**✅ TASK 2: Simple Jenkins Pipeline for CI/CD**

**🎯 Objective**

To automate the process of building, testing, and deploying a simple application using a Jenkins pipeline with Docker.

**🛠️ Tools Required**

* Jenkins (Installed locally or on a cloud server like AWS)
* Docker
* GitHub or Git repository with your app

**📂 Project Structure Example**

You can use a basic app (Node.js, Python, or HTML page). Here’s a sample structure:

perl

CopyEdit

my-app/

│

├── Dockerfile

├── index.html (or your app files)

└── Jenkinsfile

**📄 Step-by-Step Documentation**

**Step 1️⃣: Install Jenkins and Docker**

* **Jenkins Installation**: Download and install Jenkins from jenkins.io.
* **Install Required Plugins**: Git, Docker Pipeline.
* **Install Docker**: Make sure Docker is running on the same server as Jenkins.

📝 *Docker is used to build and run containers for your app.*

**Step 2️⃣: Create Jenkinsfile**

Place this file in the **root of your Git project**:

groovy

CopyEdit

pipeline {

agent any

environment {

IMAGE\_NAME = 'my-simple-app'

DOCKERHUB\_USERNAME = 'your-dockerhub-username'

}

stages {

stage('Checkout') {

steps {

echo 'Cloning the repository...'

git 'https://github.com/your-username/your-repo.git'

}

}

stage('Build') {

steps {

echo 'Building Docker image...'

sh 'docker build -t $IMAGE\_NAME .'

}

}

stage('Test') {

steps {

echo 'Running tests...'

// If test scripts exist

sh 'echo "No tests available. Skipping..."'

}

}

stage('Push to Docker Hub') {

steps {

echo 'Pushing image to DockerHub...'

withCredentials([usernamePassword(credentialsId: 'dockerhub-creds', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]) {

sh 'echo $PASSWORD | docker login -u $USERNAME --password-stdin'

sh 'docker tag $IMAGE\_NAME $DOCKERHUB\_USERNAME/$IMAGE\_NAME'

sh 'docker push $DOCKERHUB\_USERNAME/$IMAGE\_NAME'

}

}

}

stage('Deploy') {

steps {

echo 'Deploying container...'

sh 'docker run -d -p 8080:80 $DOCKERHUB\_USERNAME/$IMAGE\_NAME'

}

}

}

}

**📖 Explanation of Each Step**

**✅ pipeline { agent any }**

* Tells Jenkins to run the pipeline on any available executor (node).

**✅ environment { }**

* Stores values like image name and DockerHub username so we can reuse them easily.

**✅ stage('Checkout')**

* Clones the GitHub repository to get the latest code.

**✅ stage('Build')**

* Builds a Docker image using the Dockerfile.

**✅ stage('Test')**

* A placeholder to run your app’s tests (you can add actual test commands here).

**✅ stage('Push to Docker Hub')**

* Pushes the built image to Docker Hub (you must create credentials in Jenkins named dockerhub-creds).

**✅ stage('Deploy')**

* Runs the Docker container locally (or you can deploy to Kubernetes, AWS, etc.).

**⚙️ Step 3️⃣: Configure Jenkins to Trigger on Code Push**

1. Go to **Jenkins > New Item > Pipeline** → Name it.
2. In **Pipeline Configuration**:
   * Choose **"Pipeline script from SCM"**.
   * SCM: Git
   * Repo URL: https://github.com/your-username/your-repo.git
   * Branch: main (or whatever branch you use)
3. Click **Save**.

**🔔 Step 4️⃣: Setup Webhook in GitHub**

* Go to your GitHub repo > **Settings > Webhooks > Add Webhook**
  + Payload URL: http://your-jenkins-ip/github-webhook/
  + Content type: application/json
  + Events: Push Events
  + Save.

📝 This makes Jenkins trigger automatically when you push code.

**🧪 Step 5️⃣: Test the CI/CD Pipeline**

1. Push a code change to your GitHub repository.
2. Jenkins will:
   * Clone your repo
   * Build Docker image
   * (Optionally) Run tests
   * Push the image to DockerHub
   * Run the container

Check the **Jenkins Dashboard > Build History** for logs and stages.

**🪄 Tips**

* You must create a DockerHub account and link credentials in Jenkins.
* Ensure Jenkins has permission to run Docker (Jenkins user should be added to the docker group).
* Use port 8080 for Jenkins and 80 or 8081 for your app.

**📦 Sample Dockerfile**

Dockerfile

CopyEdit

# Simple static HTML app

FROM nginx:alpine

COPY . /usr/share/nginx/html

EXPOSE 80

**📌 1️⃣ What is Jenkins, and how is it used in CI/CD?**

**Jenkins** is an open-source automation server used to automate parts of the software development process, especially for **Continuous Integration (CI)** and **Continuous Deployment (CD)**.

**How it's used in CI/CD:**

* **Continuous Integration:** Jenkins automatically builds and tests code whenever developers push changes to the version control system (like Git).
* **Continuous Deployment/Delivery:** It can automatically deploy applications to servers, containers, or cloud services once code passes testing, making software releases faster and more reliable.

**Key Benefits:**

* Automates repetitive tasks
* Supports integration with hundreds of plugins
* Provides pipeline visualization
* Enables early bug detection via automated builds and tests

**📌 2️⃣ What is a Jenkinsfile?**

A **Jenkinsfile** is a text file that defines a **Jenkins pipeline** and contains the stages and steps to build, test, and deploy an application.

**Key Points:**

* Written in **Groovy-based DSL** (Domain Specific Language)
* Stored in the project’s source code repository
* Enables **Pipeline as Code** — version-controlling the CI/CD process
* Can be either **Declarative** (simpler and structured) or **Scripted** (more flexible and detailed)

**Example (Declarative Jenkinsfile):**

groovy

CopyEdit

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building the app...'

}

}

}

}

**📌 3️⃣ How do you create and configure Jenkins pipelines?**

**Steps to create and configure a Jenkins pipeline:**

1. **Install Jenkins** and necessary plugins (e.g., Pipeline, Git).
2. From Jenkins dashboard → Click **New Item**.
3. Enter job name → Select **Pipeline** → Click OK.
4. In the job configuration:
   * Choose **Pipeline script** (or **Pipeline script from SCM** for Jenkinsfile in repo)
   * Add your pipeline script or point it to your Jenkinsfile.
5. (Optional) Set triggers like **Poll SCM** or **GitHub webhook** for automated builds.
6. Save and run the job.

**📌 4️⃣ What are some common stages in a Jenkins pipeline?**

Typical stages in a CI/CD pipeline:

* **Checkout**: Clone or pull the latest code from the repository.
* **Build**: Compile the application or build a Docker image.
* **Test**: Run unit tests, integration tests, or security scans.
* **Deploy**: Deploy the application to staging, production, or a Docker container.
* **Cleanup**: Remove temporary files, containers, or other resources.
* **Notification**: Send build status notifications via email, Slack, or other services.

**📌 5️⃣ What is the difference between a Declarative and Scripted Jenkins pipeline?**

| **🔍 Feature** | **Declarative Pipeline** | **Scripted Pipeline** |
| --- | --- | --- |
| **Syntax Style** | High-level, structured, and predefined syntax | Groovy-based scripting syntax |
| **Readability** | Easier for beginners and standardized | More flexible but harder to read |
| **Flexibility** | Limited to predefined blocks and statements | Fully programmable and customizable using Groovy |
| **Error Handling** | Built-in post block for notifications and cleanup | Manual try-catch blocks needed |
| **Recommended for** | Simple to moderately complex pipelines | Highly complex or dynamic pipelines |

**Example — Declarative**

groovy

CopyEdit

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building...'

}

}

}

}

**Example — Scripted**

groovy

CopyEdit

node {

stage('Build') {

echo 'Building...'

}

}