REPORT

MADE BY :

YASHIKA NAGDEV

23FE10CSE00213

SECTION A

DEVOPS

TOPIC: Jenkins Job Configuration and Multibranch Pipeline Implementation

1. Aim / Objective

The objective of this lab is to understand and implement Continuous Integration (CI) using Jenkins by configuring different types of Jenkins jobs, integrating them with GitHub, and automating the build and test process using Maven and pipelines.

2. Tools & Technologies Used

- **Jenkins (CI Server)**
- **Git & GitHub**
- **Apache Maven**
- **Java JDK 17**
- **Jenkins Pipeline & Multibranch Pipeline**
- **Windows OS**

3. Theory

3.1 Continuous Integration (CI)

Continuous Integration is a software development practice where developers frequently integrate their code into a shared repository. Each integration is verified by an automated build and test process, allowing teams to detect errors early.

Benefits of CI:

- Early bug detection
- Faster development cycles
- Improved code quality
- Automated testing and builds

3.2 Jenkins

Jenkins is an open-source automation server used to implement CI/CD pipelines. It supports automation of building, testing, and deploying software projects.

3.3 Jenkins Jobs

a) Freestyle Job

A Freestyle job is a simple Jenkins job that allows users to configure build steps, triggers, and post-build actions through the Jenkins UI.

b) Multibranch Pipeline

A Multibranch Pipeline automatically detects branches in a Git repository and executes a pipeline based on a Jenkinsfile present in each branch.

Task 4: Create Different Types of Jenkins Jobs

4.1 Freestyle Project Configuration

Source Code Management

- **Connected Jenkins to GitHub repository:**

  https://github.com/Yashika-web16/CILabProject.git

Build Triggers

Poll SCM: Configured Jenkins to check for changes periodically

GitHub Webhook (configured conceptually)

Build Steps

Execute Windows Batch Command

Run Maven build commands

Post-build Actions

Archive artifacts

Publish JUnit test results

Email notification (configured)

4.2 Multibranch Pipeline Configuration

- Jenkins automatically discovered the following branches:
  - main
  - feature/login
  - release/v1.0

- Each branch followed a **different CI strategy**:
  - **Main branch:** Full build and test pipeline
  - **Feature branch:** Test-only execution
  - **Release branch:** Test and security scan simulation

5. Jenkinsfile (Pipeline Script)

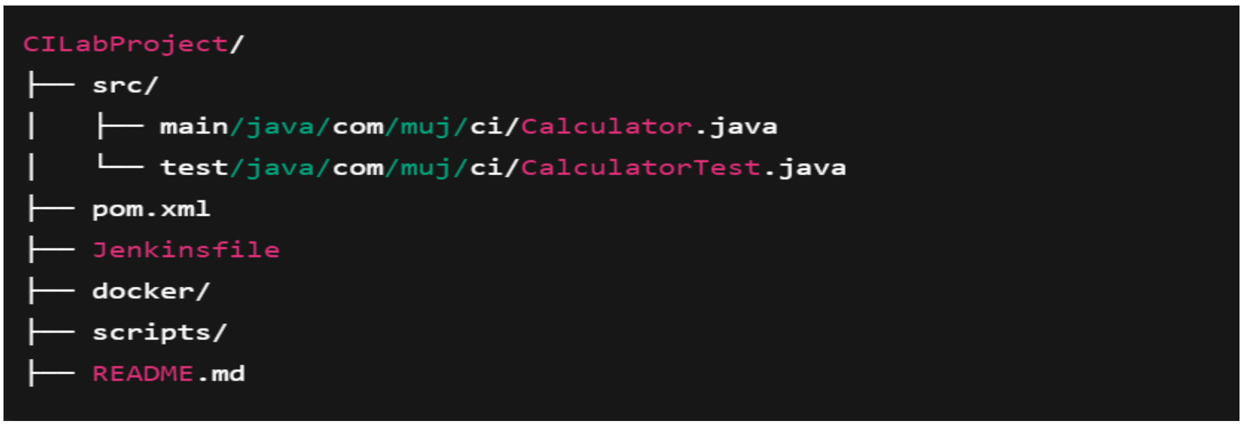
```
pipeline {
    agent any
  stages {
       stage('Build & Test - Main') {
          when { branch 'main' }
          steps {
             echo 'Running full CI pipeline for MAIN branch'
             bat
'"C:\\ProgramData\\Jenkins\\.jenkins\\tools\\hudson.tasks.Maven_MavenInstallation\\Mav
en-3\\bin\\mvn.cmd" clean package'
          }     }
stage('Test Only - Feature Branch') {
          when { branch 'feature/login' }
          steps {
             echo 'Running tests only for FEATURE branch'
             bat
'"C:\\ProgramData\\Jenkins\\.jenkins\\tools\\hudson.tasks.Maven_MavenInstallation\\Mav
en-3\\bin\\mvn.cmd" test'
          }
       } stage('Test & Security Scan - Release Branch') {
          when { branch 'release/v1.0' }
          steps {
             echo 'Running tests and security scan for RELEASE branch'
   bat
'"C:\\ProgramData\\Jenkins\\.jenkins\\tools\\hudson.tasks.Maven_MavenInstallation\\Mav
en-3\\bin\\mvn.cmd" test'

             echo 'Security scan simulated'  }  } }

  post {

       success { echo 'Build successful' }

       failure { echo 'Build failed' }

    }}
```

6. Plugin Management (Task 5)

Installed Jenkins Plugins

- Git
- GitHub
- Pipeline
- Multibranch Pipeline
- Maven Integration
- JUnit
- Workspace Cleanup
- Email Extension

7. Repository Structure



```
CILabProject/
├── src/
|   ├── main/java/com/muj/ci/Calculator.java
|   └── test/java/com/muj/ci/CalculatorTest.java
├── pom.xml
├── Jenkinsfile
├── docker/
├── scripts/
├── README.md
```

8. Results

- Jenkins successfully detected all branches automatically.
- Multibranch pipeline executed different stages based on branch type.
- CI workflow was automated using Jenkins and Maven.
- Builds and tests were executed without manual intervention.

9. Troubleshooting

Issue	**Solution**
Maven not recognized	Used absolute path to mvn.cmd
Jenkinsfile not detected	Placed Jenkinsfile at repository root
Old failures showing	Rebuilt pipeline jobs

10. Conclusion

This experiment successfully demonstrated the implementation of Continuous Integration using Jenkins. Freestyle and Multibranch pipeline jobs were configured to automate the build and test processes. Branch-based CI strategies improved code quality and automation efficiency.