Super Mario Game Deployment on AWS EKS with Terraform

Step 1: Set Up AWS Credentials

Configure the AWS CLI with your credentials by running:
`aws configure`

1. *Enter your* `AWS Access Key ID`, `AWS Secret Access Key`, `Region`, *and* `Output format`.
2. *Ensure you have permission to create EKS clusters, EC2 instances, and IAM roles.*

Step 2: Create Terraform Configuration Files

Terraform will manage the infrastructure, including EKS and related resources. Follow the steps below to create Terraform files. To create Terraform files, follow

1. Create Directory for Terraform Files

Create a new directory for your Terraform files:

```
mkdir mario-eks-deployment
cd mario-eks-deployment
```

2. Create a `main.tf` file

In this file, we define the AWS provider, EKS cluster, and necessary resources.

```
provider "aws" {
  region = "us-west-2" # Change to your preferred region
}

resource "aws_eks_cluster" "mario_cluster" {
  name     = "mario-eks-cluster"
  role_arn = aws_iam_role.eks_cluster_role.arn
  vpc_config {
    subnet_ids = aws_subnet.mario_subnet[*].id
  }
}

resource "aws_iam_role" "eks_cluster_role" {
  name = "eks-cluster-role"

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [{
      Action    = "sts:AssumeRole"
      Effect    = "Allow"
      Principal = {
        Service = "eks.amazonaws.com"
      }
    }]
  })
}

resource "aws_iam_role_policy_attachment"
"eks_cluster_policy" {
```

```
  role       = aws_iam_role.eks_cluster_role.name
  policy_arn =
"arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
}
```

3. Create `variables.tf` file

This file contains configurable variables for the resources.

```
variable "aws_region" {
  description = "AWS region to deploy the resources"
  default     = "us-west-2"
}
```

4. Create `outputs.tf` file

This file outputs important information after deployment, such as the EKS cluster name.

hcl
CopyEdit

```
output "eks_cluster_name" {
  value = aws_eks_cluster.mario_cluster.name
}
```

Step 3: Initialize and Apply Terraform

***Initialize Terraform**: Initialize the directory to download the necessary Terraform providers:*

```
terraform init
```

1.

Apply Terraform Configuration*: Run the following command to apply the Terraform configuration:*

```
terraform apply
```

2. *Terraform will prompt you to approve the changes. Type* `yes` *to proceed. Terraform will create the EKS cluster and other resources.*

Step 4: Set Up kubectl for EKS

Once the EKS cluster is created, configure `kubectl` *to interact with it.*

Run the following command to get the EKS cluster credentials:

```
aws eks --region us-west-2 update-kubeconfig --name
mario-eks-cluster
```

1. *This command updates the* `~/.kube/config` *file with the necessary configuration to access the EKS cluster.*

Verify the configuration by running:

```
kubectl get svc
```

2. *This should display the services running in your cluster, if any.*

Step 5: Containerize the Super Mario Game

If you already have a Docker image for the Super Mario game, you can skip this step. Otherwise, you can create a Docker container by following these steps:

***Create a Dockerfile** for the game: Example `Dockerfile` for a simple Python-based Super Mario game:*
Dockerfile

```
FROM python:3.8-slim

WORKDIR /app

COPY . /app

RUN pip install -r requirements.txt

CMD ["python", "game.py"]
```

***Build the Docker Image**:*

```
docker build -t super-mario-game .
```

***Push the Docker Image to Amazon ECR** (Elastic Container Registry):*
First, create an ECR repository:

```
aws ecr create-repository --repository-name
super-mario-game
```

Tag and push the Docker image to ECR:

```
$(aws ecr get-login --no-include-email --region
us-west-2)
docker tag super-mario-game:latest
<aws_account_id>.dkr.ecr.us-west-2.amazonaws.com/super-m
ario-game
docker push
<aws_account_id>.dkr.ecr.us-west-2.amazonaws.com/super-m
ario-game
```

Push the Docker Image to Amazon ECR *(Elastic Container Registry): First, create an ECR repository:*

```
aws ecr create-repository --repository-name
super-mario-game
```

Tag and push the Docker image to ECR:

```
$(aws ecr get-login --no-include-email --region
us-west-2)

docker tag super-mario-game:latest
<aws_account_id>.dkr.ecr.us-west-2.amazonaws.com/super-m
ario-game

docker push
<aws_account_id>.dkr.ecr.us-west-2.amazonaws.com/super-m
ario-game
```

Step 6: Deploy Super Mario Game on EKS

Now that the Docker image is available, deploy the game to EKS using Kubernetes.

Create Kubernetes Deployment Configuration *(`deployment.yaml`):*

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: super-mario-game
spec:
  replicas: 1
  selector:
    matchLabels:
      app: super-mario-game
  template:
    metadata:
      labels:
        app: super-mario-game
    spec:
      containers:
      - name: super-mario-game
        image:
<aws_account_id>.dkr.ecr.us-west-2.amazonaws.com/super-m
ario-game:latest
        ports:
        - containerPort: 8080
```

***Apply the Deployment to Kubernetes**:*

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

***Expose the Game via a LoadBalancer**:*

Create a Kubernetes service to expose the game on the internet.

```
apiVersion: v1
kind: Service
metadata:
  name: super-mario-game-service
spec:
  selector:
    app: super-mario-game
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

Apply the service:

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

Once the service is created, Kubernetes will provision an external IP for the game. You can check the external IP with:

```
kubectl get svc super-mario-game-service
```

***Expose the Game via a LoadBalancer**:*

Create a Kubernetes service to expose the game on the internet.

```
apiVersion: v1
kind: Service
```

```
metadata:
  name: super-mario-game-service
spec:
  selector:
    app: super-mario-game
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

Apply the service:

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

Once the service is created, Kubernetes will provision an external IP for the game. You can check the external IP with:

```
kubectl get svc super-mario-game-service
```

Step 7: Verify the Deployment

After a few moments, the game should be accessible via the external IP address provided by the LoadBalancer. Open a web browser and navigate to the URL to access the Super Mario game.