

Project 01

Problem Statement:

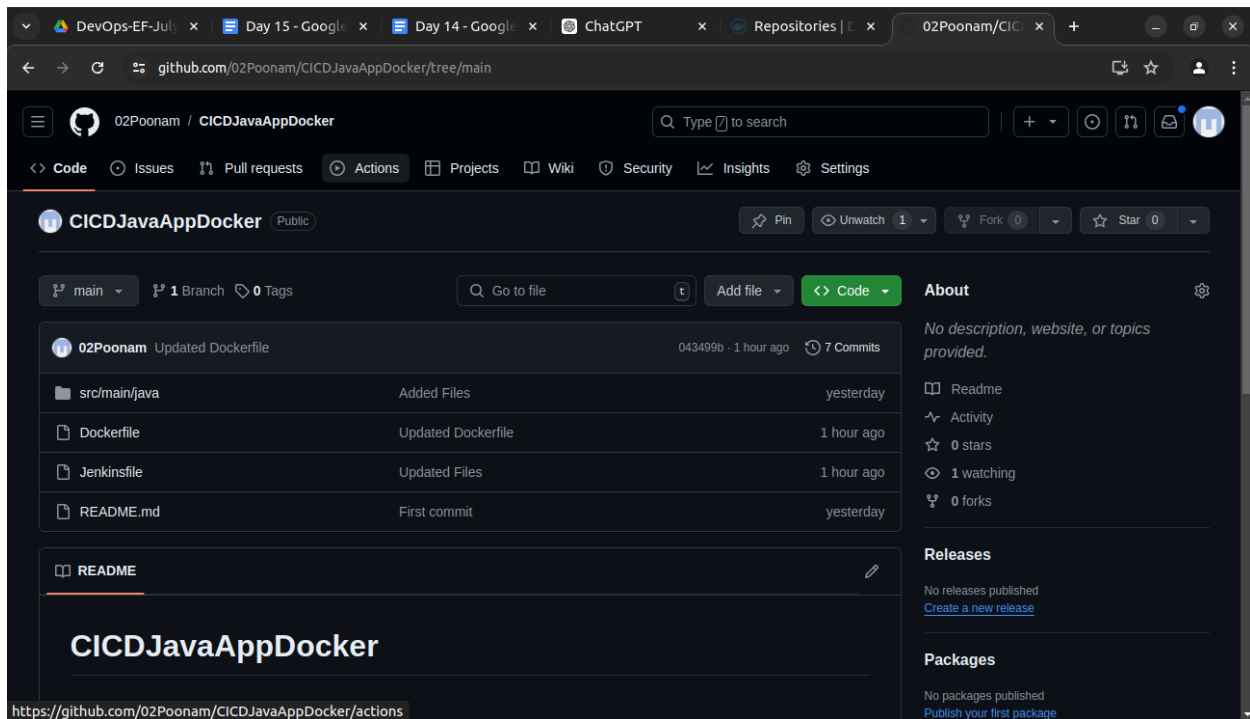
You are tasked with setting up a CI/CD pipeline using Jenkins to streamline the deployment process of a simple Java application. The pipeline should accomplish the following tasks:

1. **Fetch the Dockerfile:** The pipeline should clone a GitHub repository containing the source code of the Java application and a Dockerfile.
2. **Create a Docker Image:** The pipeline should build a Docker image from the fetched Dockerfile.
3. **Push the Docker Image:** The pipeline should push the created Docker image to a specified DockerHub repository.
4. **Deploy the Container:** The pipeline should deploy a container using the pushed Docker image.

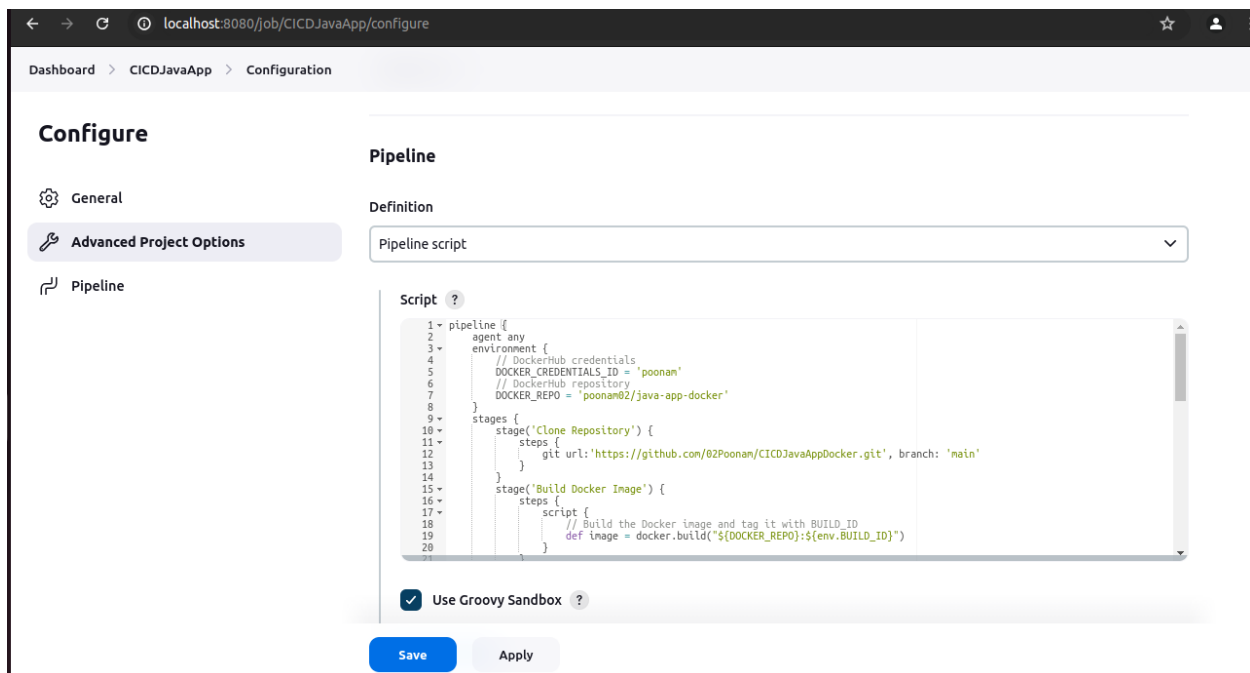
Deliverables:

5. **GitHub Repository:** A GitHub repository containing:
 - The source code of a simple Java application
 - A Dockerfile for building the Docker image
6. **Jenkins Pipeline Script:** A Jenkinsfile (pipeline script) that:
 - Clones the GitHub repository.
 - Builds the Docker image.
 - Pushes the Docker image to DockerHub.
 - Deploys a container using the pushed image.
7. **DockerHub Repository:** A DockerHub repository where the Docker images will be stored.
8. **Jenkins Setup:**
 - Jenkins installed and configured on a local Ubuntu machine.
 - Required plugins installed (e.g., Git, Docker, Pipeline).
9. **Documentation:** Detailed documentation explaining:
 - How to set up the local Jenkins environment.
 - Configuration steps for the pipeline.
 - Instructions for verifying the deployment.

Github Repository



Create a new Jenkins pipeline and configure it as follows



Also set up the local Jenkins Environment by adding the required plugins and credentials of Dockerhub

Manage Jenkins > Credentials > System > Global Credentials > Add Credentials

localhost:8080/manage/credentials/store/system/domain/_/newCredentials

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind
Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
|

☐ Treat username as secret ?

Password ?

ID ?

Create

Add your Username, password and id of Dockerhub

Then you will be able to see your Credentials as follows.

localhost:8080/manage/credentials/

Jenkins Search (CTRL+K) Poonam Mahesh Bhavsar log out

Dashboard > Manage Jenkins > Credentials

Credentials

T	P	Store	Domain	ID	Name
		System	(global)	poonam02	poonam/*****
		System	(global)	poonam	poonam02/*****

Stores scoped to Jenkins

P	Store	Domains
	System	(global)

Icon: S M L

Jenkins Pipeline Script

Jenkinsfile (pipeline script) that clones the GitHub repository, builds the Docker image, pushes the Docker image to Dockerhub and deploys the container using the pushed image

```

GNU nano 6.2
pipeline {
  agent any
  environment {
    // DockerHub credentials
    DOCKER_CREDENTIALS_ID = 'poonan'
    // DockerHub repository
    DOCKER_REPO = 'poonan82/java-app-docker'
  }
  stages {
    stage('Clone Repository') {
      steps {
        git url:'https://github.com/82Poonan/CICDJavaAppDocker.git', branch: 'main'
      }
    }
    stage('Build Docker Image') {
      steps {
        script {
          // Build the Docker image and tag it with BUILD_ID
          def image = docker.build("${DOCKER_REPO}:${env.BUILD_ID}")
        }
      }
    }
    stage('Push Docker Image') {
      steps {
        script {
          // Push the image with both the BUILD_ID tag and the 'latest' tag
          docker.withRegistry("https://index.docker.io/v1/", "${DOCKER_CREDENTIALS_ID}") {
            def image = docker.image("${DOCKER_REPO}:${env.BUILD_ID}")
            image.push("${env.BUILD_ID}") // Push with BUILD_ID tag
            image.push('latest') // Optionally, also push with 'latest' tag
          }
        }
      }
    }
    stage('Deploy Container') {
      steps {
        script {
          sh """
          #!/bin/bash
          docker pull ${DOCKER_REPO}:${env.BUILD_ID}
          docker stop java-app-docker || true
          docker rm java-app-docker || true
          docker run -d --name java-app-docker -p 8881:8881 ${DOCKER_REPO}:${env.BUILD_ID}
          """
        }
      }
    }
    stage('Print Docker Logs') {
      steps {
        script {
          sh """
          #!/bin/bash
          echo "Fetching logs for java-app-docker..."
          docker logs java-app-docker
          """
        }
      }
    }
  }
  post {
    always {
      cleanWs()
    }
  }
}

```

Dockerfile

```
GNU nano 6.2 Dockerfile
FROM eclipse-temurin:11-jdk

# Set the working directory in the container
WORKDIR /app

# Copy the Java source code into the container
COPY src/main/java/HelloWorld.java /app/

# Compile the Java source code
RUN javac HelloWorld.java

# Specify the command to run the application and keep the container running
CMD ["sh", "-c", "java HelloWorld & tail -f /dev/null"]
```

Docker Hub Repository

The screenshot shows the Docker Hub interface for the repository `poonam02/java-app-docker`. The page is divided into several sections:

- Header:** Includes the Docker Hub logo, navigation links (Explore, Repositories, Organizations), a search bar, and user profile icons.
- Breadcrumbs:** `poonam02 / Repositories / java-app-docker / General`.
- General Tab:** The active tab, showing repository details.
 - Repository Name:** `poonam02/java-app-docker` (Updated 16 minutes ago).
 - Description:** "This repository does not have a description" (INCOMPLETE).
 - Category:** "This repository does not have a category" (INCOMPLETE).
 - Docker commands:** A box showing the command `docker push poonam02/java-app-docker:tagname` with a "Public View" button.
- Tags Section:** A table listing the repository's tags.

Tag	OS	Type	Pulled	Pushed
latest	linux	Image	44 minutes ago	16 minutes ago
11	linux	Image	44 minutes ago	17 minutes ago
- Automated Builds:** A section explaining how to connect GitHub or Bitbucket for automated builds, with a link to "Read more about automated builds".

Then build the job in Jenkins.

Console Output

The screenshot shows the Jenkins web interface for build #11 of the CICDJavaApp. The left sidebar contains navigation links: Status, Changes, Console Output (selected), View as plain text, Edit Build Information, Delete build '#11', Timings, Git Build Data, Pipeline Overview, Pipeline Console, Restart from Stage, and Replay. The main area displays the 'Console Output' with a green checkmark icon. The output text shows the pipeline starting, cloning a repository, and running a script that prints Docker logs. The build concludes with a successful status.

```
Started by user Poonam Mahesh Bhavsar
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/CICDJavaApp
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Clone Repository)
[Pipeline] git
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/02Poonam/CICDJavaAppDocker.git
> git init /var/lib/jenkins/workspace/CICDJavaApp # timeout=10
Fetching upstream changes from https://github.com/02Poonam/CICDJavaAppDocker.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/02Poonam/CICDJavaAppDocker.git
[Pipeline] stage
[Pipeline] { (Print Docker Logs)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ echo Fetching logs for java-app-docker...
Fetching logs for java-app-docker...
+ docker logs java-app-docker
Hello, World from docker!
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```