Jenkins CI Pipeline using Git and Maven (Complete Step-by-Step Guide)

STEP 1: Create a Simple Maven Project (Git Side)

Project Name: simple-maven-app

Folder Structure (MANDATORY Maven Standard):

```
simple-maven-app/
 ├── pom.xml
 └── src/
    ├── main/java/com/example/App.java
    └── test/java/com/example/AppTest.java
```

STEP 2: Create App.java (Main Application)

File Path:
src/main/java/com/example/App.java

Code:

```
package com.example;

public class App {
   public int add(int a, int b) {
      return a + b;
   }
}
```

Explanation:
- Defines business logic
- add() method returns sum of two numbers

STEP 3: Create AppTest.java (JUnit Test Case)

File Path:
src/test/java/com/example/AppTest.java

Code:

```
package com.example;

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

public class AppTest {

   @Test
   public void testAdd() {
      App app = new App();
      assertEquals(5, app.add(2, 3));
   }
}
```

Explanation:
- @Test marks test method
- assertEquals compares expected and actual value
- Test failure causes build failure

STEP 4: Create pom.xml (FULL XML WITH DEPENDENCIES)

File Path:
simple-maven-app/pom.xml

Code:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
       http://maven.apache.org/xsd/maven-4.0.0.xsd">

   <modelVersion>4.0.0</modelVersion>

   <groupId>com.example</groupId>
   <artifactId>simple-maven-app</artifactId>
   <version>1.0-SNAPSHOT</version>

   <dependencies>
      <dependency>
         <groupId>junit</groupId>
```

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

</project>
```

Explanation:
- pom.xml defines project configuration
- JUnit dependency enables unit testing

STEP 5: Test Project Locally

Command:
mvn clean test

Expected Output:
BUILD SUCCESS

STEP 6: Push Project to GitHub

Commands:

```
git init
git add .
git commit -m "Initial Maven project with JUnit test"
git branch -M main
git remote add origin <your-repository-url>
git push -u origin main
```

STEP 7: Install Maven on Windows

1. Download apache-maven-3.9.12-bin.zip
2. Extract to:
   C:\Program Files\Apache\apache-maven-3.9.12
3. Set System Variable:
   MAVEN_HOME = C:\Program Files\Apache\apache-maven-3.9.12

4. Add to PATH:
   %MAVEN_HOME%\bin
5. Verify:
   mvn -version

STEP 8: Configure Maven in Jenkins

Path:
Manage Jenkins → Tools → Maven Installations

Configuration:
Name: M3
MAVEN_HOME: C:\Program Files\Apache\apache-maven-3.9.12
Install Automatically: UNCHECKED

STEP 9: Jenkins Pipeline Script (Declarative Pipeline)

```
pipeline {
    agent any

    tools {
        maven 'M3'
    }

    stages {
        stage('Checkout Git') {
            steps {
                git branch: 'main',
                    url: '<your-repository-url>'
            }
        }

        stage('Build and Test') {
            steps {
                bat 'mvn clean test'
            }
        }
    }
}
```

STEP 10: Run Jenkins Pipeline

Steps:
1. Save pipeline
2. Click Build Now
3. Open Console Output

Expected:
- Tests run successfully
- BUILD SUCCESS
- Finished: SUCCESS

FINAL RESULT:

Jenkins checks out the Maven project from GitHub, builds it using Maven, runs JUnit unit tests automatically, and reports build success or failure.

This completes a full CI pipeline.