

## 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

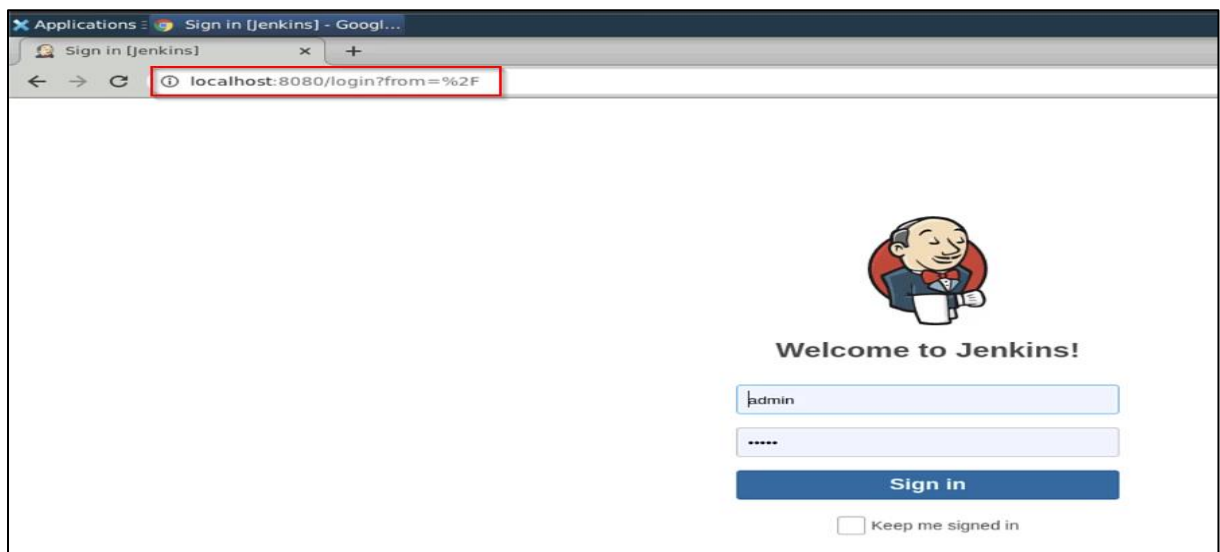
**Prerequisites:** None

#### Steps to be followed:

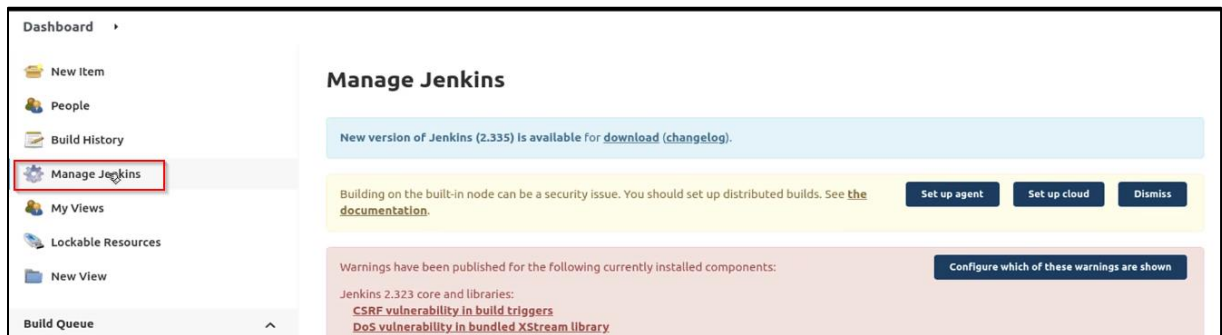
1. Configuring the plugin
2. Creating a Maven project in Jenkins

#### Step 1: Configuring the plugin

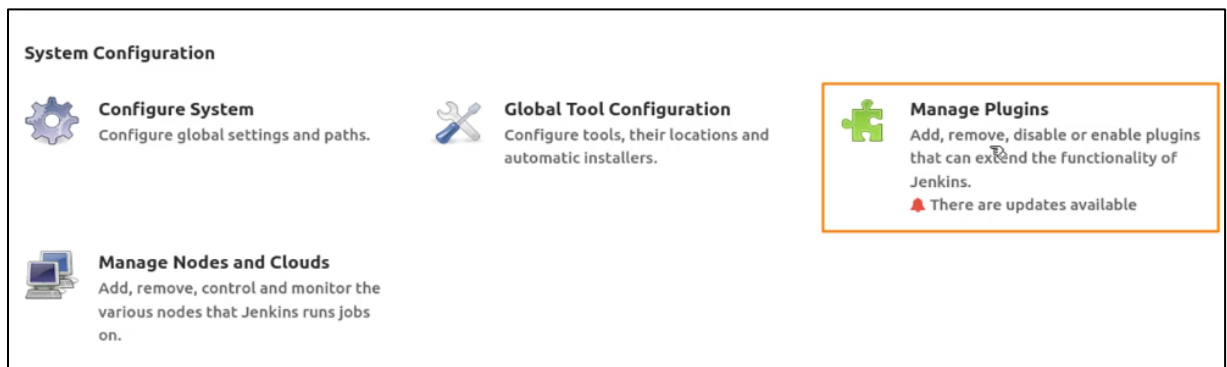
- 1.1 Open the browser in the lab and enter **localhost:8080**, then click on **Sign in**



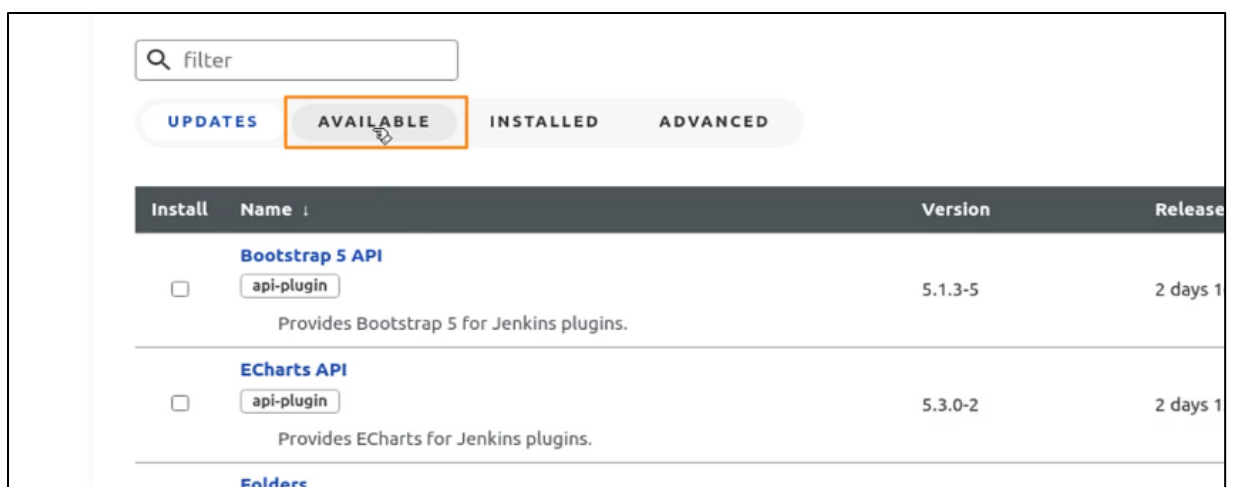
## 1.2 Click on **Manage Jenkins**



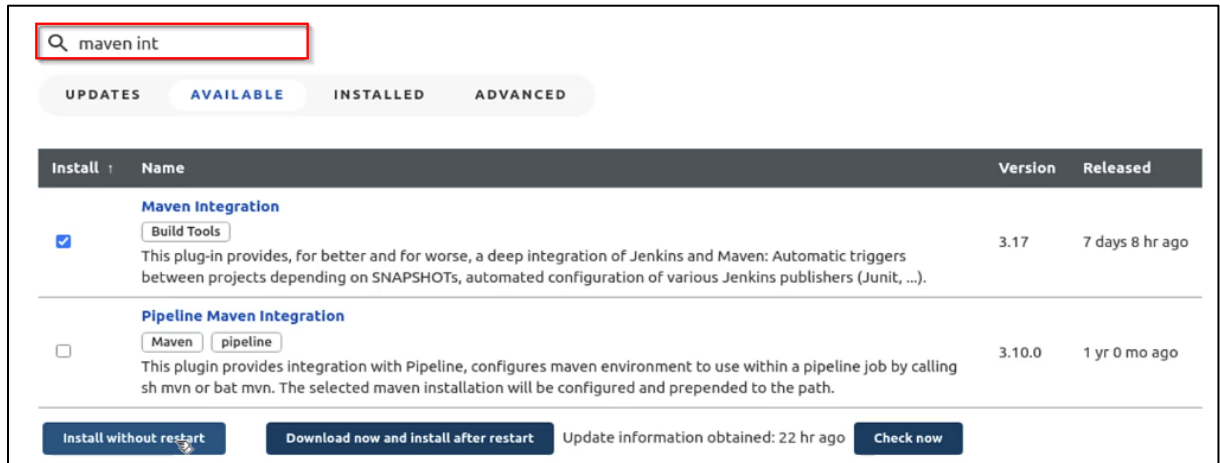
## 1.3 To add or remove Jenkins files, click on **Manage Plugins**



## 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**

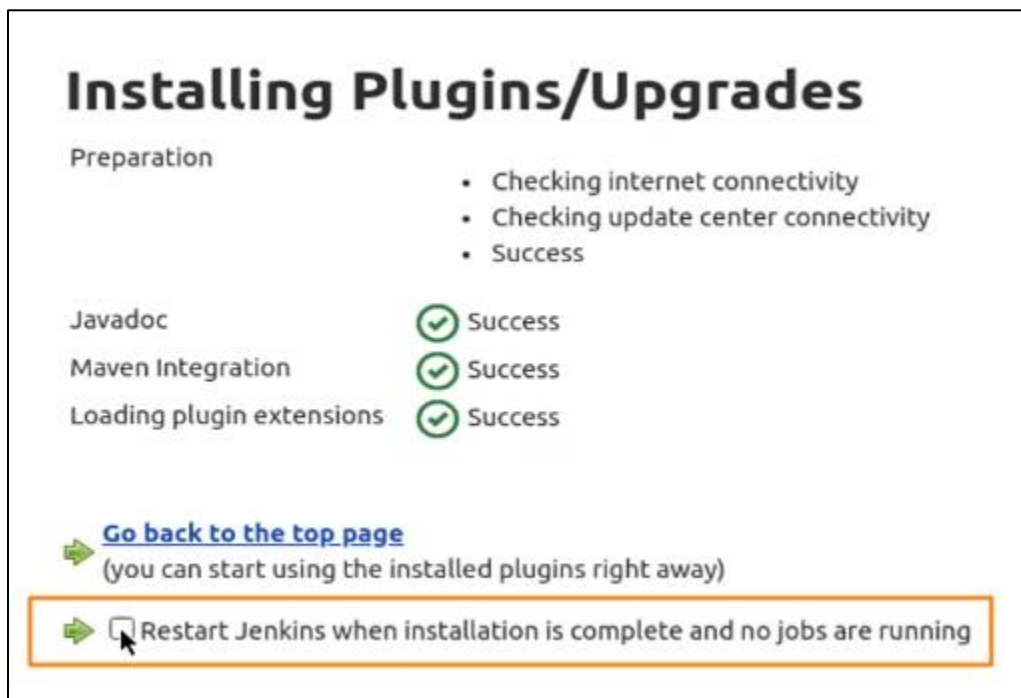


The screenshot shows the Jenkins plugin manager interface. A search bar at the top contains the text 'maven int'. Below the search bar are tabs for 'UPDATES', 'AVAILABLE', 'INSTALLED', and 'ADVANCED'. The 'AVAILABLE' tab is selected. A table lists the search results:

Install	Name	Version	Released
<input checked="" type="checkbox"/>	<b>Maven Integration</b> Build Tools This plug-in provides, for better and for worse, a deep integration of Jenkins and Maven: Automatic triggers between projects depending on SNAPSHOTS, automated configuration of various Jenkins publishers (Junit, ...).	3.17	7 days 8 hr ago
<input type="checkbox"/>	<b>Pipeline Maven Integration</b> Maven pipeline This plugin provides integration with Pipeline, configures maven environment to use within a pipeline job by calling sh mvn or bat mvn. The selected maven installation will be configured and prepended to the path.	3.10.0	1 yr 0 mo ago

At the bottom of the table, there are three buttons: 'Install without restart' (highlighted), 'Download now and install after restart', and 'Check now'. A note indicates 'Update information obtained: 22 hr ago'.

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



The screenshot shows the 'Installing Plugins/Upgrades' page in Jenkins. The title is 'Installing Plugins/Upgrades'. Under the 'Preparation' section, there is a list of steps:

- Checking internet connectivity
- Checking update center connectivity
- Success

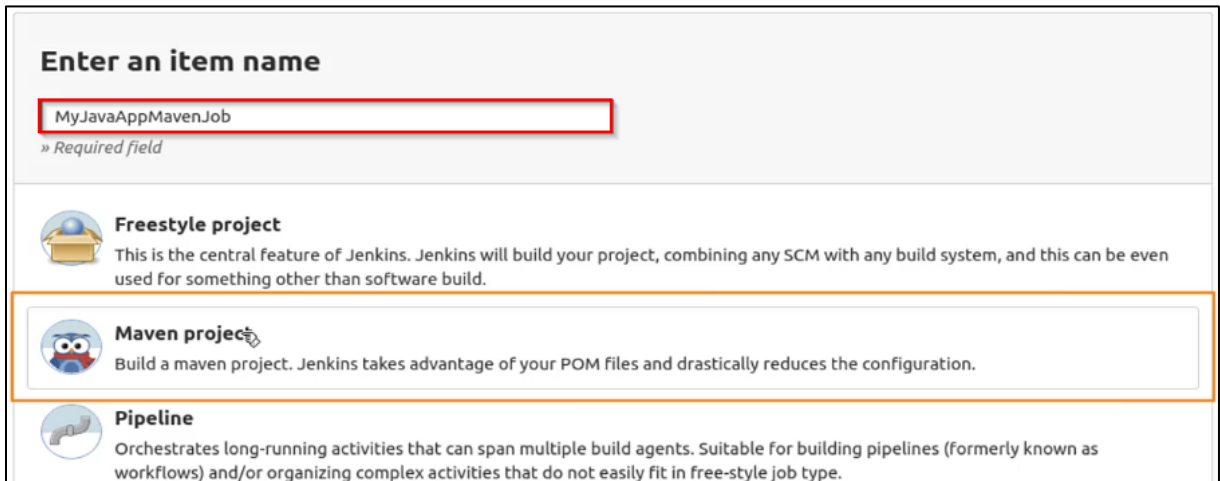
Below this, there are three rows showing the status of different components:

Javadoc	✓ Success
Maven Integration	✓ Success
Loading plugin extensions	✓ Success

At the bottom, there is a link 'Go back to the top page' with a green arrow icon. Below this, there is a checkbox labeled 'Restart Jenkins when installation is complete and no jobs are running', which is currently unchecked. The checkbox and its label are highlighted with an orange border.

## Step 2: Creating a Maven project in Jenkins

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



**Enter an item name**

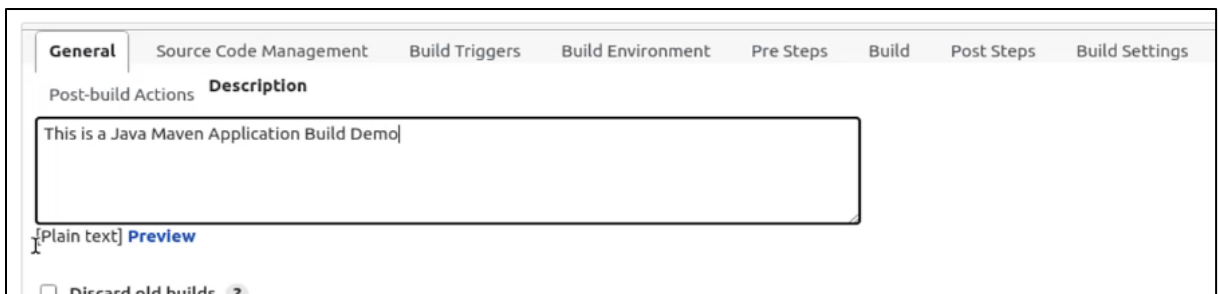
MyJavaAppMavenJob  
» Required field

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

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



**General** Source Code Management Build Triggers Build Environment Pre Steps Build Post Steps Post Steps Build Settings

Post-build Actions **Description**

This is a Java Maven Application Build Demo

[Plain text] [Preview](#)

☐ Discard old builds ?

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



**Source Code Management**

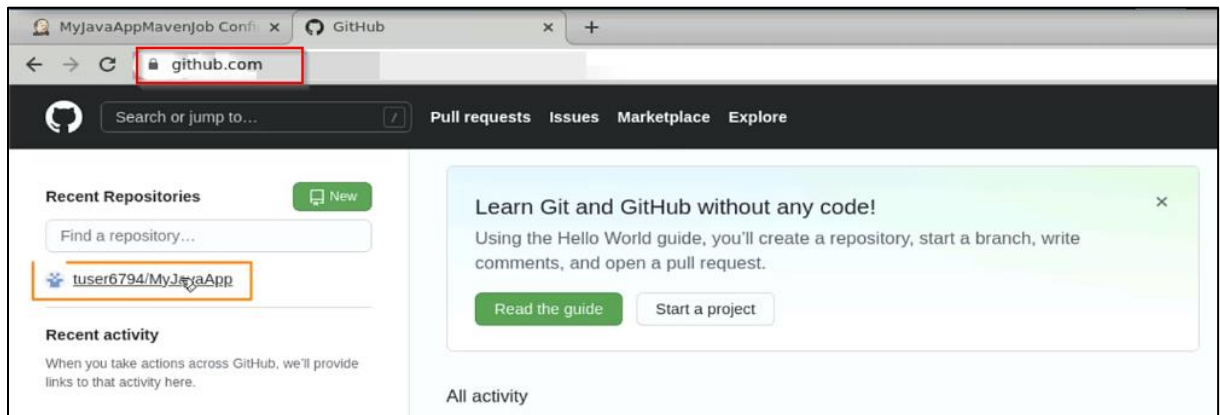
☐ None

☒ Git ?

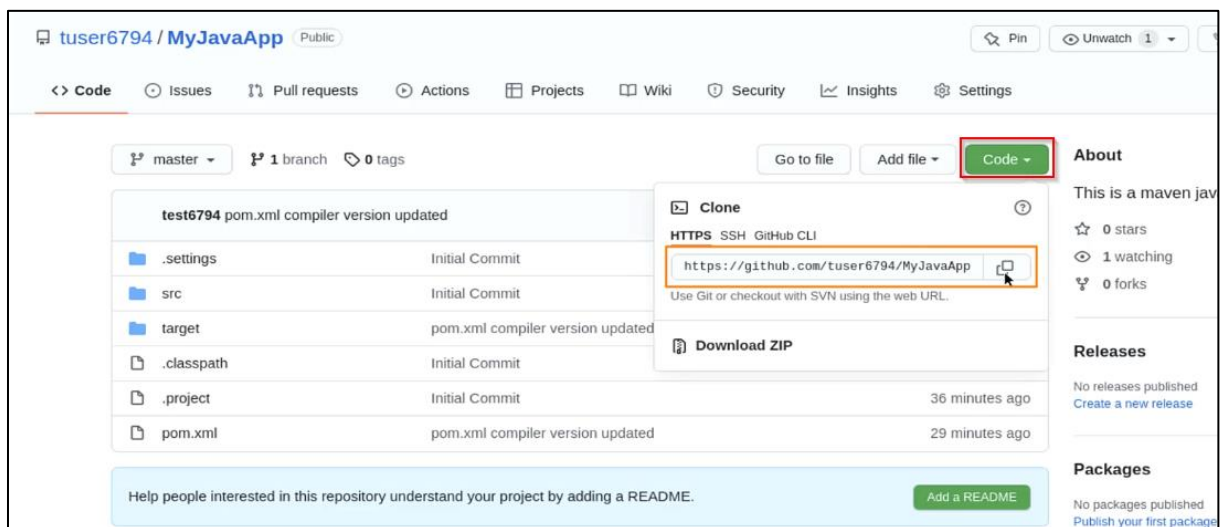
**Repositories** ?

**Repository URL** ?

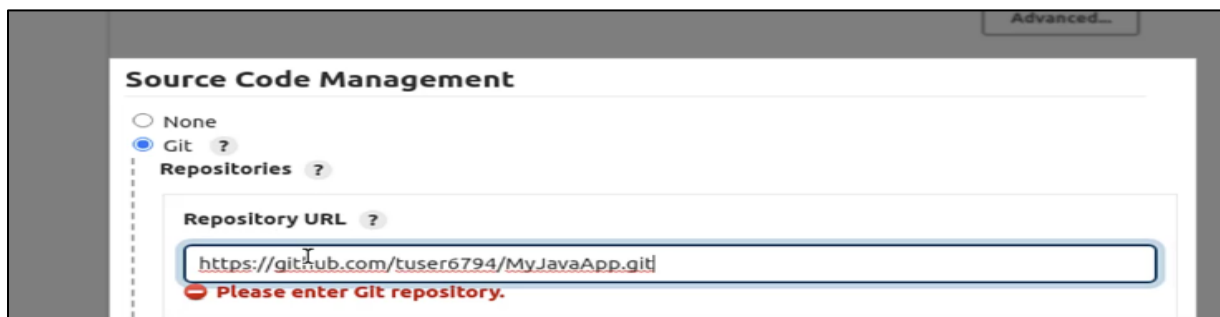
## 2.4 Navigate to the **GitHub** URL and click on the profile for **tuser6794/MyJavaApp**



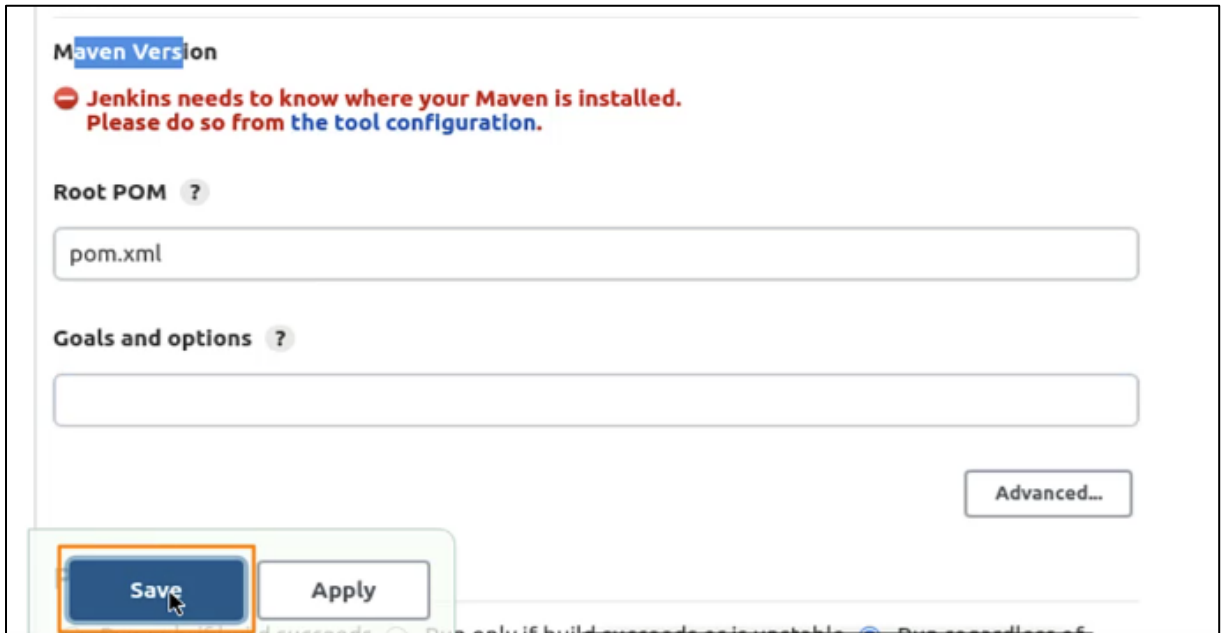
## 2.5 Click on **Code** and copy the **HTTPS** clone **GitHub** link



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



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



**Maven Version**

❌ Jenkins needs to know where your Maven is installed. Please do so from the tool configuration.

**Root POM** ?

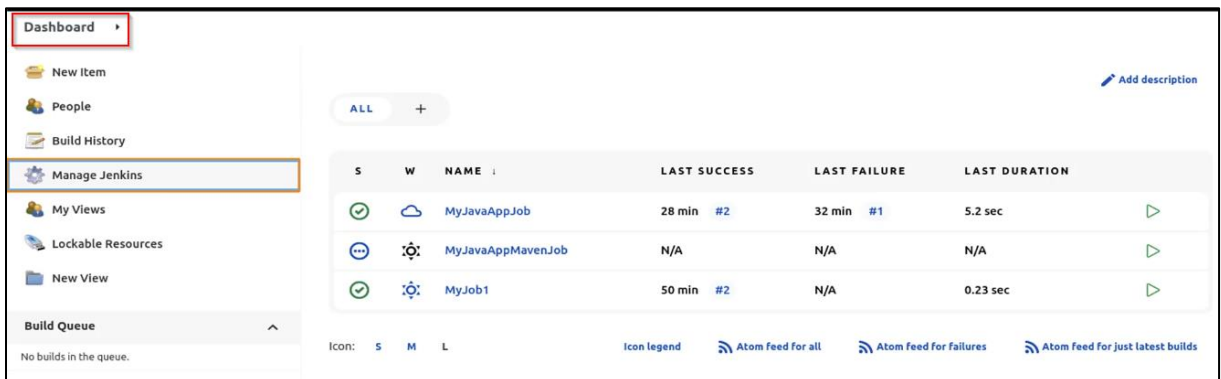
pom.xml

**Goals and options** ?

Advanced...

**Save** **Apply**

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



**Dashboard**

- New Item
- People
- Build History
- Manage Jenkins**
- My Views
- Lockable Resources
- New View

**Build Queue**

No builds in the queue.

**ALL** +

S	W	NAME	LAST SUCCESS	LAST FAILURE	LAST DURATION
✓	☁	MyJavaAppJob	28 min #2	32 min #1	5.2 sec
⋮	⚙	MyJavaAppMavenJob	N/A	N/A	N/A
✓	⚙	MyJob1	50 min #2	N/A	0.23 sec

Icon: S M L

Icon legend


Atom feed for all

Atom feed for failures


Atom feed for just latest builds

## 2.9 Select **Global Tool Configuration**


**System Configuration**




**Configure System**  
Configure global settings and paths.



**Global Tool Configuration**  
Configure tools, their locations and automatic installers.



**Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.  
🔔 There are updates available



**Manage Nodes and Clouds**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

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

### Ant

**Ant installations**

[Add Ant](#)

List of Ant installations on this system

### Maven

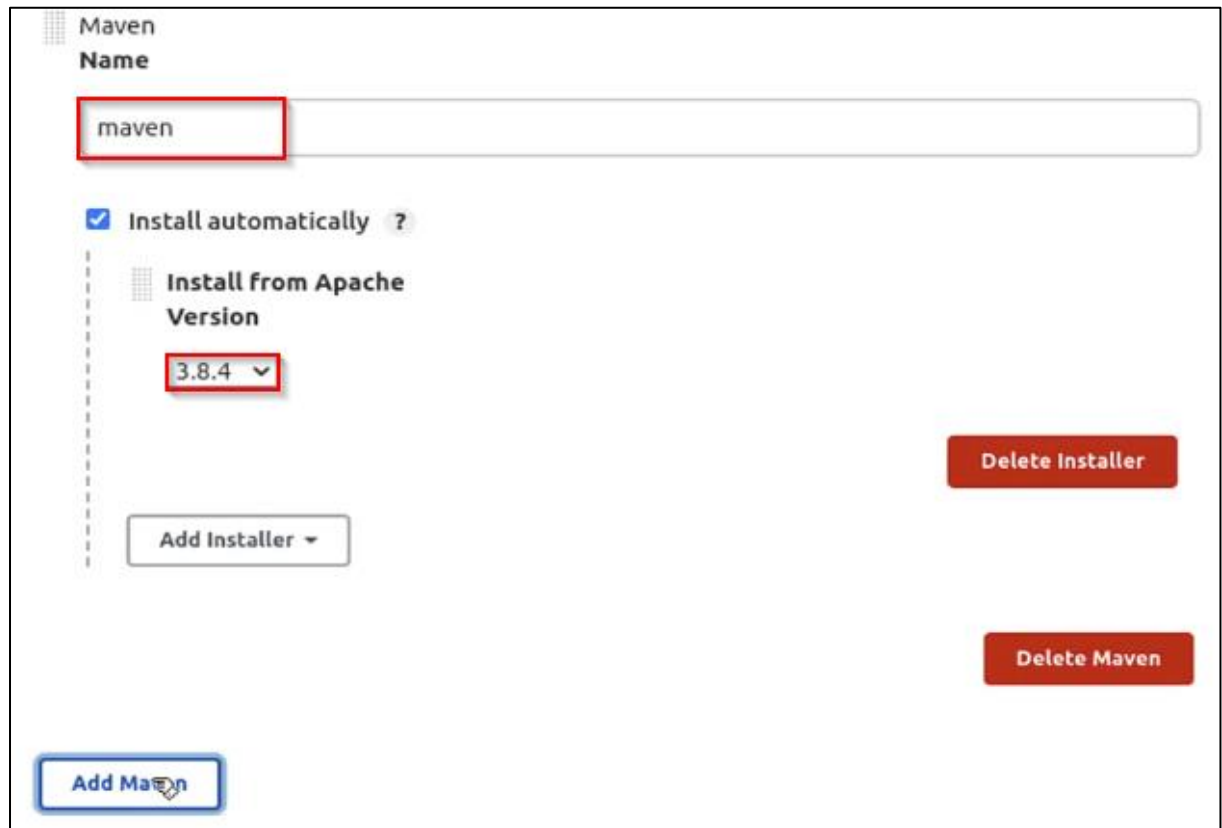
**Maven installations**

[Add Maven](#)

List of Maven installations on this system

[Save](#) [Apply](#)

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



Maven  
Name

maven

☒ Install automatically ?

Install from Apache  
Version

3.8.4 ▾

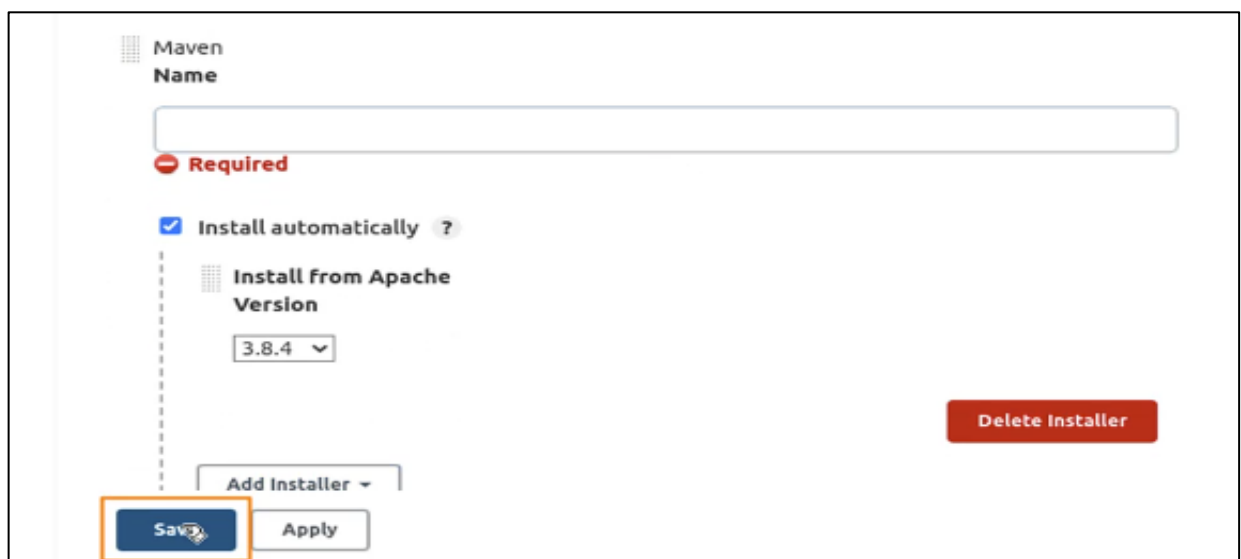
Add Installer ▾

Delete Installer

Delete Maven

Add Maven

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



Maven  
Name

Required

☒ Install automatically ?

Install from Apache  
Version

3.8.4 ▾

Add Installer ▾

Delete Installer

Save

Apply



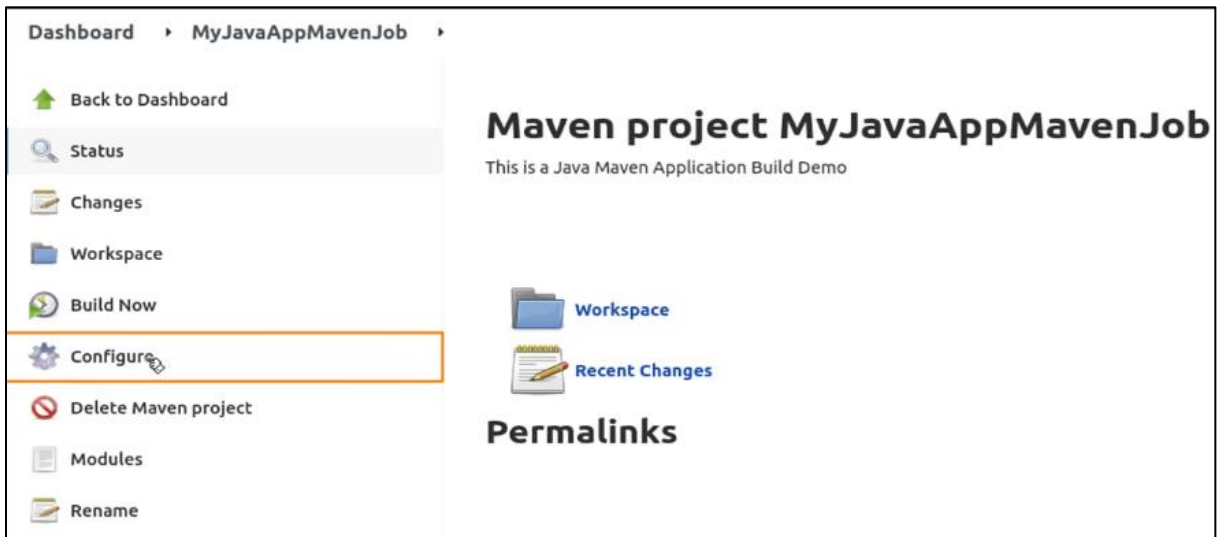
2.13 Go back to the Jenkins **Dashboard** and click **MyJavaAppMavenJob**



The screenshot shows the Jenkins Dashboard. On the left, there is a sidebar with navigation links: New Item, People, Build History, Manage Jenkins, My Views, Lockable Resources, and New View. The main area displays a table of jobs. The 'MyJavaAppMavenJob' entry is highlighted with an orange box. The table has columns for status (S), workspace (W), name (NAME), last success, last failure, last duration, and a play button icon.

S	W	NAME	LAST SUCCESS	LAST FAILURE	LAST DURATION
✓	☁	MyJavaAppJob	29 min #2	33 min #1	5.2 sec
⋮	⚙	MyJavaAppMavenJob	N/A	N/A	N/A
✓	⚙	MyJob1	51 min #2	N/A	0.23 sec

2.14 Click **Configure**



The screenshot shows the Jenkins 'Configure' page for the 'MyJavaAppMavenJob'. The left sidebar contains links: Back to Dashboard, Status, Changes, Workspace, Build Now, Configure (highlighted with an orange box), Delete Maven project, Modules, and Rename. The main area displays the title 'Maven project MyJavaAppMavenJob' and a subtitle 'This is a Java Maven Application Build Demo'. Below this, there are links for 'Workspace' and 'Recent Changes', and a section titled 'Permalinks'.

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

## Build

Root POM ?

Goals and options ?

Advanced...

### Post Steps

☐ Run only if build succeeds
 ☐ Run only if build succeeds or is unstable
 ☒ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

Save

Apply

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

Dashboard > MyJavaAppMavenJob >

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now**
- Build scheduled
- Delete Maven project
- Modules
- Rename
- Build History trend ^

## Maven project MyJavaAppMavenJob

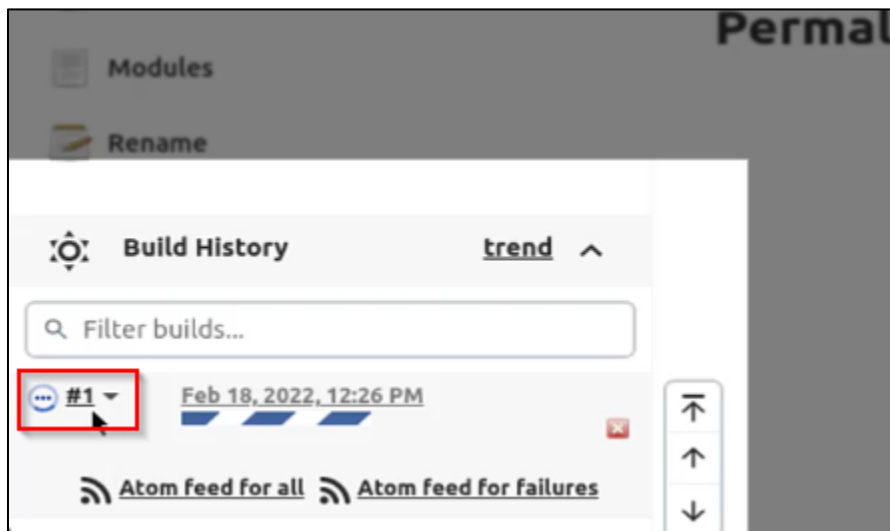
This is a Java Maven Application Build Demo

Workspace

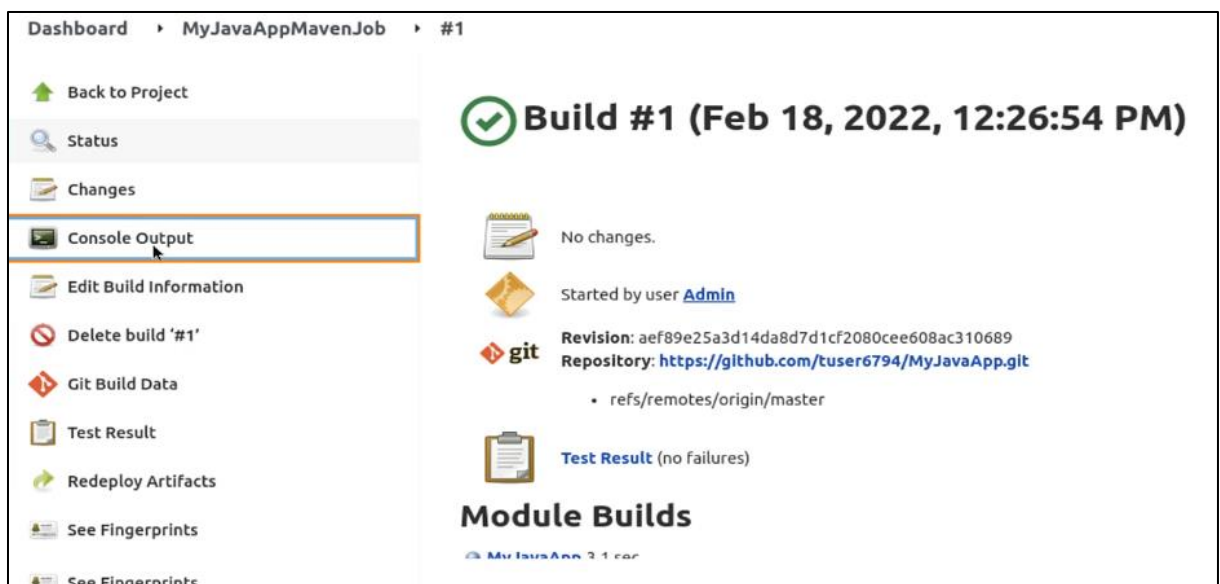
Recent Changes

### Permalinks

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.