Module 6: kubernetes project - 1

You have been hired as a Sr. DevOps Engineer at Abode Software, tasked with implementing a DevOps lifecycle. The company provided the following requirements:

1. **Install necessary software on machines using a configuration management tool (Ansible).**
2. **Implement a Git workflow:**
   - Push to master branch triggers build, test, and deploy to production.
   - Push to develop branch triggers build and test only.
3. **Automate builds and tests using AWS CodeBuild and CodePipeline:**
   - Automatically trigger builds on commits to master or develop branches.
4. **Containerize the application using Docker and use the** hshar/webapp **container as a base.**
5. **Define the entire process in a Jenkins pipeline with the following jobs:**
   - **Job 1**: Build the application.
   - **Job 2**: Test the application.
   - **Job 3**: Deploy to production (for master branch).

Step-by-Step Solution

Step 1: Install Necessary Software Using Ansible

Create an EC2 Instance:

- Launch an EC2 instance using Amazon Linux 2 or Ubuntu AMI.
- Configure security group to allow SSH access.

Install Ansible:

- Update package index:

- sudo yum update -y
- Install EPEL repository and Ansible:

- sudo amazon-linux-extras install epel -y
- sudo yum install ansible -y

Create Inventory File (hosts**):**

[servers]

server1 ansible_host=54.241.188.83

[servers:vars]

ansible_user=ec2-user

ansible_ssh_private_key_file=C:/Users/harip/Desktop/intellipaat 24-09-2024/Dev ops/module 7 - kubernetes/project 1/key/jenkinsprojectonee.pem

Create Ansible Playbook (setup.yml**):**

yaml

```
---
- name: Setup Environment
  hosts: servers
  become: yes
  tasks:
    - name: Install Docker
      yum:
        name: docker
        state: present
        update_cache: yes
    - name: Install Git
      yum:
        name: git
        state: present
        update_cache: yes
```

Run Ansible Playbook:

```
ansible-playbook -i hosts setup.yml
```

Step 2: Implement Git Workflow

Clone the Repository:

```
git clone https://github.com/hshar/website.git
cd website
```

Set Up Branches:

```
git checkout -b develop
git push origin develop
```

Commit and Push Changes:

```
git add .
git commit -m "Made changes"
git push origin master
```

Step 3: Configure CodeBuild and CodePipeline

Set Up AWS CodeBuild:

- Create a new CodeBuild project in the AWS Management Console.
- Configure the source to point to your GitHub repository.

Create/Update buildspec.yml **in Repository:**

```
yaml
version: 0.2

phases:
  install:
    runtime-versions:
      docker: 19
```

```
  commands:
    - echo "Installing Dependencies"
    - apt-get update
    - apt-get install -y docker.io

  pre_build:
    commands:
    - echo "Starting Pre-Build Phase"
    - aws ecr get-login-password --region us-west-1 | docker login --username AWS --password-stdin 140023376730.dkr.ecr.us-west-1.amazonaws.com

  build:
    commands:
    - echo "Building Docker Image"
    - docker build -t webapp .

  post_build:
    commands:
    - echo "Build Completed"
    - echo "Pushing Docker Image"
    - docker push $DOCKERHUB_USERNAME/webapp:latest

artifacts:
  files:
   - '**/*'
```

Set Up AWS CodePipeline:

- Create a new pipeline in AWS CodePipeline.
- Define source stage with GitHub repository.
- Add build stage using CodeBuild project.
- Optionally add deploy stage.

Step 4: Dockerize the Application

Create Dockerfile **in Repository:**

dockerfile

```
# Use an official webserver image as the base image

FROM hshar/webapp



# Copy application code to the container

COPY . /var/www/html
```

Add and Commit the Dockerfile:

```
git add Dockerfile

git commit -m "Added Dockerfile"

git push origin master
```

Step 5: Set Up Jenkins Pipeline

Install Jenkins and Plugins:

- Install Jenkins on your server.
- Install necessary plugins (Git, Docker, AWS CodeBuild).

Create Jenkins Pipeline Job:

- Open Jenkins and create a new pipeline job.
- Define the pipeline script:
- groovy
- pipeline {
- agent any
- environment {
- DOCKERHUB_CREDENTIALS = credentials('dockerhub-credentials')
- }
- stages {
- stage('Build') {

```
        steps {
            script {
                def branch = env.GIT_BRANCH
                if (branch == 'origin/master' || branch == 'origin/develop') {
                    sh 'docker build -t hshar/webapp .'
                }
            }
        }
    }
    stage('Test') {
        steps {
            script {
                def branch = env.GIT_BRANCH
                if (branch == 'origin/master' || branch == 'origin/develop') {
                    sh 'docker run -d -p 80:80 hshar/webapp'
                    sh 'curl -f http://localhost:80'
                }
            }
        }
    }
    stage('Prod') {
        when {
            branch 'master'
        }
        steps {
            script {
                sh 'docker tag hshar/webapp hshar/webapp:latest'
                sh 'docker push hshar/webapp:latest'
            }
        }
    }
}
```