Project 01

Problem Statement:

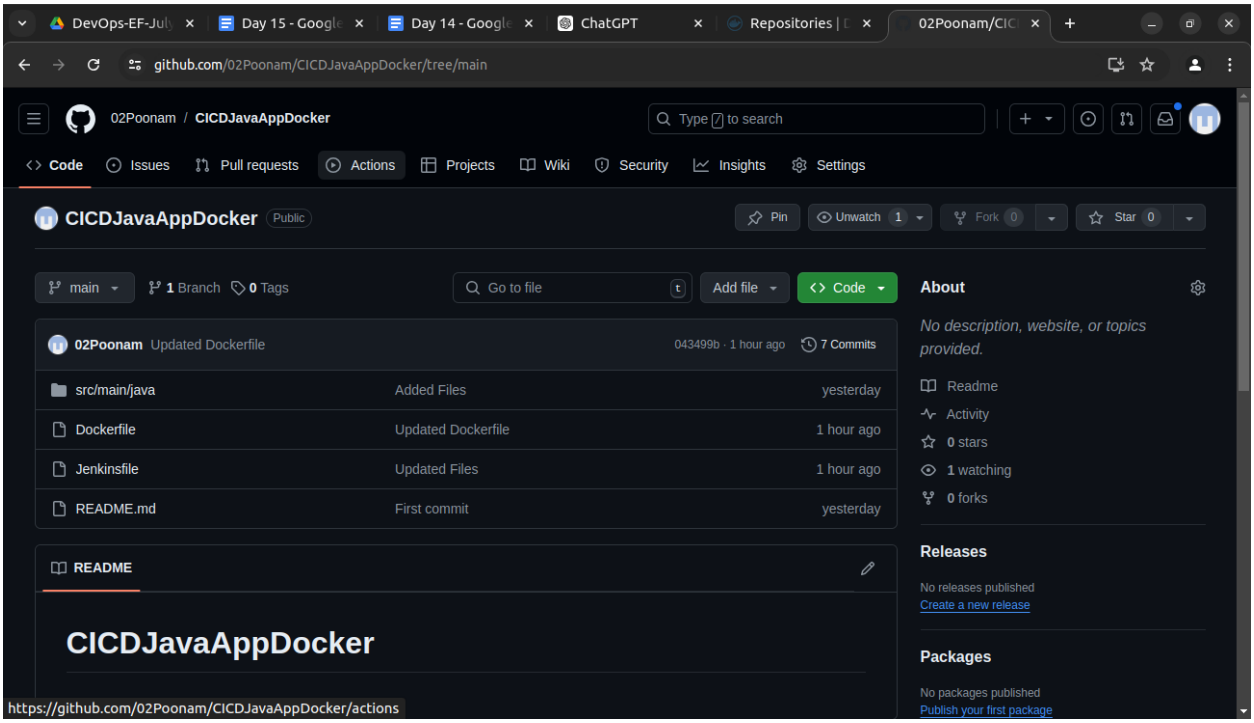
You are tasked with setting up a CI/CD pipeline using Jenkins to streamline the deployment process of a simple Java application. The pipeline should accomplish the following tasks:

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.

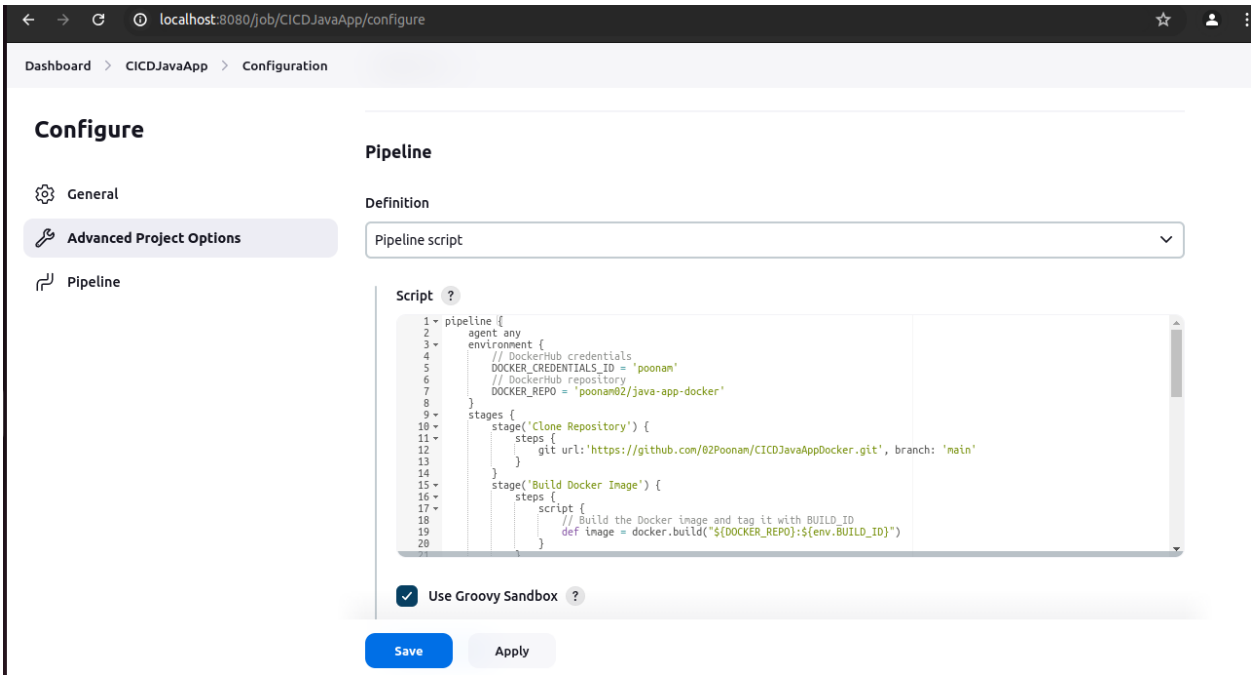
Deliverables:

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

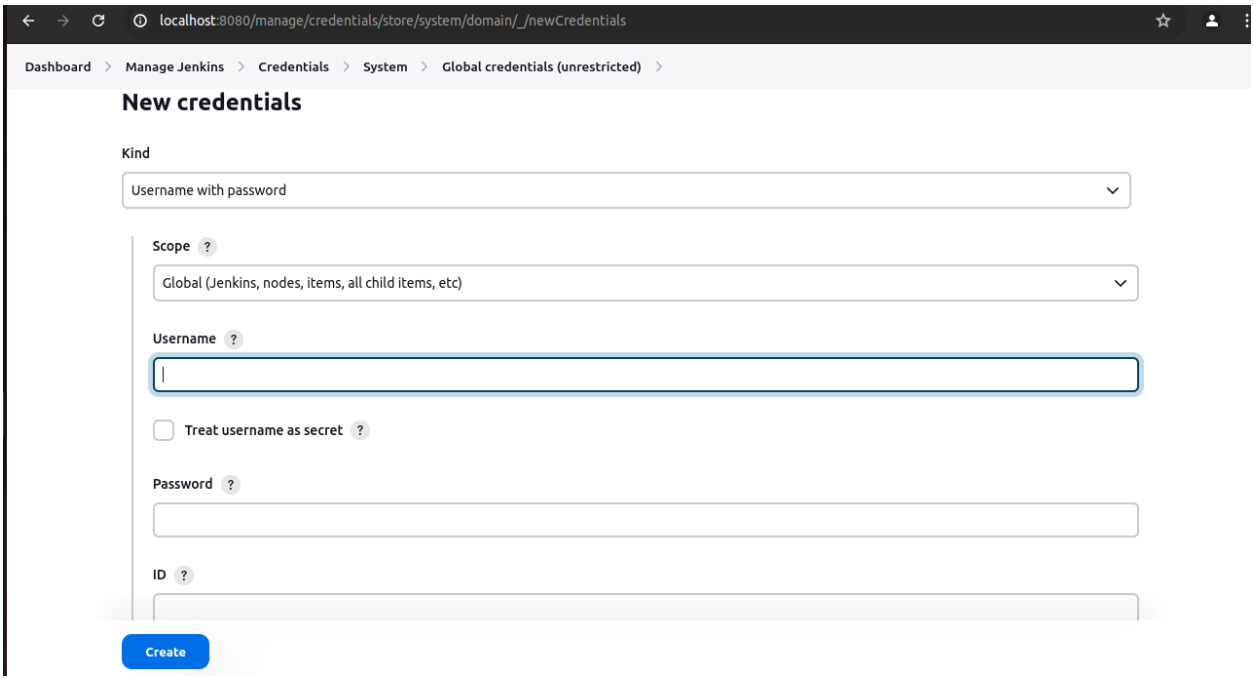


Create a new Jenkins pipeline and configure it as follows



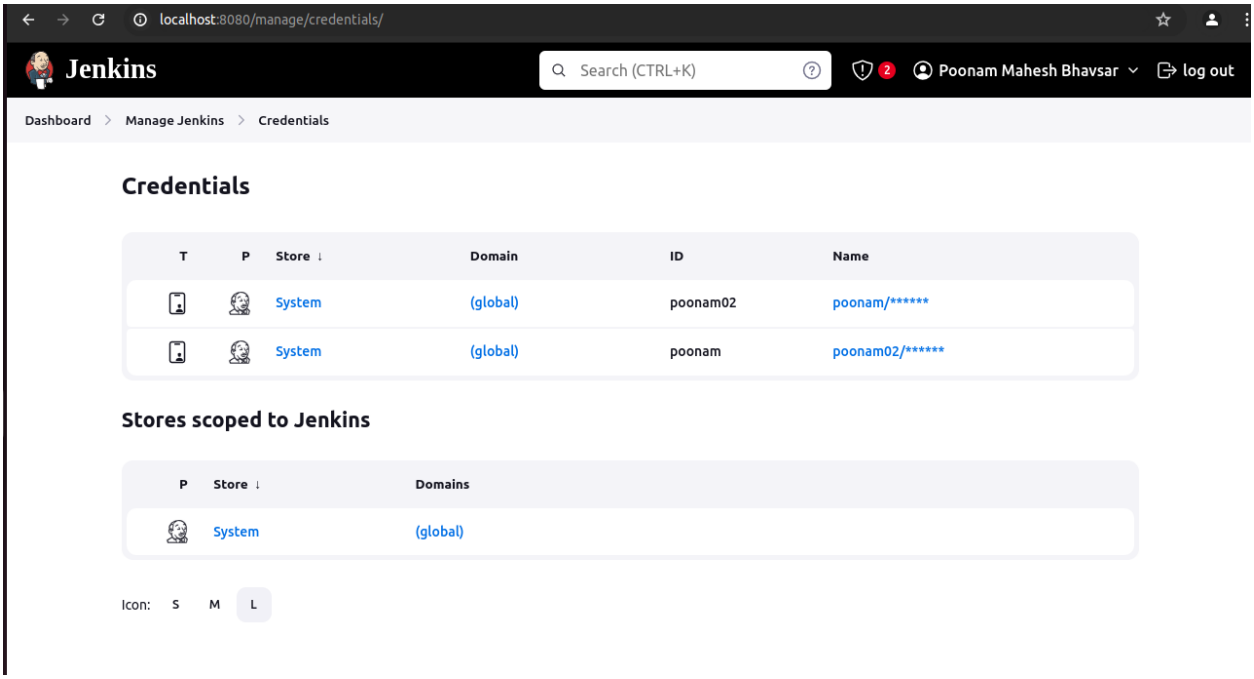
Also set up the local Jenkins Environment by adding the required plugins and credentials of Dockerhub

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



Add your Username, password and id of Dockerhub

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



Jenkins Pipeline Script

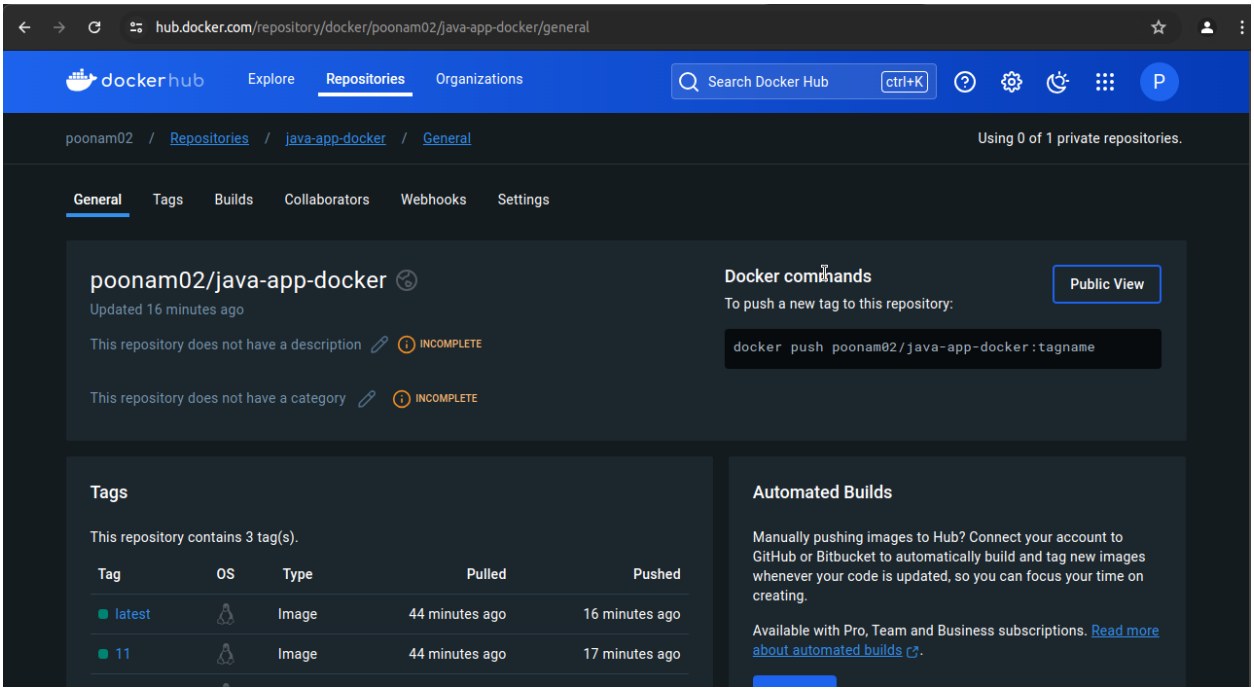
Jenkinsfile (pipeline script) that clones the GitHub repository, builds the Docker image, pushes the Docker image to Dockerhub and deploys the container using the pushed image

```
GNU nano 6.2
pipeline {
    agent any
    environment {
        // DockerHub credentials
        DOCKER_CREDENTIALS_ID = 'poonan'
        // DockerHub repository
        DOCKER_REPO = 'poonam02/java-app-docker'
    }
    stages {
        stage('Clone Repository') {
            steps {
                git url:'https://github.com/02Poonan/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 8081:8081 ${DOCKER_REPO}:${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



Then build the job in Jenkins.

Console Output