Upgrade Project on AWS

I got one project after completion of AWS course at Upgrade. This project divide into 3 part.

**Part I:**

**Step1**. I create 3 EC2 machine named “bastion” , “Jenkins”, “Appserver”. To create these 3 EC2 machine I use terraform code (main.tf) file are available in git repo. “Bastion” running on public IP but “Jenkins” & “appserver” are running on private Ip.

**Step2.**: Download .ppk file from “Bastion” to login the machine through putty. After login bastion I generate .pem file & Scp to “Bastion”. Now I can ssh both the private machine with private key.

**Step 3**: now install Ansible on bastion. Then I create “inventory.ini” file to create connection for ansible with “Jenkins” & “appserver”.

Then we create ansible playbook file “Install\_docker.yaml” .Then run below command to install docker on both the machine.

**ubuntu@ip-10-0-1-87:~$ ansible-playbook -i inventory.ini install\_docker.yam**

**Part II:**

**Step1**. Now SSH to Jenkins machine. Now install Jenkins in Jenkins EC2 machine. But machine on private subnet . So to open from outside loadbalancer (Application load balancer) need to create.

Step2: log on to portal & first create “Target Group”. After create Target Group need to create ALB to expose. Then need to add listener. Also input path for login “/login”.

After that need to expose port “80” & “8080” in “inbound Roll”.

After completion of ALB now I can access Jenkins from outside.

**Part III**

Step 1: Now login to Jenkins & Create appserver as worker Node. Then install JAVA in appserver.

Step 2: Create credential for access github repo where voting app are the .Also add credential for docker hub for “docker push” Image

Step 3: Now write 1 docker file & Jenkins file

DOCKERFILE

FROM python:3.9-slim

RUN ls

RUN apt-get update \

    && apt-get install -y --no-install-recommends \

    curl \

    && rm -rf /var/lib/apt/lists/\*

WORKDIR /app

COPY requirements.txt /app/requirements.txt

RUN pip install -r requirements.txt

COPY . .

EXPOSE 80

CMD ["gunicorn", "app:app", "-b", "0.0.0.0:80", "--log-file", "-", "--access-logfile", "-", "--workers", "4", "--keep-alive", "0"]

Jenkins file

pipeline {

agent { label 'appserver-agent' }

environment {

IMAGE\_NAME = 'jaydeepc1985/voting-app'

TAG = 'latest'

}

stages {

stage('Checkout Code') {

steps {

git branch: 'master', url: 'https://github.com/Jaydeepc1985/voting-app.git'

}

}

stage('Build Docker Image') {

steps {

sh 'docker build -t $IMAGE\_NAME:$TAG vote/'

}

}

stage('Push Docker Image') {

steps {

withCredentials([usernamePassword(credentialsId: 'dockerhub-creds', usernameVariable: 'DOCKER\_USER', passwordVariable: 'DOCKER\_PASS')]) {

sh '''

echo "$DOCKER\_PASS" | docker login -u "$DOCKER\_USER" --password-stdin

docker push $IMAGE\_NAME:$TAG

'''

}

}

}

stage('Remove Old Container') {

steps {

sh 'docker rm -f voting-app || true'

}

}

stage('Run New Container') {

steps {

sh 'docker run -d --name voting-app -p 80:80 $IMAGE\_NAME:$TAG'

}

}

}

post {

always {

echo 'Pipeline execution complete.'

}

failure {

echo 'Pipeline failed!'

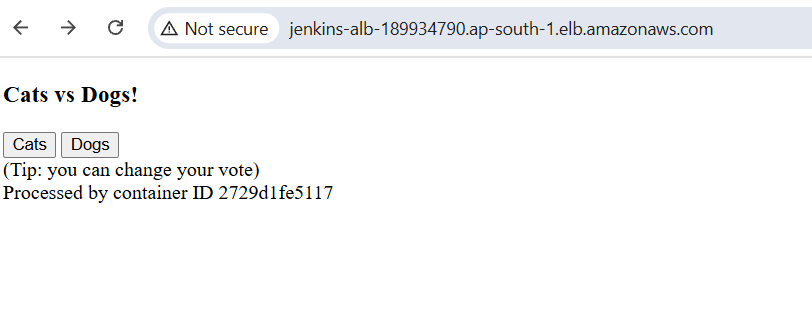
}

}

}

Then run the build manually & complete the build task.

Step 3: Create target group for Appserver & connect with ALB to expose it on internet.



After Exposing our Vote app our I complete my whole project on Aws