DevOps Deployment Task

This repository contains instructions and configurations for deploying a web application using modern DevOps practices and tools including Docker, Kubernetes (EKS), AWS, and Terraform.

Table of Contents

Prerequisites

- AWS Account with appropriate permissions
- AWS CLI installed and configured
- kubectl installed
- Terraform installed
- Git installed
- Basic knowledge of Docker, Kubernetes, and AWS services

Deployment Steps

Step 1: Clone Repository

```bash
# SSH into your EC2 instance
ssh -i "your-key.pem" ubuntu@<Your-EC2-Public-IP>

# Install Git (if not installed)
sudo apt-get update
sudo apt-get install git

# Clone this repository
git clone https://github.com/sadman-tanim/devops-deploy-task.git
cd devops-deploy-task
```

Step 2: Docker Setup

Install Docker on your EC2 instance:

```bash
# Update and install Docker
sudo apt update
sudo apt install docker.io

# Start Docker and enable it on boot
sudo systemctl start docker
sudo systemctl enable docker

# Verify Docker installation
sudo docker --version
sudo docker run hello-world
```

If you encounter any issues with Docker installation, try the following steps:

bash

```bash
# Add Docker's official GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

# Add Docker repository
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_releas

# Update and install Docker CE
sudo apt-get update
sudo apt-get install docker-ce
```

Step 3: Application Dockerization

1. Create a Dockerfile in your project directory:

Dockerfile

```
# Use official Nginx image as a base
FROM nginx:alpine

# Copy your application files into the container's Nginx HTML directory
COPY . /usr/share/nginx/html

# Expose port 80
EXPOSE 80

# Run Nginx in the foreground
CMD ["nginx", "-g", "daemon off;"]
```

2. Build and run the Docker image:

bash

```bash
# Build the Docker image
sudo docker build -t devops-app .

# Run the Docker container
sudo docker run -d -p 80:80 devops-app
```

3. Verify the application by visiting `http://<Your-EC2-Public-IP>` in your browser.

Step 4: Kubernetes (EKS) Setup

Install required tools for managing Kubernetes:

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

# Configure AWS CLI
aws configure

# Install kubectl
curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.g
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin/kubectl
```

Create an EKS cluster (via AWS Console or AWS CLI), then:

```bash
# Configure kubectl to use the EKS cluster
aws eks --region <your-region> update-kubeconfig --name <your-cluster-name>

# Verify connection to the cluster
kubectl get nodes
```

Step 5: Deploy to Kubernetes

1. Create a deployment.yaml file:

yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: devops-app-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: devops-app
  template:
    metadata:
      labels:
        app: devops-app
    spec:
      containers:
      - name: devops-app
        image: devops-app
        ports:
        - containerPort: 80
```

2. Apply the deployment and expose it:

bash

```bash
# Apply the deployment
kubectl apply -f deployment.yaml

# Expose the deployment via LoadBalancer
kubectl expose deployment devops-app-deployment --type=LoadBalancer --name=devops-app-service

# Check the external IP for the service
kubectl get svc
```

3. Access the application at the LoadBalancer's external IP.

Step 6: Infrastructure Automation with Terraform

1. Install Terraform:

bash

```bash
# Download and install Terraform
curl -LO https://releases.hashicorp.com/terraform/latest_version/terraform_0.14.7_linux_amd64.z
unzip terraform_0.14.7_linux_amd64.zip
sudo mv terraform /usr/local/bin/
```

2. Create a main.tf file for provisioning an EKS cluster:

hcl

```hcl
provider "aws" {
  region = "us-west-2"
}

resource "aws_eks_cluster" "example" {
  name     = "example-cluster"
  role_arn = "arn:aws:iam::123456789012:role/eksServiceRole"

  vpc_config {
    subnet_ids = ["subnet-12345", "subnet-67890"]
  }
}

resource "aws_eks_node_group" "example" {
  cluster_name = aws_eks_cluster.example.name
  node_role_arn = "arn:aws:iam::123456789012:role/eksNodeRole"
  subnet_ids   = ["subnet-12345", "subnet-67890"]
  scaling_config {
    desired_size = 2
    max_size     = 3
    min_size     = 1
  }
}
```

3. Initialize and apply Terraform configuration:

```bash
# Initialize Terraform
terraform init

# Apply the configuration
terraform apply
```

4. Set up kubectl for the new cluster:

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

Step 7: Scaling and Monitoring

Set up auto-scaling for your Kubernetes pods:

```bash
# Horizontal Pod Autoscaler
kubectl autoscale deployment devops-app-deployment --cpu-percent=50 --min=1 --max=5
```

Enable CloudWatch for monitoring:

```bash
# Install and start CloudWatch Logs agent
sudo apt-get install awslogs
sudo service awslogs start
```

Set up CloudWatch alarms for metrics like CPU usage and memory usage via the AWS Console.

Additional Configuration

CloudFront Setup

For improved content delivery, set up CloudFront:

1. Go to CloudFront → Create Distribution
2. Set the origin to the LoadBalancer URL or EC2 public IP

Route 53 Domain Management

1. Create a Hosted Zone in Route 53 for your domain

2. Set DNS records to point to your LoadBalancer:

```bash
aws route53 change-resource-record-sets --hosted-zone-id <zone-id> --change-batch file://record
```

Example records.json:

```json
{
  "Changes": [{
    "Action": "UPSERT",
    "ResourceRecordSet": {
      "Name": "yourdomain.com",
      "Type": "A",
      "AliasTarget": {
        "DNSName": "<ELB-DNS-Name>",
        "EvaluateTargetHealth": false,
        "HostedZoneId": "Z3DZXE0EIN2L7G"
      }
    }
  }]
}
```

Clean Up Resources

To avoid unnecessary AWS charges, delete resources when not in use:

```bash
# Delete Kubernetes resources
kubectl delete deployment devops-app-deployment
kubectl delete svc devops-app-service

# Destroy Terraform resources
terraform destroy

# Terminate EC2 instances if needed
aws ec2 terminate-instances --instance-ids <instance-id>
```

Troubleshooting

Docker Issues

If you encounter Docker repository issues, try:

```bash
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Jenkins Repository Error

If Jenkins installation fails due to repository issues:

```bash
sudo rm /etc/apt/sources.list.d/jenkins.list
sudo wget -q -O - https://pkg.jenkins.io/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable/ / > /etc/apt/sources.list.d/jenkins.l
sudo apt-get update
sudo apt-get install jenkins
```

Connection Issues

If you encounter connection issues with repositories:

1. Check your internet connection
2. Try changing Ubuntu mirrors in `/etc/apt/sources.list`
3. Check for proxy issues with `echo $http_proxy`