-cluster --name <cluster-name>

**Step 4: Create Node Group**

EKS clusters consist of control plane nodes and worker nodes. You need to create a node group to run your workloads on the worker nodes. You can create a node group using the following command:

aws eks create-nodegroup --cluster-name <cluster-name> --nodegroup-name <node-group-name> --subnets <subnet-ids> --security-groups <security-group-ids> --instance-types <instance-types> --ami-type AL2\_x86\_64

Replace <cluster-name> with the name of your cluster, <node-group-name> with a name for your node group, <subnet-ids> with a comma-separated list of the subnet IDs that we created earlier, <security-group-ids> with the ID of the security group that we created, and <instance-types> with a comma-separated list of the instance types that you want to use for your worker nodes.

After you have run the create-nodegroup command, you need to wait for the node group to be created. This can take several minutes. You can check the status of the node group using the following command:

aws eks describe-nodegroup --cluster-name <cluster-name> --nodegroup-name <node-group-name>

Replace <cluster-name> with the name of your cluster and <node-group-name> with the name of your node group.

Once the node group has been created, you should now have a fully functional EKS cluster that you can use to run your containers.

Step 5: Install and Configure Kubectl

Now, we need to configure the **kubectl** command-line tool to use the cluster. You can do this using the following command:

**update Kubeconfig file**

aws eks update-kubeconfig --region <region-name> --name <cluster-name>

Replace <cluster-name> with the name of your cluster and <region-name> with your region name.

**Install kubectl**

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

chmod +x kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

kubectl version --short --client

Now, **kubectl** is also installed and configured and you’re ready to use the EKS cluster.

**Check Cluster Info**

kubectl cluster-info