

SUPER MARIO GAME DEPLOYMENT

(AWS DEVOPS PROJECT)

Prerequisite →

1. We will use AWS account .
2. Then we configure Terraform inside an ec2 instance
3. we also need an IAM ec2 role that provide necessary permission to ec2

Completion steps →

Step 1 → Login and basics setup

Step 2 → Setup Docker ,Terraform ,aws cli , and Kubectl

Step 3 → IAM Role for EC2

Step 4 → Attach IAM role with your EC2

Step 5 → Building Infrastructure Using terraform

Step 6 → Creation of deployment and service for EKS

Step 7 → Destroy all the Insrastucture

Steps:

Step 1 : First we will Launch EC2 machine

Go to aws account

Go to Instances ->Launch instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

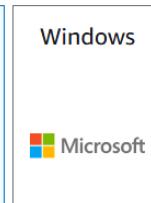
Name

[Add additional tags](#)

▼ Application and OS Images (Amazon Machine Image) Info

Use AMI as ubuntu

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

[Recents](#)[Quick Start](#)[Browse more AMIs](#)

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

ami-0866a3c8686eaeeba (64-bit (x86)) / ami-0325498274077fac5 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Instance type : c7i-flex.large

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t3.medium

Family: t3 2 vCPU 4 GiB Memory Current generation: true
On-Demand SUSE base pricing: 0.0979 USD per Hour
On-Demand Windows base pricing: 0.06 USD per Hour
On-Demand Linux base pricing: 0.0416 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0451 USD per Hour
On-Demand RHEL base pricing: 0.0704 USD per Hour

All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

Create a new key pair

Search [Alt+S] All generations N. Virginia ▾

Create key pair

Key pair name The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type RSA RSA encrypted private and public key pair ED25519 ED25519 encrypted private and public key pair

Private key file format .pem For use with OpenSSH .ppk For use with PuTTY

[Cancel](#) [Create key pair](#) [Preview code](#)

In security group allow ssh and http traffic

[Additional charges apply](#) when outside of free tier allowance

Firewall (security groups) | [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

We'll create a new security group called 'launch-wizard-17' with the following rules:

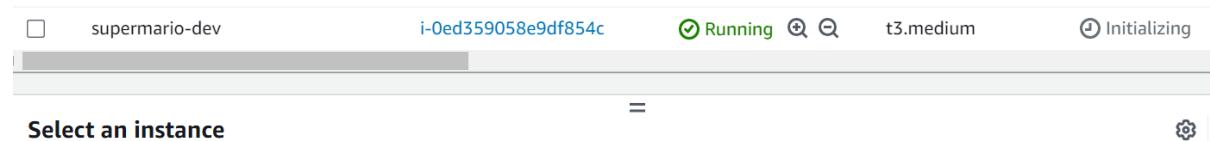
Allow SSH traffic from
Helps you connect to your instance

Anywhere
0.0.0.0/0

Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

Go ahead and launch the instance

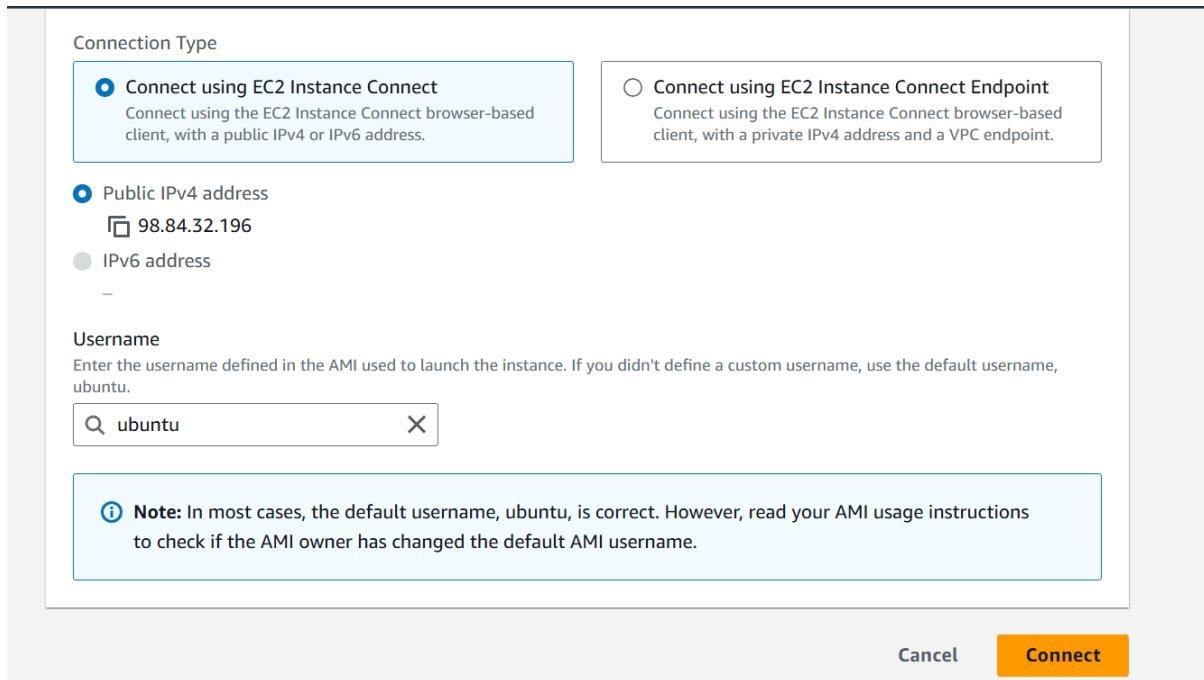


2) Now we will connect to the machine . For that we will first select the machine

And then click on connect button

Click on Ec2 instance connect tab

And click on connect button



3) Run the following commands

a. sudo su

b. apt update

4) Setup Docker ,Terraform ,aws cli , and Kubectl

1. **apt install docker.io**

```
root@ip-172-31-35-40:/home/ubuntu# docker --version
Docker version 24.0.7, build 24.0.7-0ubuntu4.1
root@ip-172-31-35-40:/home/ubuntu#
```

2. **usermod -aG docker \$USER** # Replace with your username e.g 'ubuntu' **usermod -aG docker ubuntu**

3. **newgrp docker**

5) We will now setup terraform in the machine

```
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.d/hashicorp.list
```

```
sudo apt update && sudo apt install terraform
```

6) Now we will setup the AWS CL

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
apt install unzip -y
```

```
unzip awscliv2.zip
```

```
sudo ./aws/install
```

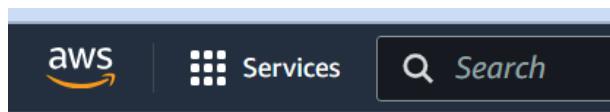
7) Now we will setup Kubectl

1. sudo apt install curl -y
2. curl -LO [https://dl.k8s.io/release/\\$\(curl -L -s https://dl.k8s.io/release/stable.txt\)/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)
3. sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

```
root@ip-172-31-35-40:/home/ubuntu# kubectl version --client
Client Version: v1.31.2
Kustomize Version: v5.4.2
```

8) Every thing is setup and installed let's make a IAM EC2 Role

Go to IAM -> Roles



Identity and Access Management (IAM)

Search IAM

Dashboard

▼ Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

click on create role and choose EC2 from the dropdown

The screenshot shows the "Create Role" dialog box. At the top, there is a "Use case" section with a description: "Allow an AWS service like EC2, Lambda, or others to perform actions in this account." Below this is a "Service or use case" dropdown menu where "EC2" is selected. Underneath the dropdown, there is a section titled "Choose a use case for the specified service." It contains three options:

- EC2**
Allows EC2 instances to call AWS services on your behalf.
- EC2 Role for AWS Systems Manager**
Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- EC2 Spot Fleet Role**
Allows EC2 Spot Fleet to request and terminate Spot Instances on your behalf.

Choose administrator access

Step 2

Add permissions

Step 3

Name, review, and create

Permissions policies (1/1022) [Info](#)

Choose one or more policies to attach to your new role.

Filter by Type

Policy name	Type	Description
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function	Provides full access to AWS services an...
<input type="checkbox"/> AdministratorAcce...	AWS managed	Grants account administrative permissi...
<input type="checkbox"/> AdministratorAcce...	AWS managed	Grants account administrative permissi...
<input type="checkbox"/> AlexaForBusinessD...	AWS managed	Provide device setup access to AlexaFo...
<input type="checkbox"/> AlexaForBusinessF...	AWS managed	Grants full access to AlexaForBusiness ...
<input type="checkbox"/> AlexaForBusinessG...	AWS managed	Provide gateway execution access to A...
<input type="checkbox"/> AlexaForBusinessLi...	AWS managed	Provide access to Lifesize AVS devices

Next

Step 2

[Add permissions](#)

Step 3

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

Maximum 64 characters. Use alphanumeric and '+=_,.@-' characters.

Description
Add a short explanation for this role.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/[\{\}!#\$%^&*()~`"{},";

Click on Create role

The screenshot shows the 'Permissions policy summary' section of the AWS IAM console. It lists a single policy named 'AdministratorAccess' which is an 'AWS managed - job function' type and is attached as a 'Permissions policy'. Below this, a step-by-step wizard titled 'Step 3: Add tags' is displayed. It includes instructions about adding optional tags, a note that no tags are currently associated with the resource, and a button to 'Add new tag'. A note also states that up to 50 more tags can be added. At the bottom right of the wizard are 'Cancel', 'Previous', and 'Create role' buttons.

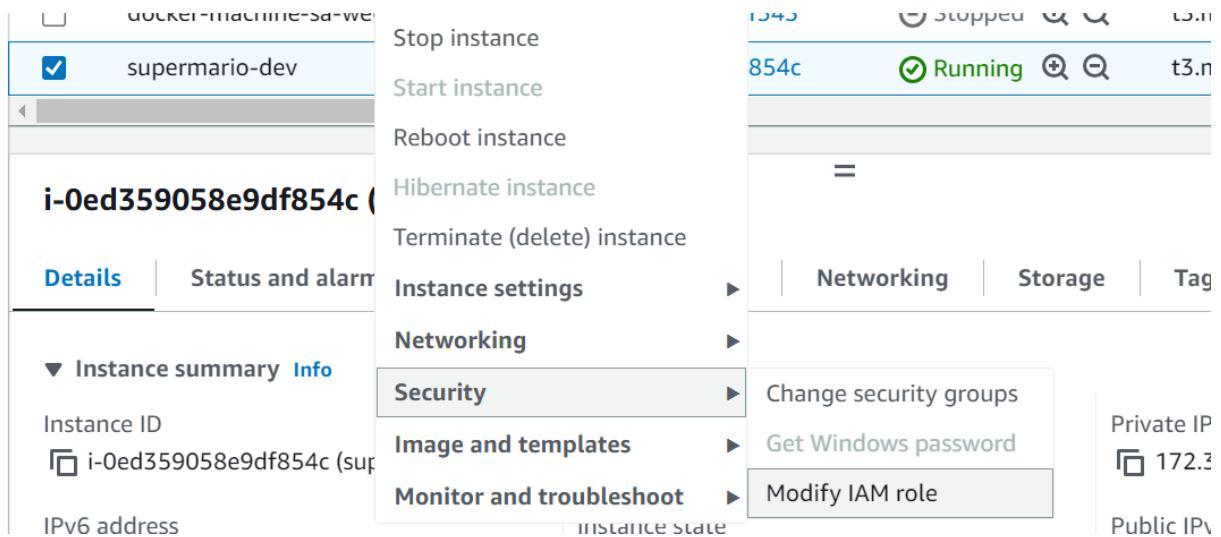
9) Now we will attach the IAM role to Ec2 machine

Go to ec2 machine and select the machine

The screenshot shows the 'Instances' section of the AWS EC2 console. A single instance named 'supermario-dev' is listed, showing its status as 'Running', instance type as 't3.medium', and state as 'Initializing'. Below the main list, the specific instance 'i-0ed359058e9df854c (supermario-dev)' is selected. Its detailed view is shown with tabs for 'Details', 'Status and alarms', 'Monitoring', 'Security', 'Networking', 'Storage', and 'Tags'. Under the 'Details' tab, the 'Instance summary' section is expanded, showing fields for 'Instance ID' (i-0ed359058e9df854c), 'Public IPv4 address', and 'Private IPv4 addresses'.

Right click on the machine

Click on Security -> modify IAM role



Select IAM role which we created previously

The screenshot shows the 'Modify IAM role' dialog box. It displays the selected IAM role 'supermario-project-role' in a dropdown menu. At the bottom, there are 'Cancel' and 'Update IAM role' buttons, with 'Update IAM role' being the active button.

Click on update IAM role

10) Building Infrastructure Using terraform

First we will clone the github repo

Go to the machine

```
mkdir supermario
```

```
cd supermario
```

```
kustomize version: v3.4.2
root@ip-172-31-35-40:/home/ubuntu# mkdir supermario
root@ip-172-31-35-40:/home/ubuntu# cd supermario
root@ip-172-31-35-40:/home/ubuntu/supermario# pwd
/home/ubuntu/supermario
root@ip-172-31-35-40:/home/ubuntu/supermario#
```

```
i-0ed359058e9df854c (supermario-dev)
```

```
git clone https://github.com/akshu20791/supermario-game
```

```
root@ip-172-31-35-40:/home/ubuntu/supermario# git clone https://github.com/akshu20791/supermario-game
Cloning into 'supermario-game'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 8 (delta 0), reused 8 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (8/8), done.
root@ip-172-31-35-40:/home/ubuntu/supermario#
```

```
i-0ed359058e9df854c (supermario-dev)
```

```
root@ip-172-31-35-40:/home/ubuntu/supermario# ls
supermario-game
root@ip-172-31-35-40:/home/ubuntu/supermario#
```

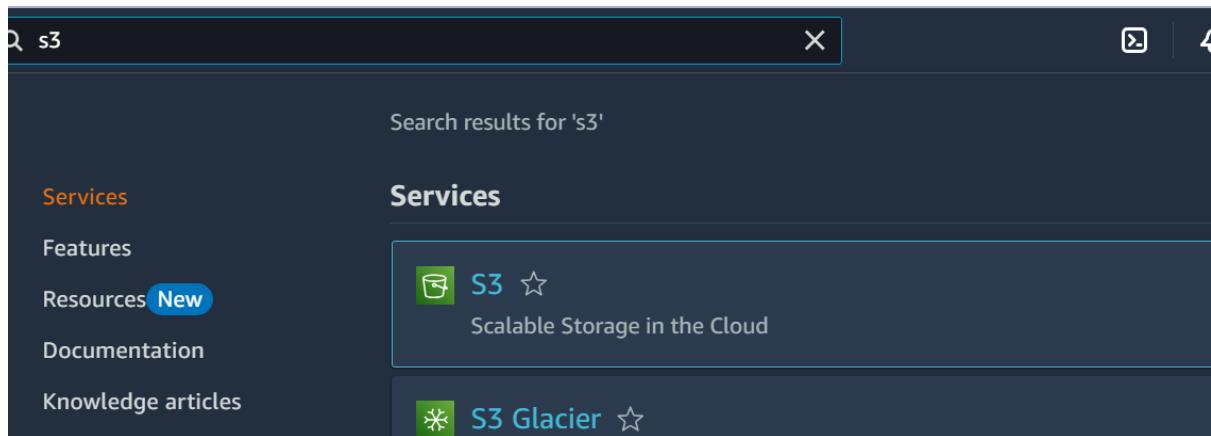
```
cd supermario-game
```

```
root@ip-172-31-35-40:/home/ubuntu/supermario# cd supermario-game
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game#
```

```
cd EKS-TF
```

```
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game# cd EKS-TF
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game/EKS-TF# ls
backend.tf  main.tf  provider.tf
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game/EKS-TF#
```

12) Go to aws -> S3 -> Create a s3 bucket with some unique name



Screenshot of the AWS Management Console showing the Amazon S3 service page. The left sidebar includes links for Buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, and Block Public Access settings. The main content area displays an account snapshot (updated every 24 hours) and a list of General purpose buckets (50). A prominent 'Create bucket' button is visible.

Screenshot of the AWS Management Console showing the 'Create bucket' wizard. It asks for a bucket name ('mariobucketakshat') and provides options for bucket type: 'General purpose' (selected) or 'Directory'. It also includes sections for copying settings from an existing bucket and specifying a prefix.

Create bucket

13) Go back to Ec2 machine

edit the backend.tf file by → **vim backend.tf**

```
terraform {
  backend "s3" {
    bucket = "mariobucketakshat" # Replace with your actual S3 bucket name
    key    = "EKS/terraform.tfstate"
    region = "us-east-1"
  }
}
~
```

Note →make sure to provide your bucket and region name in this file otherwise it doesn't work and IAM role is also associated with your ec2 which helps ec2 to use other services such S3 bucket

```
>> nano main.tf
```

Change instance type → c7i-flex.large

14) Now we will run

terraform init

```
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game/EKS-TF# 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 v5.75.1...
- Installed hashicorp/aws v5.75.1 (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
```

terraform validate

```
Commands will detect it and remind you to do so if necessary.
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game/EKS-TF# terraform validate
Success! The configuration is valid.
```

go to AWS ACCOUNT -> GO TO VPC -> GO TO SUBNETS -> DELETE subnet in availability zone us-east-1e

terraform plan

```
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game/EKS-TF# 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 2s [id=vpc-0b918116e453a16a9]
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 indicated with the following symbols:
+ create

Terraform will perform the following actions:

  # aws_eks_cluster.example will be created
```

Terraform will perform the following actions:

```
# aws_eks_cluster.example will be created
+ resource "aws_eks_cluster" "example" {
    + arn                      = (known after apply)
    + bootstrap_self_managed-addons = true
    + 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)
```

terraform apply --auto-approve

```
Plan: 8 to add, 0 to change, 0 to destroy.
aws_iam_role.example: Creating...
aws_iam_role.example1: Creating...
aws_iam.role.example: Creation complete after 0s [id=eks-cluster-cloud]
aws_iam.role_policy_attachment.example-AWSContainerPolicy: Creating...
aws_iam.role.example1: Creation complete after 0s [id=eks-node-group-cloud]
aws_iam.role_policy_attachment.example-AWSWorkerNodePolicy: creating...
aws_iam.role_policy_attachment.example-AWS_CNI_Policy: creating...
aws_iam.role_policy_attachment.example-AWSContainerRegistryReadOnly: Creating...
aws_iam.role_policy_attachment.example-AWSContainerPolicy: Creation complete after 0s [id=eks-cluster-cloud-20240114062220905000000001]
aws_eks_cluster.example: Creating...
aws_iam.role_policy_attachment.example-AWSContainerRegistryReadOnly: Creation complete after 0s [id=eks-node-group-cloud-20240114062220947300000002]
aws_iam.role_policy_attachment.example-AWS_CNI_Policy: Creation complete after 0s [id=eks-node-group-cloud-20240114062220978900000003]
aws_iam.role_policy_attachment.example-AWSWorkerNodePolicy: Creation complete after 0s [id=eks-node-group-cloud-20240114062220986200000004]
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]
```

It will take 10-15 minutes to complete

```
aws_eks_cluster.example: Still creating... [5m40s elapsed]
aws_eks_cluster.example: Still creating... [5m50s elapsed]
aws_eks_cluster.example: Still creating... [6m0s elapsed]
aws_eks_cluster.example: Creation complete after 6m9s [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: Creation complete after 1m51s [id=EKS_CLOUD:Node-cloud]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.
```

15) Now we will update the configuration of EKS

```
aws eks update-kubeconfig --name EKS_CLOUD --region us-east-1
```

```
# ensure that you take care of the region you are working in
```

```
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game/EKS-TF# aws eks update-kubeconfig --name EKS_CLOUD --region ap-south-1
Added new context arn:aws:eks:ap-south-1:975050323630:cluster/EKS_CLOUD to /root/.kube/config
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game/EKS-TF#
```

16) Creation of deployment and service for EKS

1. change the directory where deployment and service files are stored use the command → `cd ..`

```
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game/EKS-TF# cd ..
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game# ls
EKS-TF  deployment.yaml  service.yaml
```

2. create the deployment

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

```
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game# kubectl apply -f deployment.yaml
deployment.apps/mario-deployment created
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game#
```

```
kubectl apply -f service.yaml
```

```
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game# kubectl apply -f service.yaml
service/mario-service created
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game#
```

run → `kubectl get all`

```

root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game# kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/mario-deployment-6bfd49df4b-tgzrd   1/1    Running   0          64s
pod/mario-deployment-6bfd49df4b-vxf7q   1/1    Running   0          64s

NAME                TYPE        CLUSTER-IP      EXTERNAL-IP
service/kubernetes  ClusterIP   10.100.0.1    <none>
13m
service/mario-service  LoadBalancer  10.100.68.49  a086c1c645bc8444f943afe7a74c47a5-1256226015.ap-south-1.elb.amazonaws.com  80:3155
3/TCP  25s

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mario-deployment   2/2     2           2           65s

NAME             DESIRED   CURRENT   READY   AGE
replicaset.apps/mario-deployment-6bfd49df4b  2         2         2         65s
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game# [REDACTED]

i-Oed359058e9df854c (supermario-dev) [REDACTED]
Public IPs: 98.84.32.196 Private IPs: 172.31.35.40

```

16) Now Run the following command to get the load balancer ingress

This command tells all the details of your application

kubectl describe service mario-service

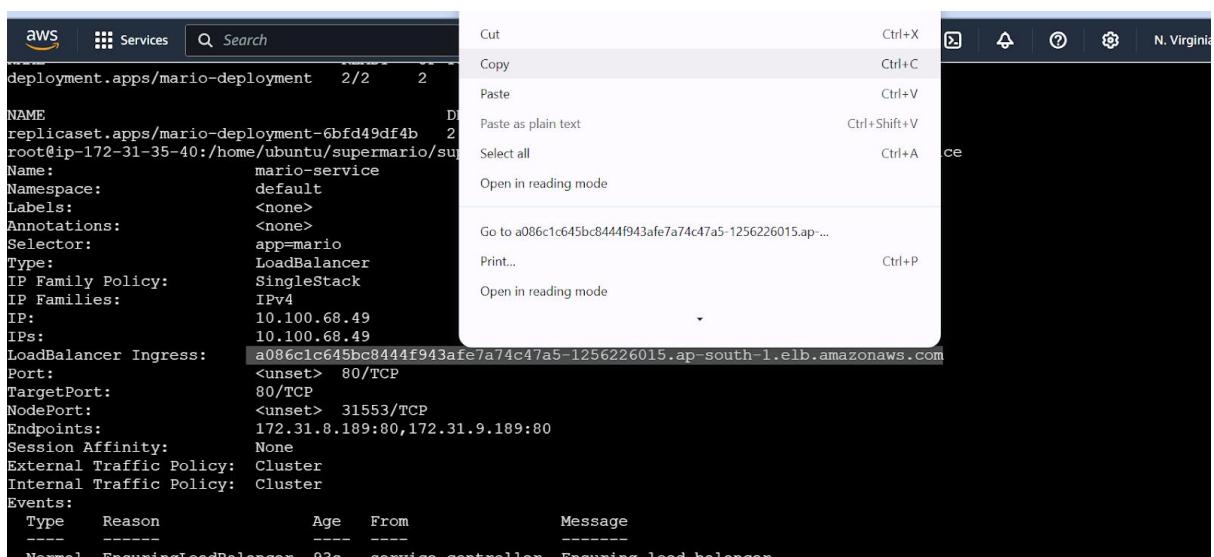
```

root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game# 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:                      10.100.68.49
IPs:                     10.100.68.49
LoadBalancer Ingress:    a086c1c645bc8444f943afe7a74c47a5-1256226015.ap-south-1.elb.amazonaws.com
Port:                    <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                <unset>  31553/TCP
Endpoints:               172.31.8.189:80,172.31.9.189:80
Session Affinity:        None
External Traffic Policy: Cluster
Internal Traffic Policy: Cluster
Events:
  Type  Reason     Age   From            Message
  ----  --::--  ----  --::--  -----
  Normal  EnsuringLoadBalancer  93s  service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer  89s  service-controller  Ensured load balancer
root@ip-172-31-35-40:/home/ubuntu/supermario/supermario-game# [REDACTED]

```

Here you will see load balancer ingress link

Copy that



17) paste it in the browser

Ensure that when you copy paste ...it should be http not https

