## Breakdown of the pipeline:

#### 1. Checkout Code

o checkout scm: This ensures that the code is checked out from your Git repository, as configured in Jenkins.

#### 2. Build with Maven

o The command mvn clean package will clean any previous builds and create a new package (usually a .jar or .war file) from the source code.

### 3. Run Unit Tests

o mvn test runs the unit tests as part of the build process.

## 4. Dockerize Application

This stage builds a Docker image using the Dockerfile present in the repository and tags it with the DOCKER IMAGE environment variable.

### 5. Push Docker Image to Registry

 This stage logs into Docker Hub using stored credentials and pushes the Docker image to your Docker Hub repository.

## 6. **Deploy Locally**

o Stops and removes any existing banking-app container, and runs a new container using the newly built Docker image.

## 7. Post-Deployment Verification

o This checks the health of the deployed app by querying the health endpoint (/health on port 8080). If the health check fails, it triggers a failure in the pipeline.

This pipeline performs the following steps:

- **Checkout**: Pulls the latest code from your GitHub repository.
- Build: Builds your Spring Boot application using Maven (mvn clean package).
- **Build Docker Image**: Creates a Docker image for your app.
- **Push Docker Image**: Pushes the Docker image to your Docker registry (e.g., Docker Hub or a private registry).
- **Deploy to Server**: SSHs into your server and deploys the Docker container.

```
git init

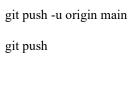
ls -al

git status

git add --all

git commit -m "initial commit"

git remote add origin https://github.com/supun-chandana/Banking-App.git
```



#### **Account Controller**

Spring Boot REST Controller named AccountController for managing banking account operations.

#### Account

defines the Account entity class for a bank application, mapping it to a database table using JPA annotations. Java Persistence API (JPA) is a specification in Java that provides a standard way to map Java objects to relational database tables, enabling developers to interact with databases using object-oriented programming principles rather than SQL queries.

## **Account Repository Interface**

The AccountRepository interface is a Spring Data JPA repository for managing Account entities.

Extends JpaRepository:

• Inherits CRUD operations

#### **Account Service Interface**

The Account Service interface defines the core operations for managing bank accounts in the application.

### AccountServiceImpl

The AccountServiceImpl class implements the AccountService interface and provides the business logic for managing bank accounts. It uses AccountRepository to interact with the database.

## **Menu Application**

# BankApplication

The BankApplication class serves as the entry point for the Spring Boot application and integrates a user interface for managing bank-related functionalities.