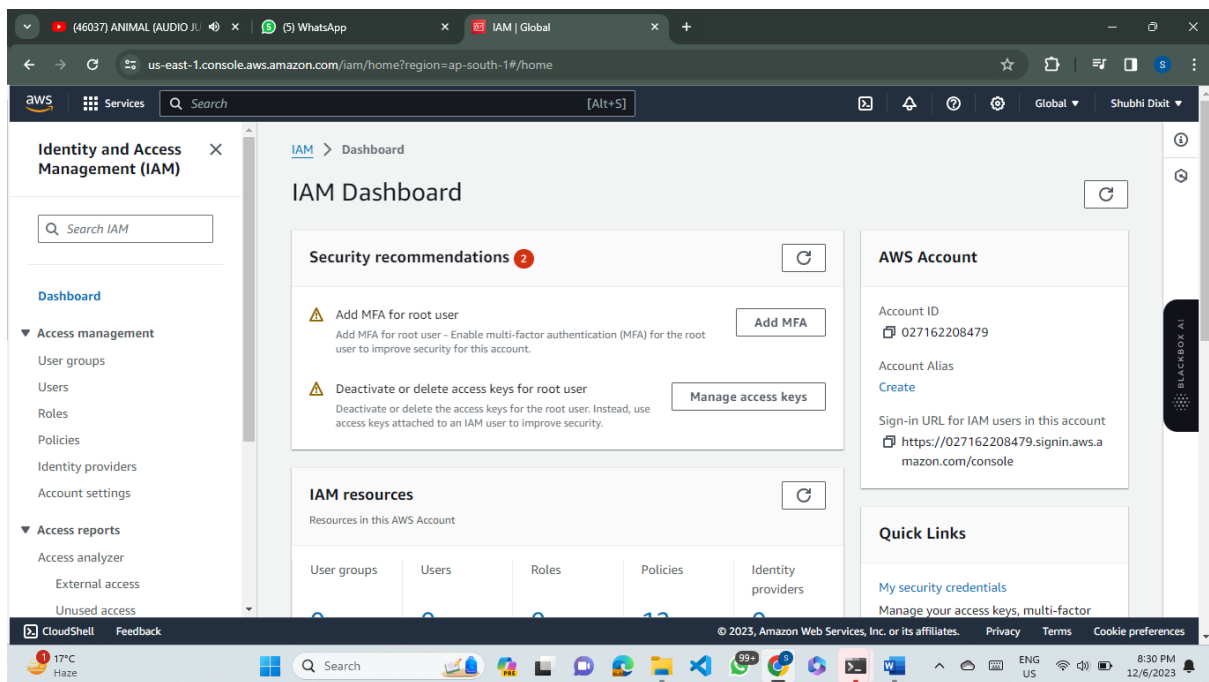
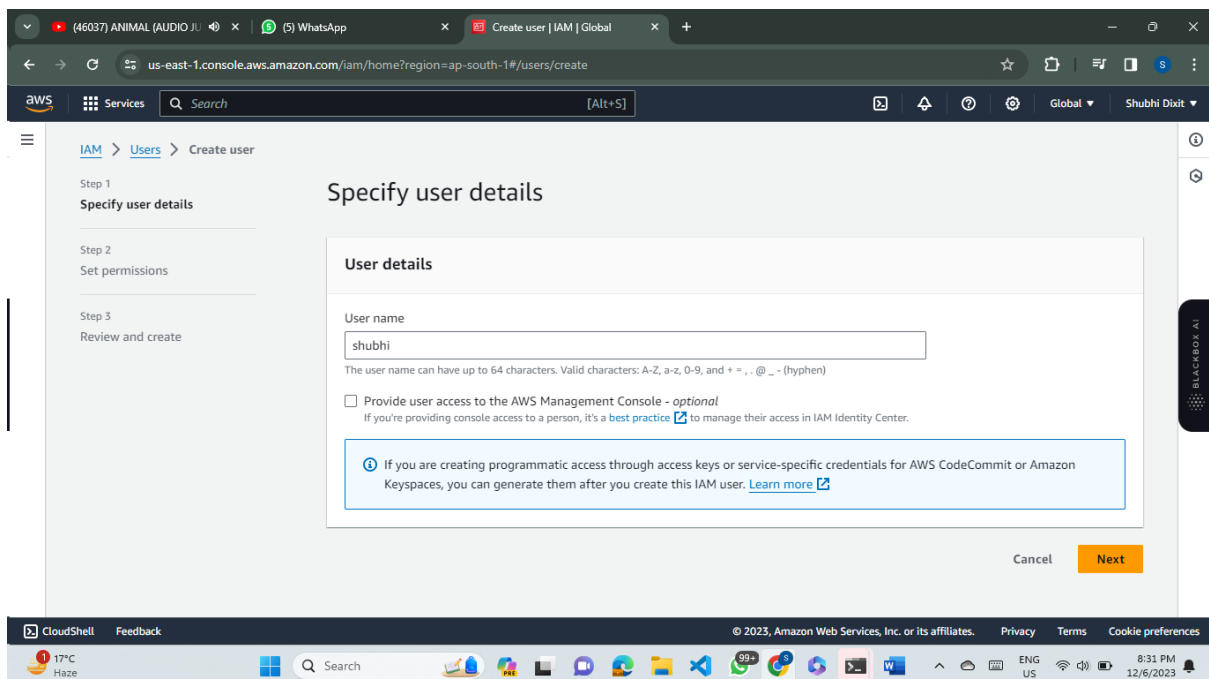


# AWS EKS

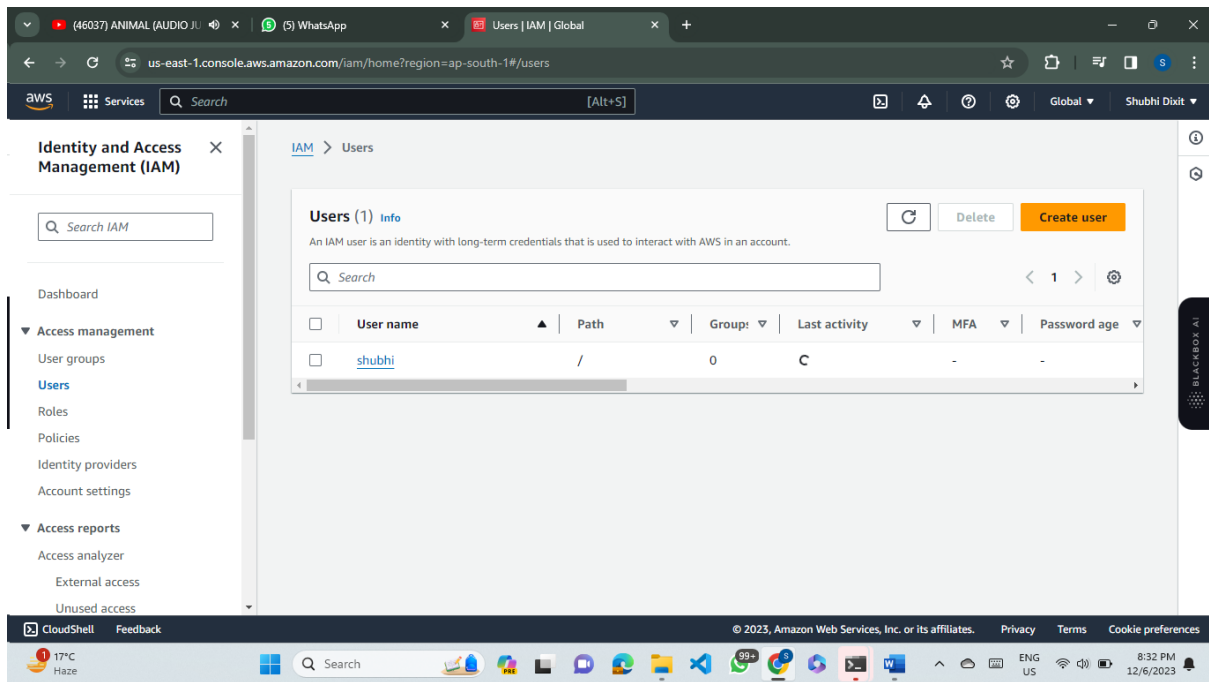
**Step-1:** Go to **IAM console** and create a user.



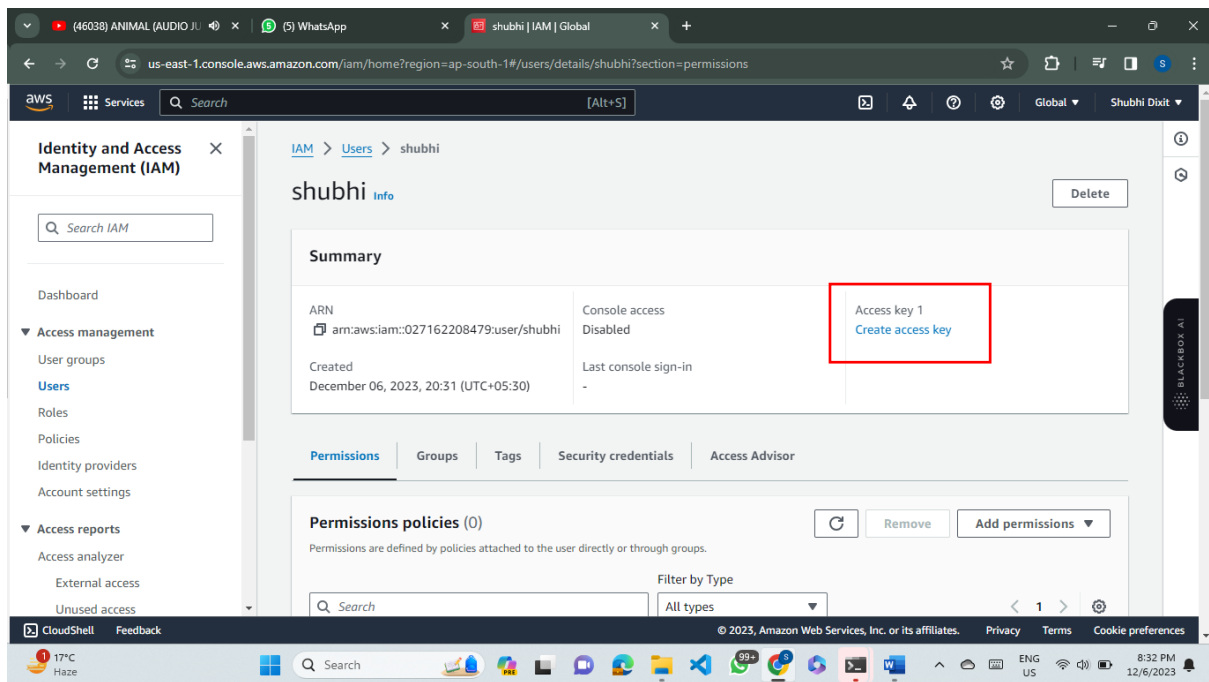
**Step-2:** Name the user.



**Step-3:** Now click on the user that you just created. Here it is “eks”.



**Step-4:** Click on “Create access key”.



**Step-5:** Choose the option “Other” and click on next. Leave the other details as default.

### Access key best practices & alternatives [info](#)

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

☐ **Command Line Interface (CLI)**  
You plan to use this access key to enable the AWS CLI to access your AWS account.

☐ **Local code**  
You plan to use this access key to enable application code in a local development environment to access your AWS account.

☐ **Application running on an AWS compute service**  
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

☐ **Third-party service**  
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

☐ **Application running outside AWS**  
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

☒ **Other**  
Your use case is not listed here.

**Step-6:** Now copy the “Access key” and “Secret Access Key” and save it somewhere as it can’t be retrieved later.

### Retrieve access keys [info](#)

**Access key**  
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAUOR2PLEMOI4CFXG3	***** <a href="#">Show</a>

**Access key best practices**

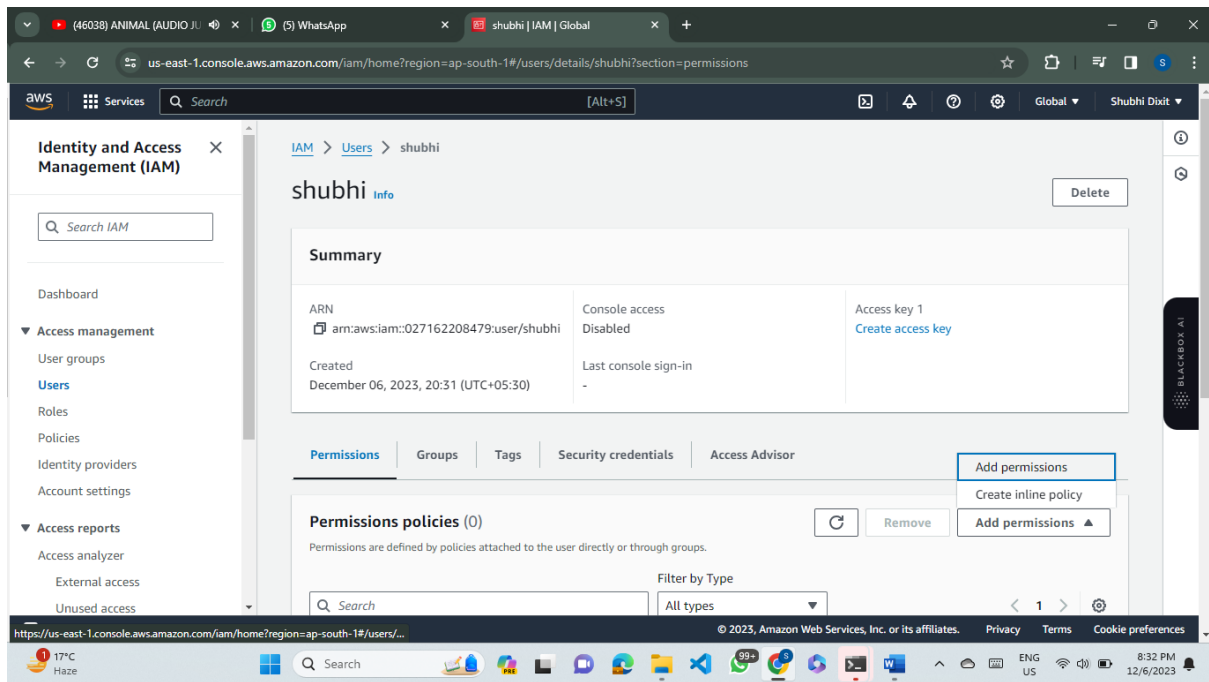
- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

[Download .csv file](#) [Done](#)

**Step-7:** Now go to “Add Permissions”.

**Note:** You can attach the policies while creating the user or after creating the user as well.

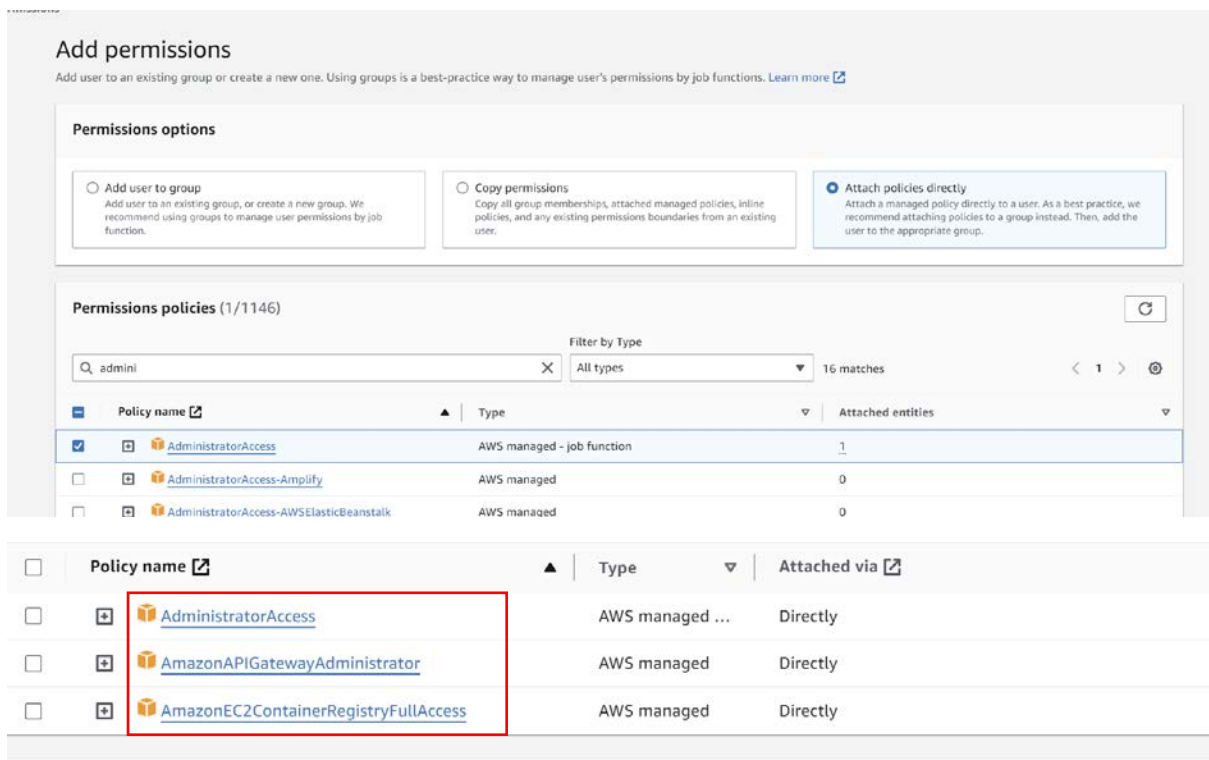


**Step-8:** Click on “Attach policies directly” and choose the following permissions and click on next and save them.

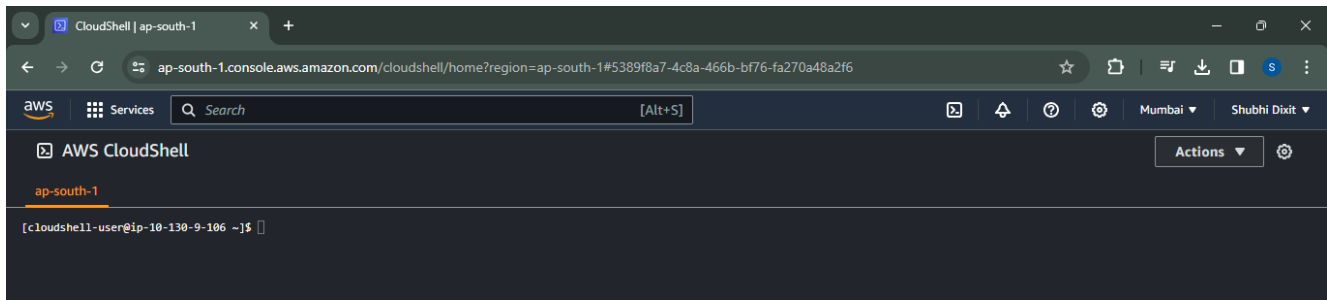
**AdministratorAccess**

**AmazonAPIGatewayAdministrator**

**AmazonEC2ContainerRegistryFullAccess**



**Step-9:** Now go to AWS Console and search “AWS Cloudshell”.



**Step-10:** First step is to configure AWS CLI. Write the command “**aws configure**”. After that provide the access key and secret access key that you have copied earlier. Write your region and the format. Keep the format in “**json**” for example.

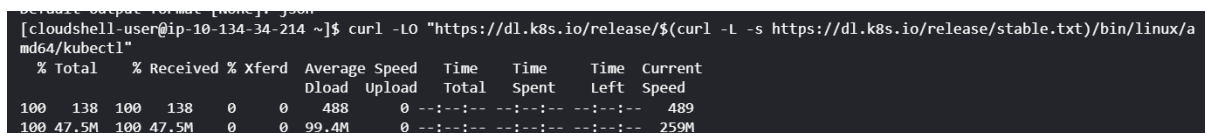


## Installing the required tools

**Step-11:** Now install **kubectl** using the following command:

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

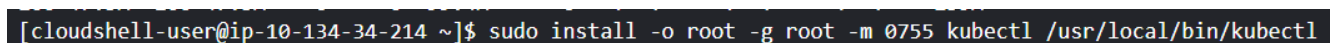
➔ **kubectl** is the command-line tool used to interact with Kubernetes clusters.



**Step-12:** After the installation is complete, run the following command:

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

➔ This command is installing the **kubectl** binary into the **/usr/local/bin** directory with **root** ownership and executable permissions, allowing users to run **kubectl** commands globally on the system.



**Step-13:** In order to install the “**eksctl**” command line tool, we need to create a script. Create a file called “**eksctl.sh**” using vim or nano or any other text editor.

```
vim eksctl.sh
```

- ➔ eksctl is a command-line utility for creating, managing, and interacting with Amazon Elastic Kubernetes Service (EKS) clusters.

```
[cloudshell-user@ip-10-134-34-214 ~]$ vim eksctl.sh
```

**Step-14:** Now in order to insert text, press “**I**” and then copy the below given commands inside it.

```
ARCH=amd64
```

```
PLATFORM=$(uname -s)_$ARCH
```

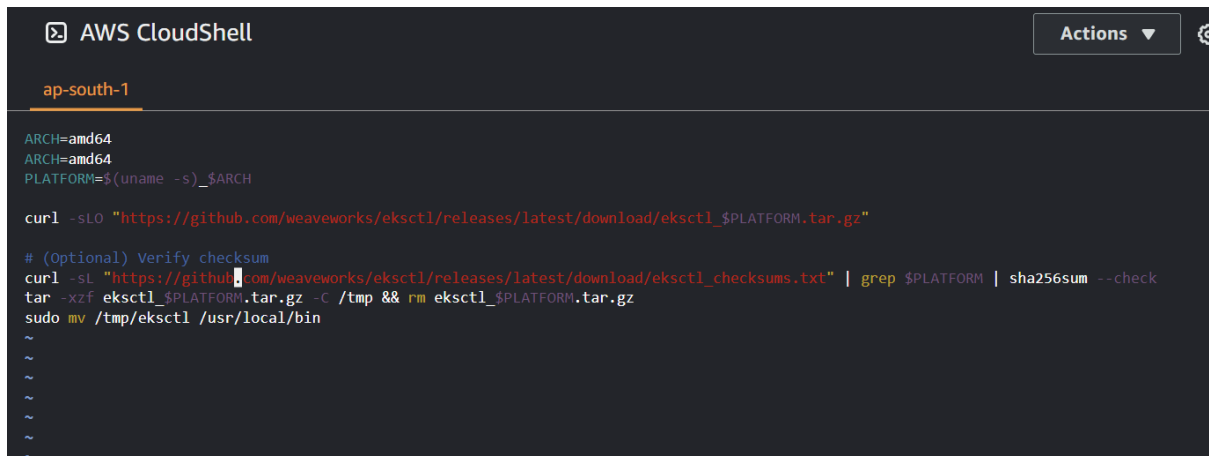
```
curl -sLO "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_${PLATFORM}.tar.gz"
```

```
# (Optional) Verify checksum
```

```
curl -sL "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_checksums.txt" | grep $PLATFORM | sha256sum --check
```

```
tar -xzf eksctl_${PLATFORM}.tar.gz -C /tmp && rm eksctl_${PLATFORM}.tar.gz
```

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

A screenshot of the AWS CloudShell interface. The title bar shows 'AWS CloudShell' and 'Actions' with a dropdown arrow. Below the title bar, the region 'ap-south-1' is selected. The terminal window displays the following commands and their output:

```
ARCH=amd64
ARCH=amd64
PLATFORM=$(uname -s)_$ARCH

curl -sLO "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_${PLATFORM}.tar.gz"

# (Optional) Verify checksum
curl -sL "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_checksums.txt" | grep $PLATFORM | sha256sum --check
tar -xzf eksctl_${PLATFORM}.tar.gz -C /tmp && rm eksctl_${PLATFORM}.tar.gz
sudo mv /tmp/eksctl /usr/local/bin
~
~
~
~
~
~
```

**Step-15:** Run the below commands:

```
chmod +x eksctl.sh
```

```
sudo sh eksctl.sh
```

- ➔ The command **chmod +x eksctl.sh** is used to grant execute permissions (+x) to the file named eksctl.sh.
- ➔ The command **sudo sh eksctl.sh** is attempting to run the script named **eksctl.sh** that we created.

```
[cloudshell-user@ip-10-134-34-214 ~]$ chmod +x eksctl.sh
[cloudshell-user@ip-10-134-34-214 ~]$ sudo sh eksctl.sh
eksctl_linux_amd64.tar.gz: OK
```

## Creating an EKS Cluster

**Step-16:** Now create a new Amazon EKS cluster using the `eksctl` command:

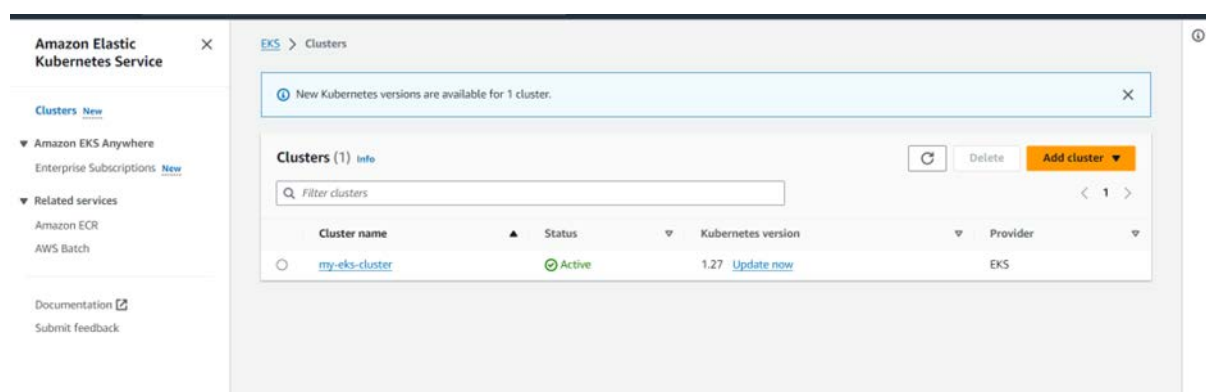
```
eksctl create cluster --name my-eks-cluster --region ap-south-1 --nodegroup-name my-nodegroup --node-type t2.small --nodes 3 --nodes-min 1 --nodes-max 5 --managed
```

- ➔ The `eksctl create cluster` command is creating an Amazon EKS cluster named **"my-eks-cluster"** in the **Asia Pacific (Mumbai) region** with a managed node group named **"my-nodegroup,"** using **t2.small** instances with a **desired** count of **3 nodes**, a **minimum** of **1 node**, and a **maximum** of 5 nodes.

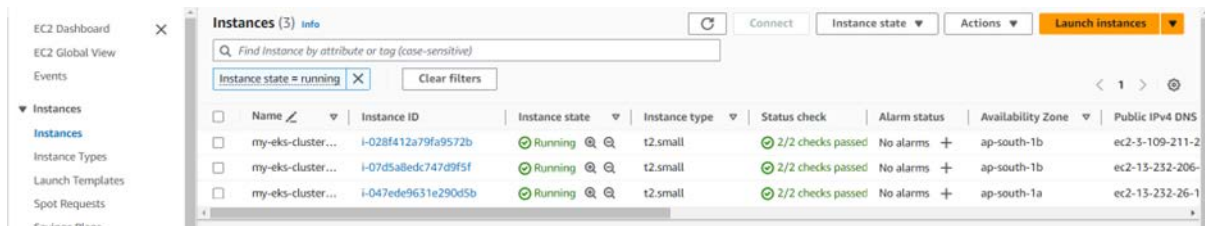
**Note:** Make sure to change the region name to your specific region in the command portion that is highlighted.

```
[cloudshell-user@ip-10-134-34-214 ~]$ eksctl create cluster --name my-eks-cluster --region ap-south-1 --nodegroup-name my-nodegroup --node-type t2.small --nodes 3 --nodes-min 1 --nodes-max 5 --managed
2023-11-18 08:34:39 [i] eksctl version 0.164.0
2023-11-18 08:34:39 [i] using region ap-south-1
2023-11-18 08:34:39 [i] skipping ap-south-1c from selection because it doesn't support the following instance type(s): t2.small
2023-11-18 08:34:39 [i] setting availability zones to [ap-south-1a ap-south-1b]
2023-11-18 08:34:39 [i] subnets for ap-south-1a - public:192.168.0.0/19 private:192.168.0.0/19
2023-11-18 08:34:39 [i] subnets for ap-south-1b - public:192.168.0.0/19 private:192.168.0.0/19
2023-11-18 08:34:39 [i] nodegroup "my-nodegroup" will use "" [AmazonLinux2/1.27]
2023-11-18 08:34:39 [i] using Kubernetes version 1.27
2023-11-18 08:34:39 [i] creating EKS cluster "my-eks-cluster" in "ap-south-1" region with managed nodes
2023-11-18 08:34:39 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2023-11-18 08:34:39 [i] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=my-eks-cluster'
2023-11-18 08:34:39 [i] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "my-eks-cluster" in "ap-south-1"
2023-11-18 08:34:39 [i] CloudWatch logging will not be enabled for cluster "my-eks-cluster" in "ap-south-1"
2023-11-18 08:34:39 [i] you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-south-1 --cluster=my-eks-cluster'
2023-11-18 08:34:39 [i]
2 sequential tasks: [ create cluster control plane "my-eks-cluster",
2 sequential sub-tasks: (
    wait for control plane to become ready,
    create managed nodegroup "my-nodegroup",
)
2023-11-18 08:34:39 [i] building cluster stack "eksctl-my-eks-cluster-cluster"
2023-11-18 08:34:39 [i] deploying stack "eksctl-my-eks-cluster-cluster"
2023-11-18 08:35:09 [i] waiting for cloudformation stack "eksctl-my-eks-cluster-cluster"
2023-11-18 08:35:39 [i] waiting for cloudformation stack "eksctl-my-eks-cluster-cluster"
```

**Step-17:** Now after the creation is complete in the above step, go to **AWS console** and search **“EKS”**. You can see that the cluster is created.



**Step-18:** Also go to **EC2 Dashboard** and click on instances. We can see that the specified number of node instances are created i.e. 3.



## Deploying a sample application

**Step-19:** Now create a “**deployment.yaml**” file using any editor and copy the following yaml code.

NOTE: MAKE SURE THE INDENTATION OF THE YAML FILE IS SAME AS SHOWN OTHERWISE IT WILL THROW MAPPING ERROR.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

```
[cloudshell-user@ip-10-134-34-214 ~]$ vim deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```



**Step-20:** Deploy the sample application to your EKS cluster:

```
kubectl apply -f deployment.yaml
```

```
[cloudshell-user@ip-10-134-34-214 ~]$ kubectl apply -f deployment.yaml
deployment.apps/nginx-deployment created
```

## Exposing the Application

**Step-21:** Now expose the application using a Kubernetes service and get the external IP address of the LoadBalancer:

```
kubectl expose deployment nginx-deployment --type=LoadBalancer --name=my-service
```

- ➔ The **kubectl expose** command is creating a new Kubernetes Service named "**my-service**" and exposing the "**nginx-deployment**" Deployment to the external network using a **LoadBalancer** type service.

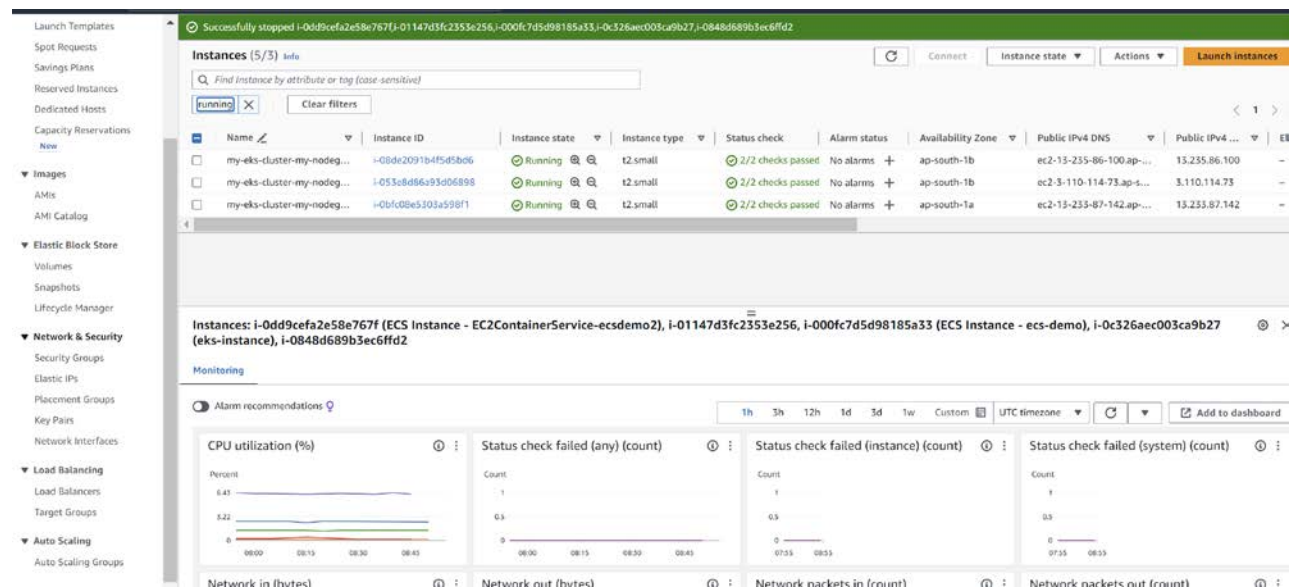
```
kubectl get services my-service
```

- ➔ The **kubectl get services my-service** command is used to retrieve information about the Kubernetes Service named "**my-service**," displaying details such as the **service's IP address**, **ports**, and other relevant information.

```
[cloudshell-user@ip-10-134-34-214 ~]$ kubectl expose deployment nginx-deployment --type=LoadBalancer --name=my-service
service/my-service exposed
[cloudshell-user@ip-10-134-34-214 ~]$ kubectl get services my-service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
my-service	LoadBalancer	10.100.114.51	af45aabc5fce24064b7e8a878aba3d9a-595339015.ap-south-1.elb.amazonaws.com	80:32291/TCP	29s

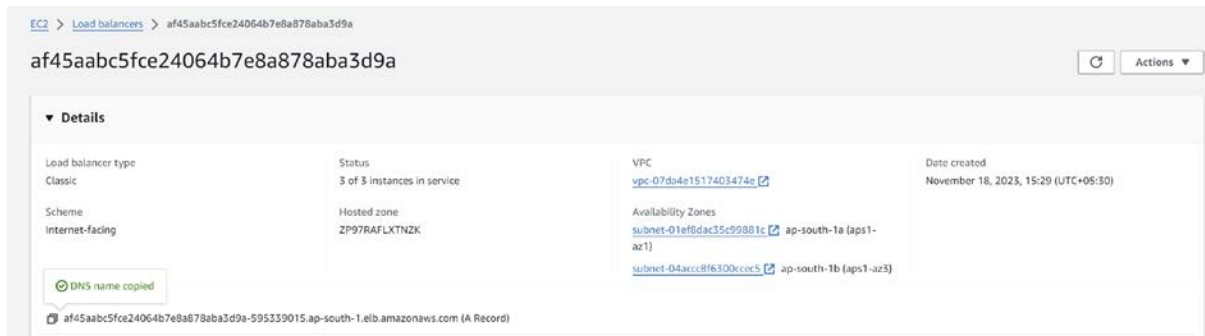
**Step-22:** Now go to EC2 dashboards and click on load balancers.



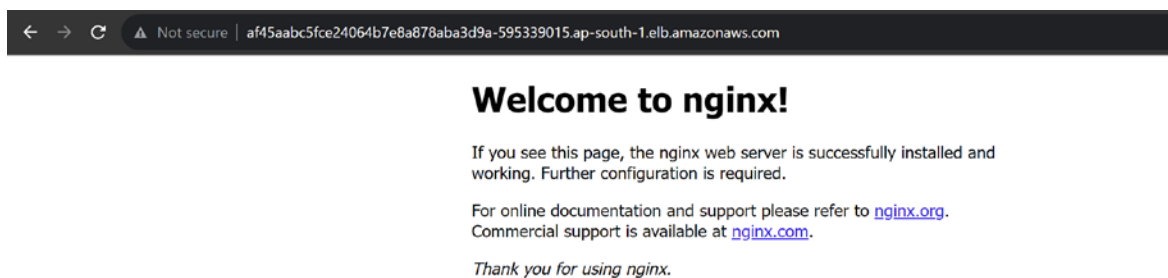
**Step-23:** We can see on the dashboard a “classic” load balancer that is made by default after running the command.



**Step-24:** Copy the **DNS name** and paste it in browser to check whether the service is accessible or not.



We can see that it is accessible on browser.



## Setting up Horizontal Pod Autoscaler

**Step-25:** Now create a new file named `hpa.yaml` (just like we made the `deployment.yaml`) with the following content:

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-deployment-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
  minReplicas: 3
```

```
maxReplicas: 10
```

```
targetCPUUtilizationPercentage: 50
```

ap-south-1

```
apiVersion: autoscaling/v1
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: nginx-deployment-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx-deployment
  minReplicas: 3
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

```
~
~
~
~
~
~
~
~
~
~
```

## Step-26:

```
kubectl apply -f hpa.yaml
```

```
[cloudshell-user@ip-10-134-34-214 ~]$ kubectl apply -f hpa.yaml
horizontalpodautoscaler.autoscaling/nginx-deployment-hpa created
```

Step-27: After performing the above commands, you can delete the service and cluster using the following commands:

```
kubectl delete svc my-service
```

```
eksctl delete cluster --name my-eks-cluster
```