# SOFTWARE ENGINEERING LAB

## EXERCISE – 6

## TOPIC – 3

# BUILDING THE CI/CD SCRIPTED PIPELINE USING JENKINS FOR MAVEN JAVA PROJECT WITH POLL SCM

**Note: At every step take screenshots and save in a document**

## 1. Introduction to Jenkins for CI/CD Pipelines

Jenkins is an open-source automation server that helps in building, testing, and deploying code through continuous integration (CI) and continuous delivery (CD). For software development teams, Jenkins ensures that changes in code are automatically built and tested, facilitating a streamlined workflow.

## 2. Creating a New Pipeline in Jenkins

To begin, navigate to the Jenkins dashboard:

- **Step 1:** Click on "New Item".
- **Step 2:** Enter a project name.
- **Step 3:** Select the third option, "Pipeline", and click "OK".

**Why choose "Pipeline"?**

Jenkins Pipeline is a suite of plugins that supports implementing and integrating continuous delivery pipelines. It provides a robust and flexible framework for modelling complex workflows.

## 3. Configuring Build Triggers in Jenkins

Scroll to the "Build Triggers" section:

- **Select "Build Periodically"**: This option allows Jenkins to trigger builds at specified intervals, even without any code changes.
- **Why "Build Periodically"?**

  Automating builds at regular intervals helps ensure that scheduled tasks such as code compilation, tests, or deployments occur consistently. This keeps the project up-to-date and highlights issues promptly.

## 4. Understanding the Scheduling Syntax

The scheduling syntax uses a crontab format:

- Example: ****H/15 * * * ***** - Triggers a build every 15 minutes.
- Example: ****H * * 3 ***** - Triggers a build on the first day of every third month.

**Explanation of the Script Syntax**:

- The "H" symbol represents a hashed value to distribute load evenly.
- Numbers separated by spaces represent: minute, hour, day of the month, month, and day of the week.

## 5. Adding the Pipeline Script

Navigate to the "Pipeline" section and paste the following script:

```
pipeline {
    agent any
    tools {
        maven 'MAVEN_HOME'
    }
    stages {
        stage('git repo & clean') {
            steps {
```

```
                bat "rmdir  /s /q SampleMavenJavaProject"
                bat "git clone https://github.com/budarajumadhurika/SampleMavenJavaProject.git"
                bat "mvn clean -f SampleMavenJavaProject"
            }
        }
        stage('install') {
            steps {
                bat "mvn install -f SampleMavenJavaProject"
            }
        }
        stage('test') {
            steps {
                bat "mvn test -f SampleMavenJavaProject"
            }
        }
        stage('package') {
            steps {
                bat "mvn package -f SampleMavenJavaProject"
            }
        }
    }
}
```

**Explanation of the Script**:

- **'pipeline' block**: Declares the start of the Jenkins pipeline.
- **'agent any'**: Specifies that Jenkins can run the pipeline on any available agent.
- **'tools' block**: Ensures the specified Maven version is available.
- **Stages**:
    - **Git Repo & Clean Stage**: Deletes any existing project directory using `rmdir` and clones the Git repository. Runs `mvn clean` to clean the project.

- **Install Stage**: Runs `mvn install` to compile, test, and install the package into the local repository.
- **Test Stage**: Executes `mvn test` to run unit tests and validate code functionality.
- **Package Stage**: Runs `mvn package` to bundle the compiled code into a deployable format (e.g., a JAR or WAR file).

## 6. Applying Changes and Running the Pipeline

- **Click "Apply" and "Save"**: Save your pipeline configuration.
- **Click "Build Now"**: Initiates the pipeline run.
- **Successful Build Verification**: Confirm a successful build by checking the console output for completion messages.

## 7. Poll SCM for Automatic Triggers

Polling the Source Code Management (SCM) ensures that the pipeline triggers when new changes are committed:

- **Enable Poll SCM** under "Build Triggers".
- **Why use Poll SCM?**

  It automates the process of detecting changes in the repository, making builds more efficient by triggering them only when updates are made.

- **Example Schedule:** `H/1 * * * *` triggers a build check every minute.

## Conclusion:

Using Jenkins with Poll SCM and pipelines streamlines CI/CD processes, automates builds, and enhances overall project efficiency. This approach minimizes manual intervention, supports consistent testing, and ensures prompt feedback.