Experiment 7: Implementing CI/CD with Java and Jenkins Module Name: Implementation of CICD with Java and open source stack Configure the Jenkins tool with the required paths, path variables, users and pipeline views.

Step 1: Install and Start Jenkins

1. **Install Jenkins:**
   - If you haven't installed Jenkins yet, install it.
     - Go with **"Run service as Local System"** for a smoother and simpler setup.
   - After installation, start Jenkins and access it via your web browser at http://localhost:8080.
2. **Unlock Jenkins:**
   - Enter the initial admin password found in the secrets/initialAdminPassword file as prompted.
   - Complete the setup by creating your admin user.

Step 2: Install Necessary Plug-ins

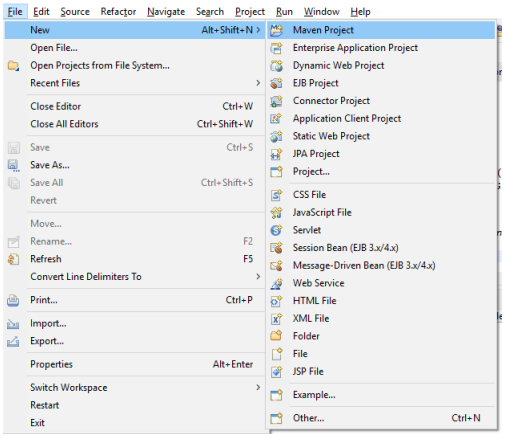
1. **Access Jenkins:**
   - Go to http://localhost:8080 and log in with your admin credentials.
2. **Install Plug-ins:**
   - Navigate to "Manage Jenkins" > "Manage Plug-ins."
   - Under the "Available" tab, search for and install the following plug-ins:
     - **Git plug-in**: Provides Git integration.
     - **Maven Integration plug-in**: Integrates Maven with Jenkins.
     - **Pipeline plug-in**: Enables Jenkins Pipeline as code.
     - **Stage View Plug-in:** Provides a UI to visualize the pipeline stages and their status (success/failure).
     - **SCM Step Plug-in:** Provides support for using source control management (SCM) systems like Git within your pipeline.
     - **Build Step Plug-in:** Allows you to add build steps in the pipeline script.
     - **GitHub Plug-in:** Adds support for integrating GitHub with Jenkins pipelines.

Step 3: Configure Global Tool Settings

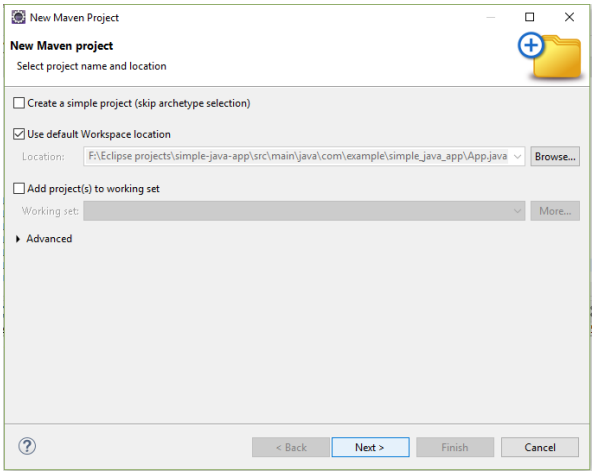
1. **Configure Java:**
   - Go to "Manage Jenkins" > "Global Tool Configuration."
   - Scroll to the "JDK" section and click "Add JDK."
   - Name it JDK8 and uncheck "Install automatically."
   - Set JAVA_HOME to the path where JDK is installed on your machine (e.g., C:\Program Files\Java\jdk1.8.0_291).
2. **Configure Maven:**
   - In the same "Global Tool Configuration" page, scroll to the "Maven" section and click "Add Maven."
   - Name it Maven3 and uncheck "Install automatically."
   - Set MAVEN_HOME to the path where Maven is installed on your machine (e.g., C:\Program Files\Apache\maven-3.6.3).

Step 4: Create a Simple Java Application

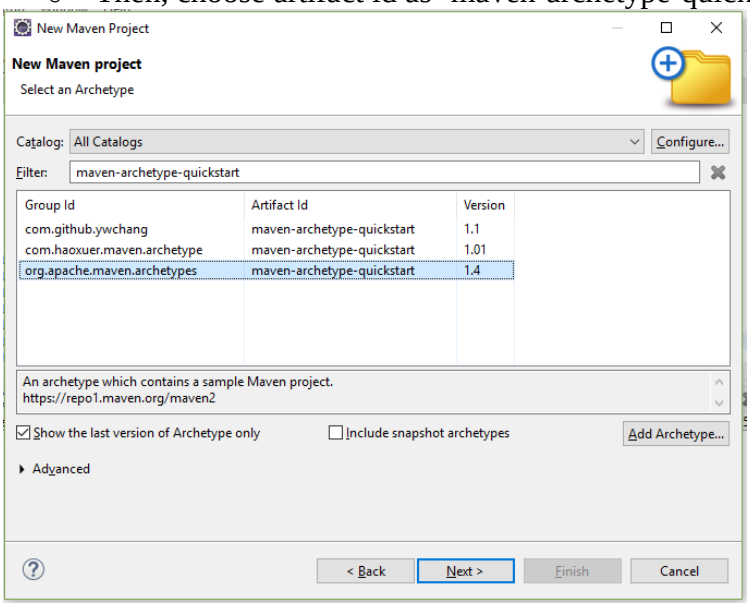
1. **Create a Maven Project:**
   - Select File →New →Maven Project



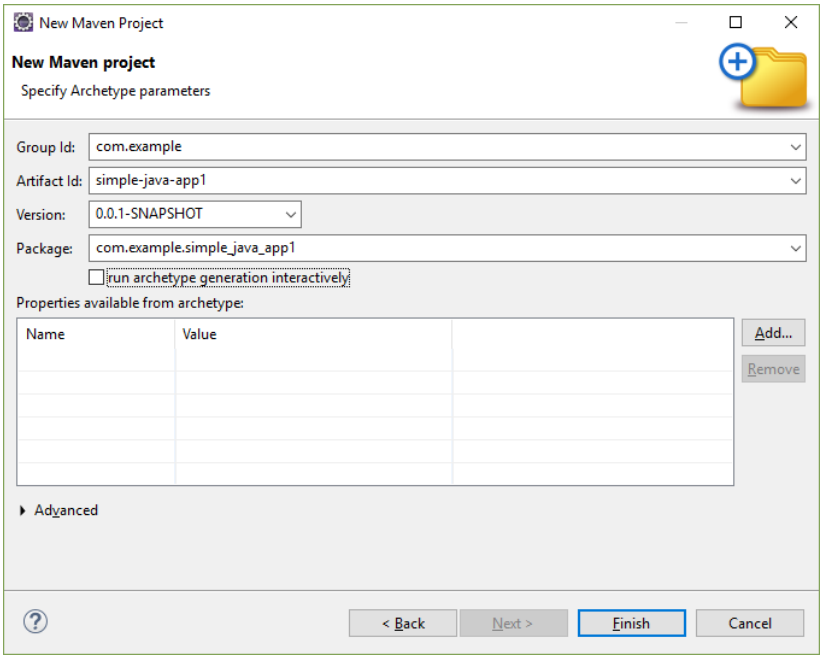
   - Select ‘use default workspace location’ and click ‘Next’.



   - Then, choose artifact id as ‘maven-archetype-quickstart’ and click ‘Next’.



- Give values for 'Group id' as 'com.example', 'Artifact id' as 'simple-java-app', 'version' as '0.0.1-SNAPSHOT' and 'package' as 'com.example.simple_java_app'.
- Deselect 'run archetype generation interactively' and click 'finish'.



2. **Update the Java Source Code:**
   - Replace the contents of src/main/java/com/example/ simple_java_app/App.java with the following:

```
package com.example.simple_java_app;

public class App {
    public static String getGreeting() {
        return "Hello, CI/CD World!";
    }
    public static void main(String[] args) {
        System.out.println(getGreeting());
    }
}
```

   - Replace the contents of src/test/java/com/example/ simple_java_app/AppTest.java with:

```
package com.example.simple_java_app;

import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class AppTest {
    @Test
```

```
    public void testGetGreeting() {
        assertEquals("Hello, CI/CD World!", App.getGreeting());
    }
}
```

3. **Update the pom.xml File:**
   - Update pom.xml to include the latest JUnit and Maven Surefire Plugin.

```
<project>
  ...
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.13.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M5</version>
      </plugin>
    </plugins>
  </build>
  ...
</project>
```

4. **Initialize Git Repository:**
   - Initialize a Git repository and commit your code.

```
git init
git add .
git commit -m "Initial commit"
```

5. **Push to GitHub:**
   - Create a repository on GitHub (or another Git hosting service).
   - Push your code to GitHub:

```
git remote add origin https://github.com/yourusername/simple-java-app.git
git push -u origin master
```

Step 5: Create a Jenkins Pipeline

1. **Create a New Pipeline Job:**
   - On the Jenkins dashboard, click "New Item."
   - Enter a name (e.g., simple-java-app-pipeline), select "Pipeline," and click "OK."
2. **Configure the Pipeline:**
   - Scroll down to the "Pipeline" section.
   - Select "Pipeline script" and enter the following script:

```
pipeline {
    agent any

    tools {
        jdk 'JDK8'
        maven 'Maven3'
    }

    stages {
        stage('Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/YourUserName/simple-java-app'
            }
        }
        stage('Build') {
            steps {
                bat 'mvn clean compile'
            }
        }
        stage('Test') {
            steps {
                bat 'mvn test'
            }
            post {
                always {
                    junit '**/target/surefire-reports/TEST-*.xml'
                }
            }
        }
        stage('Package') {
            steps {
                bat 'mvn package'
            }
            post {
                success {
                    archiveArtifacts 'target/*.jar'
                }
                failure {
                    echo 'Package stage failed. Check the logs for more information.'
                }
            }
```

```
        }
    stage('Run Application') {
        steps {
            bat 'java -jar target/simple-java-app-1.0-SNAPSHOT.jar'
        }
    }
    }
    }
```

Replace 'https://github.com/yourusername/simple-java-app.git' with your actual GitHub repository URL.

3. **Save and Execute the Pipeline:**
    - Click "Save" to save the pipeline configuration.
    - On the project page, click "Build Now" to execute the pipeline.

Step 6: Monitor the Pipeline Execution

- **Build Process:**
    - Jenkins will check out the code from your repository.
    - It will compile the Java application, run tests, and package the application.
- **View Results:**
    - You can monitor the progress on the Jenkins dashboard.
    - After the build, click on the build number to see the build details, including **console output**: to understand each step of your build process and **test results**: to ensure all your tests passed.
- **Artifacts:**
    - If the build and tests succeed, Jenkins will archive the generated .jar file as an artifact.

Step 7: Verify the JAR

- Download the JAR file from the build artifacts.
- You can try running it locally to ensure it works as expected:

```
java -jar your-artifact-name.jar
```

Output: "Hello, CI/CD World!"