**KELP Assignment**

GitURL: https://github.com/technicalhrushi/DevOps-Assignment.git  
  
**Part1 – Installation for local Setup**  
Step1 : Install Docker

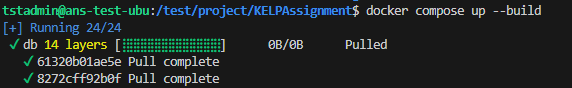
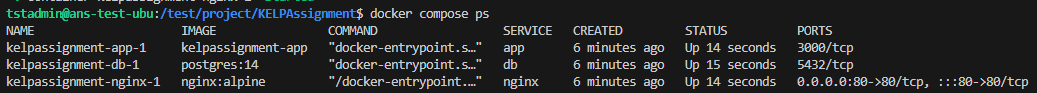
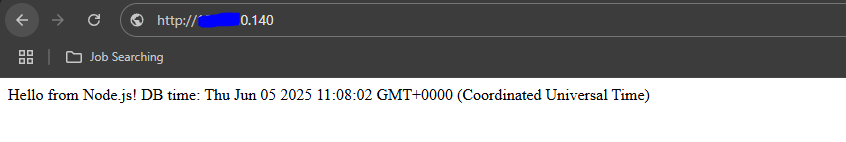
|  |
| --- |
| sudo apt-get update sudo apt install docker.io |

Step2: Install Docker Compose  
  
**Part2 – Containerize the Application**Step1: Files used for the containerzing the Application

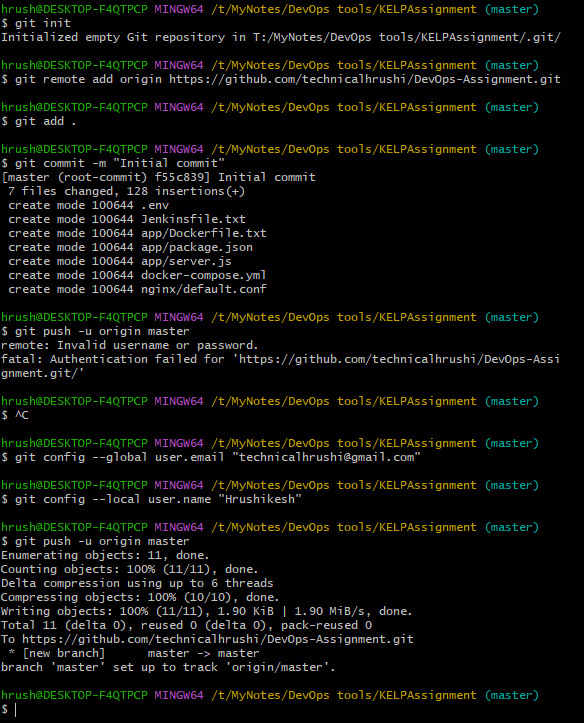
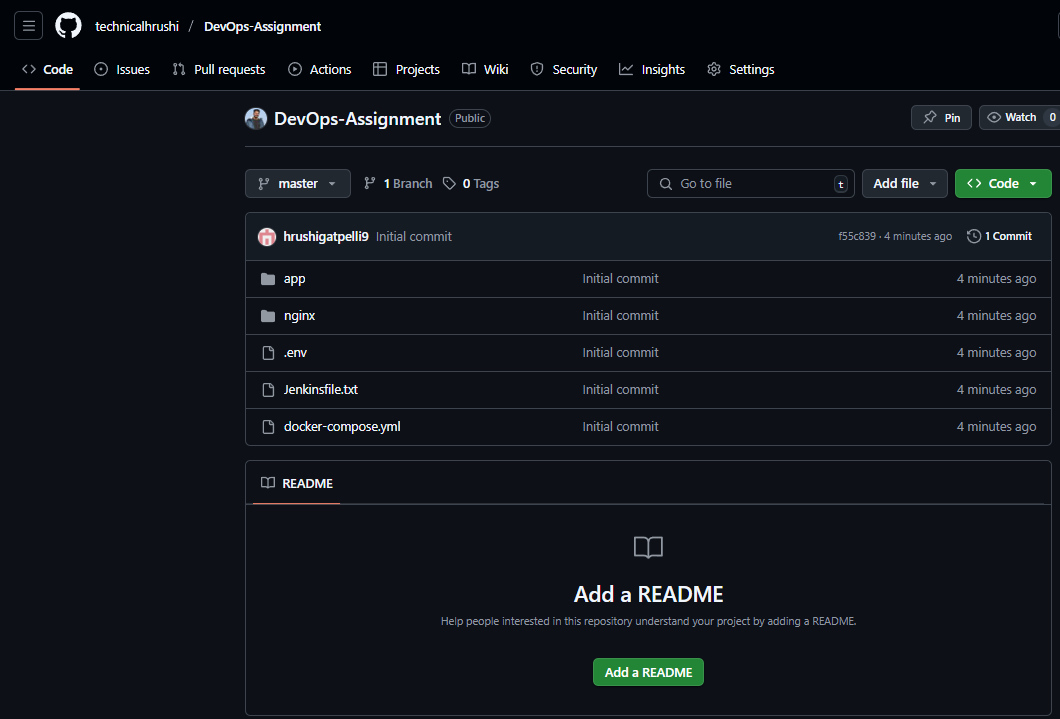
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | --------------------------------------------------------server.js------------------------------------------------------------ const express = require('express');  const { Pool } = require('pg');  const app = express();  const port = 3000;  const pool = new Pool({  user: process.env.POSTGRES\_USER,  host: 'db',  database: process.env.POSTGRES\_DB,  password: process.env.POSTGRES\_PASSWORD,  port: 5432,  });  app.get('/', async (req, res) => {  const result = await pool.query('SELECT NOW()');  res.send(`Hello from Node.js! DB time: ${result.rows[0].now}`);  });  app.listen(port, () => {  console.log(`App running on http://localhost:${port}`);  }); |      |  | | --- | | --------------------------------------------------------Dockerfile--------------------------------------------------------- FROM node:18  WORKDIR /usr/src/app  COPY package\*.json ./  RUN npm install  COPY . .  EXPOSE 3000  CMD ["node", "server.js"] |      |  | | --- | | ------------------------------------------------------package.json-------------------------------------------------------- {  "name": "simple-node-app",  "version": "1.0.0",  "main": "server.js",  "scripts": {  "start": "node server.js"  },  "dependencies": {  "express": "^4.18.2",  "pg": "^8.10.0"  }  } |      |  | | --- | | -----------------------------------------------------default.conf -------------------------------------------------------- server {  listen 80;  location / {  proxy\_pass http://app:3000;  }  } |      |  | | --- | | ----------------------------------------------------docker-compose.yml----------------------------------------------- version: '3.8'  services:  app:  build: ./app  environment:  - POSTGRES\_USER=${POSTGRES\_USER}  - POSTGRES\_PASSWORD=${POSTGRES\_PASSWORD}  - POSTGRES\_DB=${POSTGRES\_DB}  depends\_on:  - db  db:  image: postgres:14  environment:  POSTGRES\_USER: ${POSTGRES\_USER}  POSTGRES\_PASSWORD: ${POSTGRES\_PASSWORD}  POSTGRES\_DB: ${POSTGRES\_DB}  volumes:  - db\_data:/var/lib/postgresql/data  nginx:  image: nginx:alpine  ports:  - "80:80"  volumes:  - ./nginx/default.conf:/etc/nginx/default.conf  depends\_on:  - app  volumes:  db\_data: |  |  | | --- | | ------------------------------------------------------------.env------------------------------------------------------------- POSTGRES\_USER=postgres  POSTGRES\_PASSWORD=secret  POSTGRES\_DB=mydb |      |  | | --- | | --------------------------------------------------------JenkinsFile--------------------------------------------------------- pipeline {  agent any  parameters {  string(name: 'EC2\_PUBLIC\_IP', defaultValue: '', description: 'Enter EC2 Public IP')  }    environment {  DOCKER\_IMAGE = 'technicalhrushi/devops-app'  }  stages {  stage('Build') {  steps {  sh 'docker-compose build'  }  }  stage('Test') {  steps {  sh '''  docker-compose up -d  docker-compose ps  docker-compose logs app  docker compose exec -T app curl -v http://localhost:3000 || exit 1  '''  }  }  stage('Push') {  steps {  withCredentials([usernamePassword(credentialsId: 'dockerhub', usernameVariable: 'USER', passwordVariable: 'PASS')]) {  sh '''  echo "$PASS" | docker login -u "$USER" --password-stdin  docker tag kelpassignment-app $DOCKER\_IMAGE  docker push $DOCKER\_IMAGE  '''  }  }  }  stage('Deploy to AWS') {  steps {  withCredentials([  usernamePassword(credentialsId: 'dockerhub', usernameVariable: 'USER', passwordVariable: 'PASS'),  sshUserPrivateKey(credentialsId: 'aws-ec2-ssh', keyFileVariable: 'KEY')  ]) {  sh '''  chmod 600 "$KEY"  ssh -o StrictHostKeyChecking=no -i "$KEY" ubuntu@${EC2\_PUBLIC\_IP} <<EOF  echo "$PASS" | docker login -u "$USER" --password-stdin  mkdir -p ~/devops-deploy && cd ~/devops-deploy  cat > docker-compose.yml <<EOL  services:  app:  image: technicalhrushi/devops-app  environment:  - POSTGRES\_USER=postgres  - POSTGRES\_PASSWORD=secret  - POSTGRES\_DB=mydb  depends\_on:  - db  db:  image: postgres:14  environment:  - POSTGRES\_USER=postgres  - POSTGRES\_PASSWORD=secret  - POSTGRES\_DB=mydb  volumes:  - db\_data:/var/lib/postgresql/data  nginx:  image: nginx:alpine  ports:  - "80:80"  volumes:  - ./nginx/default.conf:/etc/nginx/conf.d/default.conf  depends\_on:  - app  volumes:  db\_data:  EOL  mkdir -p nginx  cat > nginx/default.conf <<NGINX  server {  listen 80;  location / {  proxy\_pass http://app:3000;  }  }  NGINX  docker compose down || true  docker compose up -d  EOF  '''  }  }  }  }  } | |

Step2 : Create A Directory and place the below files inside it

|  |  |
| --- | --- |
| devops-assignment/  │  ├── app/  │ ├── server.js  │ ├── package.json  │ └── Dockerfile  │  ├── nginx/  │ └── default.conf  │  ├── docker-compose.yml  ├── .env  └── Jenkinsfile |  |

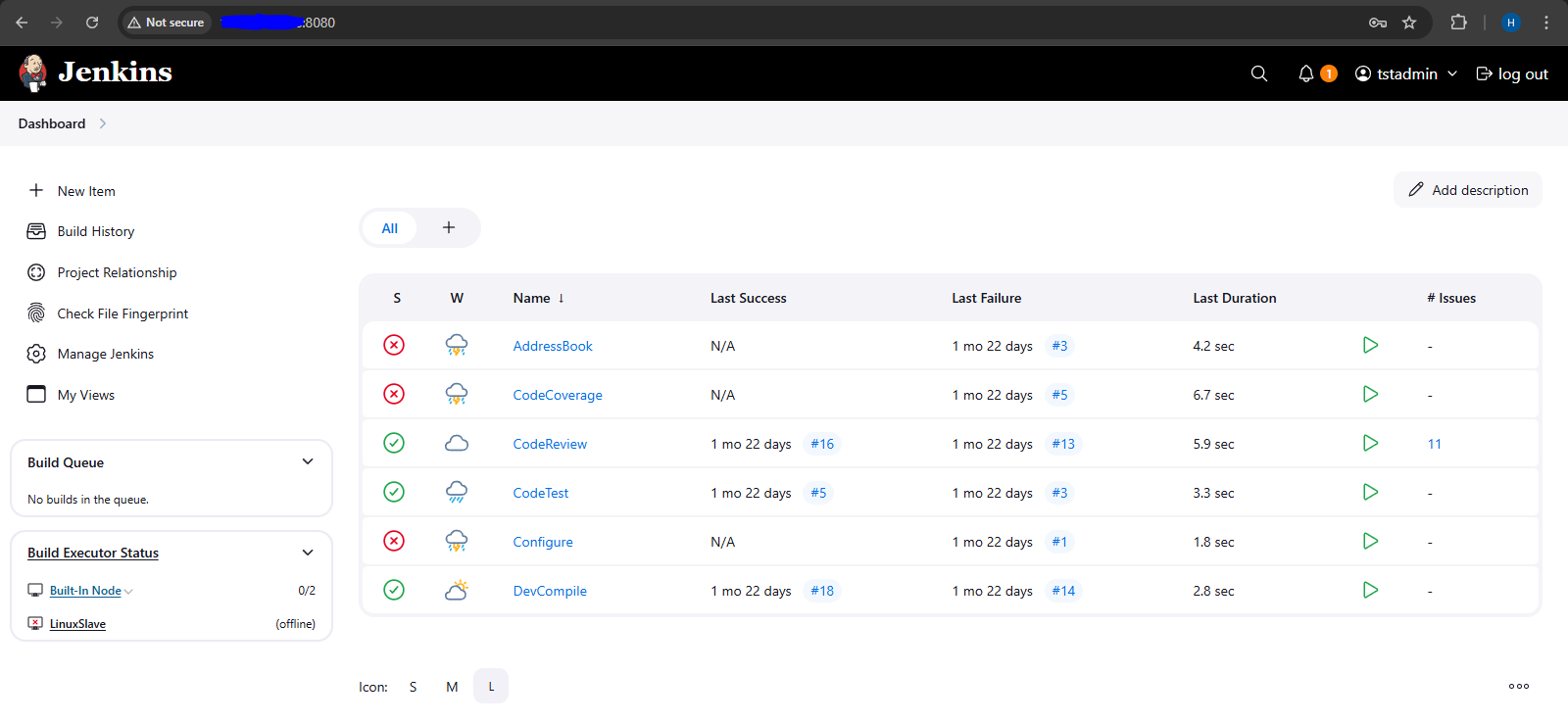
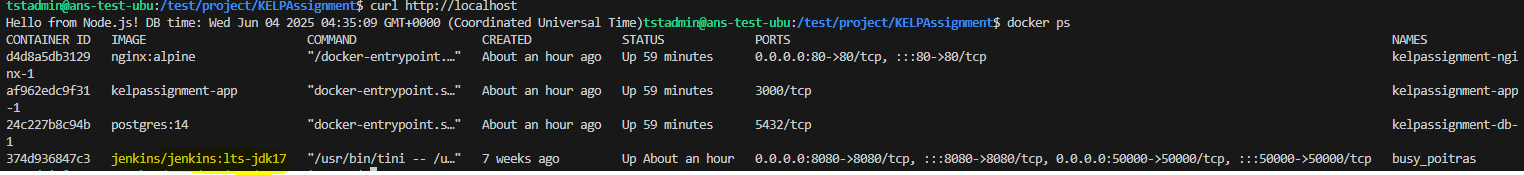
Step3: Execute the docker compose up --build  
  
  
Step4: check the present state  
  
  
Step5: check if the application is Running  
  


**Part3**  
Step1: Git Setup

Step2: Setup Jenkins Container

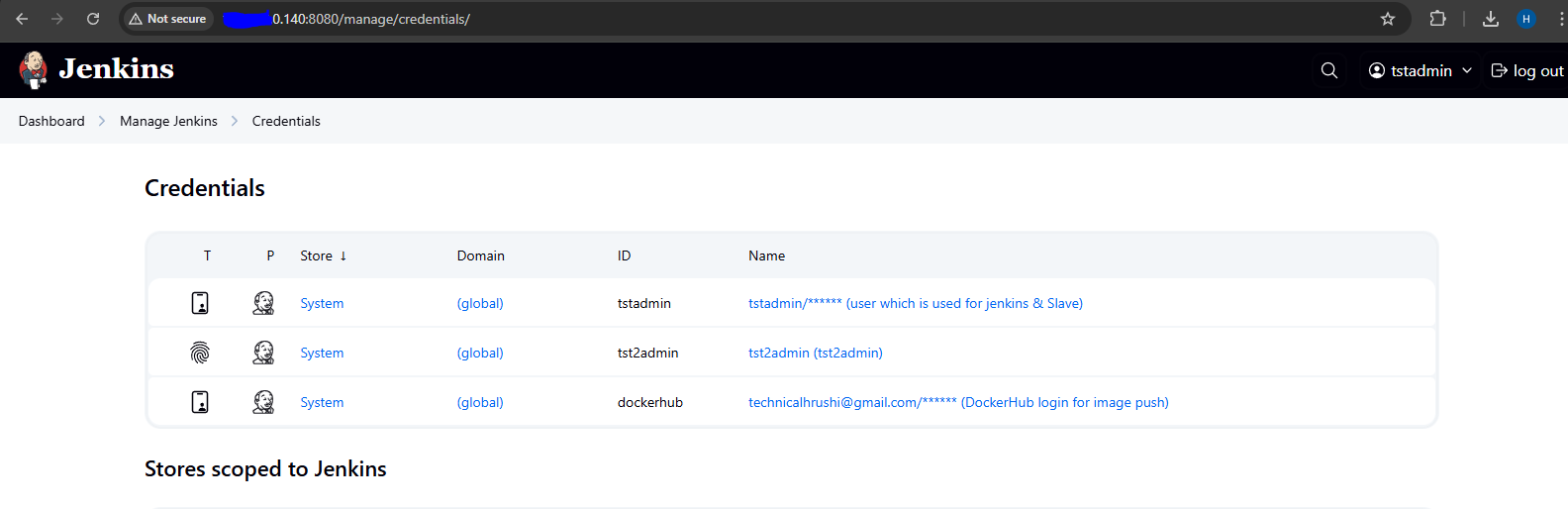
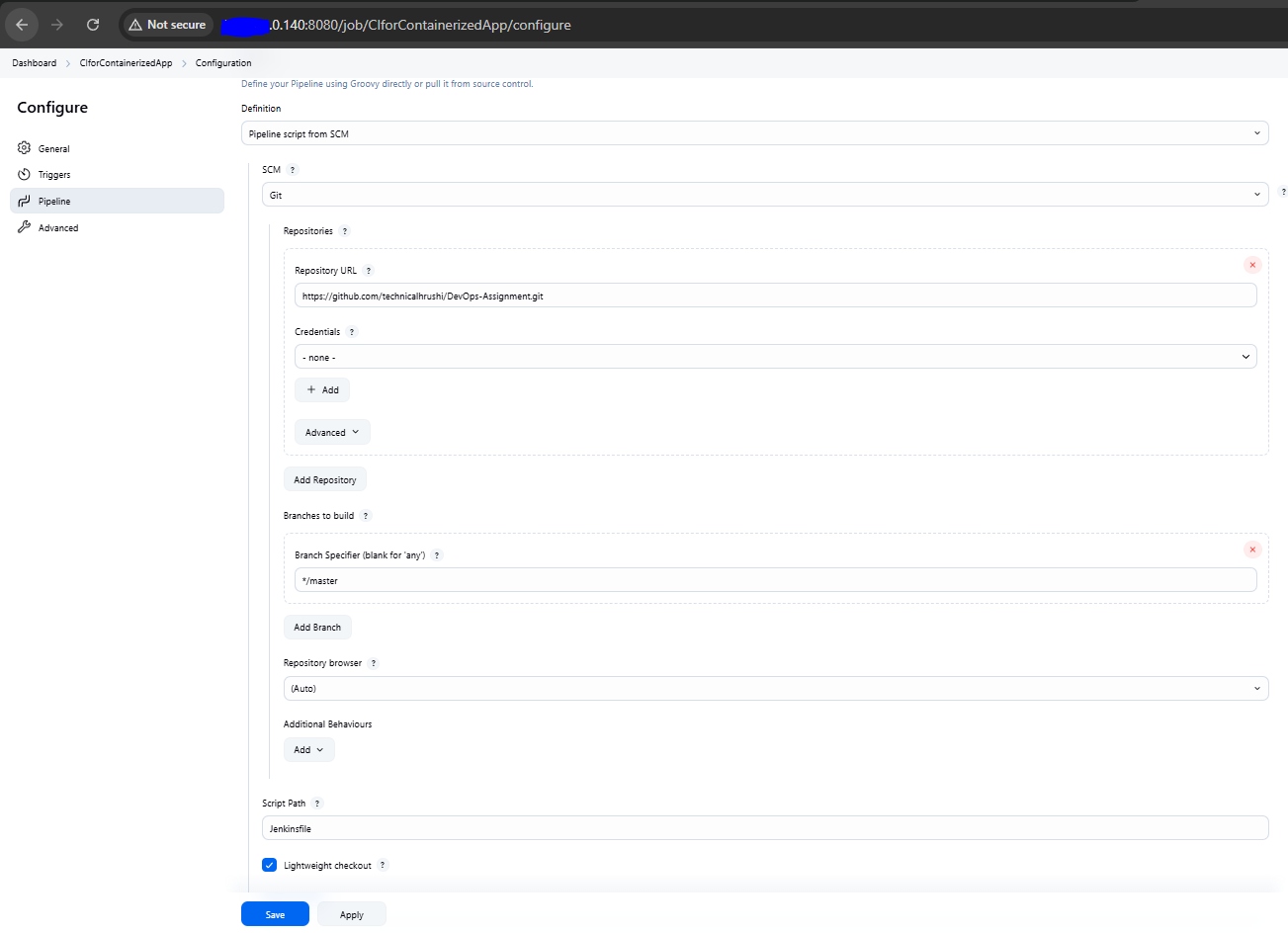
|  |
| --- |
| docker run -d --user root --name jenkins -p 8080:8080 -p 50000:50000 -v jenkins\_home:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock --restart=on-failure jenkins/jenkins:lts-jdk17 |

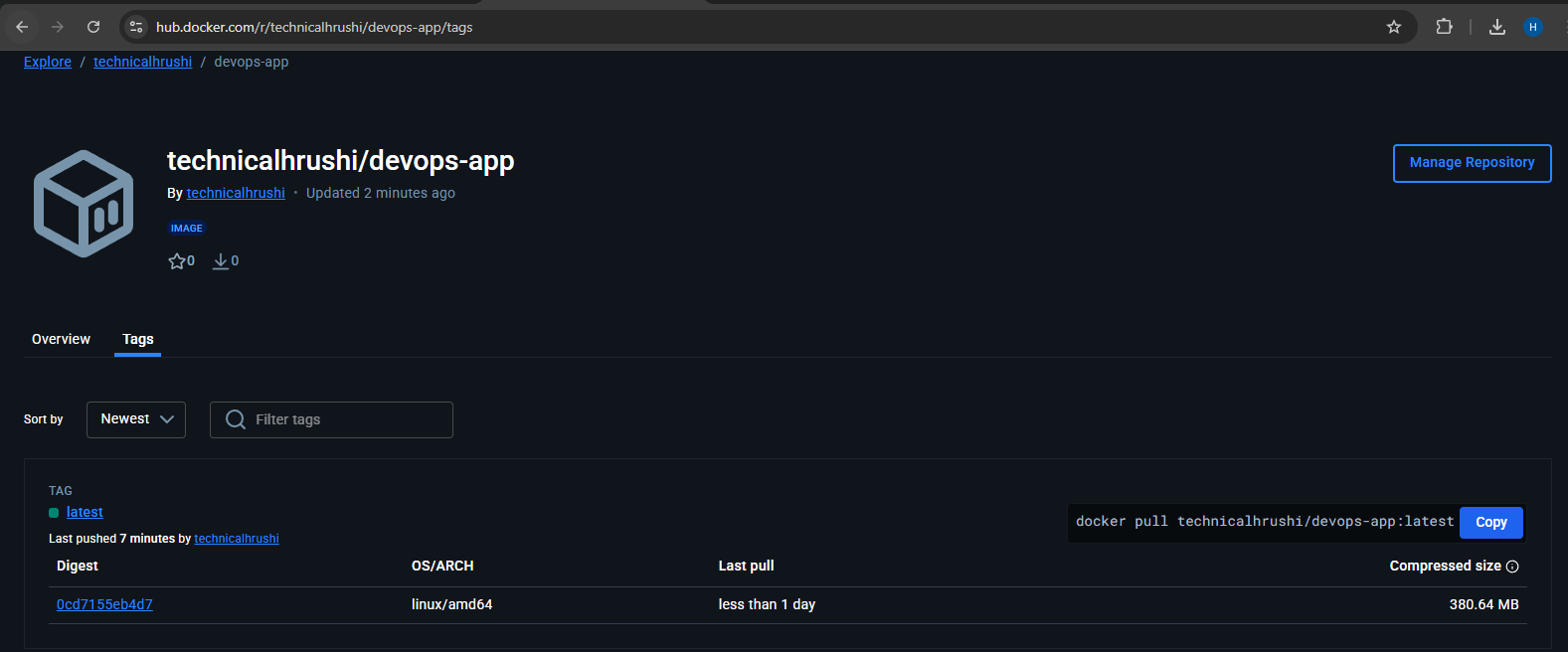
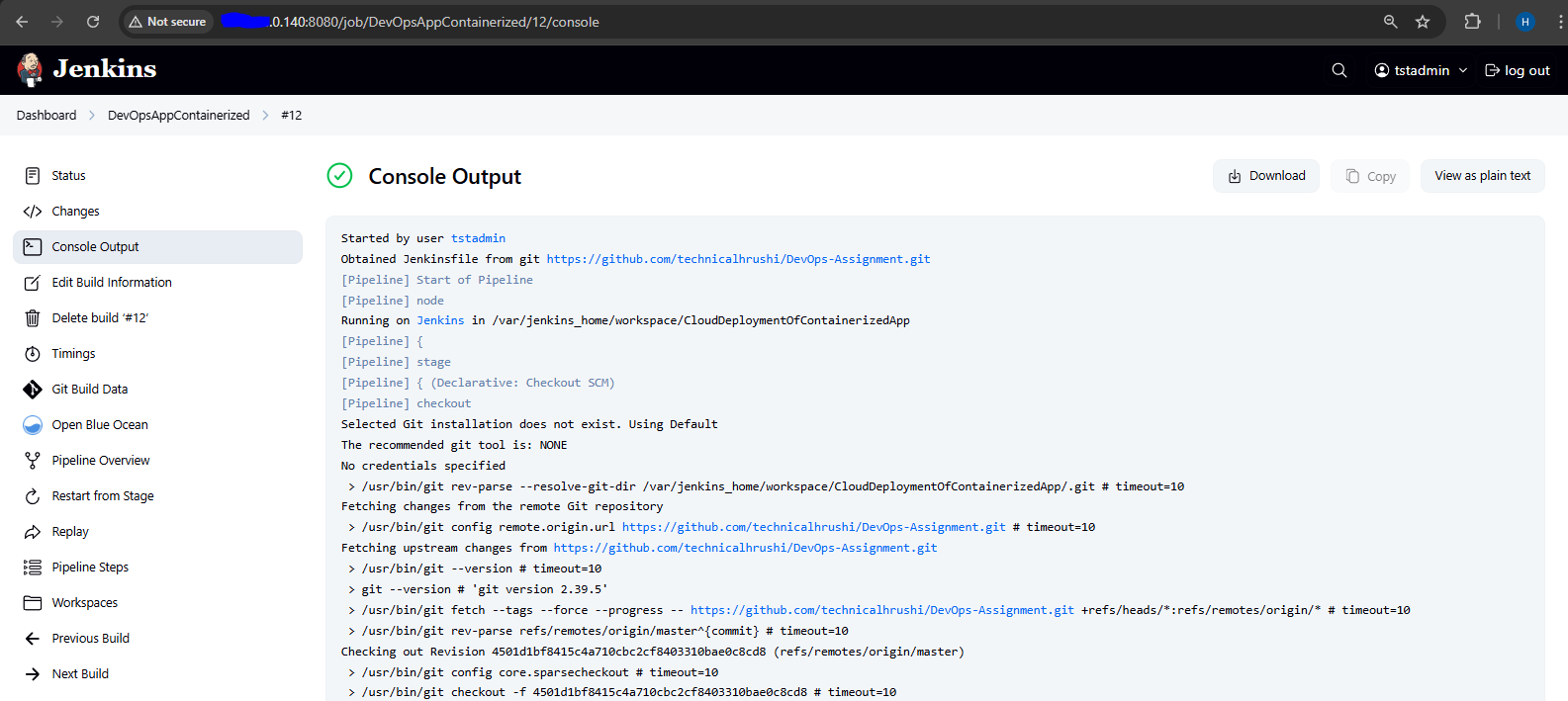


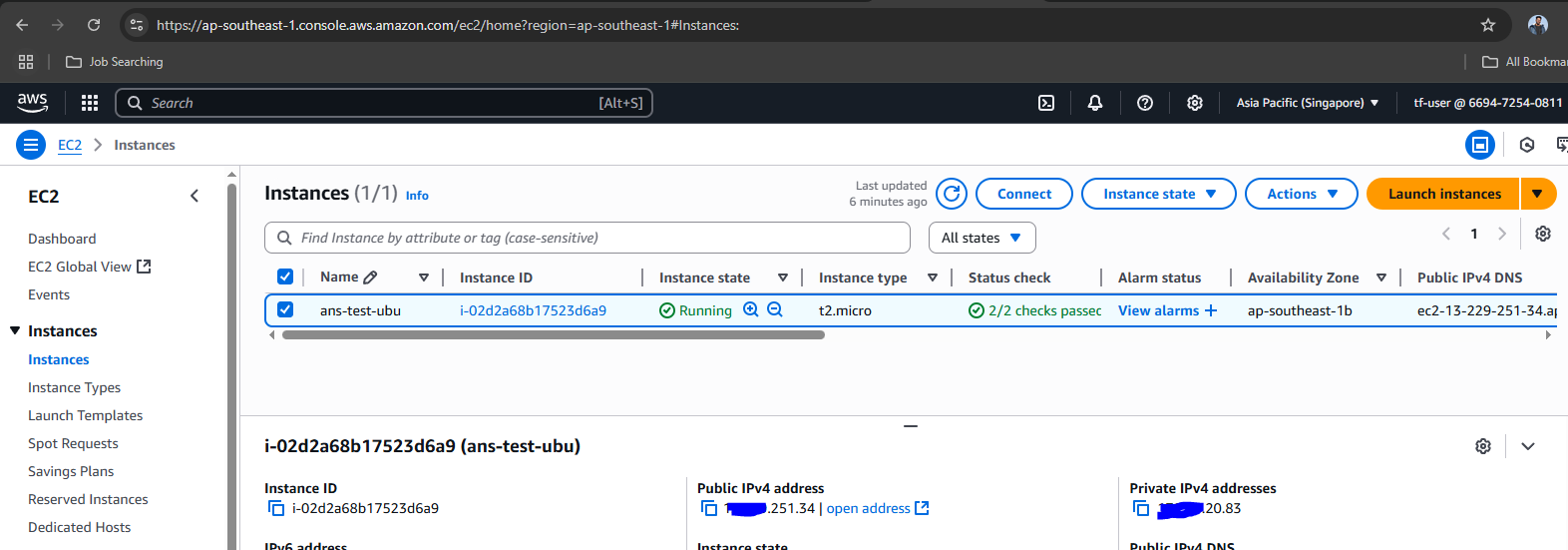
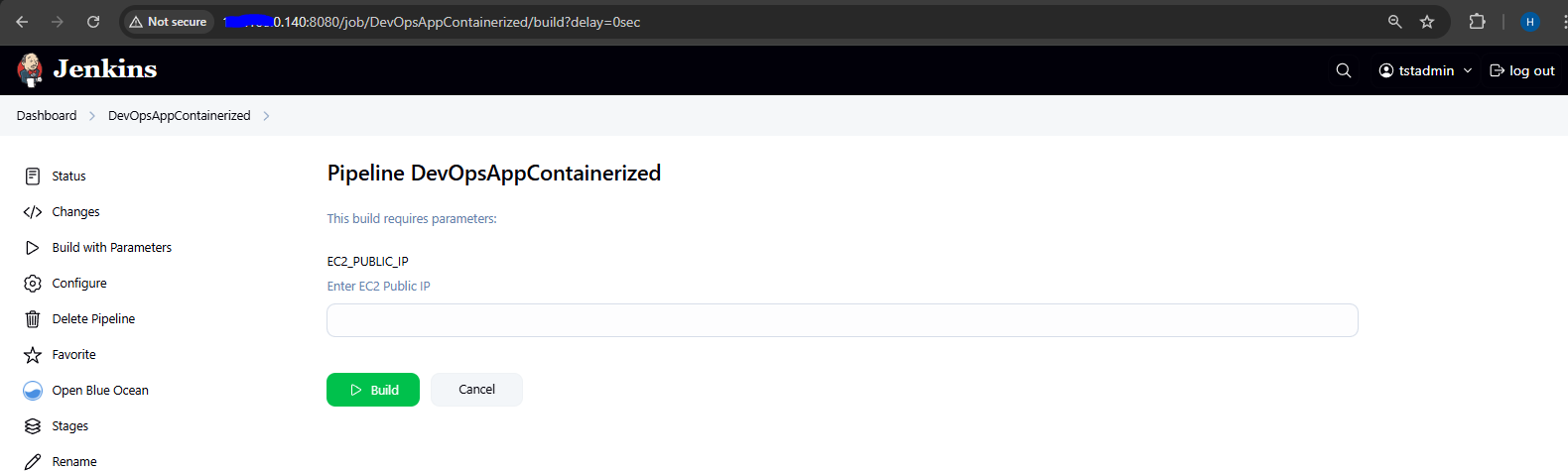
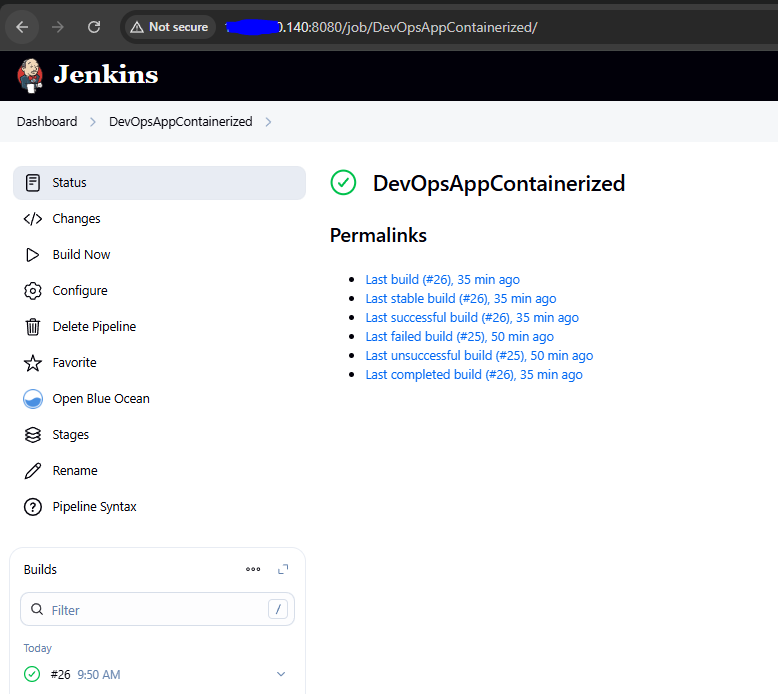
Step3: Setup Docker and DockerCompose in Jenkins container, need to execute both the below bash files

|  |
| --- |
| ------------------------------------------------Dockerconfigurationfile--------------------------------------------------- #!/bin/bash  # 1. Install prerequisites  apt-get update && apt-get install -y \  ca-certificates \  curl \  gnupg \  lsb-release  # 2. Add Docker’s official GPG key  mkdir -p /etc/apt/keyrings  curl -fsSL https://download.docker.com/linux/debian/gpg | \  gpg --dearmor -o /etc/apt/keyrings/docker.gpg  # 3. Set up the Docker repository  echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \  https://download.docker.com/linux/debian $(lsb\_release -cs) stable" | \  tee /etc/apt/sources.list.d/docker.list > /dev/null  # 4. Update the package index again  apt-get update  # 5. Install Docker CLI  apt-get install -y docker-ce-cli |

|  |
| --- |
| -------------------------------------------DockerComposeConfigurationfile------------------------------------------ #/bin/bash  DOCKER\_COMPOSE\_VERSION=$(curl -s https://api.github.com/repos/docker/compose/releases/latest | grep tag\_name | cut -d '"' -f 4)  curl -L "https://github.com/docker/compose/releases/download/${DOCKER\_COMPOSE\_VERSION}/docker-compose-$(uname -s)-$(uname -m)" \  -o /usr/local/bin/docker-compose  chmod +x /usr/local/bin/docker-compose  docker-compose version |

Step4: Setup the Docker Creds on Jenkins  
  
 **Part4: Setting up Jenkins Pipeline**

**SuccessResult** **PART5: Pushing the same code to AWS EC2 Cloud**

Step1: Created EC2Step2: Configure it in Jenkins FileStep3: Trigger the Job

# Step4: Check the output Part 6: AWS EC2 + Auto Scaling + ALB (Overview)

1. Create custom AMI from EC2 instance with Docker & Compose pre-installed
2. Create Launch Template using the AMI
3. Create Auto Scaling Group:
4. Attach ALB
5. Set min=1, desired=2, max=4
6. Use CPU-based scaling policy
7. Confirm EC2s scale and ALB health checks pass