TASK-2 Simple Jenkins Pipeline for CI/CD

Objective:

Set up a basic Jenkins pipeline to automate the process of building and deploying an application.

Tools Used:

Jenkins: Automation server for CI/CD

Docker: For containerizing the app

AWS EC2: Server to host Jenkins and run your app

GitHub: Source code repository

Steps 1:

1. Launch ec2 instance(linux or ubuntu)
2. Enable security groups
3. Install Jenkins for this cmds:
    - sudo yum update -y
    - sudo yum install git -y
    - sudo yum install java-17-amazon-corretto.x86_64 (first install java before jenkins)
    - sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
    - sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
    - sudo yum install jenkins -y
    - sudo systemctl enable jenkins
    - sudo systemctl start jenkins
    - Access Jenkins on http://<EC2-Public-IP>:8080
4. **Install Jenkins Plugins:**
    - Docker Pipeline
    - GitHub Integration
    - Stage view
    - SSH Agent(optional)
5. **Install Docker for this cmds:**
    - sudo yum install docker -y
    - sudo usermod -aG docker ec2-user
    - sudo usermod -aG docker Jenkins
    - sudo service docker start
    - sudo systemctl enable docker
    - sudo docker --version

Steps 2:

Create a new repo in GitHub and the clone the repo

connect to ec2 server and clone the repo

Create a sample code of nodejs like --> app.js, package.js

#Create a Dockerfile and Jenkinsfile

Jenkinsfile

```
pipeline {
    agent any

    environment {
        APP_NAME = "nodejs-demo-app"
        DOCKER_IMAGE = " your-dockerhub-username /nodejs-demo-app:latest"
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/ your-username / your-nodejs-repo.git '
            }
        }

        stage('Build Docker Image') {
            steps {
                sh 'docker build -t $DOCKER_IMAGE .'
            }
        }

        stage('Test') {
            steps {
                sh 'docker run --rm $DOCKER_IMAGE npm test || echo "No tests found"'
            }
        }

        stage('Push to DockerHub') {
            steps {
                withCredentials([usernamePassword(
                    credentialsId: 'dockerhub-creds',
                    usernameVariable: 'USERNAME',
                    passwordVariable: 'PASSWORD'
                )]) {
                    sh '''
                        echo "$PASSWORD" | docker login -u "$USERNAME" --password-stdin
```

```
                docker push $DOCKER_IMAGE
            '''
          }
        }
      }

      stage('Deploy to EC2') {
        steps {
          sh '''
            docker stop $APP_NAME || true
            docker rm $APP_NAME || true
            docker pull $DOCKER_IMAGE
            docker run -d --name $APP_NAME -p 3001:3000 $DOCKER_IMAGE
          '''
        }
      }
    }
}
```

Step 3:

```
# push the nodejs app and Dockerfile and Jenkinsfile to your git repo
# Go to your Github repo ---> setting -->webhooks
# Payload URL: http://<ec2-ip>:8080/github-webhook/
# Content type: application/json
```

Step 4:

```
# Go to Jenkins Dashboard → Manage Jenkins → Credentials → Global.
# Click Add Credentials:
```

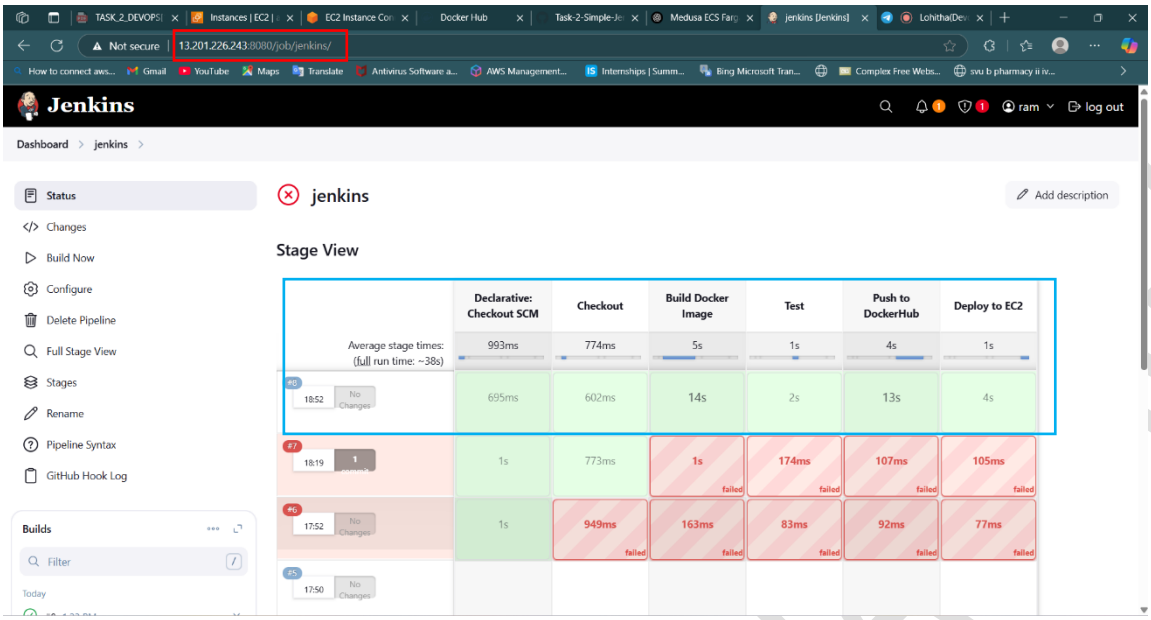
- **Kind**: Username & Password
- **Username**: Your GitHub/DockerHub username
- **Password**:
  1. For GitHub → use a **Personal Access Token (PAT)**
  2. For DockerHub → use your DockerHub password (or token)
- **ID**: e.g., dockerhub-creds(docker) or github-creds(git)
- Use this ID in your Jenkinsfile.

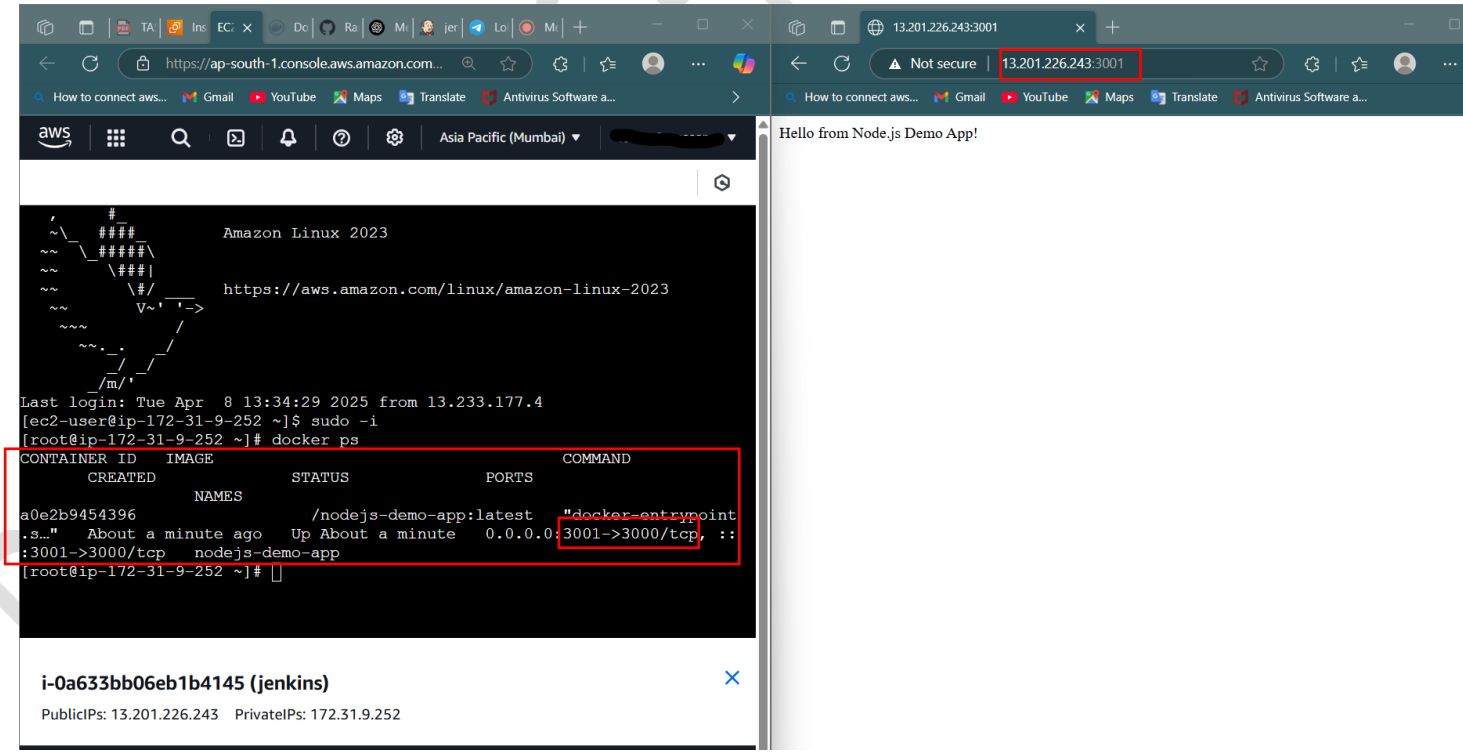
```
# Click "OK" or "Save"
```

Steps 5: **Test It**

1. Push changes to GitHub repo.

2. Jenkins should pull code, build image, push to DockerHub, and deploy.
3. Go to ec2 server check container ------docker ps





Tips:

→ Use Secrets in Jenkins instead of hardcoding passwords.

→ Always add logging to your stages for easier debugging.

→ Use tags for Docker images for version control.