**How to set up our eks cluster step by step**

<https://github.com/stacksimplify>

we have option in cloud so we will take t2.micro Ubuntu instance and do on that

### Step01:-Install and configure AWS CLI

We are Linux user so we have follow this steps

**1.For the current version of the AWS CLI,** use the following command:

**curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"**

**For a specific version of the AWS CLI,** append a hyphen and the version number to the filename. For this example the filename for version *2.0.30* would be awscli-exe-linux-x86\_64-2.0.30.zip resulting in the following command:

**curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64-2.0.30.zip" -o "awscliv2.zip"**

for reference please follow below link

<https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html>

2. Now unzip the installer

If unzip is not available in your system please install it using

**sudo apt install unzip**

**unzip awscliv2.zip**

3.Run the install program. The installation command uses a file named install in the newly unzipped aws directory. By default, the files are all installed to /usr/local/aws-cli, and a symbolic link is created in /usr/local/bin. The command includes sudo to grant write permissions to those directories

**sudo ./aws/install**

4. Confirm the installation.

**aws --version**

### Step02: Configure AWS Command Line using Security Credentials

* Go to AWS Management Console --> Services --> IAM
* Select the IAM User: anis

If you don’t have IAM user created lets create it first because Never ever use Root User. (Highly not recommended) which is by default in creditionals

* Click on **Security credentials** tab
* Click on **Create access key**
* Copy Access ID and Secret access key
* Go to command line and provide the required detail

**aws configure**

AWS Access Key ID [None]:- paste your access key here

AWS Secret Access Key [None]:- paste your secret key

Default region name [None]: us-east-1 choose as per your region

Default output format [None]: json

After that check your AWS CLI working or not

**aws ec2 describe-vpcs**

### Step-03: Install kubectl CLI

* **IMPORTANT NOTE:** Kubectl binaries for EKS please prefer to use from Amazon (**Amazon EKS-vended kubectl binary**)
* This will help us to get the exact Kubectl client version based on our EKS Cluster version. You can use the below documentation link to download the binary.
* Reference: https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html

**1.**To install kubectl on Linux follow commands

So in this case ill take kubernetes 1.18 version if you want to take some other version follow reference link which will provide on start

**curl -o kubectl https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl**

**2.**now we will Download the SHA-256 sum for your cluster's Kubernetes version for Linux

**curl -o kubectl.sha256 https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl.sha256**

**3.** Now Check the SHA-256 sum for your downloaded binary**.**

**openssl sha1 -sha256 kubectl**

Compare the generated SHA-256 sum in the command output against your downloaded SHA-256 file. The two should match

**4.** 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

**5.** Add the $HOME/bin path to your shell initialization file so that it is configured when you open a shell.

**echo 'export PATH=$PATH:$HOME/bin' >> ~/.bashrc**

you can check with nano .bashrc so export path will shown in configuration

**6.** After you install kubectl , you can verify its version with the following command:

**kubectl version --short –client**

## **Step-04: Install eksctl CLI**

To install or upgrade eksctl on Linux using curl

For reference follow this link

https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html

* 1. Download and extract the latest release of eksctl with the following command.

**curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp**

* 1. Move the extracted binary to /usr/local/bin.

sudo mv /tmp/eksctl /usr/local/bin

* 1. Test that your installation was successful with the following command

**eksctl version**

# **Create EKS Cluster & Node Groups**

## Step-01: Create EKS Cluster using eksctl

* Please note that It will take 15 to 20 minutes to create the Cluster Control Plane so be patience

**eksctl create cluster --name=kubercluster \**

**--region=us-east-1 \**

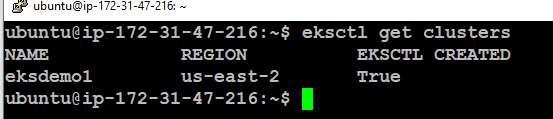
**--zones=us-east-1a,us-east-1b \**

**--without-nodegroup**

After the cluster is ready then check with following commands

# Get List of clusters

**eksctl get clusters**

****

Now cluster is ready

## **Step-05: Create & Associate IAM OIDC Provider for our EKS Cluster**

* To enable and use AWS IAM roles for Kubernetes service accounts on our EKS cluster, we must create & associate OIDC identity provider.
* To do so using eksctl we can use the below command.
* Use latest eksctl version (as on today the latest version is 0.35.0)

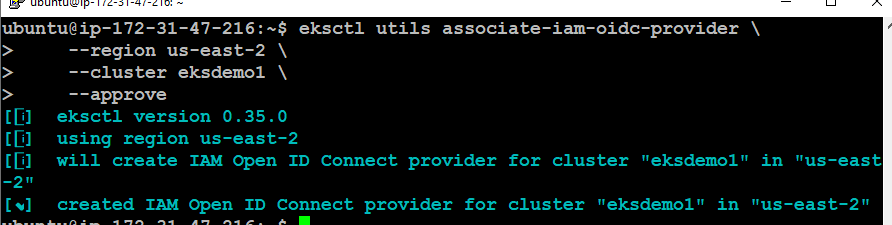
# Replace with region & cluster name

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

**--region us-east-1 \**

**--cluster kubercluster \**

**--approve**

****

## **Step-06**: **Create EC2 Keypair**

If you already have key then use those key in my case I have already created key which is

Techcloud.pem use format as per your need for ex:- .pem .ppk if you don’t have key then create it

* Create a new EC2 Keypair with name as techcloud
* This keypair we will use it when creating the EKS NodeGroup.
* This will help us to login to the EKS Worker Nodes using Terminal.

## **Step-07: Create Node Group with additional Add-Ons in Public Subnets**

* These add-ons will create the respective IAM policies for us automatically within our Node Group role.
* # Create Public Node Group

eksctl create nodegroup --cluster= kubercluster \

--region=us-east-1 \

--name=eksdemo1-ng-public1 \

--node-type=t2.large \

--nodes=2 \

--nodes-min=2 \

--nodes-max=4 \

--node-volume-size=20 \

--ssh-access \

--ssh-public-key=myekskey \

--managed \

--asg-access \

--external-dns-access \

--full-ecr-access \

--appmesh-access \

--alb-ingress-access

## **Step-08: Verify Cluster & Nodes**

### Verify NodeGroup subnets to confirm EC2 Instances are in Public Subnet

* Verify the node group subnet to ensure it created in public subnets
  + Go to Services -> EKS ->eksdemo -> eksdemo1-ng1-public
  + Click on Associated subnet in **Details** tab
  + Click on **Route Table** Tab.
  + We should see that internet route via Internet Gateway (0.0.0.0/0 ->igw-xxxxxxxx)

### Verify Cluster, NodeGroup in EKS Management Console

* Go to Services -> Elastic Kubernetes Service -> eksdemo1

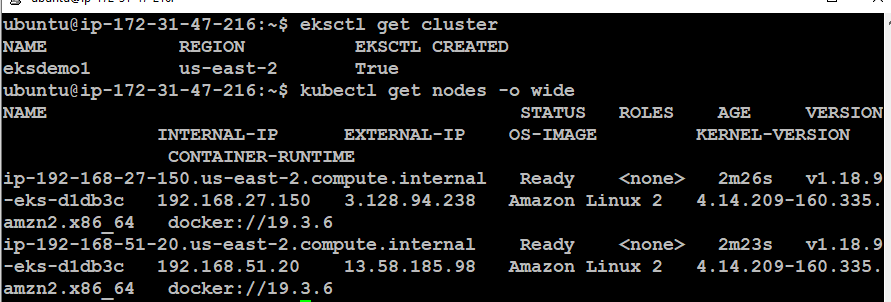
### List Worker Nodes

# List EKS clusters

**eksctl get cluster**

# List Nodes in current kubernetes cluster

**kubectl get nodes -o wide**

****

# Our kubectl context should be automatically changed to new cluster

**kubectl config view –minify**

### Verify Worker Node IAM Role and list of Policies

* Go to Services -> EC2 -> Worker Nodes
* Click on **IAM Role associated to EC2 Worker Nodes**

### Verify Security Group Associated to Worker Nodes

* Go to Services -> EC2 -> Worker Nodes
* Click on **Security Group** associated to EC2 Instance which contains remote in the name.

### Verify CloudFormation Stacks

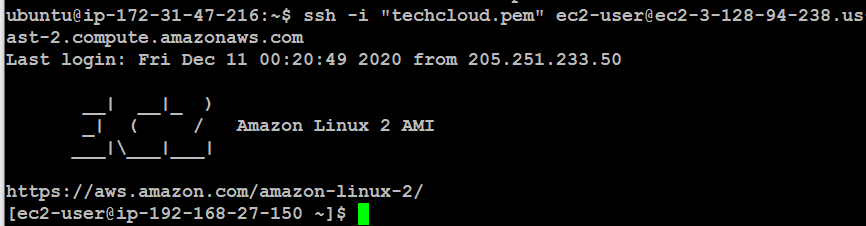
* Verify Control Plane Stack & Events
* Verify NodeGroup Stack & Events

### Login to Worker Node using Keypair techcloud.pem

* Login to worker node

**ssh -i "techcloud.pem"** [ec2-user@ec2-3-128-94-238.us-east-2.compute.amazonaws.com](mailto:ec2-user@ec2-3-128-94-238.us-east-2.compute.amazonaws.com)

**now we are successfully on worker node**

****

Once you done it with all then now we see how to delete eks cluster

## **Step-09: Delete Node Group**

* We can delete a nodegroup separately using below eksctl delete nodegroup
* List EKS Clusters

**eksctl get clusters**

# Capture Node Group name

**eksctl get nodegroup --cluster=eksdemo1**

# Delete Node Group

**eksctl delete nodegroup --cluster=eksdemo1 --name=eksdemo1-ng-public1**

## **Step-10: Delete Cluster**

* We can delete cluster using eksctl delete cluster

Ill strongly recommend that while deleting cluster use this command because we will practicing

# Delete Cluster

**eksctl delete cluster ekscluster**

## Important Notes

### Note-1: Rollback any Security Group Changes

* When we create a EKS cluster using eksctl it creates the worker node security group with only port 22 access.
* When we progress through the course, we will be creating many **NodePort Services** to access and test our applications via browser.
* During this process, we need to add an additional rule to this automatically created security group, allowing access to our applications we have deployed.
* So the point we need to understand here is when we are deleting the cluster using eksctl, its core components should be in same state which means roll back the change we have done to security group before deleting the cluster.
* In this way, cluster will get deleted without any issues, else we might have issues and we need to refer cloudformation events and manually delete few things. In short, we need to go to many places for deletions.

### Note-2: Rollback any EC2 Worker Node Instance Role - Policy changes

* When we are doing EBS Storage Section with EBS CSI Driver we will add a custom policy to worker node IAM role.
* When you are deleting the cluster, first roll back that change and delete it.
* This way we don't face any issues during cluster deletion.