devops project - 2

Project Overview:

You are hired as a DevOps Engineer for Analytics Pvt Ltd to implement a DevOps lifecycle for automating deployment, scaling, and operations of application containers across clusters of hosts. The product is available on GitHub: https://github.com/hshar/website.git.

Tasks To Be Performed:

1. **Git Workflow:**
   - Implement version control with Git. Ensure releases happen only on the 25th of every month.
2. **CodeBuild:**
   - Trigger CodeBuild once commits are made in the master branch.
3. **Docker Containerization:**
   - Create a Dockerfile and build a custom Docker image every time there's a push to GitHub.
4. **Kubernetes Deployment:**
   - Deploy the containerized code from Docker Hub to a Kubernetes cluster with 2 replicas. Create a NodePort service on port 30008.
5. **Jenkins Pipeline:**
   - Create a Jenkins Pipeline script to accomplish the above tasks.
6. **Configuration Management:**
   - **Use Ansible to install Kubernetes on the servers. [Corrected]**
7. **Terraform for AWS Infrastructure:**
   - Use Terraform to create infrastructure in the AWS cloud provider.

Architectural Advice:

- **Worker1:** Jenkins, Java
- **Worker2:** Docker, Kubernetes
- **Worker3:** Java, Docker, Kubernetes
- **Worker4:** Docker, Kubernetes

Solution:

1. Git Workflow

Initialize Git Repository:

git init

git remote add origin https://github.com/hshar/website.git

Set Up Version Control:

git add .

git commit -m "Initial commit"

git push origin master

Schedule Release (Cron Job Example):

0 0 25 * * cd /path/to/repo && git push origin master

2. CodeBuild

Create buildspec.yml File:

yaml

```
version: 0.2

phases:

install:

 commands:

   - echo Installing dependencies...

build:

 commands:

   - echo Build started on `date`

    - echo Build completed on `date`
```

Trigger CodeBuild:

Set up a webhook in GitHub to trigger CodeBuild on commits to the master branch.

3. Docker Containerization

- **Create Dockerfile:**
- Dockerfile
- FROM node:14
- WORKDIR /app
- COPY . .
- RUN npm install
- CMD ["npm", "start"]
- **Build and Push Docker Image:**

- bash
- docker build -t yourdockerhubusername/website:latest .
- docker push yourdockerhubusername/website:latest

4. Kubernetes Deployment

Create Deployment YAML:

yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: website-deployment
spec:
replicas: 2
 selector:
   matchLabels:
    app: website
 template:
  metadata:
    labels:
    app: website
  spec:
    containers:
    - name: website
      image: yourdockerhubusername/website:latest
      ports:
      - containerPort: 3000
```

Create NodePort Service:

yaml

```
apiVersion: v1
kind: Service
metadata:
  name: website-service
spec:
  type: NodePort
  selector:
    app: website
  ports:
  - protocol: TCP
    port: 3000
    nodePort: 30008
```

Apply Configurations:

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

5. Jenkins Pipeline

Create Jenkinsfile:

```
groovy
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                script {
                    def app = docker.build("yourdockerhubusername/website:latest")
                    app.push()
```

```
            }
        }
    }
    stage('Deploy to Kubernetes') {
        steps {
            sh 'kubectl apply -f deployment.yaml'
            sh 'kubectl apply -f service.yaml'
        }
    }
  }
}
```

6. Configuration Management

Ansible Playbook for Worker1 (Jenkins and Java):

```
yaml
- hosts: worker1
  tasks:
    - name: Install Jenkins
      apt:
        name: jenkins
        state: present
    - name: Install Java
      apt:
        name: default-jdk
        state: present
```

Ansible Playbook for Worker2, Worker3, and Worker4 (Docker and Kubernetes):

```
yaml
```

```
- hosts: worker2:worker3:worker4
  tasks:
    - name: Install Docker
      apt:
        name: docker.io
        state: present
    - name: **Install Kubernetes**
      shell: |
        curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
        apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
        apt-get update
        apt-get install -y kubelet kubeadm kubectl
    - name: Start Docker Service
      service:
        name: docker
        state: started
        enabled: true
    - name: Start Kubernetes Service
      service:
        name: kubelet
        state: started
        enabled: true
```

7. Terraform for AWS Infrastructure

Terraform Configuration File (main.tf):

```
hcl
provider "aws" {
```

```
  region = "us-west-2"
}

resource "aws_instance" "app" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"

  tags = {
    Name = "AppInstance"
  }
}
```

Initialize and Apply Terraform:

```
terraform init
terraform apply
```