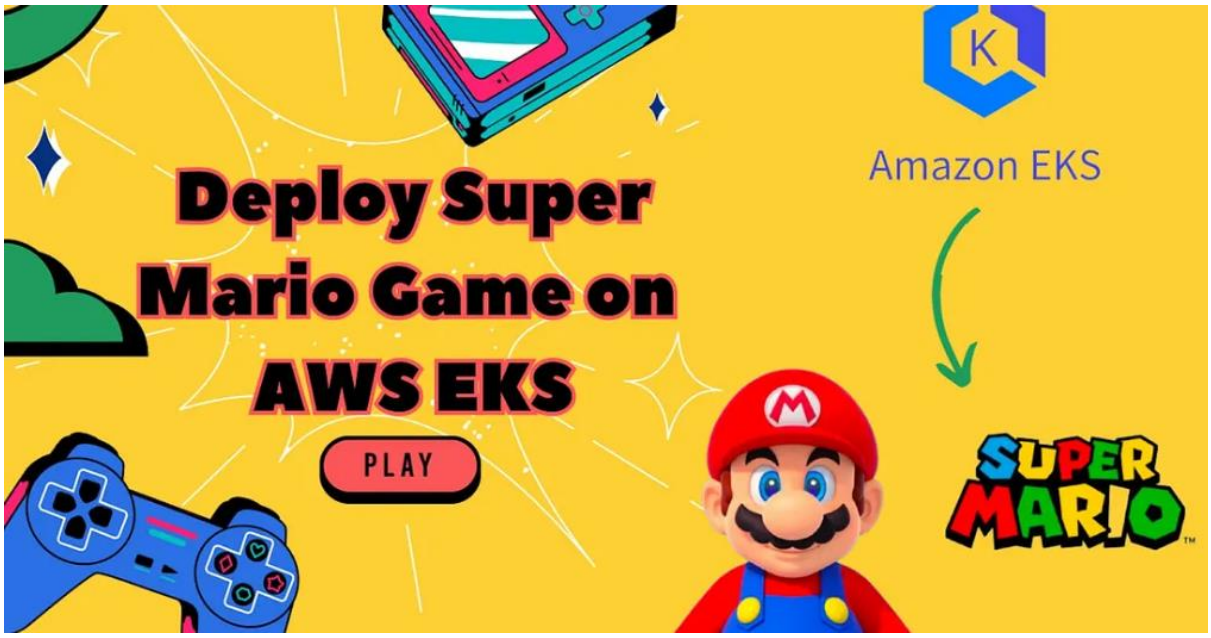
NAME – VAIBHAV NAVNEET JORVEKAR

CDEC – B24

DEPLOYING SUPER MARIO ON KUBERNETES



Deploying Super Mario on Kubernetes

1. An Ubuntu Instance
2. IAM role
3. Terraform should be installed on instance
4. AWS CLI and KUBECTL on Instance

LET'S DEPLOY

STEP 1: Launch Ubuntu instance

1. Sign in to AWS Console: Log in to your AWS Management Console.
2. Navigate to EC2 Dashboard: Go to the EC2 Dashboard by selecting “Services” in the top menu and then choosing “EC2” under the Compute section.
3. Launch Instance: Click on the “Launch Instance” button to start the instance creation process.
4. Choose an Amazon Machine Image (AMI): Select an appropriate AMI for your instance. For example, you can choose Ubuntu image.
5. Choose an Instance Type: In the “Choose Instance Type” step, select `t2.micro` as your instance type. Proceed by clicking "Next: Configure Instance Details."
6. Configure Instance Details:

- For “Number of Instances,” set it to 1 (unless you need multiple instances).
- Configure additional settings like network, subnets, IAM role, etc., if necessary.
- For “Storage,” click “Add New Volume” and set the size to 8GB (or modify the existing storage to 8GB).
- Click “Next: Add Tags” when you’re done.

7. Add Tags (Optional): Add any desired tags to your instance. This step is optional, but it helps in organizing instances.

8. Configure Security Group:

- Choose an existing security group or create a new one.
- Ensure the security group has the necessary inbound/outbound rules to allow access as required.

9. Review and Launch: Review the configuration details. Ensure everything is set as desired.

10. Select Key Pair:

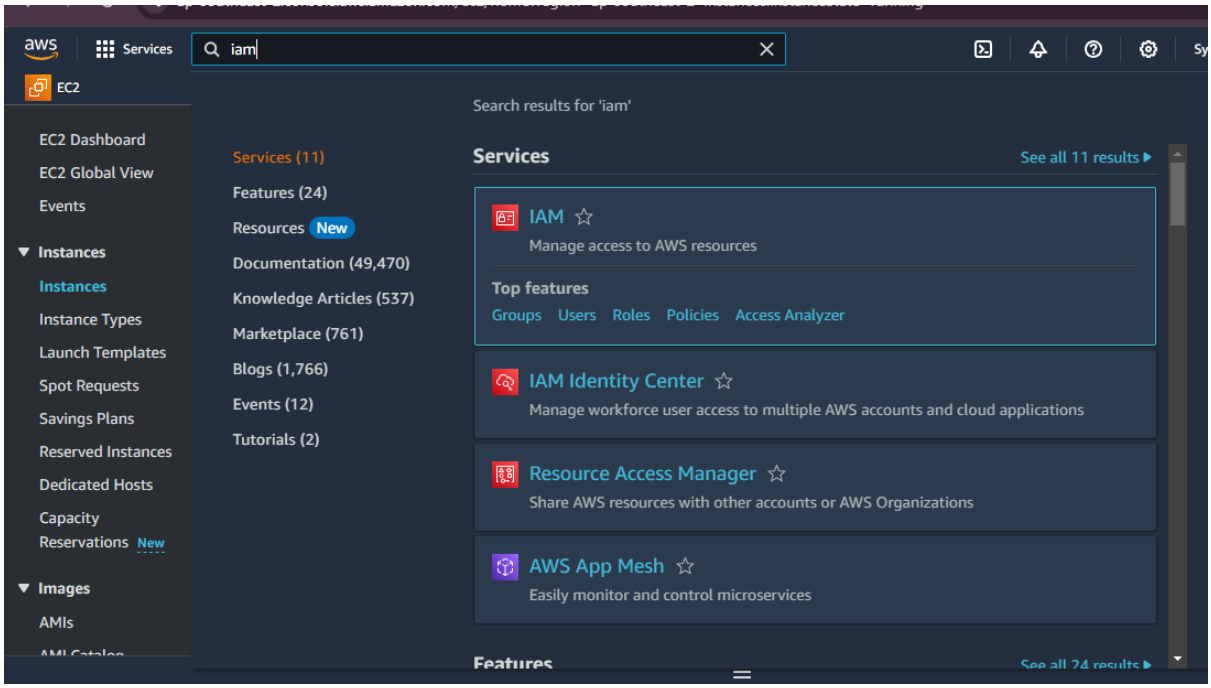
- Select “Choose an existing key pair” and choose the key pair from the dropdown.
- Acknowledge that you have access to the selected private key file.
- Click “Launch Instances” to create the instance.

11. Access the EC2 Instance: Once the instance is launched, you can access it using the key pair and the instance’s public IP or DNS.

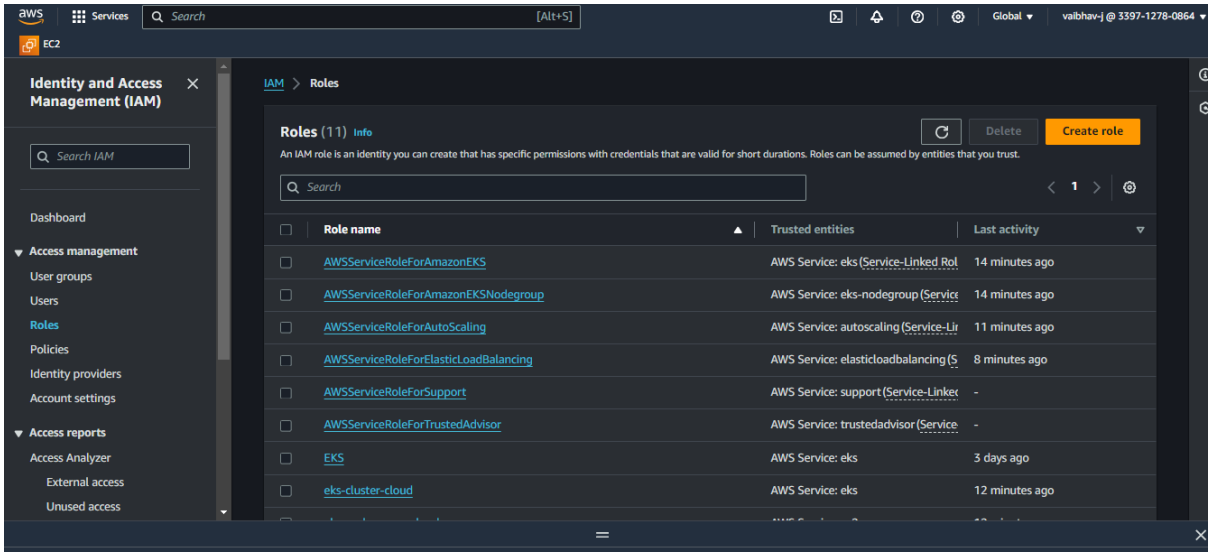
Ensure you have necessary permissions and follow best practices while configuring security groups and key pairs to maintain security for your EC2 instance.

STEP 2: Create IAM role

Search for IAM in the search bar of AWS and click on roles.

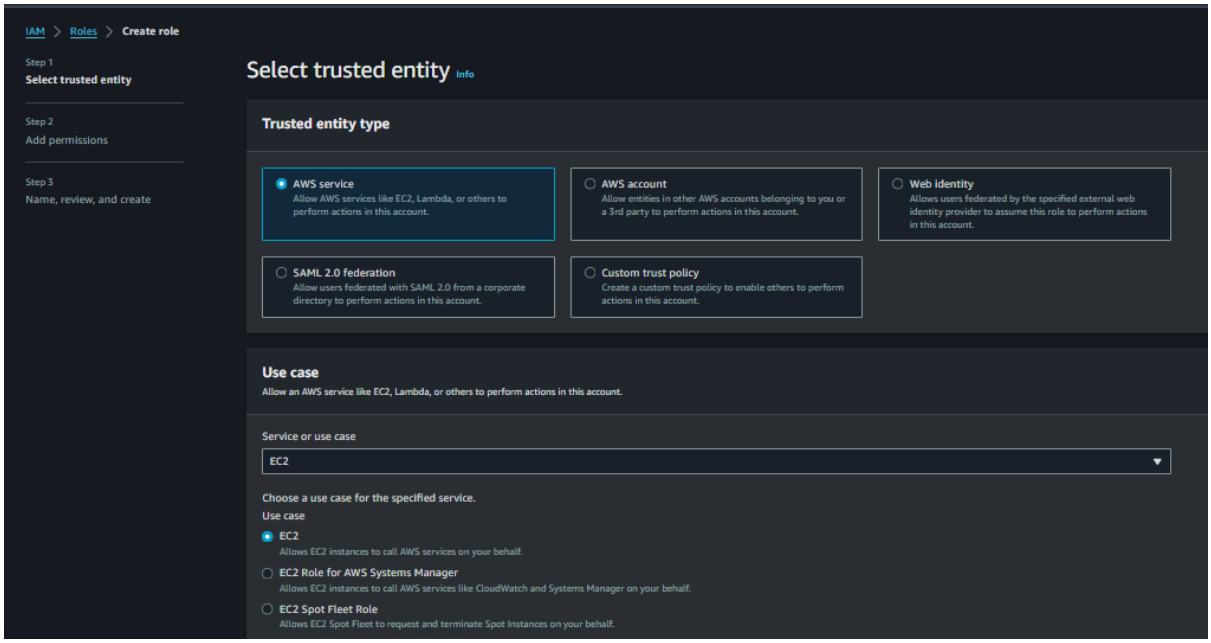


Click on Create Role

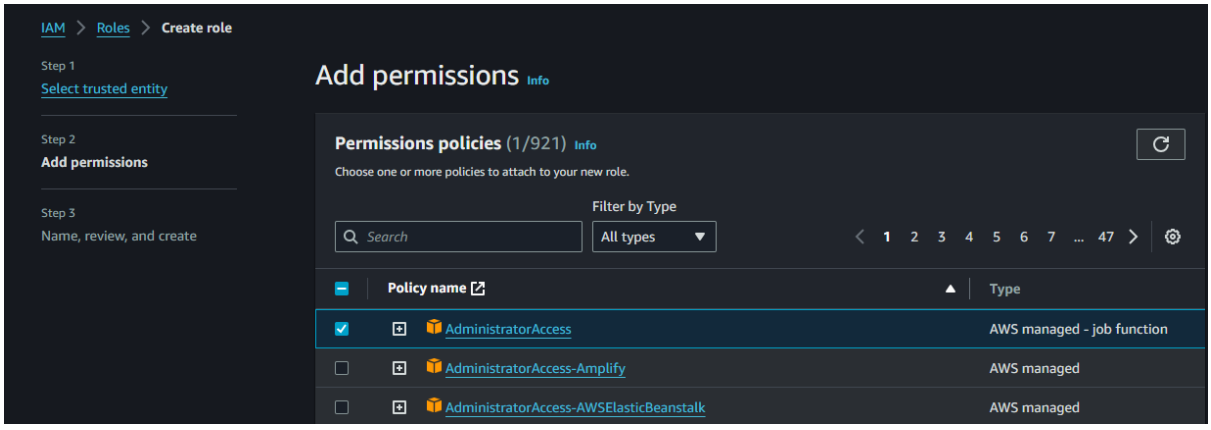


Select entity type as AWS service

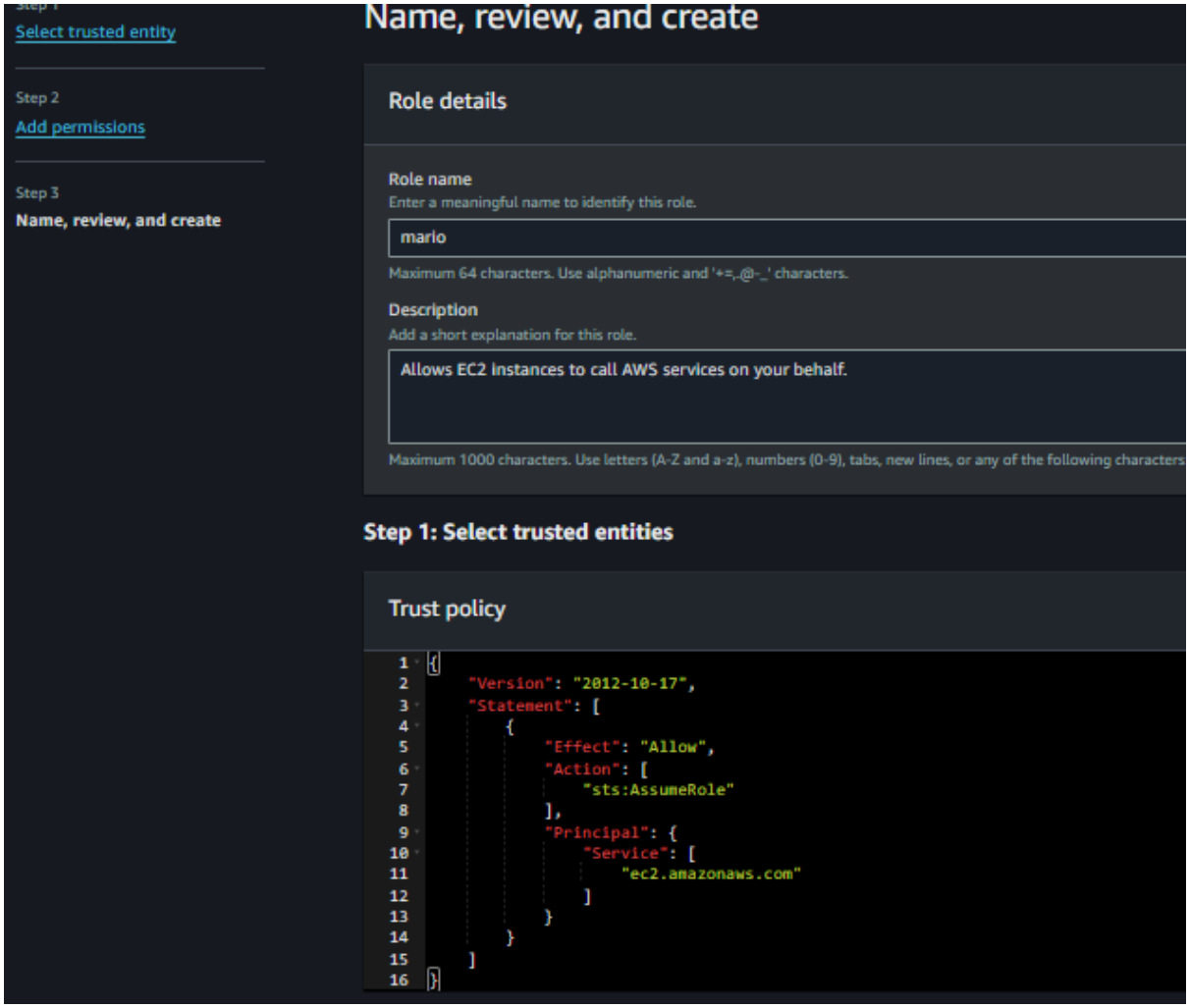
Use case as EC2 and click on Next.



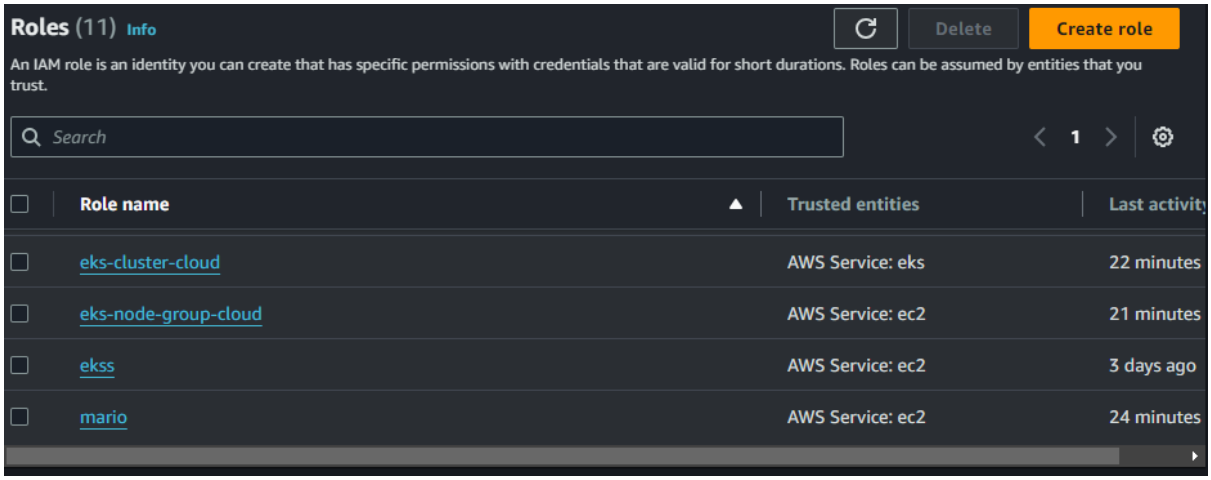
For permission policy select Administrator Access (Just for learning purpose), click Next.



Provide a Name for Role and click on Create role.



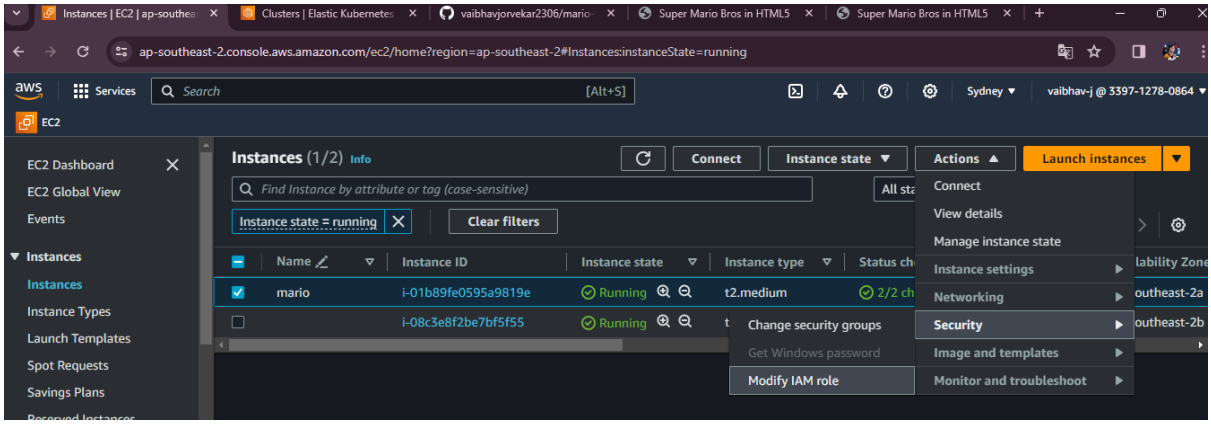
Role is created.



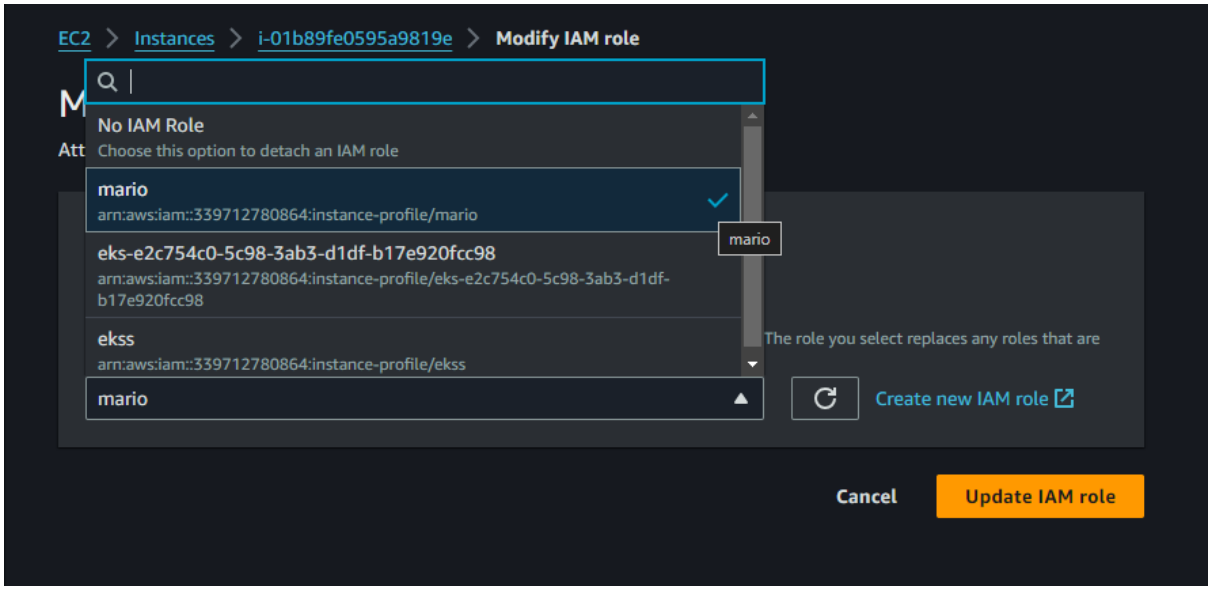
Now Attach this role to Ec2 instance that we created earlier, so we can provision cluster from that instance.

Go to EC2 Dashboard and select the instance.

Click on Actions → Security → Modify IAM role.



Select the Role that created earlier and click on Update IAM role.



Connect the instance to Mobaxtreme or Putty

STEP 3: Cluster provision

1. Now clone this Repo.
   git clone https://github.com/vaibhavjorvekar2306/mario-vj.git

```
root@ip-172-31-13-124:~# git clone https://github.com/vaibhavjorvekar2306/mario-vj.git
Cloning into 'mario-vj'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 25 (delta 7), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (25/25), 7.62 KiB | 1.90 MiB/s, done.
Resolving deltas: 100% (7/7), done.
root@ip-172-31-13-124:~# ls
mario-vj  snap
```

2. change directory
cd k8s-mario

3. Provide the executable permission to script.sh file, and run it.
sudo chmod +777 script.sh
sh script.sh

```
root@ip-172-31-13-124:~/mario-vj# ll
total 32
drwxr-xr-x 4 root root 4096 Apr  4 14:14 ./
drwx------ 5 root root 4096 Apr  4 14:14 ../
drwxr-xr-x 8 root root 4096 Apr  4 14:14 .git/
drwxr-xr-x 2 root root 4096 Apr  4 14:14 EKS-TF/
-rw-r--r-- 1 root root   10 Apr  4 14:14 README.md
-rw-r--r-- 1 root root  407 Apr  4 14:14 deployment.yaml
-rw-r--r-- 1 root root  911 Apr  4 14:14 script.sh
-rw-r--r-- 1 root root  192 Apr  4 14:14 service.yaml
root@ip-172-31-13-124:~/mario-vj# chmod +777 script.sh
root@ip-172-31-13-124:~/mario-vj# ll
total 32
drwxr-xr-x 4 root root 4096 Apr  4 14:14 ./
drwx------ 5 root root 4096 Apr  4 14:14 ../
drwxr-xr-x 8 root root 4096 Apr  4 14:14 .git/
drwxr-xr-x 2 root root 4096 Apr  4 14:14 EKS-TF/
-rw-r--r-- 1 root root   10 Apr  4 14:14 README.md
-rw-r--r-- 1 root root  407 Apr  4 14:14 deployment.yaml
-rwxrwxrwx 1 root root  911 Apr  4 14:14 script.sh*
-rw-r--r-- 1 root root  192 Apr  4 14:14 service.yaml
```

This script will install `Terraform, AWS cli, Kubectl, Docker.`

4. Check versions

```
Docker --version

Terraform --version

Aws --version

Kubectl version --client
```

For e.g.

```
root@ip-172-31-13-124:~/mario-vj# docker --version
Docker version 26.0.0, build 2ae903e
root@ip-172-31-13-124:~/mario-vj# terraform --version
Terraform v1.7.5
on linux_amd64
```

5. Now change directory into the EKS-TF

6. Run Terraform init

NOTE: Don't forgot to change the s3 bucket name in the backend.tf file

```
cd EKS-TF
terraform init
```

```
root@ip-172-31-13-124:~/mario-vj# cd EKS-TF/
root@ip-172-31-13-124:~/mario-vj/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 v5.43.0...
- Installed hashicorp/aws v5.43.0 (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.
```

7. Now run terraform validate and terraform plan

```
terraform validate
terraform plan
```

```
root@ip-172-31-13-124:~/mario-vj/EKS-TF# terrafrom validate
Command 'terrafrom' not found, did you mean:
  command 'terraform' from snap terraform (1.7.5)
See 'snap info <snapname>' for additional versions.
root@ip-172-31-13-124:~/mario-vj/EKS-TF# terraform validate
Success! The configuration is valid.

root@ip-172-31-13-124:~/mario-vj/EKS-TF# terraform plan
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 0s [id=vpc-028a1281f2b8b4fef]
data.aws_subnets.public: Reading...
data.aws_subnets.public: Read complete after 0s [id=ap-southeast-2]

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
  + 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 {
```

8. Now Run terraform apply to provision cluster.

```
terraform apply --auto-approve
```

```
root@ip-172-31-13-124:~/mario-vj/EKS-TF# terraform apply --auto-approve
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-028a1281f2b8b4fef]
data.aws_subnets.public: Reading...
data.aws_subnets.public: Read complete after 0s [id=ap-southeast-2]

Terraform used the selected providers to generate the following execution plan. Res
symbols:
  + 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 {
```

Completed in 10 min

```
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... [2m10s elapsed]
aws_eks_cluster.example: Still creating... [2m20s elapsed]
aws_eks_cluster.example: Still creating... [2m30s elapsed]
aws_eks_cluster.example: Still creating... [2m40s elapsed]
aws_eks_cluster.example: Still creating... [2m50s elapsed]
aws_eks_cluster.example: Still creating... [3m0s elapsed]
aws_eks_cluster.example: Still creating... [3m10s elapsed]
aws_eks_cluster.example: Still creating... [3m20s elapsed]
aws_eks_cluster.example: Still creating... [3m30s elapsed]
aws_eks_cluster.example: Still creating... [3m40s elapsed]
aws_eks_cluster.example: Still creating... [3m50s elapsed]
aws_eks_cluster.example: Still creating... [4m0s elapsed]
aws_eks_cluster.example: Still creating... [4m10s elapsed]
aws_eks_cluster.example: Still creating... [4m20s elapsed]
aws_eks_cluster.example: Still creating... [4m30s elapsed]
aws_eks_cluster.example: Still creating... [4m40s elapsed]
aws_eks_cluster.example: Still creating... [4m50s elapsed]
aws_eks_cluster.example: Still creating... [5m0s elapsed]
aws_eks_cluster.example: Still creating... [5m10s elapsed]
aws_eks_cluster.example: Still creating... [5m20s elapsed]
aws_eks_cluster.example: Still creating... [5m30s elapsed]
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: Still creating... [6m10s elapsed]
aws_eks_cluster.example: Still creating... [6m20s elapsed]
aws_eks_cluster.example: Still creating... [6m30s elapsed]
aws_eks_cluster.example: Creation complete after 6m31s [id=EKS_CLOUD]
```

9. Update the Kubernetes configuration

Make sure change your desired region

```
aws eks update-kubeconfig --name EKS_CLOUD --region ap-southeast-2
```

```
root@ip-172-31-13-124:~/mario-vj/EKS-TF# aws eks update-kubeconfig --name EKS_CLOUD --region ap-southeast-2
Added new context arn:aws:eks:ap-southeast-2:339712780864:cluster/EKS_CLOUD to /root/.kube/config
```

10 .Now change directory back to k8s-mario
Cd ..

11. Let’s apply the deployment and service

Deployment

```
kubectl apply -f deployment.yaml
#to check the deployment
kubectl get all
```

```
root@ip-172-31-13-124:~/mario-vj# kubectl apply -f deployment.yaml
deployment.apps/mario-deployment created
root@ip-172-31-13-124:~/mario-vj# kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/mario-deployment-78cbc65cb-mtf55    1/1     Running   0          26s
pod/mario-deployment-78cbc65cb-qsk5t    1/1     Running   0          26s

NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.100.0.1    <none>        443/TCP   7m33s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mario-deployment    2/2     2            2           26s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/mario-deployment-78cbc65cb     2         2         2       26s
```

12. Now let’s apply the service

Service

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

```
root@ip-172-31-13-124:~/mario-vj# kubectl apply -f service.yaml
service/mario-service created
root@ip-172-31-13-124:~/mario-vj# kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/mario-deployment-78cbc65cb-mtf55    1/1     Running   0          76s
pod/mario-deployment-78cbc65cb-qsk5t    1/1     Running   0          76s

NAME                 TYPE           CLUSTER-IP      EXTERNAL-IP                                                              POR
T(S)        AGE
service/kubernetes   ClusterIP      10.100.0.1      <none>                                                                   443
/TCP        8m23s
service/mario-service   LoadBalancer   10.100.113.30   ad5ba9f4b431645bd83174426d7c38ad-1600352451.ap-southeast-2.elb.amazonaws.com   80:
32208/TCP   11s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mario-deployment    2/2     2            2           76s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/mario-deployment-78cbc65cb     2         2         2       76s
```

13. Now let's describe the service and copy the LoadBalancer Ingress

```
kubectl describe service mario-service
```

```
root@ip-172-31-13-124:~/mario-vj# 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:                       10.100.113.30
IPs:                      10.100.113.30
LoadBalancer Ingress:     ad5ba9f4b431645bd83174426d7c38ad-1600352451.ap-southeast-2.elb.amazonaws.com
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  32208/TCP
Endpoints:                172.31.36.9:80,172.31.47.107:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason                Age   From                Message
  ----    ------                ----  ----                -------
  Normal  EnsuringLoadBalancer  55s   service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer   51s   service-controller  Ensured load balancer
root@ip-172-31-13-124:~/mario-vj#
```

Paste the ingress link in a browser and you will see the Mario game.

Let's Go back to 1985 and play the game like children.