Lesson 01 Demo 03

Maven Plugin Integration in Jenkins

Objective: To demonstrate the integration of the Maven plugin in Jenkins and create a Maven project for building and managing a Java application

Tool required: GitHub

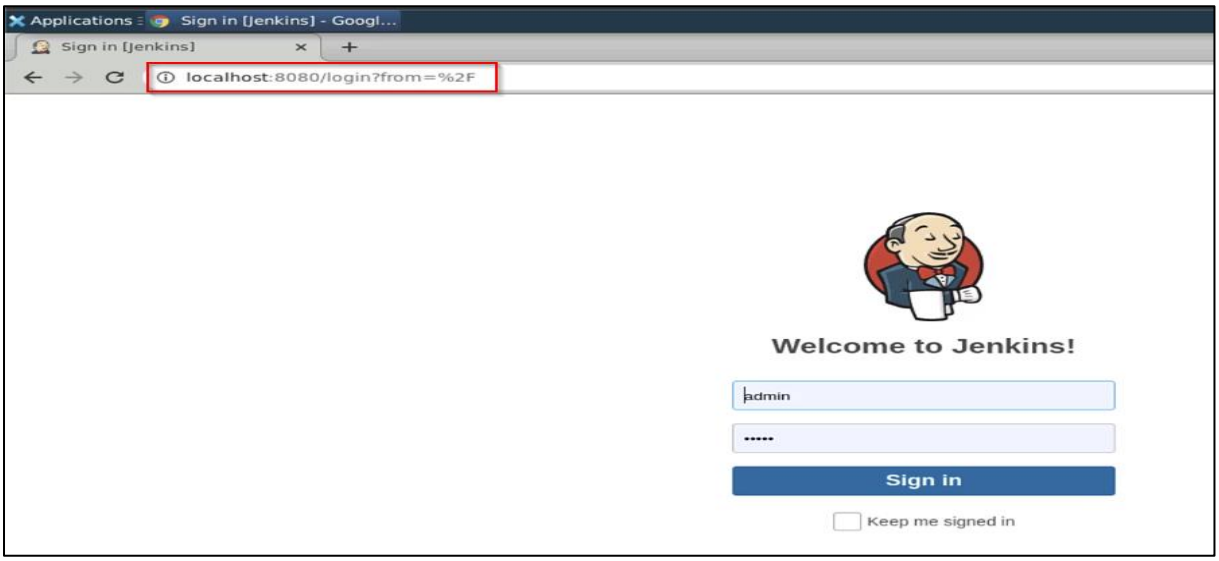
Prerequisites: None

Steps to be followed:

1. Configuring the plugin
2. Creating a Maven project in Jenkins

Step 1: Configuring the plugin

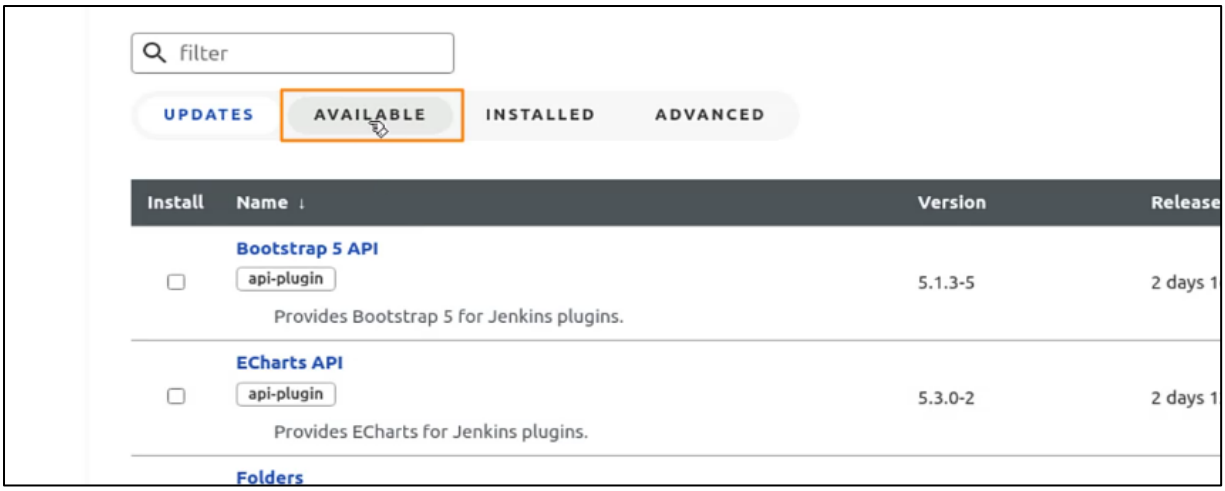
1.1 Open the browser in the lab and enter **localhost:8080**, then click on **Sign in**



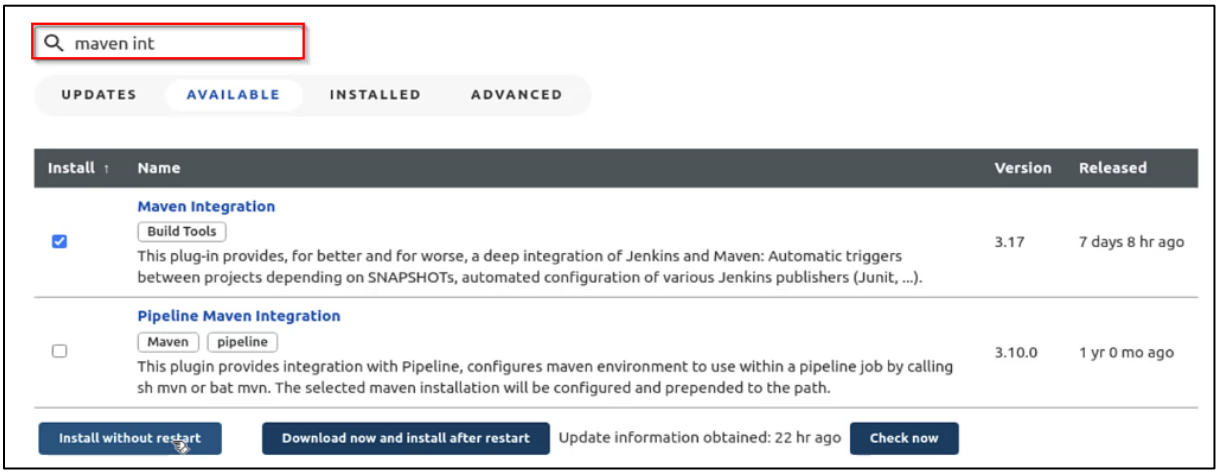
1.2 Click on **Manage Jenkins**

1.3 To add or remove Jenkins files, click on **Manage Plugins**

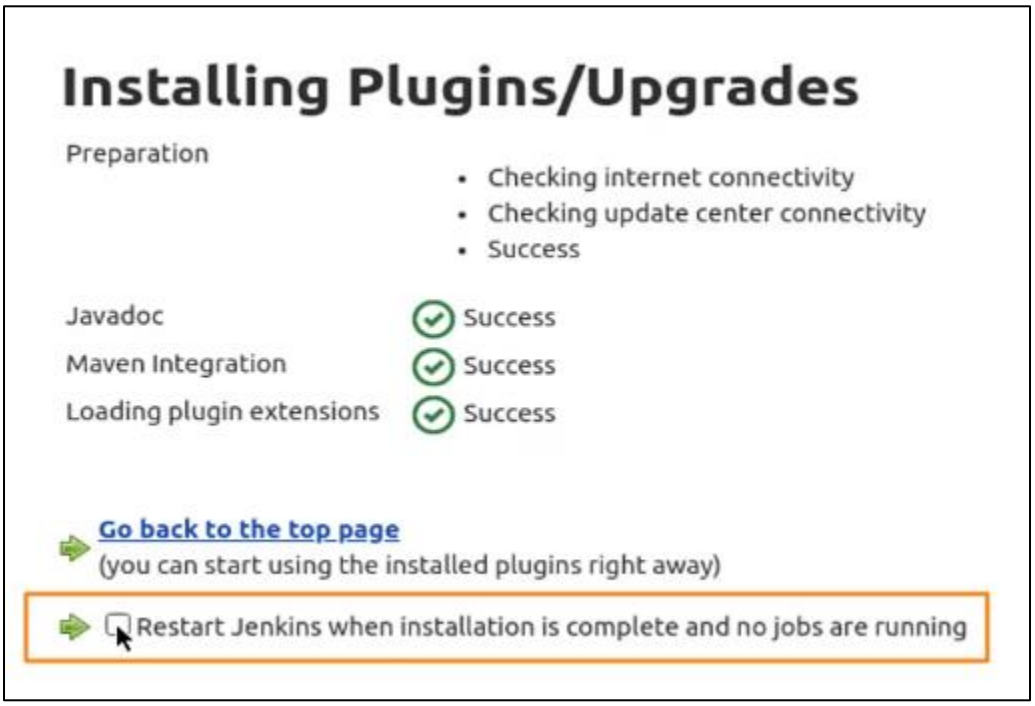
1.4 Click on the **AVAILABLE** button to access the available plugins



1.5 Search for **Maven Integration** under the **Build Tools** section and select **Maven Integration > Install without restart**

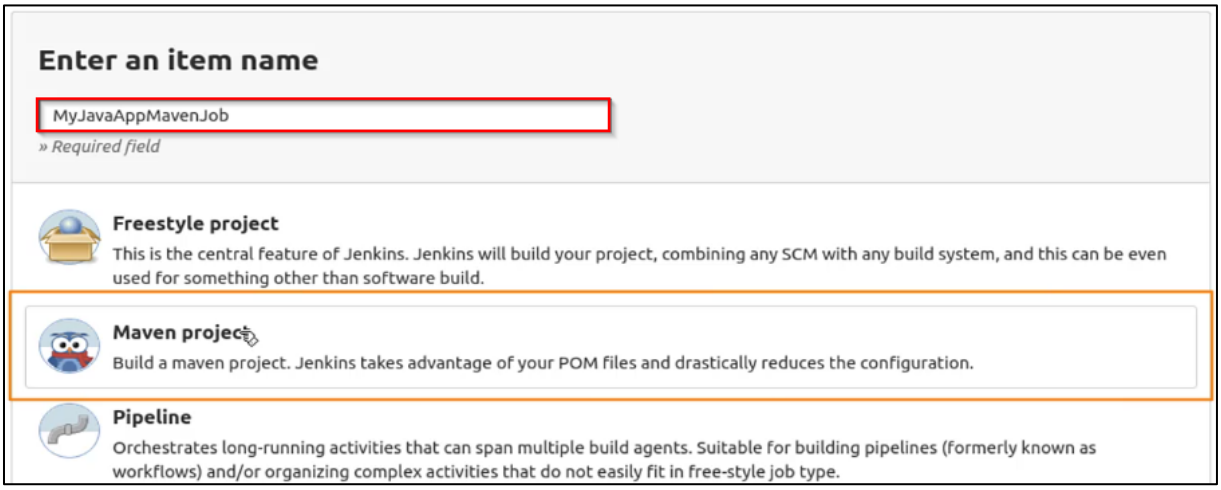


1.6 After the plugins are installed successfully, select **Restart Jenkins**



Step 2: Creating a Maven project in Jenkins

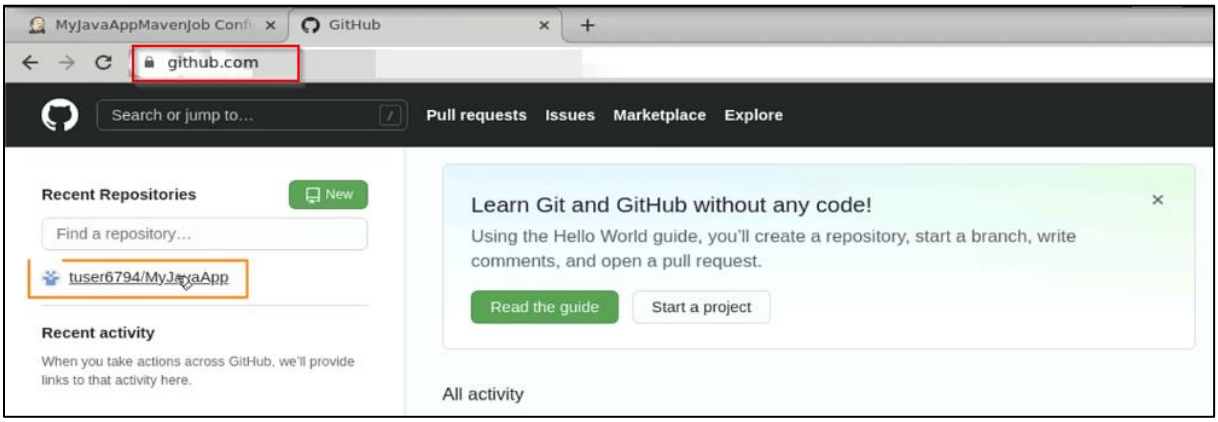
2.1 Enter the item name as **MyJavaAppMavenJob** and select **Maven project. Now,** click **OK**



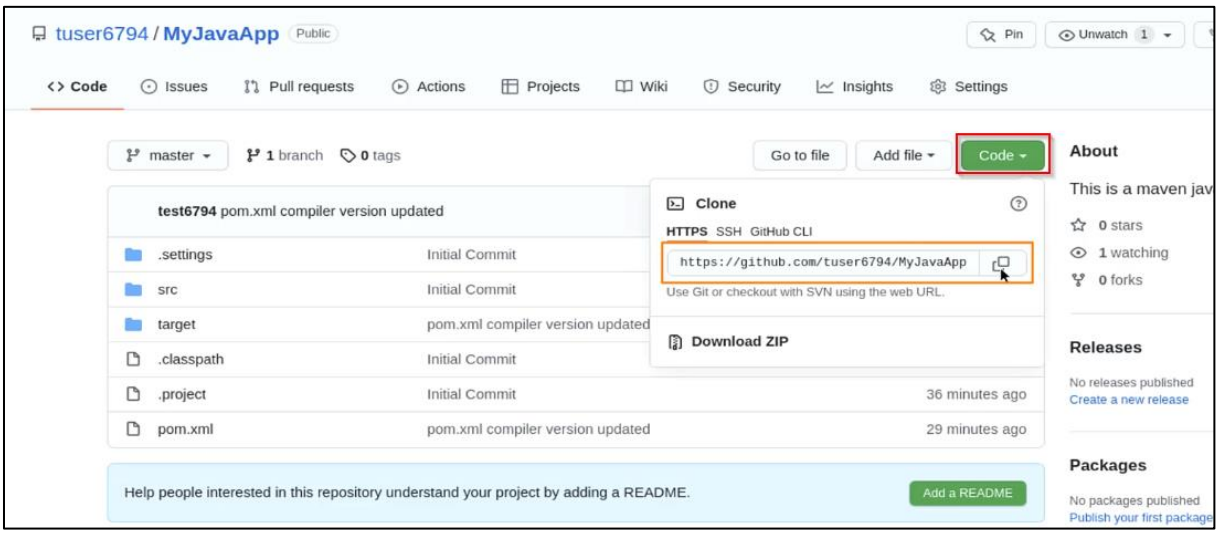
2.2 Provide the given description in the **General** section

2.3 Select the **Git** option under **Source Code Management**

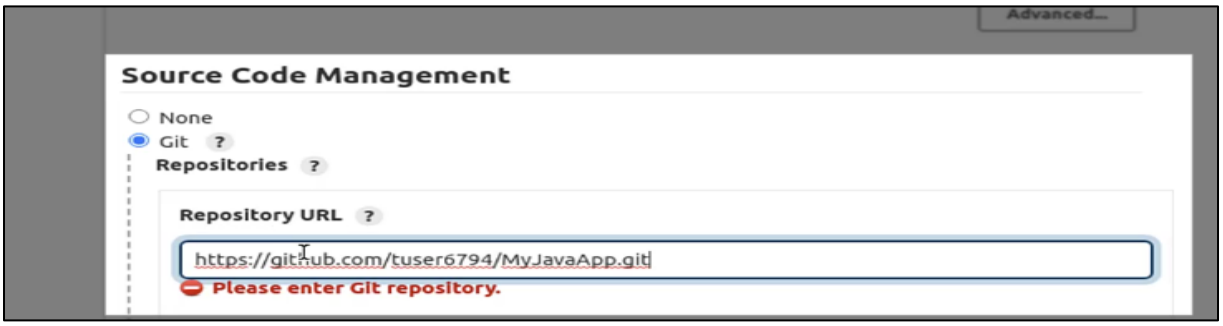
2.4 Navigate to the **GitHub URL** and click on the profile for **tuser6794/MyJavaApp**



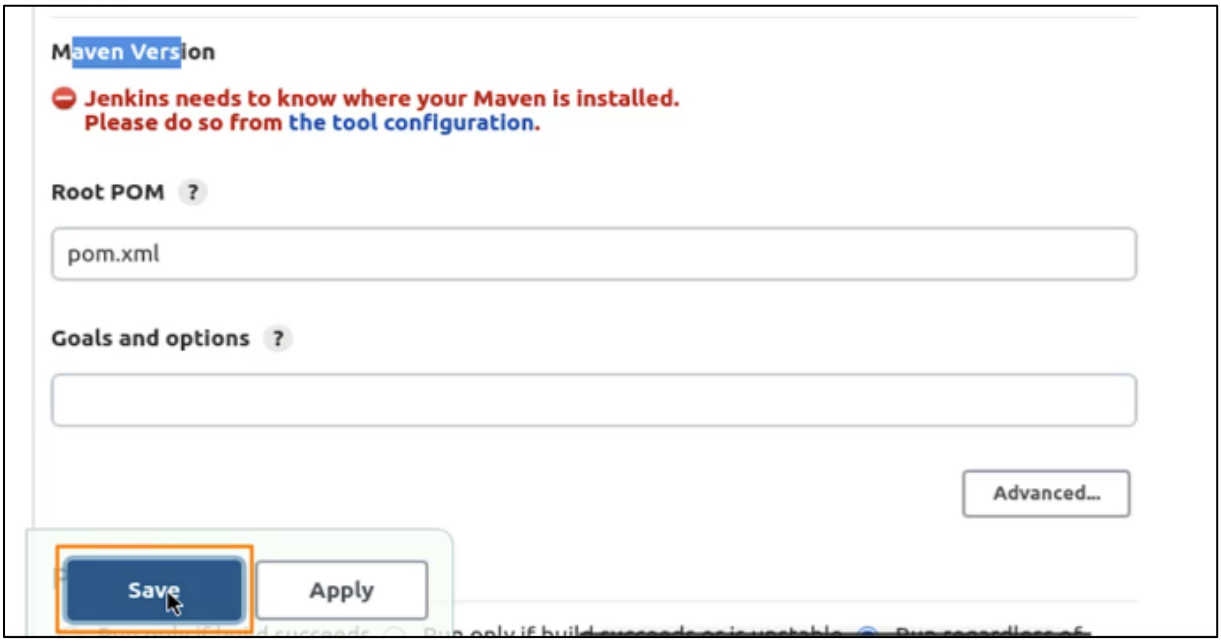
2.5 Click on **Code** and copy the **HTTPS** clone **GitHub** link



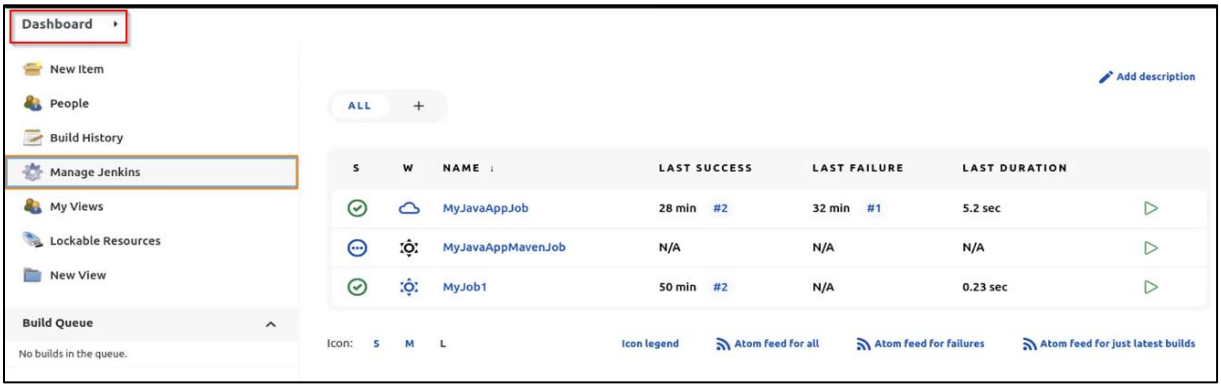
2.6 Paste the copied link into the **Repository URL** field



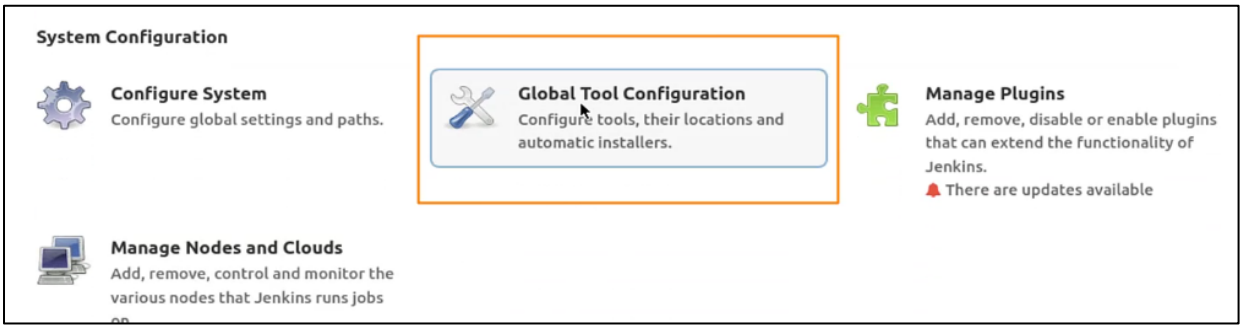
2.7 Scroll and click on the **Save** button



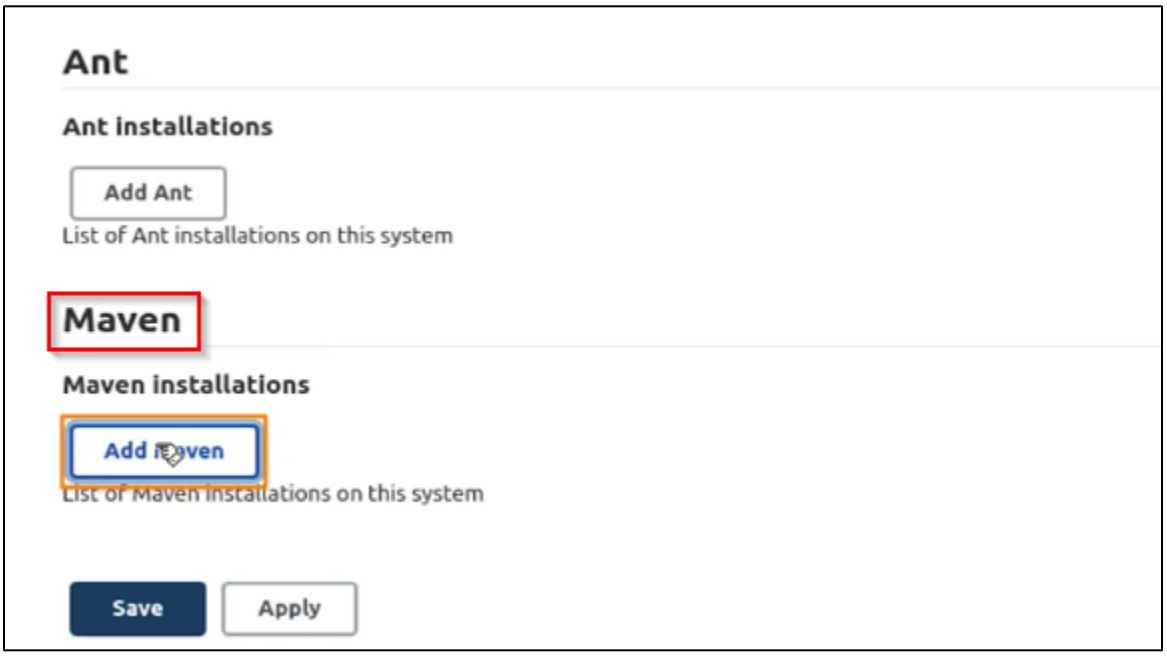
2.8 Navigate to the **Dashboard** and click **Manage Jenkins**



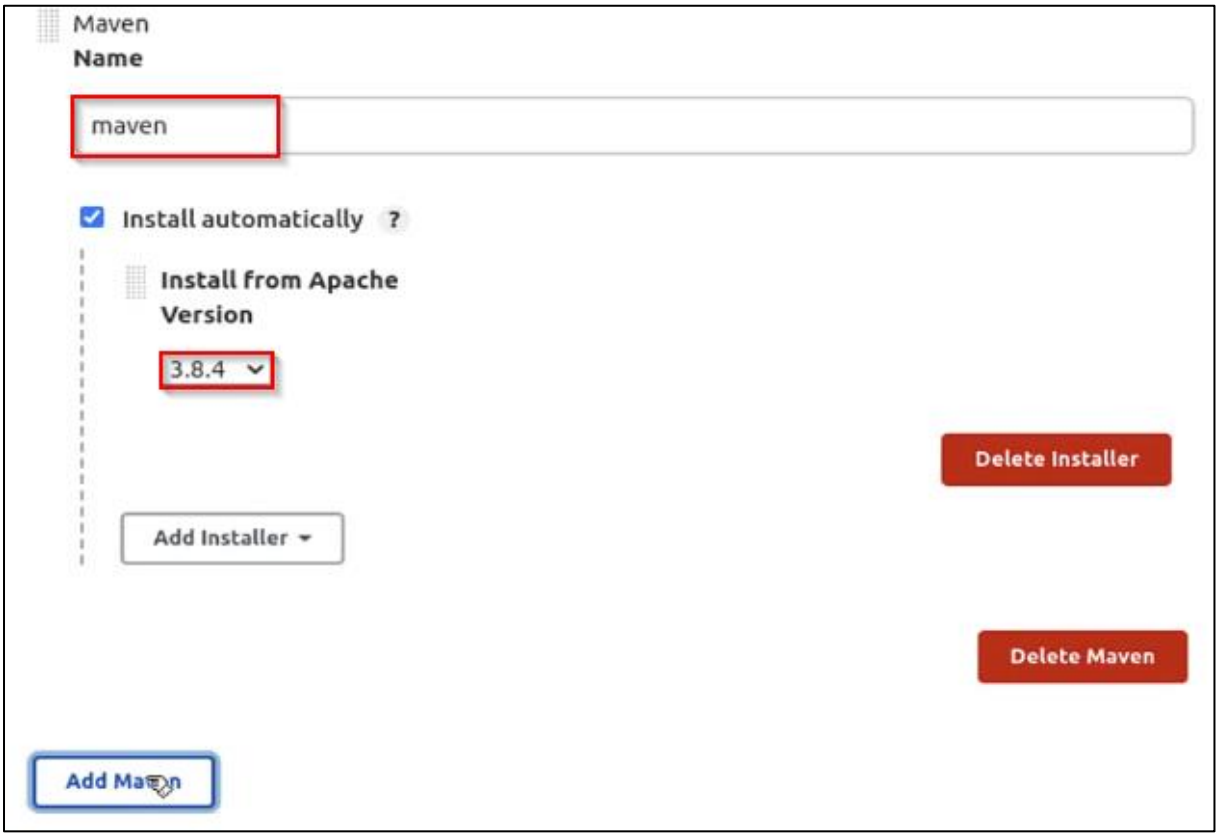
2.9 Select **Global Tool Configuration**



2.10 Scroll and click **Maven > Add Maven**



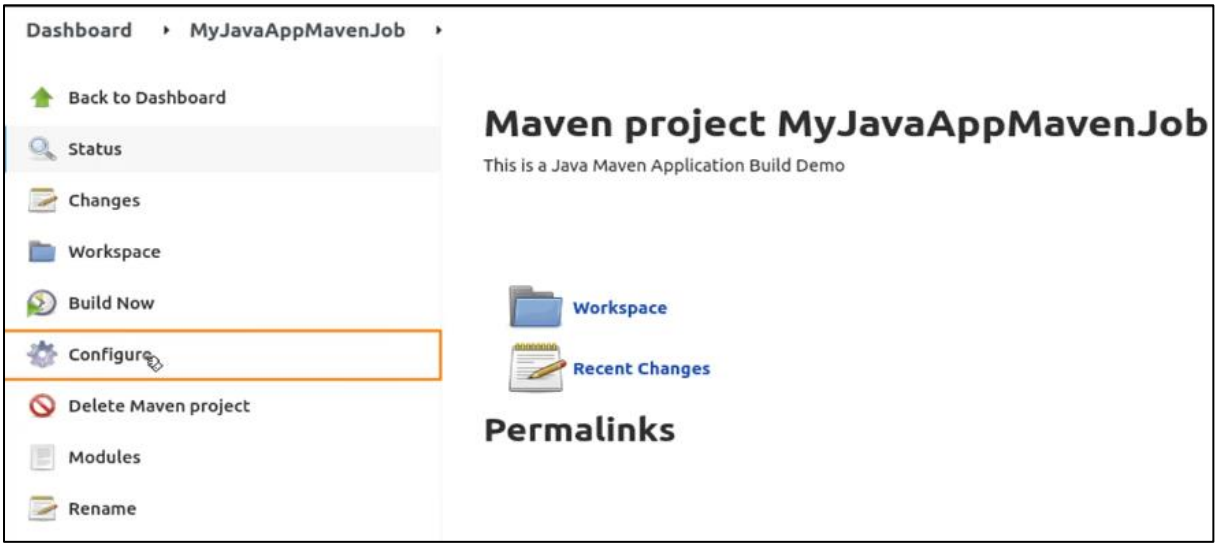
2.11 Enter the Name as **maven,** install **Apache Version 3.8.4**, and click **Add Maven**



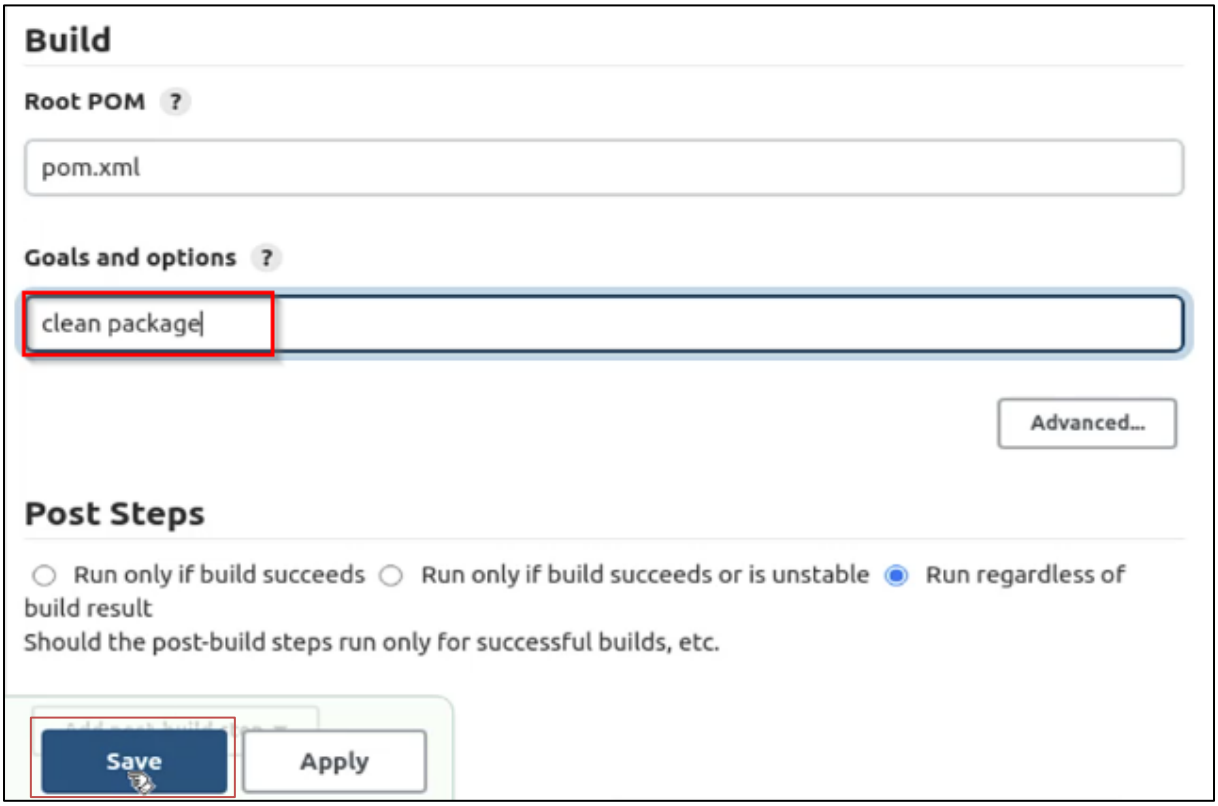
2.12 Select **Maven name** and click on **Save**

2.13 Go back to the Jenkins **Dashboard** and click **MyJavaAppMavenJob**

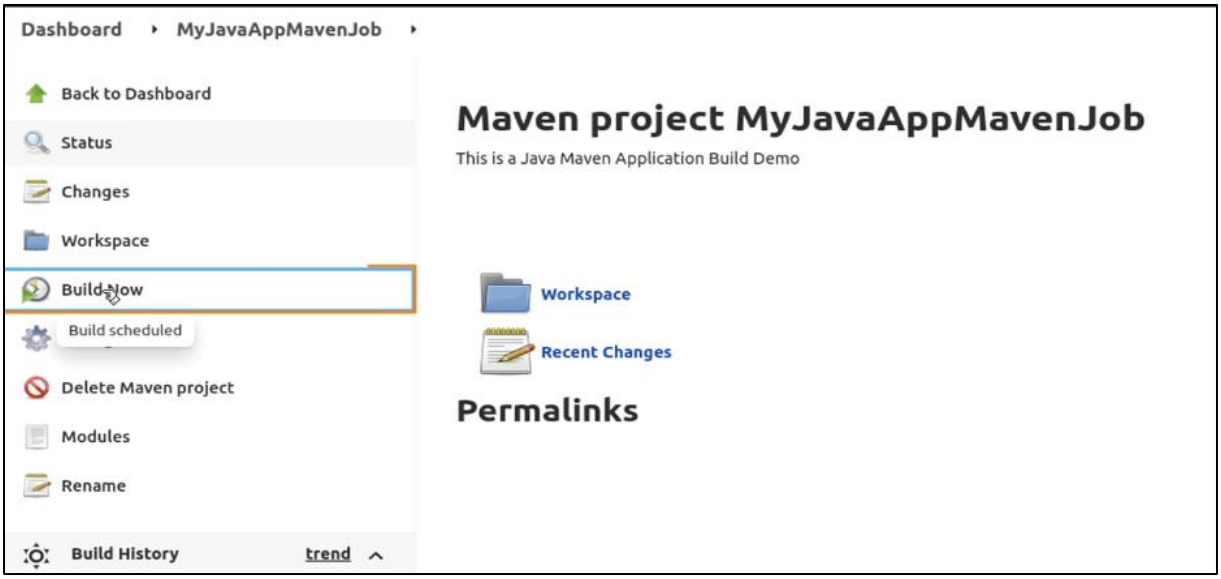
2.14 Click **Configure**



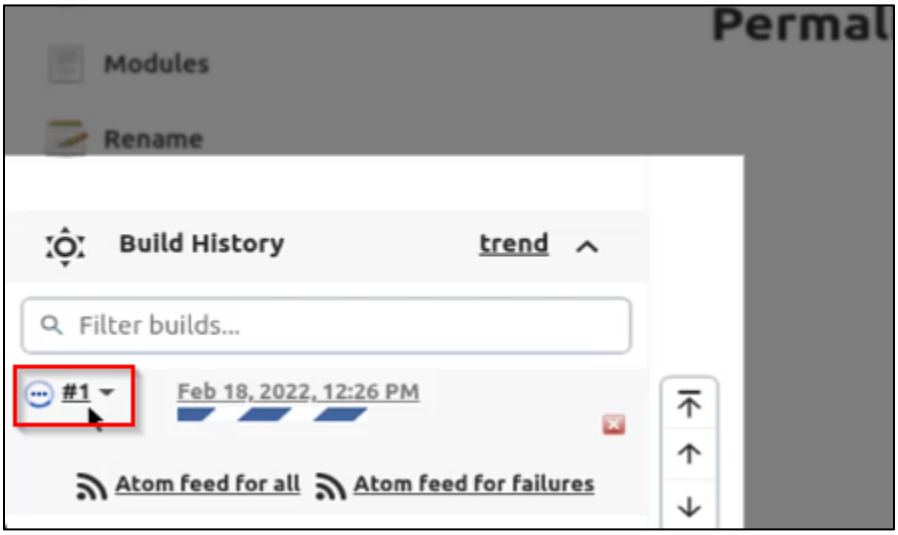
2.15 In the **Goal and options** section, name it as a **clean package** and click **Save**



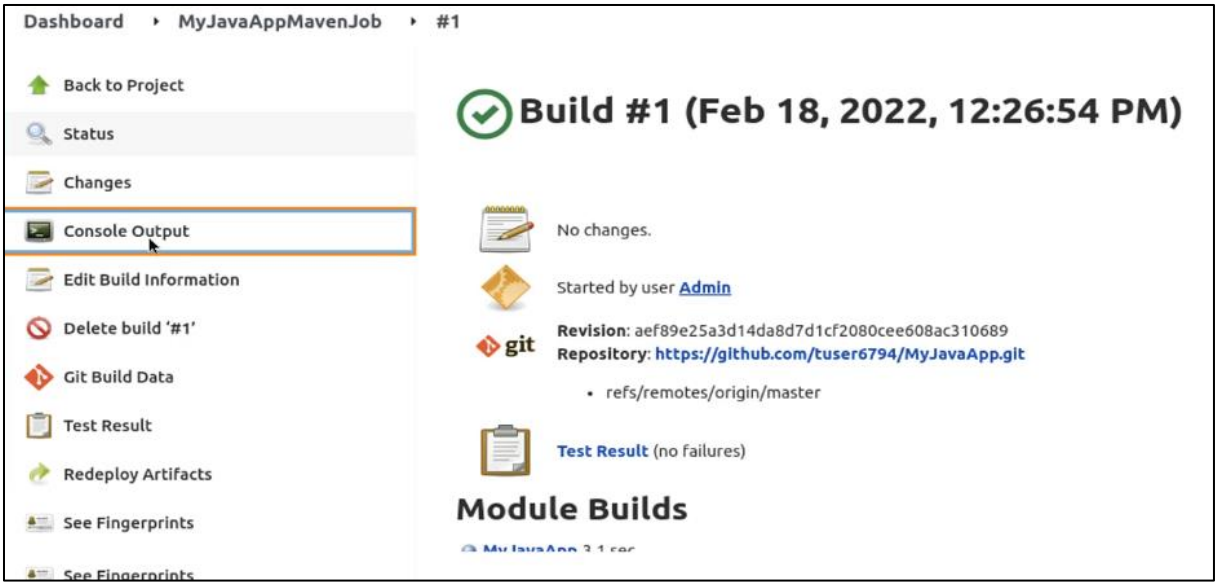
2.16 Click on the **Build Now** option for the Maven project **MyJavaAppMavenJob**



2.17 Click on **Build** option **#1** and open it



2.18 Open the **Console Output** for Build **#1**



Maven Plugin has been successfully installed in **Jenkins.**

```
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ MyJavaApp ---
[INFO] Surefire report directory: /var/lib/jenkins/workspace/MyJavaAppMavenJob/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.example.MyJavaApp.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.009 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[JENKINS] Recording test results
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ MyJavaApp ---
[INFO] Building jar: /var/lib/jenkins/workspace/MyJavaAppMavenJob/target/MyJavaApp-0.0.1-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.572 s
[INFO] Finished at: 2022-02-18T12:27:05Z
[INFO] ------------------------------------------------------------------------
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/MyJavaAppMavenJob/pom.xml to com.example/MyJavaApp/0.0.1-SNAPSHOT/MyJavaApp-
0.0.1-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/MyJavaAppMavenJob/target/MyJavaApp-0.0.1-SNAPSHOT.jar to
com.example/MyJavaApp/0.0.1-SNAPSHOT/MyJavaApp-0.0.1-SNAPSHOT.jar
channel stopped
Finished: SUCCESS
```

In conclusion, this demo showcased the seamless integration of the Maven plugin in Jenkins, enabling efficient management of Java applications. The configuration process involved installing the Maven Integration plugin and setting up a Maven project linked to a Git repository. Global tool configuration ensured the use of the correct Maven version.