MARIO

Project Overview: Deploying Mario Game on Amazon EKS

Project Goal:

The goal of this project is to deploy a Mario game on Amazon Elastic Kubernetes Service (EKS) using a Docker image. The Mario game is containerized using Docker, and the deployment is orchestrated on a Kubernetes cluster managed by Amazon EKS.

Technologies Used:

- Docker: Containerization of the Mario game.
- Amazon EKS: Managed Kubernetes service for orchestrating containers.
- Kubernetes: Container orchestration platform for automating deployment, scaling, and management of containerized applications.
- AWS CLI: Command-line interface for interacting with AWS resources.
- Kubectl: Command-line tool for interacting with Kubernetes clusters.
- Amazon ECR: Elastic Container Registry for storing and managing Docker images.

Project Steps:

Containerization with Docker:

- Create a Dockerfile to package the Mario game and its dependencies into a Docker image.
- Build the Docker image and push it to a container registry (e.g., Amazon ECR).

Amazon EKS Cluster Setup:

- Create an Amazon EKS cluster using the AWS Management Console, AWS CLI, or infrastructure-as-code tools.
- Configure kubectl to connect to the EKS cluster.

Kubernetes Manifests:

- Create Kubernetes Deployment and Service manifests to define the desired state of the Mario game deployment.
- The Deployment manages the pod replicas running the Mario game, and the Service exposes the game externally.

Apply Kubernetes Manifests:

- Use kubectl to apply the Deployment and Service manifests to the EKS cluster.
- This step creates the necessary resources on the cluster for running the Mario game.

Expose the Service:

- Monitor the Service to obtain the external IP address once assigned.
- Access the Mario game using the external IP address and the specified port.

Testing and Validation:

- Test the deployed Mario game to ensure proper functionality.
- Verify that the game is accessible, and interactions work as expected.

Clean-Up:

- Delete the deployed resources when done testing to avoid unnecessary costs.
- Use kubectl delete commands to delete the Deployment and Service.

Additional Considerations:

- Scaling: Explore Kubernetes scaling features to scale the Mario game based on demand.
- Monitoring: Implement monitoring and logging solutions to track the performance and health of the Mario game.
- Security: Apply security best practices for containerized applications and Kubernetes clusters.

Conclusion:

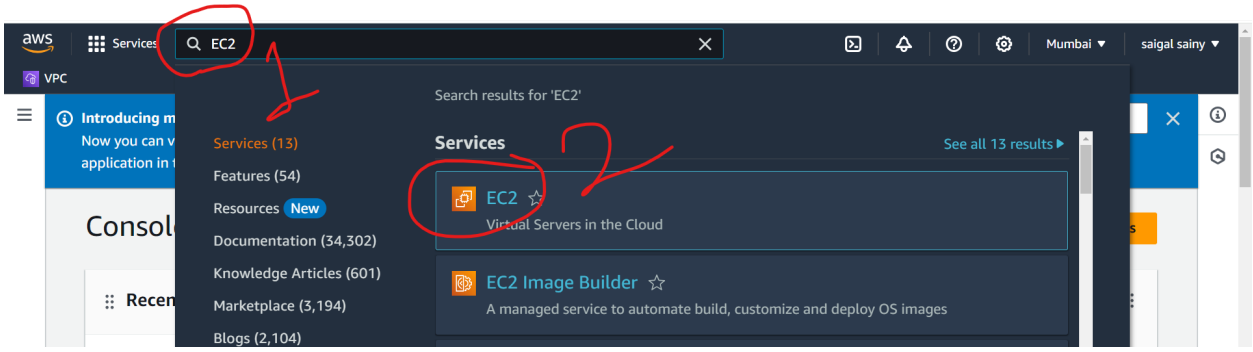
This project demonstrates the process of containerizing and deploying a Mario game on Amazon EKS, leveraging container orchestration capabilities for scalability and maintainability. It provides hands-on experience with Docker, Kubernetes, and Amazon EKS in a real-world scenario.

Project Configuration

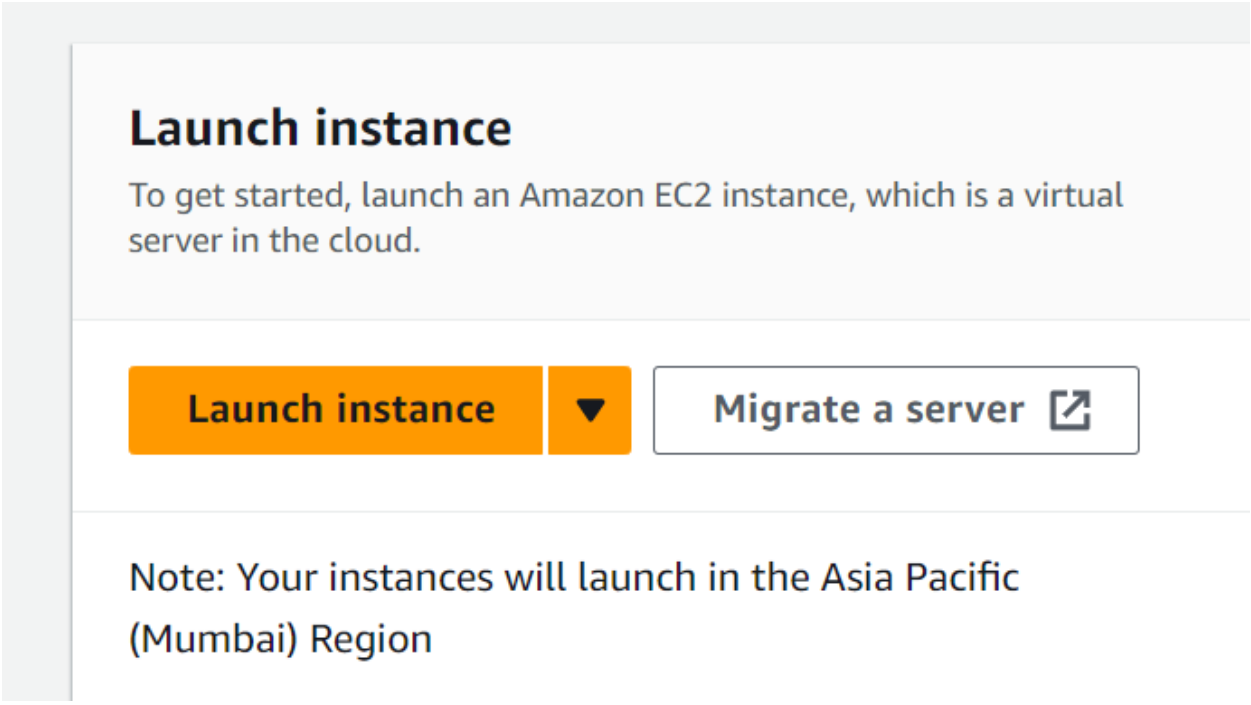
.

STEP :- 1 :- Create EC2 Instance by follow steps bellow.

1. Login your AWS account and search EC2 in search bar

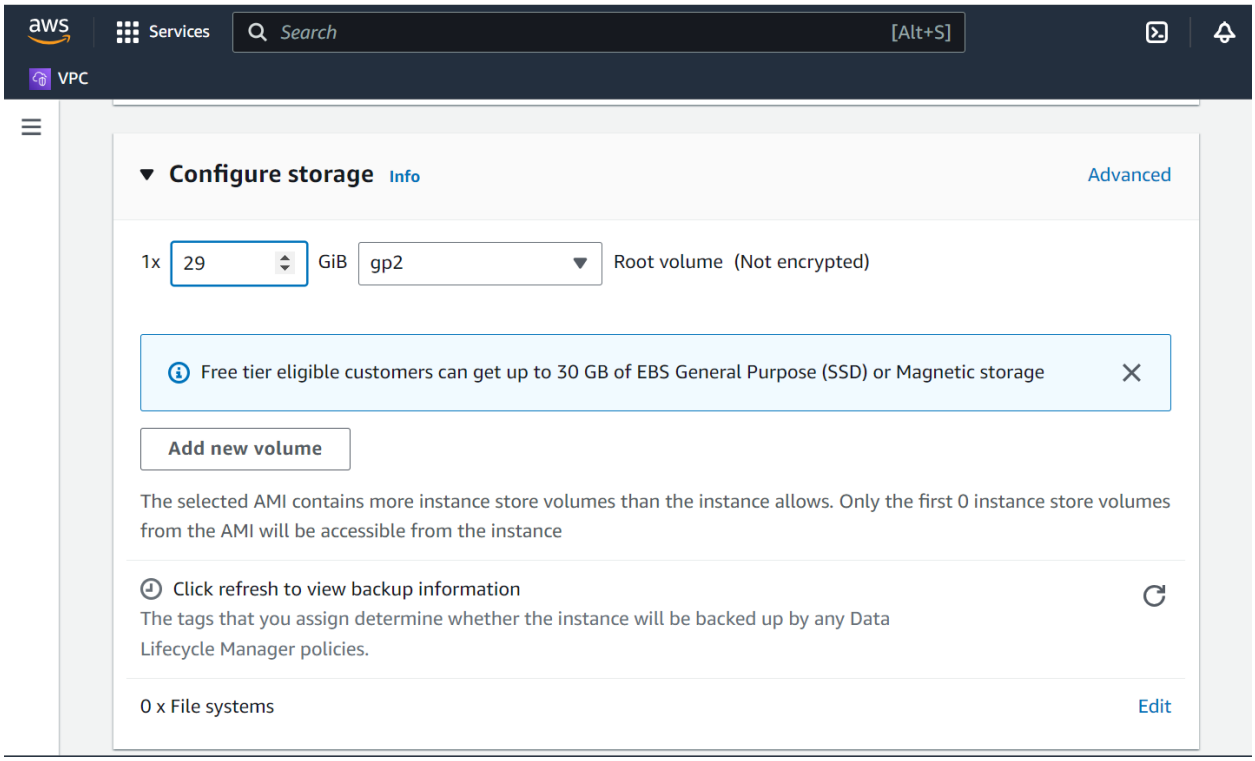


2. Go into EC2 and click on ‘Launch Instance’

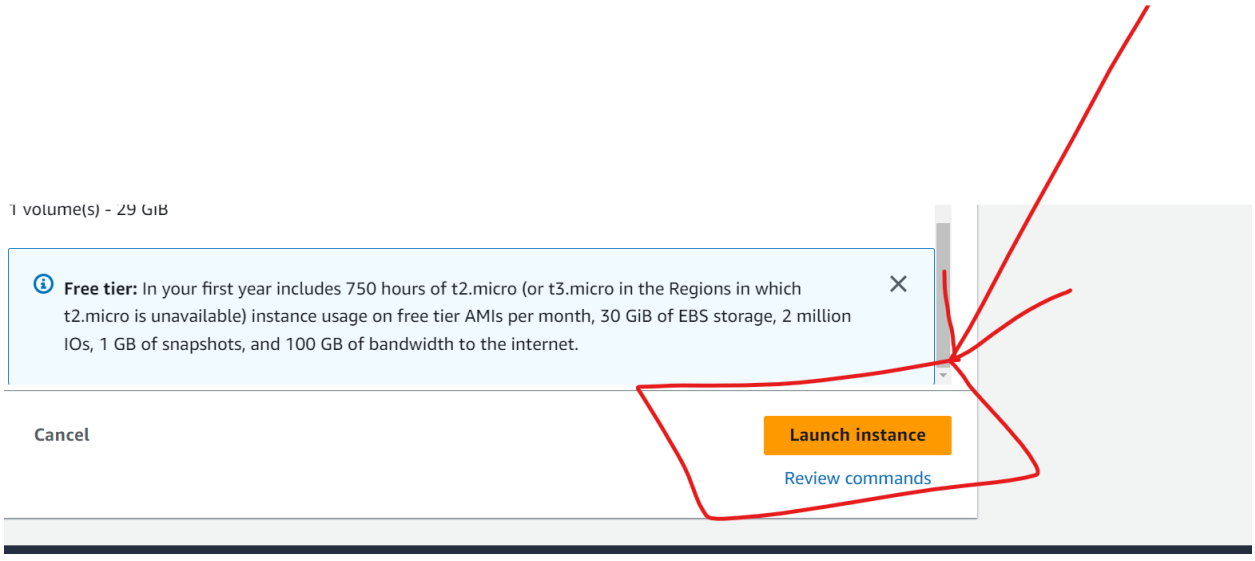


3. Instance Configuration
   Name - ‘as you want(mario)’
   AMI - ‘Ubuntu’
   Instance Type - t2.micro
   Key pair - create and select

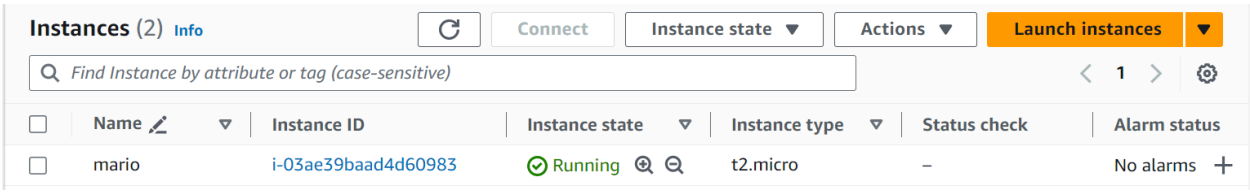
Network setting - ‘Default,Check HTTP & HTTPS’
Storage - ‘as you want(free >30GB)’



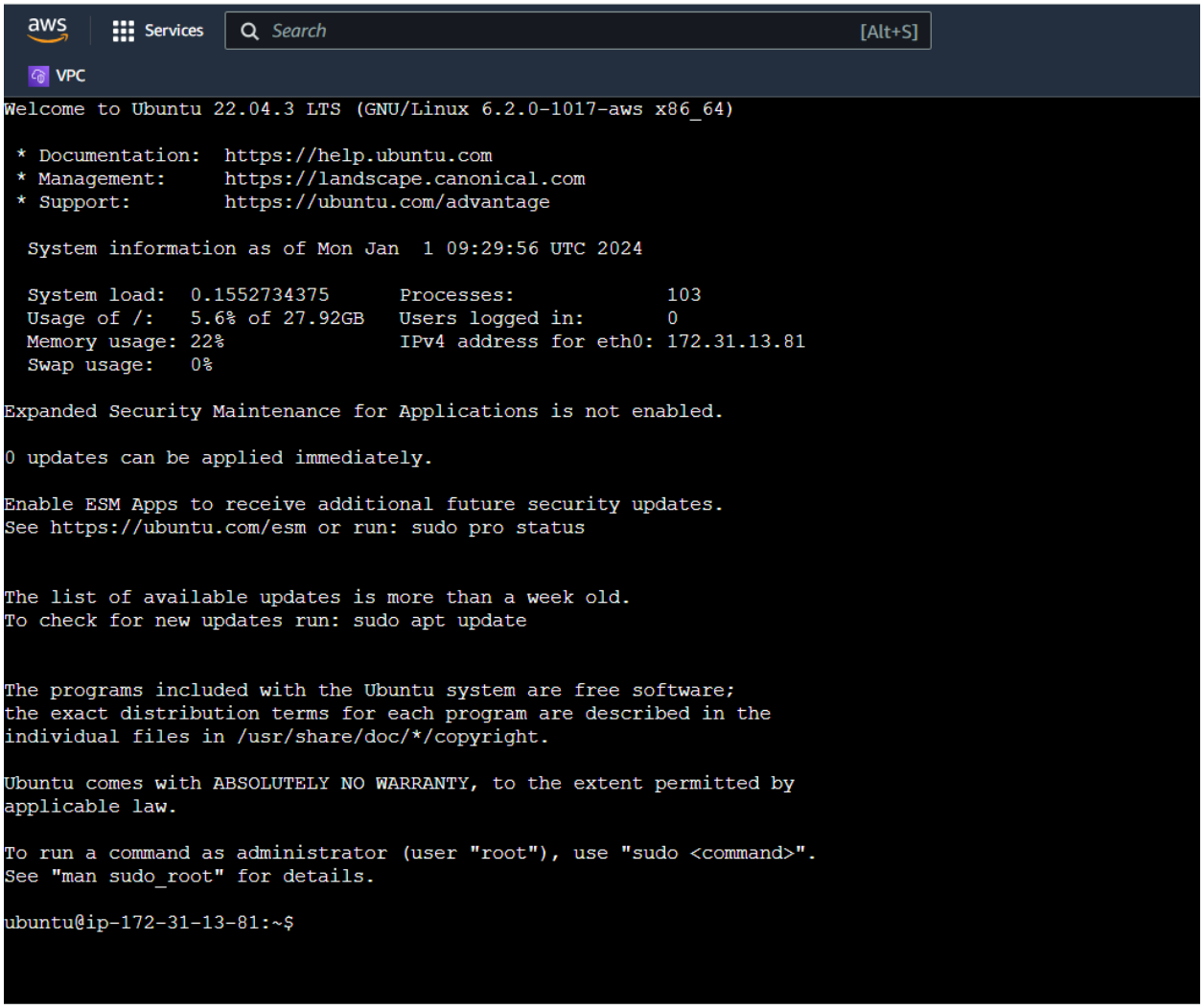
Summery(Number of instance) - 1
Click on ‘Launch Instance’



4. Instance created and running



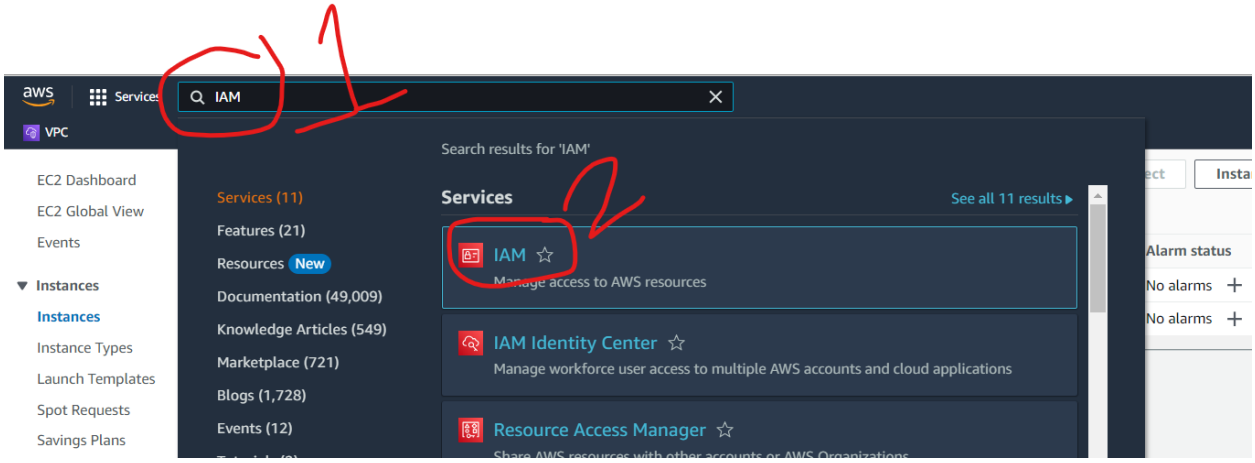
Connection medium - as you want(browser)



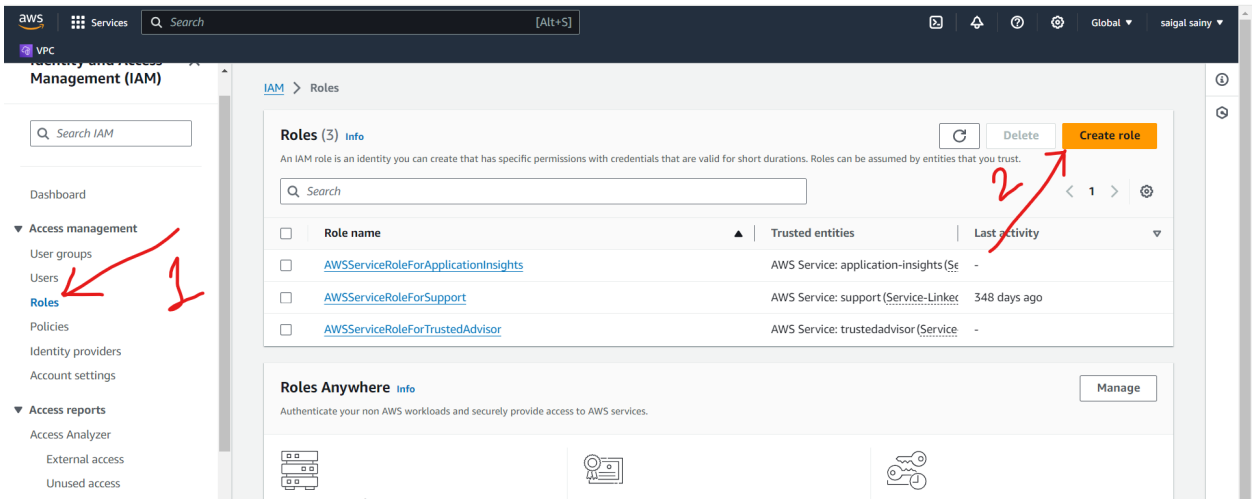
Machine connected

STEP :- 2→ Create roles from IAM

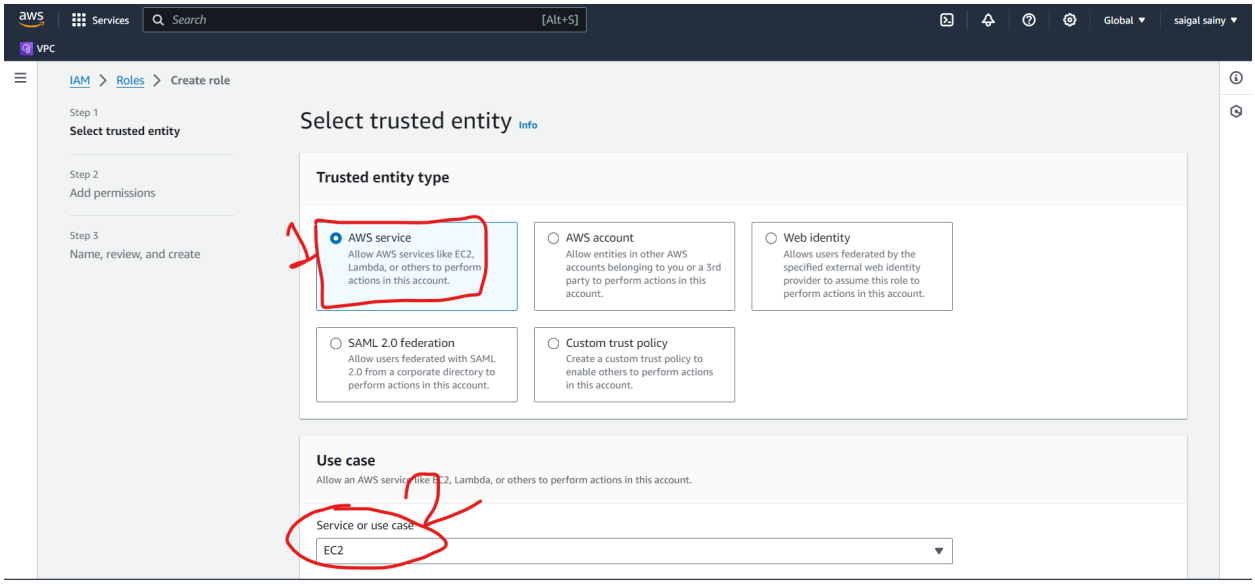
1. **Go into IAM by searching IAM on AWS Console**



2. **Click on** create role from left side bar then click on 'create role'

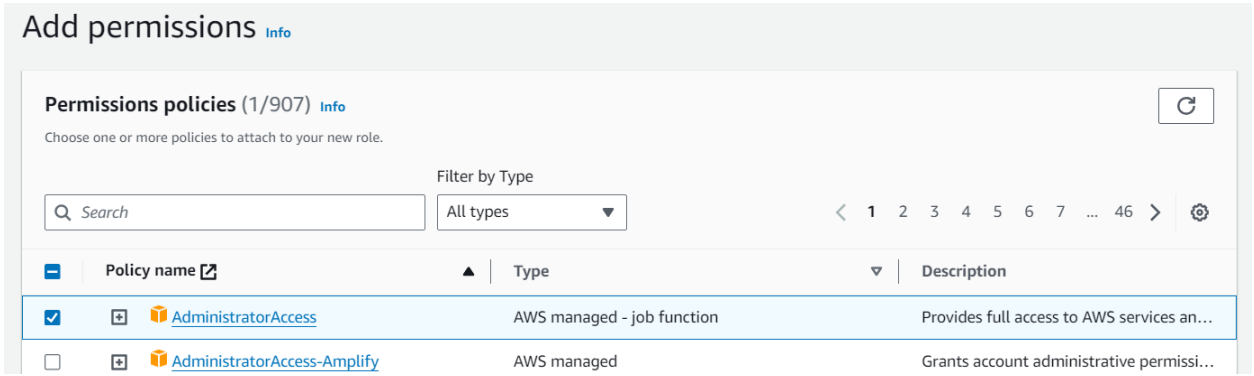


3. Role configuration
   Trusted entity - AWS service
   Use case - EC2



Press “Next”

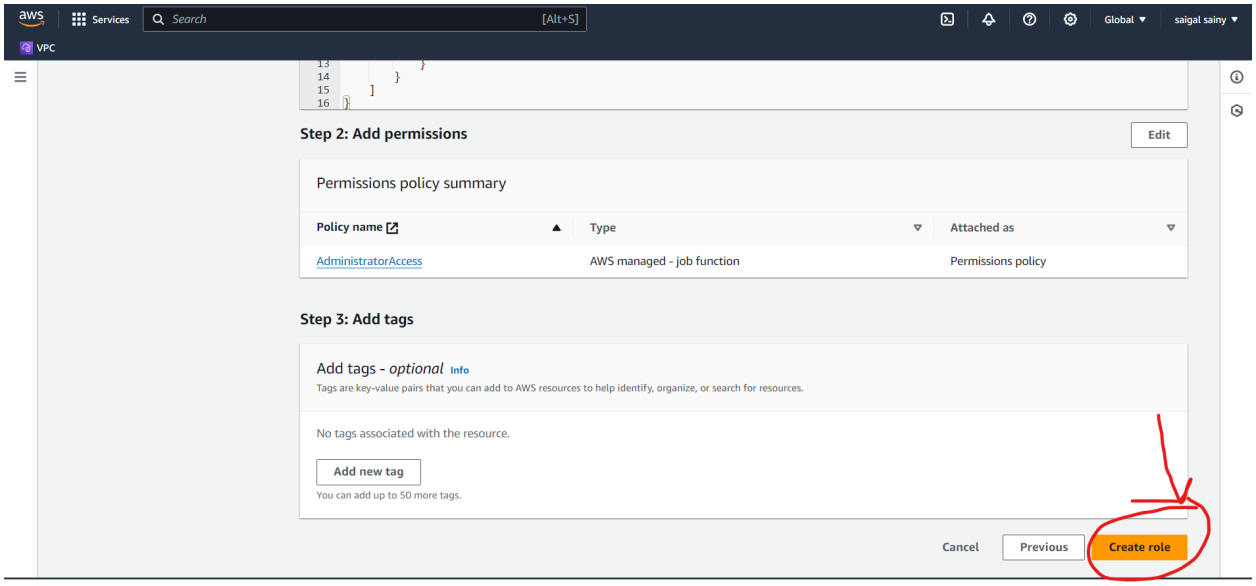
Permission policy - AdministratorAccess(for learning purpose)



Click ‘Next’

Role name - ‘as you want (Mario)’

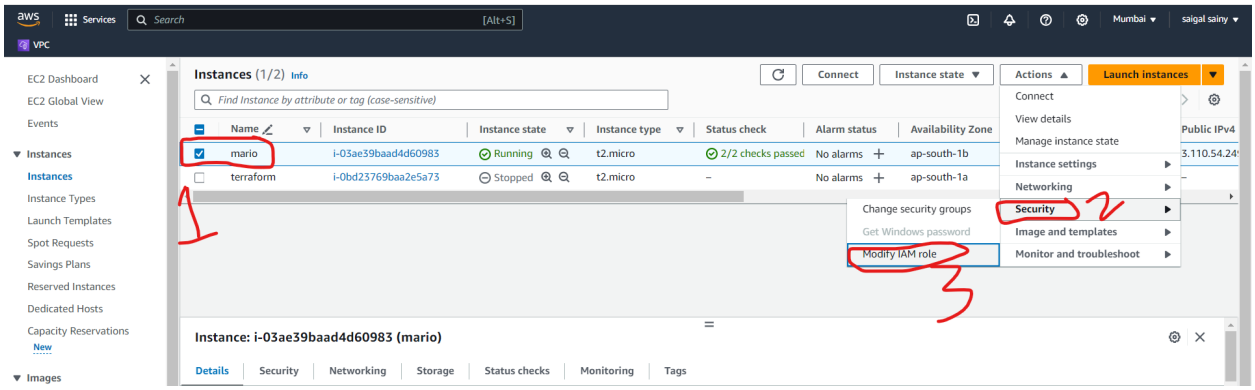
Click - ‘create role’



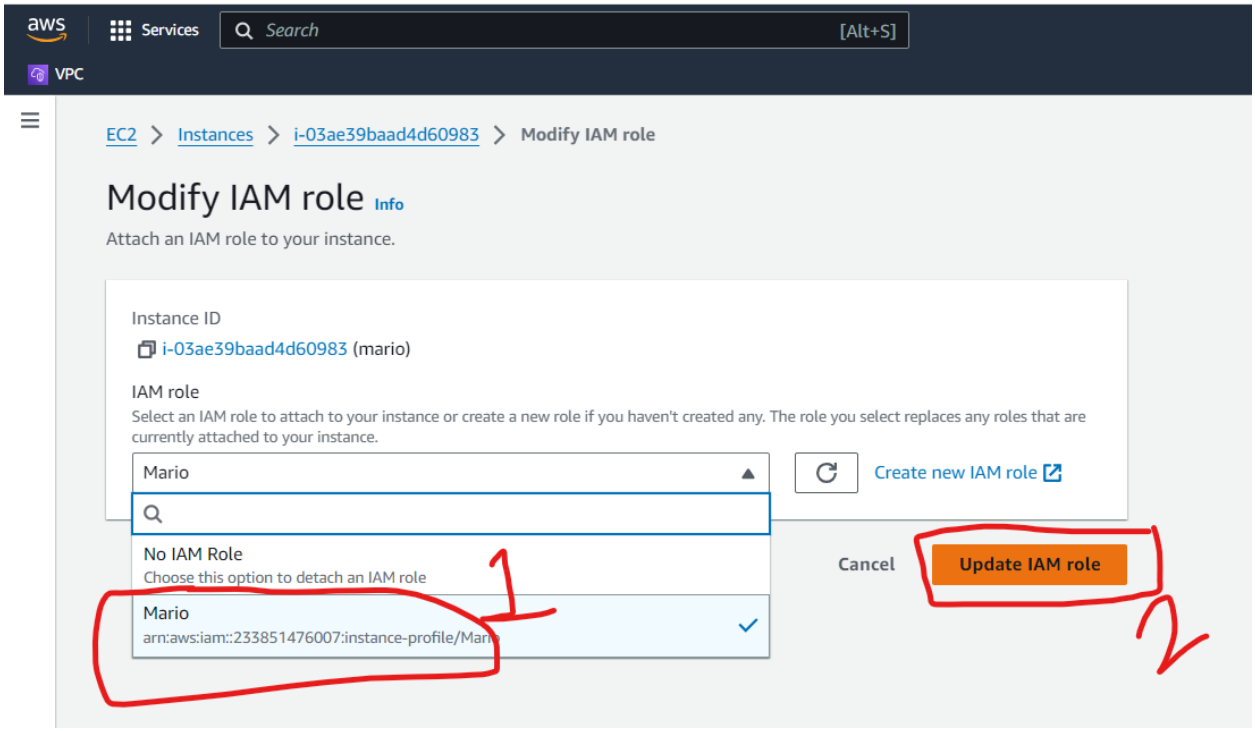
Role created

STEP :- 3→ Attach role in EC2 Instance

1. Go into EC2 Instance and select your created instance and follow steps bellow



2. Select your created role and click ‘Update IAM role’



Role updated

STEP :- 4→ Install kubectl, Terraform, AWS CLI & Docker

1. Go to EC2 instance and install it manually

Or

Write a script for install it

On Instance →

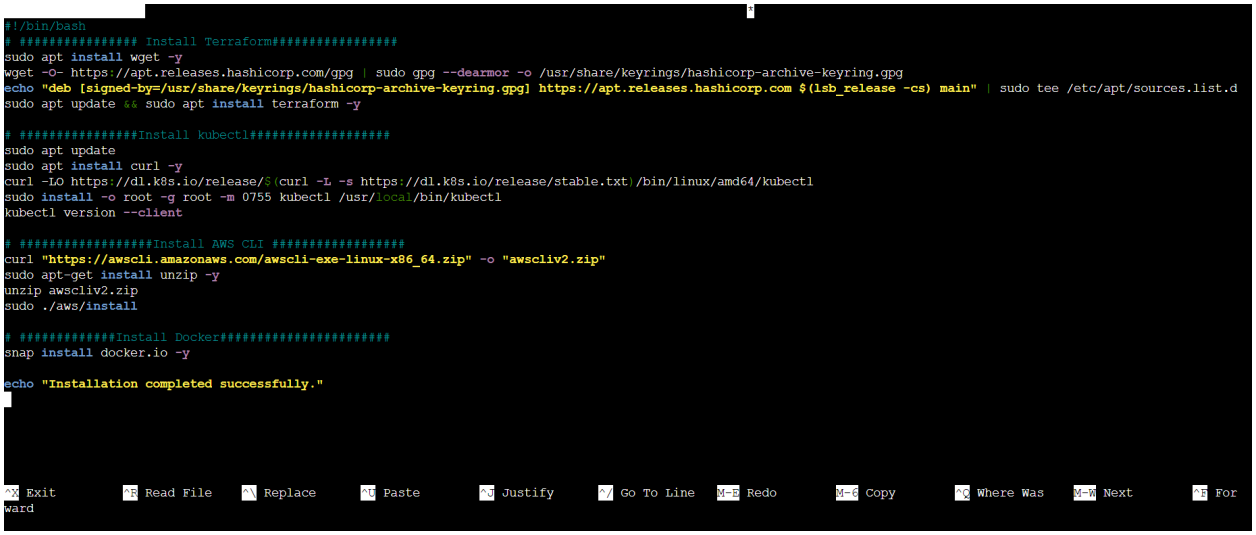
Create a file by extension '.sh' and write script

Script.sh

```
==========================================================
#!/bin/bash
# ################ Install Terraform#################
sudo apt install wget -y
wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg
--dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
echo "deb
[signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg]
https://apt.releases.hashicorp.com $(lsb_release -cs) main" |
sudo tee /etc/apt/sources.list.dsudo apt update && sudo apt
install terraform -y
```

```
# ################Install kubectl###################
sudo apt update
sudo apt install curl -y
curl -LO https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
sudo install -o root -g root -m 0755 kubectl
/usr/local/bin/kubectl
kubectl version --client

# ##################Install AWS CLI ##################
curl
"https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
sudo apt-get install unzip -y
unzip awscliv2.zip
sudo ./aws/install

# #############Install Docker#######################
snap install docker

echo "Installation completed successfully."
```

==========



Save the file and run script for install these packages
This script will install ‘Terraform’,’AWS CLI’,’Kubectl’&’Docker’

2. For check the installation run below commands

```
docker --version
aws --version
kubectl version --client
terraform --version
```

```
root@ip-172-31-13-81:~# docker --version
aws --version
kubectl version --client
terraform --version
Docker version 20.10.24, build 297e128
aws-cli/2.15.6 Python/3.11.6 Linux/6.2.0-1017-aws exe/x86_64.ubuntu.22 prompt/off
Client Version: v1.29.0
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Terraform v1.6.6
on linux_amd64
```

STEP :- 5→ Create K8s cluster on AWS by using terraform

1. **Create S3-Bucket**

   Follow this →

   https://github.com/chaudharysurya14/AWS-S3

2. **Go to** instance and create a directory

   →mkdir EKS-Terraform

   →cd EKS-Terraform

```
root@ip-172-31-13-81:~# mkdir EKS-Terraform
root@ip-172-31-13-81:~# cd EKS-Terraform/
root@ip-172-31-13-81:~/EKS-Terraform# nano K8s.tf
```

3. Create a terraform script file for make kubernetes cluster

   →nano backend.tf

```
terraform {
  backend "s3" {
    bucket = "surya0chy1bucket" # Replace with your bucket name
    key    = "EKS/terraform.tfstate"
    region = "ap-south-1" # Verify your region
  }
}
```

```
terraform {
  backend "s3" {
    bucket = "surya0chy1bucket" # Replace with your bucket name
    key    = "EKS/terraform.tfstate"
    region = "ap-south-1"
  }
}
```

4. Create a file by extension '.tf' and write terraform script for create role, EKS Cluster, VPC.

```
→nano main.tf
data "aws_iam_policy_document" "assume_role" {
  statement {
    effect = "Allow"

    principals {
      type        = "Service"
      identifiers = ["eks.amazonaws.com"]
    }

    actions = ["sts:AssumeRole"]
  }
}

resource "aws_iam_role" "example" {
  name               = "eks-cluster-cloud"
  assume_role_policy =
data.aws_iam_policy_document.assume_role.json
}

resource "aws_iam_role_policy_attachment"
"example-AmazonEKSClusterPolicy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
  role       = aws_iam_role.example.name
}

#get vpc data
data "aws_vpc" "default" {
  default = true
```

```
}
#get public subnets for cluster
data "aws_subnets" "public" {
  filter {
    name   = "vpc-id"
    values = [data.aws_vpc.default.id]
  }
}
#cluster provision
resource "aws_eks_cluster" "example" {
  name     = "EKS_CLOUD"
  role_arn = aws_iam_role.example.arn

  vpc_config {
    subnet_ids = data.aws_subnets.public.ids
  }

  # Ensure that IAM Role permissions are created before and
deleted after EKS Cluster handling.
  # Otherwise, EKS will not be able to properly delete EKS
managed EC2 infrastructure such as Security Groups.
  depends_on = [

aws_iam_role_policy_attachment.example-AmazonEKSClusterPolicy,
  ]
}

resource "aws_iam_role" "example1" {
  name = "eks-node-group-cloud"

  assume_role_policy = jsonencode({
    Statement = [{
      Action = "sts:AssumeRole"
      Effect = "Allow"
      Principal = {
        Service = "ec2.amazonaws.com"
      }
    }]
```

```
    Version = "2012-10-17"
  })
}

resource "aws_iam_role_policy_attachment"
"example-AmazonEKSWorkerNodePolicy" {
  policy_arn =
"arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
  role       = aws_iam_role.example1.name
}

resource "aws_iam_role_policy_attachment"
"example-AmazonEKS_CNI_Policy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"
  role       = aws_iam_role.example1.name
}

resource "aws_iam_role_policy_attachment"
"example-AmazonEC2ContainerRegistryReadOnly" {
  policy_arn =
"arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
  role       = aws_iam_role.example1.name
}

#create node group
resource "aws_eks_node_group" "example" {
  cluster_name    = aws_eks_cluster.example.name
  node_group_name = "Node-cloud"
  node_role_arn   = aws_iam_role.example1.arn
  subnet_ids      = data.aws_subnets.public.ids

  scaling_config {
    desired_size = 1
    max_size     = 2
    min_size     = 1
  }
  instance_types = ["t2.medium"]
```

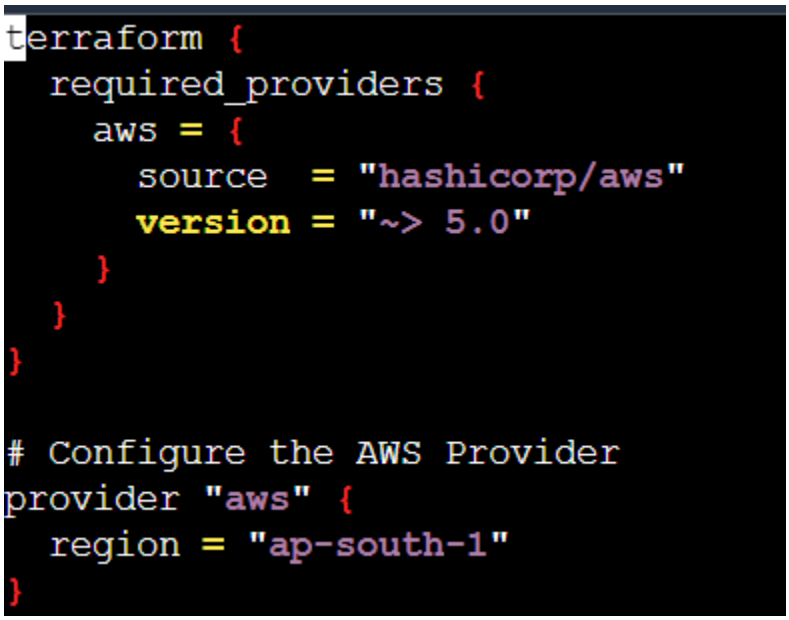
```
  # Ensure that IAM Role permissions are created before and
deleted after EKS Node Group handling.
  # Otherwise, EKS will not be able to properly delete EC2
Instances and Elastic Network Interfaces.
  depends_on = [

aws_iam_role_policy_attachment.example-AmazonEKSWorkerNodePolicy,
    aws_iam_role_policy_attachment.example-AmazonEKS_CNI_Policy,

aws_iam_role_policy_attachment.example-AmazonEC2ContainerRegistry
ReadOnly,
  ]
}
```

```
data "aws_iam_policy_document" "assume_role" {
  statement {
    effect = "Allow"

    principals {
      type        = "Service"
      identifiers = ["eks.amazonaws.com"]
    }

    actions = ["sts:AssumeRole"]
  }
}

resource "aws_iam_role" "example" {
  name               = "K8s-cluster-cloud"
  assume_role_policy = data.aws_iam_policy_document.assume_role.json
}

resource "aws_iam_role_policy_attachment" "example-AmazonEKSClusterPolicy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
  role       = aws_iam_role.example.name
}

#get vpc data
data "aws_vpc" "default" {
  default = true
-- INSERT --
```

5. Create a file by extension ‘.tf’ and write terraform script for provide AWS environment

   →nano provider.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "ap-south-1"
}
```



6. For apply terraform script and create environment run terraform commands
   →terraform init :-For install all plugins for create environment

```
root@ip-172-31-13-81:~/EKS-Terraform# terraform init

Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
root@ip-172-31-13-81:~/EKS-Terraform#
```

→terraform plan :- For see the changes that will take place on the infrastructure

```
root@ip-172-31-13-81:~/EKS-Terraform# terraform plan
data.aws_vpc.default: Reading...
data.aws_iam_policy_document.assume_role: Reading...
data.aws_iam_policy_document.assume_role: Read complete after 0s [id=3552664922]
data.aws_vpc.default: Read complete after 0s [id=vpc-03597436eda7c1e62]
data.aws_subnets.public: Reading...
data.aws_subnets.public: Read complete after 0s [id=ap-south-1]

Terraform used the selected providers to generate the following execution plan. Resource actions are indica
  + create

Terraform will perform the following actions:

  # aws_eks_cluster.example will be created
  + resource "aws_eks_cluster" "example" {
      + arn                   = (known after apply)
      + certificate_authority = (known after apply)
      + cluster_id            = (known after apply)
      + created_at            = (known after apply)
      + endpoint              = (known after apply)
      + id                    = (known after apply)
      + identity              = (known after apply)
      + name                  = "EKS_CLOUD"
      + platform_version      = (known after apply)
      + role_arn              = (known after apply)
      + status                = (known after apply)
      + tags_all              = (known after apply)
      + version               = (known after apply)

      + vpc_config {
          + cluster_security_group_id = (known after apply)
          + endpoint_private_access   = false
          + endpoint_public_access    = true
          + public_access_cidrs       = (known after apply)
```

```
  # aws_iam_role_policy_attachment.example-AmazonEC2ContainerRegistryReadOnly will be created
  + resource "aws_iam_role_policy_attachment" "example-AmazonEC2ContainerRegistryReadOnly" {
      + id         = (known after apply)
      + policy_arn = "arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly"
      + role       = "eks-node-group-cloud"
    }

  # aws_iam_role_policy_attachment.example-AmazonEKSClusterPolicy will be created
  + resource "aws_iam_role_policy_attachment" "example-AmazonEKSClusterPolicy" {
      + id         = (known after apply)
      + policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
      + role       = "K8s-cluster-cloud"
    }

  # aws_iam_role_policy_attachment.example-AmazonEKSWorkerNodePolicy will be created
  + resource "aws_iam_role_policy_attachment" "example-AmazonEKSWorkerNodePolicy" {
      + id         = (known after apply)
      + policy_arn = "arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy"
      + role       = "eks-node-group-cloud"
    }

  # aws_iam_role_policy_attachment.example-AmazonEKS_CNI_Policy will be created
  + resource "aws_iam_role_policy_attachment" "example-AmazonEKS_CNI_Policy" {
      + id         = (known after apply)
      + policy_arn = "arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy"
      + role       = "eks-node-group-cloud"
    }

Plan: 8 to add, 0 to change, 0 to destroy.


Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these acti
root@ip-172-31-13-81:~/EKS-Terraform#
```

→terraform apply :- For create infrastructure through script

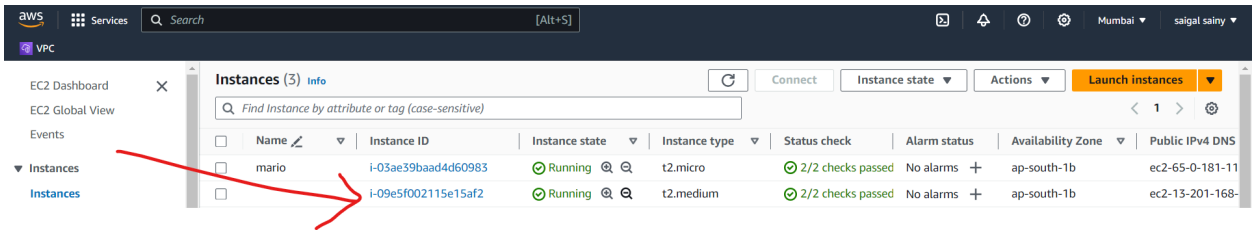
```
root@ip-172-31-13-81:~/EKS-Terraform# terraform apply
data.aws_iam_policy_document.assume_role: Reading...
data.aws_vpc.default: Reading...
data.aws_iam_policy_document.assume_role: Read complete after 0s [id=3552664922]
data.aws_vpc.default: Read complete after 1s [id=vpc-03597436eda7c1e62]
data.aws_subnets.public: Reading...
data.aws_subnets.public: Read complete after 0s [id=ap-south-1]

Terraform used the selected providers to generate the following execution plan. Resourc
  + create

Terraform will perform the following actions:

  # aws_eks_cluster.example will be created
  + resource "aws_eks_cluster" "example" {
      + arn                   = (known after apply)
      + certificate_authority = (known after apply)
      + cluster_id            = (known after apply)
      + created_at            = (known after apply)
      + endpoint              = (known after apply)
      + id                    = (known after apply)
      + identity              = (known after apply)
      + name                  = "EKS_CLOUD"
      + platform_version      = (known after apply)
      + role_arn              = (known after apply)
      + status                = (known after apply)
      + tags_all              = (known after apply)
      + version               = (known after apply)

      + vpc_config {
          + cluster_security_group_id = (known after apply)
          + endpoint_private_access   = false
          + endpoint_public_access    = true
          + public_access_cidrs       = (known after apply)
```

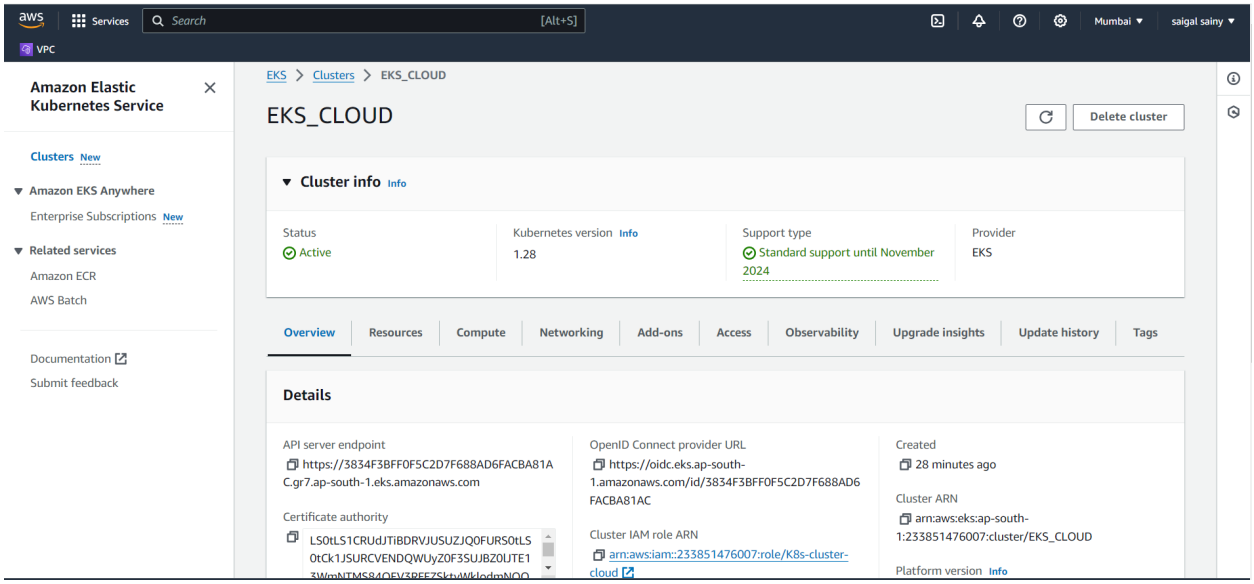
It’ll ask for confirmation then press ‘yes’

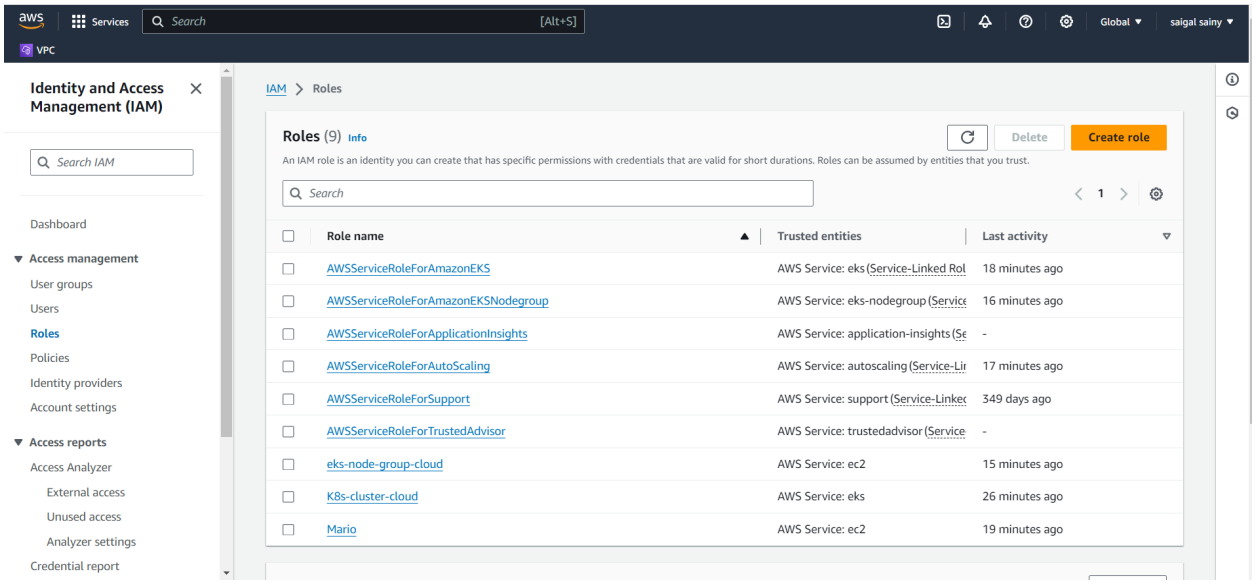
```
Plan: 8 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_iam_role.example1: Creating...
aws_iam_role.example: Creating...
aws_iam_role.example1: Creation complete after 2s [id=eks-node-group-cloud]
aws_iam_role.example: Creation complete after 2s [id=K8s-cluster-cloud]
aws_iam_role_policy_attachment.example-AmazonEKSWorkerNodePolicy: Creating...
aws_iam_role_policy_attachment.example-AmazonEC2ContainerRegistryReadOnly: Creating...
aws_iam_role_policy_attachment.example-AmazonEKS_CNI_Policy: Creating...
aws_iam_role_policy_attachment.example-AmazonEKSClusterPolicy: Creating...
aws_iam_role_policy_attachment.example-AmazonEKS_CNI_Policy: Creation complete after 1s [id=eks-node-group-cloud-20240102102822227500000001]
aws_iam_role_policy_attachment.example-AmazonEKSWorkerNodePolicy: Creation complete after 1s [id=eks-node-group-cloud-20240102102822274400000002]
aws_iam_role_policy_attachment.example-AmazonEC2ContainerRegistryReadOnly: Creation complete after 1s [id=eks-node-group-cloud-20240102102822471800000003]
aws_iam_role_policy_attachment.example-AmazonEKSClusterPolicy: Creation complete after 1s [id=K8s-cluster-cloud-20240102102822510600000004]
aws_eks_cluster.example: Creating...
aws_eks_cluster.example: Still creating... [10s elapsed]
aws_eks_cluster.example: Still creating... [20s elapsed]
aws_eks_cluster.example: Still creating... [30s elapsed]
aws_eks_cluster.example: Still creating... [40s elapsed]
aws_eks_cluster.example: Still creating... [50s elapsed]
aws_eks_cluster.example: Still creating... [1m0s elapsed]
aws_eks_cluster.example: Still creating... [1m10s elapsed]
aws_eks_cluster.example: Still creating... [1m20s elapsed]
aws_eks_cluster.example: Still creating... [1m30s elapsed]
aws_eks_cluster.example: Still creating... [1m40s elapsed]
aws_eks_cluster.example: Still creating... [1m50s elapsed]
aws_eks_cluster.example: Still creating... [2m0s elapsed]
```

```
aws_eks_cluster.example: Still creating... [11m50s elapsed]
aws_eks_cluster.example: Still creating... [12m0s elapsed]
aws_eks_cluster.example: Still creating... [12m10s elapsed]
aws_eks_cluster.example: Still creating... [12m20s elapsed]
aws_eks_cluster.example: Creation complete after 12m26s [id=EKS_CLOUD]
aws_eks_node_group.example: Creating...
aws_eks_node_group.example: Still creating... [10s elapsed]
aws_eks_node_group.example: Still creating... [20s elapsed]
aws_eks_node_group.example: Still creating... [30s elapsed]
aws_eks_node_group.example: Still creating... [40s elapsed]
aws_eks_node_group.example: Still creating... [50s elapsed]
aws_eks_node_group.example: Still creating... [1m0s elapsed]
aws_eks_node_group.example: Still creating... [1m10s elapsed]
aws_eks_node_group.example: Still creating... [1m20s elapsed]
aws_eks_node_group.example: Still creating... [1m30s elapsed]
aws_eks_node_group.example: Still creating... [1m40s elapsed]
aws_eks_node_group.example: Still creating... [1m50s elapsed]
aws_eks_node_group.example: Still creating... [2m0s elapsed]
aws_eks_node_group.example: Still creating... [2m10s elapsed]
aws_eks_node_group.example: Still creating... [2m20s elapsed]
aws_eks_node_group.example: Still creating... [2m30s elapsed]
aws_eks_node_group.example: Still creating... [2m40s elapsed]
aws_eks_node_group.example: Still creating... [2m50s elapsed]
aws_eks_node_group.example: Still creating... [3m0s elapsed]
aws_eks_node_group.example: Still creating... [3m10s elapsed]
aws_eks_node_group.example: Still creating... [3m20s elapsed]
aws_eks_node_group.example: Still creating... [3m30s elapsed]
aws_eks_node_group.example: Still creating... [3m40s elapsed]
aws_eks_node_group.example: Still creating... [3m50s elapsed]
aws_eks_node_group.example: Still creating... [4m0s elapsed]
aws_eks_node_group.example: Creation complete after 4m0s [id=EKS_CLOUD:EKS_CLUSTER]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.
root@ip-172-31-13-81:~/EKS-Terraform#
```

Cluster created







STEP :- 6→ Update cube config configuration by below command

→aws eks update-kubeconfig --name EKS_CLOUD --region ap-south-1

EKS_CLOUD → kubernetes cluster name

Ap-south-1 → Your region

```
root@ip-172-31-13-81:~/EKS-Terraform# aws eks update-kubeconfig --name EKS_CLOUD --region ap-south-1
Added new context arn:aws:eks:ap-south-1:233851476007:cluster/EKS_CLOUD to /root/.kube/config
root@ip-172-31-13-81:~/EKS-Terraform#
```

STEP :- 7→ Deploy application on EKS

1. Create a deployment.yaml file and write script for deploy application on EKS

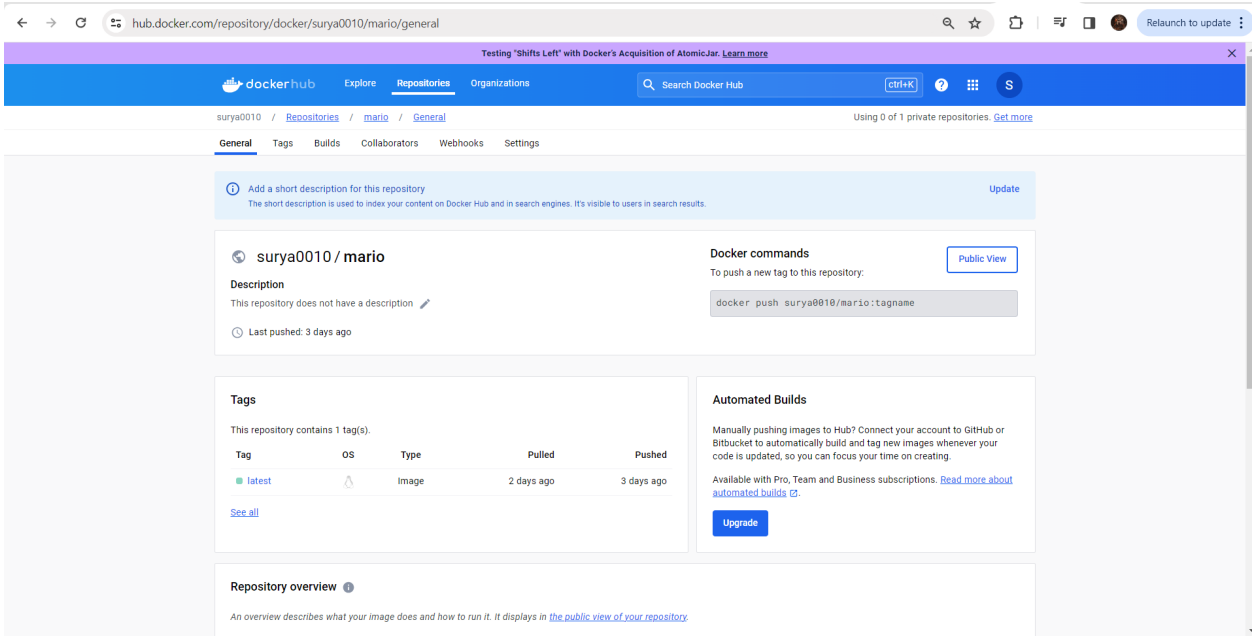
→ nano deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mario-deployment
spec:
  replicas: 2  # You can adjust the number of replicas as needed
  selector:
    matchLabels:
      app: mario
  template:
    metadata:
      labels:
        app: mario
    spec:
      containers:
      - name: mario-container
        image: surya001/mario:latest
        ports:
        - containerPort: 80
```

```
 GNU nano 6.2
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mario-deployment
spec:
  replicas: 2  # You can adjust the number of replicas as needed
  selector:
    matchLabels:
      app: mario
  template:
    metadata:
      labels:
        app: mario
    spec:
      containers:
      - name: mario-container
        image: surya001/mario:latest
        ports:
        - containerPort: 80
```

Here i'm using docker mario image

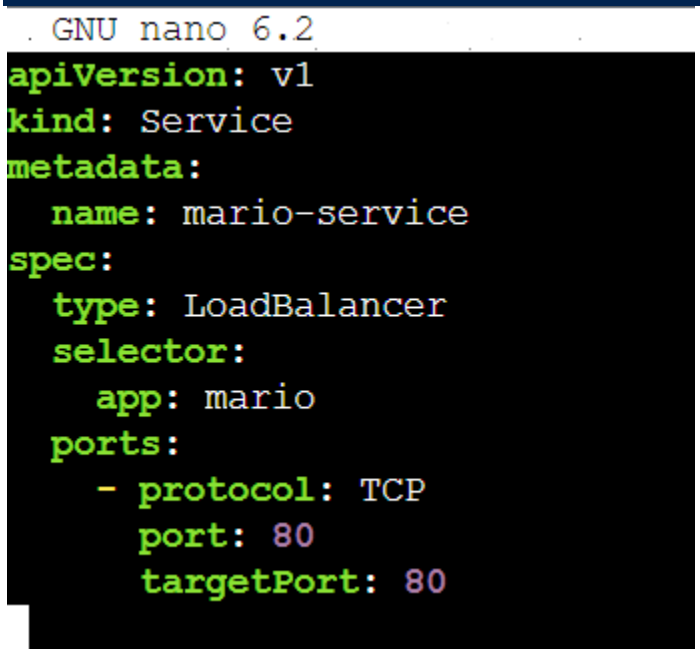
Name → "surya0010/mario:latest"



2. Create service.yaml file for configuration of K8s

→service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mario-service
spec:
  type: LoadBalancer
  selector:
    app: mario
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```



3. Run this command for deploy the mario application

→kubectl apply -f deployment.yaml

```
root@ip-172-31-13-81:~/EKS-Terraform# kubectl apply -f deployment.yaml
deployment.apps/mario-deployment created
root@ip-172-31-13-81:~/EKS-Terraform#
```

→kubectl apply -f service.yaml

```
root@ip-172-31-13-81:~/EKS-Terraform# kubectl apply -f service.yaml
service/mario-service created
root@ip-172-31-13-81:~/EKS-Terraform#
```

4. For check/confirm the deployment run below command
   →kubectl get all

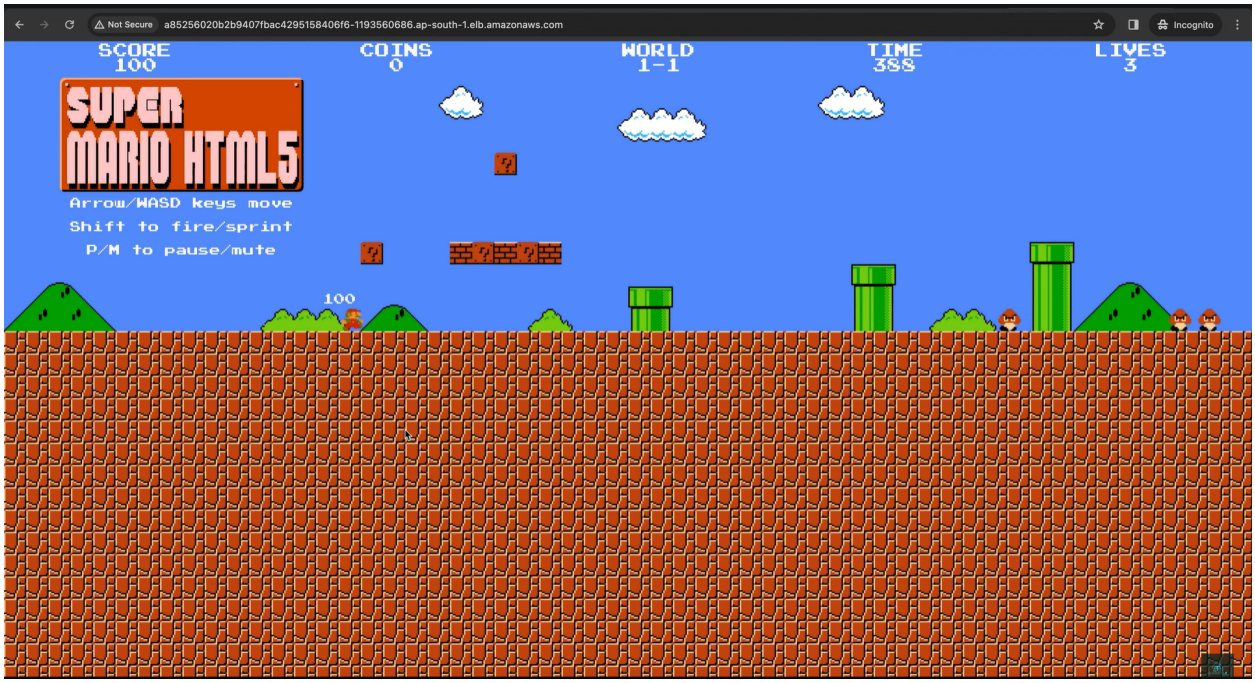
```
root@ip-172-31-41-147:/home/ubuntu/k8s-mario#
root@ip-172-31-41-147:/home/ubuntu/k8s-mario# kubectl apply -f service.yaml
service/mario-service created
root@ip-172-31-41-147:/home/ubuntu/k8s-mario# kubectl get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/mario-deployment-78cbc65cb-9n547   1/1     Running   0          47s
pod/mario-deployment-78cbc65cb-stxkr   1/1     Running   0          47s

NAME                    TYPE           CLUSTER-IP      EXTERNAL-IP                                                              PORT(S)        AGE
service/kubernetes      ClusterIP      10.100.0.1      <none>                                                                   443/TCP        7m42s
service/mario-service   LoadBalancer   10.100.119.16   afe22fd27c11b42c0a7c486290725ad8-21137843.ap-south-1.elb.amazonaws.com   80:30746/TCP   14s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mario-deployment   2/2     2            2           47s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/mario-deployment-78cbc65cb    2         2         2       47s
root@ip-172-31-41-147:/home/ubuntu/k8s-mario#
```

5. For get information about your deployed application
   →kubectl describe service mario-service

```
root@ip-172-31-13-81:~/EKS-Terraform# kubectl describe service mario-service
Name:                     mario-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=mario
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.100.45.55
IPs:                      10.100.45.55
LoadBalancer Ingress:     a063a4cb5b6de46c8926dcf1a3182f28-92974309.ap-south-1.elb.amazonaws.com
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31949/TCP
Endpoints:
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason                Age   From                Message
  ----    ------                ----  ----                -------
  Normal  EnsuringLoadBalancer  4m5s  service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer   4m1s  service-controller  Ensured load balancer
root@ip-172-31-13-81:~/EKS-Terraform#
```

6. For access the application copy this and paste on browser

```
root@ip-172-31-13-81:~/EKS-Terraform# kubectl describe service mario-service
Name:                     mario-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=mario
Type:                     LoadBalancer
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.100.45.55
IPs:                      10.100.45.55
LoadBalancer Ingress:     a063a4cb5b6de46c8926dcf1a3182f28-92974309.ap-south-1.elb.amazonaws.com
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31949/TCP
Endpoints:
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason                Age   From                Message
  ----    ------                ----  ----                -------
  Normal  EnsuringLoadBalancer  4m5s  service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer   4m1s  service-controller  Ensured load balancer
root@ip-172-31-13-81:~/EKS-Terraform#
```



STEP :- 7→ Delete your project

1. **For delete the pr**oject run below command
   →kubectl delete deployment mario-service

2. Delete EKS cluster using bellow command
   → terraform destroy --auto-approve

```
root@ip-172-31-13-81:~/EKS-Terraform# terraform destroy --auto-approve
data.aws_iam_policy_document.assume_role: Reading...
data.aws_vpc.default: Reading...
aws_iam_role.example1: Refreshing state... [id=eks-node-group-cloud]
data.aws_iam_policy_document.assume_role: Read complete after 0s [id=3552664922]
aws_iam_role.example: Refreshing state... [id=K8s-cluster-cloud]
data.aws_vpc.default: Read complete after 1s [id=vpc-03597436eda7c1e62]
data.aws_subnets.public: Reading...
data.aws_subnets.public: Read complete after 0s [id=ap-south-1]
aws_iam_role_policy_attachment.example-AmazonEKSClusterPolicy: Refreshing state... [id=K8s-cluster-cloud-20240102102822510600000004]
aws_iam_role_policy_attachment.example-AmazonEKSWorkerNodePolicy: Refreshing state... [id=eks-node-group-cloud-20240102102822274400000002]
aws_iam_role_policy_attachment.example-AmazonEKS_CNI_Policy: Refreshing state... [id=eks-node-group-cloud-20240102102822227500000001]
aws_iam_role_policy_attachment.example-AmazonEC2ContainerRegistryReadOnly: Refreshing state... [id=eks-node-group-cloud-20240102102822471800000003]
aws_eks_cluster.example: Refreshing state... [id=EKS_CLOUD]
aws_eks_node_group.example: Refreshing state... [id=EKS_CLOUD:EKS_CLUSTER]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_eks_cluster.example will be destroyed
  - resource "aws_eks_cluster" "example" {
      - arn                   = "arn:aws:eks:ap-south-1:233851476007:cluster/EKS_CLOUD" -> null
      - certificate_authority = [
          - {
              - data = "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJTE13WmNTMS84OFV3RFFZSktvWklodmNOQVFFTEJRQXdGVEVUTUJFR0ExVUUKQXhNS2EzVmlaWEp1Wl
hSbGN6QWVGdzB5TkRBeE1ESXhNREk0TURoYUZ3MHpNekV5TXpBeE1ETXpNRGhhTUJVeApFekFSQmdOVkJBTVRDbXQxWW1WeWJtVjBaWE13Z2dFaU1BMEdEU3FHU0liM0RRRUJBUUVVBQTRJQkR3QXdnZ0VLCkFvSUJBUURh
dVNwalIzSlcvN1JkM0JreUNTbGNDKytGcVJKbDJjYmxZQmY3Z0hiL2VTSjhpbHN6NkxPaHRWdisKVkhvVXdYZ09sWlFnYlZ4QzRuOVZKREdVUHVRUXcxWFNBQk05L0UwaFRqbzJQT0ZCOVA5YUJudjVYajVjZm9rTQo3WV
VMa0dLV0xpYUlMeHJHcURramVqMklURlZ0TFllUU8zNWdaWm9ETVpHTUlvN0txYy9FOUFXNFBtdk5iRFhzCkppZWdJc2RnSERKcWZia1FCWWNSZ3BrQXR3U2orNWdURUhEMEQzVUIxc3JXMEtxMVFGZ0RYZFduTFJhU3N6
bWEKanJvRmJMR3JiZ1NIWGx1Tlc1TkZTWDZ1UUxaWDF5Y1FERnQ5aitLNDNyQ0dBWmU4VE5ZKytJWXFBd0wyOXNHaQpCM2gzRTV1QVFFL0x3WWkvOHN1M3ZyRVNYcHhiQWdNQkFBR2pXVEJYTUE0R0ExVWREd0VCL3dRRU
F3SUNwREFQCkJnTlZIUk1CQWY4RUJUQURBUUgvTUIwR0ExVWREZ1FXQkJTUHNEOWJ5M2RINGpmOWc1TVFGbk11Y3dhRVJEQVYKQmdOVkhSRUVakFNZ2dwcmRXSmxjbTVsZEdWek1BMEdDU3FHU0liM0RRRUJDd1VBQTRJ
QkFRQzhCcU1ORnZkWQoxUzlnVEhHU1F4YjNGTFFpZmpuN0ZzbWNwZVJydVhiSjlHekhqbWM1WE5EMzR3aWNlOGxGZnFhVGkyb0hqTnBECnNLK0RxMnY2aGdNZ3Qzb3FVMmRFaVhCNWtVZFp5aFR1MUtzeVV3MzgrWkxSaz
cwWm9ZN1U3bDV4SDBYMDV0a1QKaXNxZ1lmRVZLM3A5OU5XWGlGakFSb24vVjNMenFqVXRVQ0xnVnhtQ2EzY1F6MnNVL2k2dVBJdWpaSHk2bnBRTQo3aTFGczFDdzM5Ukh5NW1VZFR2SEFtc2tKYnlDY3lPeEFSTVI3aTA5
```

```
aws_eks_node_group.example: Still destroying... [id=EKS_CLOUD:EKS_CLUSTER, 7m0s elapsed]
aws_eks_node_group.example: Still destroying... [id=EKS_CLOUD:EKS_CLUSTER, 7m10s elapsed]
aws_eks_node_group.example: Still destroying... [id=EKS_CLOUD:EKS_CLUSTER, 7m20s elapsed]
aws_eks_node_group.example: Still destroying... [id=EKS_CLOUD:EKS_CLUSTER, 7m30s elapsed]
aws_eks_node_group.example: Still destroying... [id=EKS_CLOUD:EKS_CLUSTER, 7m40s elapsed]
aws_eks_node_group.example: Still destroying... [id=EKS_CLOUD:EKS_CLUSTER, 7m50s elapsed]
aws_eks_node_group.example: Still destroying... [id=EKS_CLOUD:EKS_CLUSTER, 8m0s elapsed]
aws_eks_node_group.example: Still destroying... [id=EKS_CLOUD:EKS_CLUSTER, 8m10s elapsed]
aws_eks_node_group.example: Still destroying... [id=EKS_CLOUD:EKS_CLUSTER, 8m20s elapsed]
aws_eks_node_group.example: Destruction complete after 8m21s
aws_iam_role_policy_attachment.example-AmazonEKS_CNI_Policy: Destroying... [id=eks-node-group-clou
aws_iam_role_policy_attachment.example-AmazonEKSWorkerNodePolicy: Destroying... [id=eks-node-group-
aws_eks_cluster.example: Destroying... [id=EKS_CLOUD]
aws_iam_role_policy_attachment.example-AmazonEC2ContainerRegistryReadOnly: Destroying... [id=eks-n
aws_iam_role_policy_attachment.example-AmazonEC2ContainerRegistryReadOnly: Destruction complete af
aws_iam_role_policy_attachment.example-AmazonEKS_CNI_Policy: Destruction complete after 1s
aws_iam_role_policy_attachment.example-AmazonEKSWorkerNodePolicy: Destruction complete after 1s
aws_iam_role.example1: Destroying... [id=eks-node-group-cloud]
aws_iam_role.example1: Destruction complete after 1s
aws_eks_cluster.example: Still destroying... [id=EKS_CLOUD, 10s elapsed]
aws_eks_cluster.example: Still destroying... [id=EKS_CLOUD, 20s elapsed]
aws_eks_cluster.example: Still destroying... [id=EKS_CLOUD, 30s elapsed]
aws_eks_cluster.example: Still destroying... [id=EKS_CLOUD, 40s elapsed]
aws_eks_cluster.example: Still destroying... [id=EKS_CLOUD, 50s elapsed]
aws_eks_cluster.example: Still destroying... [id=EKS_CLOUD, 1m0s elapsed]
aws_eks_cluster.example: Still destroying... [id=EKS_CLOUD, 1m10s elapsed]
aws_eks_cluster.example: Destruction complete after 1m16s
aws_iam_role_policy_attachment.example-AmazonEKSClusterPolicy: Destroying... [id=K8s-cluster-cloud
aws_iam_role_policy_attachment.example-AmazonEKSClusterPolicy: Destruction complete after 1s
aws_iam_role.example: Destroying... [id=K8s-cluster-cloud]
aws_iam_role.example: Destruction complete after 0s

Destroy complete! Resources: 8 destroyed.
root@ip-172-31-13-81:~/EKS-Terraform#
```

Cluster destroyed.

Thank You

Please like and share this project to your friends
Thank you for visit 👋