

Lab Exercise 7

Integrating Maven with Jenkins

Objective: To install the Maven plugin in Jenkins for smooth integration and automation of Maven-based build processes within the Jenkins environment

Tools required: Git, GitHub, and Jenkins

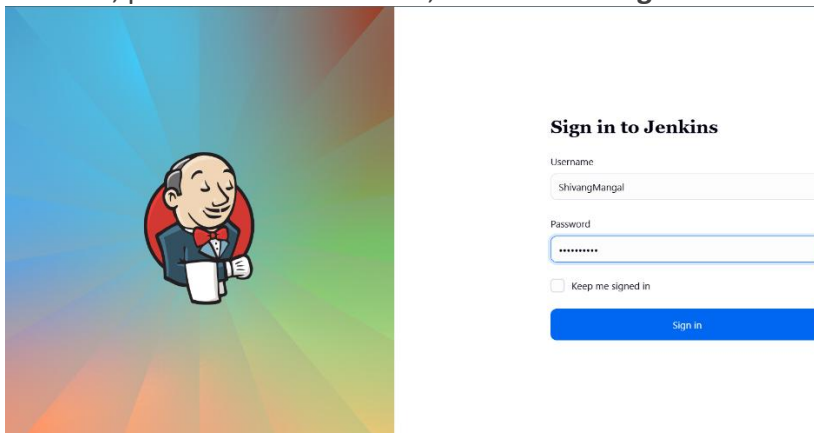
Prerequisites: None

Steps to be followed:

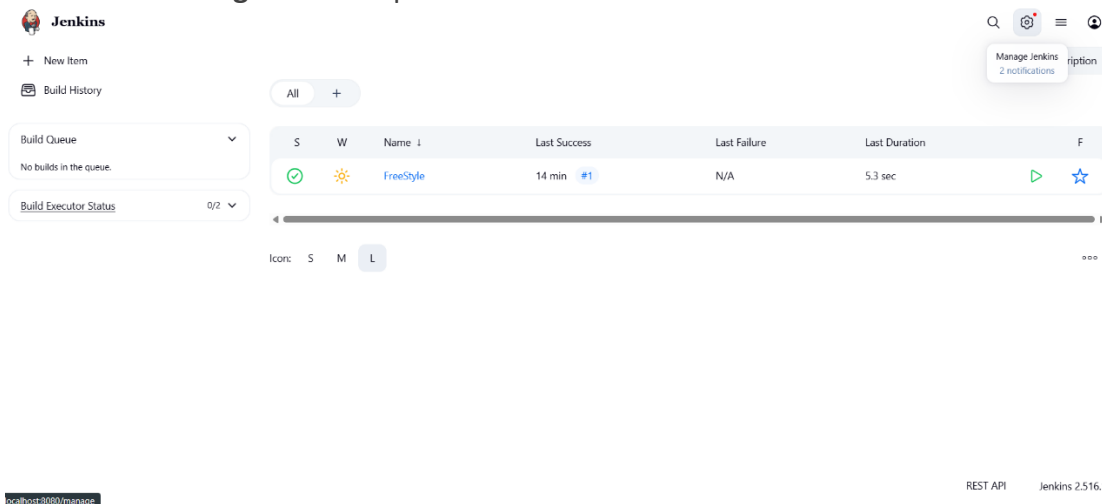
1. Install the Maven plugin
2. Set up Global Tool Configuration
3. Fork a sample repository
4. Integrate Maven with Jenkins

Step 1: Install the Maven plugin

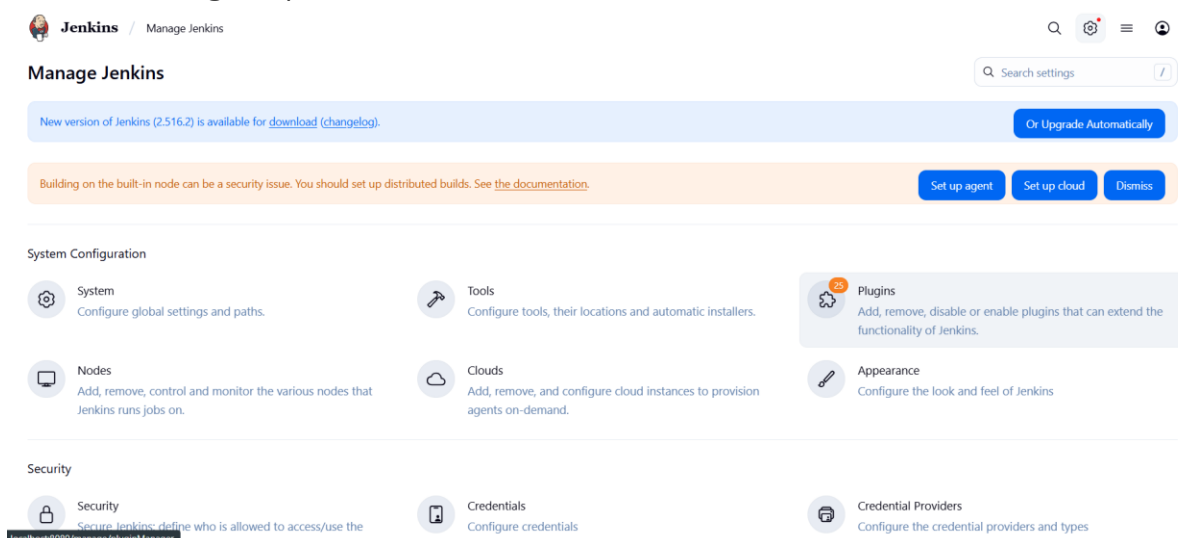
- 1.1 Open the browser, go to the Jenkins Dashboard by typing **localhost:8080** in your browser, provide the credentials, and click the **Sign in** button



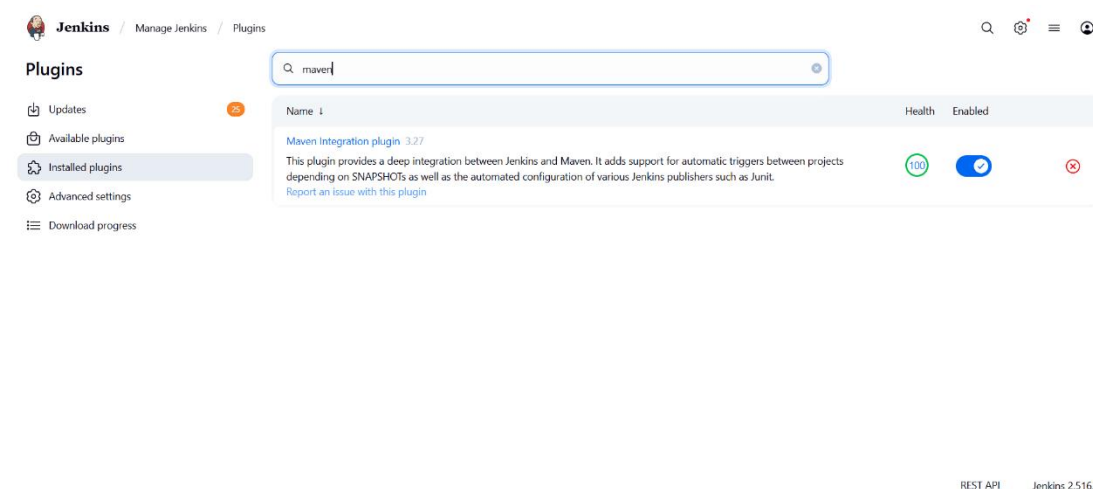
- 1.2 Click on the **Manage Jenkins** option as shown in the screenshot below:



1.3 Click on the **Plugins** option as shown in the screenshot below:



1.4 Click on **Installed plugins** to verify whether the **Maven Integration plugin** has been installed



Note: Maven is already installed in your practice lab environment. If not, click on **Available plugins**, search for the Maven Integration plugin, and install it.

1.5 Use the following command to check the Maven version:

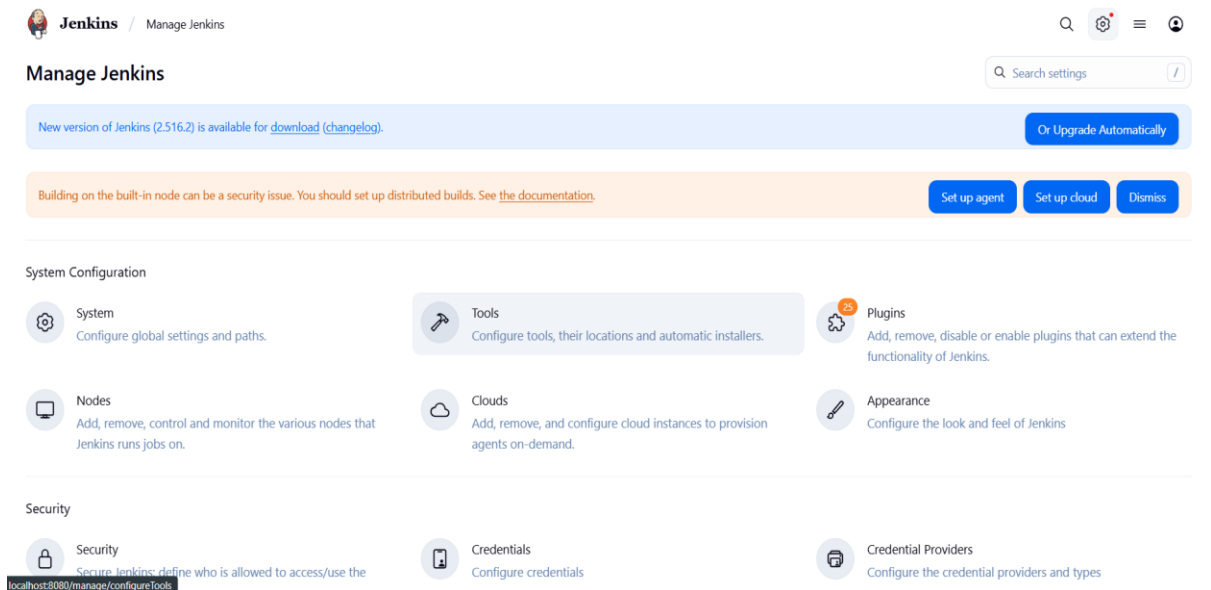
mvn -version

```
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>mvn --version
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: C:\maven-mvnd-1.0.2-windows-amd64\mvn
Java version: 23.0.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-23
Default locale: en_IN, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
```

Step 2: Set up Global Tool Configuration

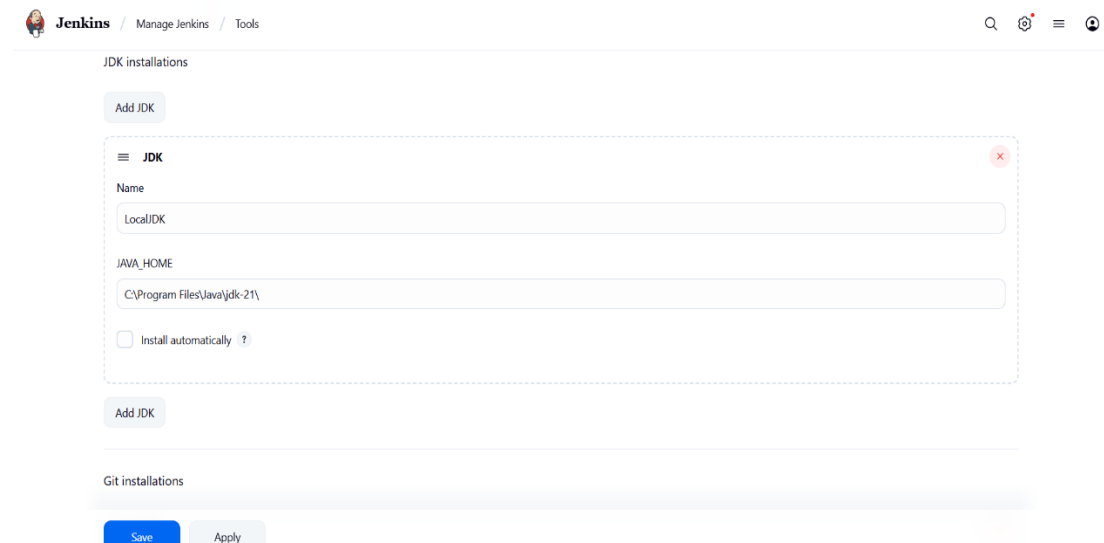
2.1 Go to the Jenkins Dashboard, click on **Manage Jenkins**, and then select **Tools** from the list of options



The screenshot shows the Jenkins 'Manage Jenkins' page. At the top, there's a navigation bar with the Jenkins logo, 'Manage Jenkins' text, and search, settings, and user icons. Below this is a 'Manage Jenkins' header with a search settings input. A blue banner indicates a new version of Jenkins (2.516.2) is available for download. An orange banner below it states that building on the built-in node can be a security issue and suggests setting up distributed builds. The main content area is divided into two sections: 'System Configuration' and 'Security'. Under 'System Configuration', there are six options: 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers), 'Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), and 'Appearance' (Configure the look and feel of Jenkins). Under 'Security', there are three options: 'Security' (Secure Jenkins: define who is allowed to access/use the Jenkins instance), 'Credentials' (Configure credentials), and 'Credential Providers' (Configure the credential providers and types).

2.2 Click on **JDK installations** and provide the **Name** and **JAVA_HOME** path

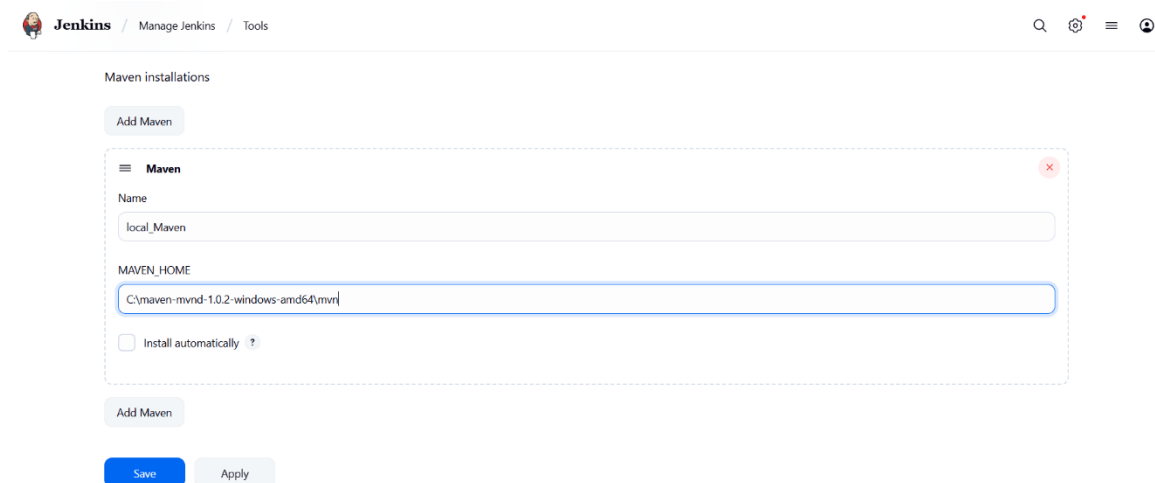
Note: Set the **JAVA_HOME** environment variable to **/usr/lib/jvm/java-11-openjdk-amd64**



The screenshot shows the Jenkins 'Tools' page, specifically the 'JDK installations' section. At the top, there's a navigation bar with the Jenkins logo, 'Manage Jenkins', and 'Tools' text. Below this is a 'JDK installations' header. A dashed box contains a form for adding a new JDK. The form has a title 'JDK' and a close button. It includes two text input fields: 'Name' (with the value 'LocalJDK') and 'JAVA_HOME' (with the value 'C:\Program Files\Java\jdk-21\'). There is also a checkbox labeled 'Install automatically' with a help icon. Below the form is an 'Add JDK' button. At the bottom of the page, there are 'Save' and 'Apply' buttons.

2.3 To configure Maven, click on the **Maven installations** button in the Maven section and enter a **Name** and **MAVEN_HOME** path

Note: Set the **MAVEN_HOME** environment variable to **/usr/share/maven**



Jenkins / Manage Jenkins / Tools

Maven installations

Add Maven

Maven

Name

local_Maven

MAVEN_HOME

C:\maven-mvnd-1.0.2-windows-amd64\mvn

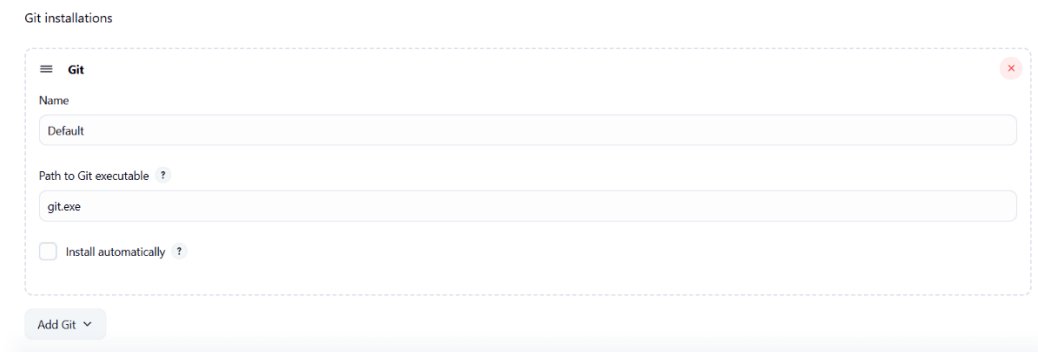
☐ Install automatically ?

Add Maven

Save Apply

2.4 To configure Git, click on **Git installations** and add the **Name** and **Path to Git executable**

Note: Set the **Path to Git executable** environment variable to **/bin/git** and click on **Save**



Git installations

Git

Name

Default

Path to Git executable ?

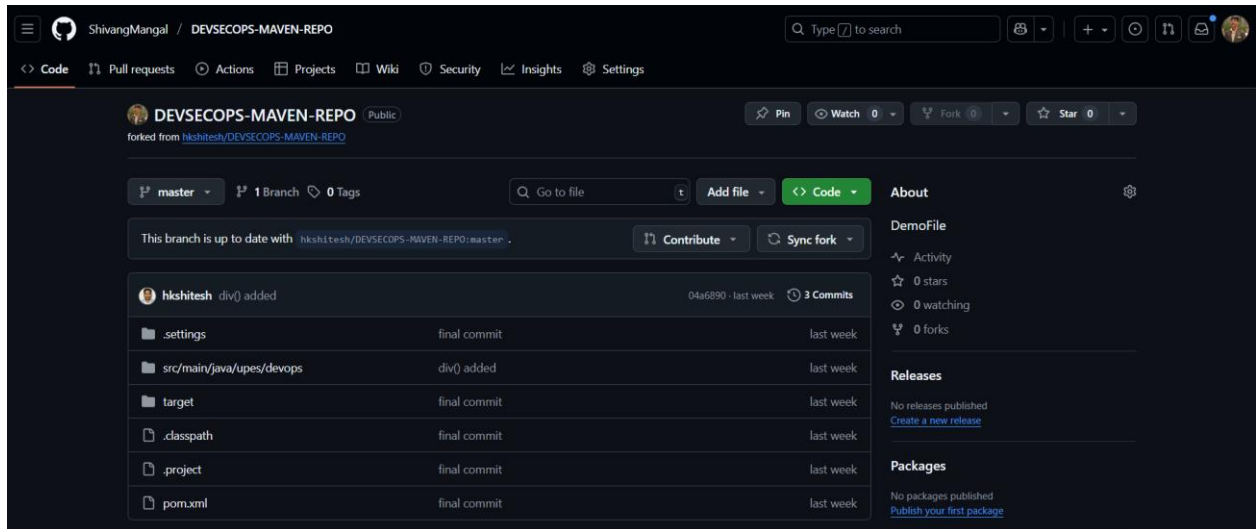
git.exe

☐ Install automatically ?

Add Git

Step 3: Fork a sample repository

3.1 Log in to your GitHub account, navigate to <https://github.com/jenkins-docs/simple-java-maven-app>, and click on **Fork**



3.2 Run `git clone` [Forked REPO URL] in the terminal to clone the repository locally

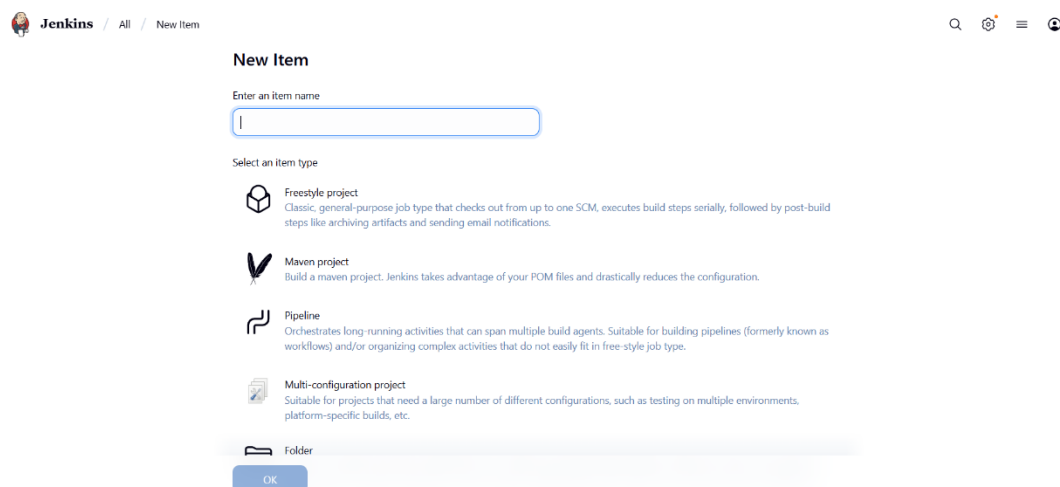
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

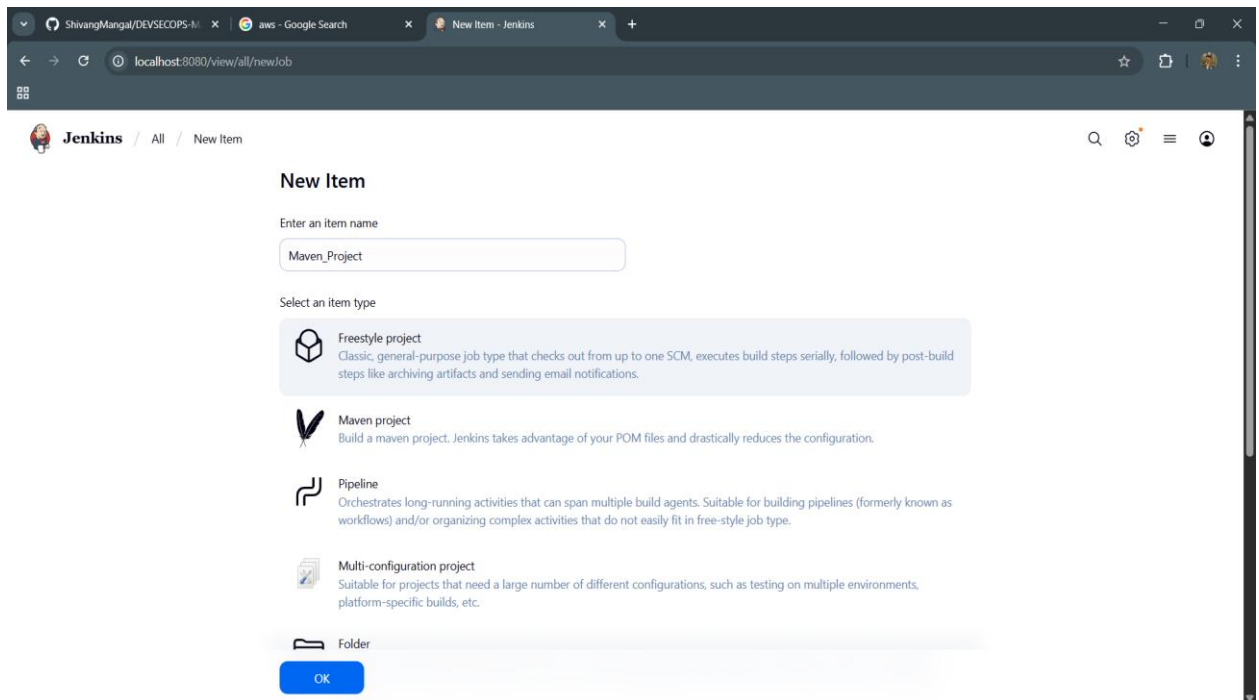
PS C:\Users\HP\Desktop\DevSecOps_Lab> git clone git@github.com:ShivangMangal/DEVSECOPS-MAVEN-REPO.git
Cloning into 'DEVSECOPS-MAVEN-REPO'...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 32 (delta 4), reused 28 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (32/32), 5.17 KiB | 755.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.
PS C:\Users\HP\Desktop\DevSecOps_Lab> |
```

Step 4: Integrate Maven with Jenkins

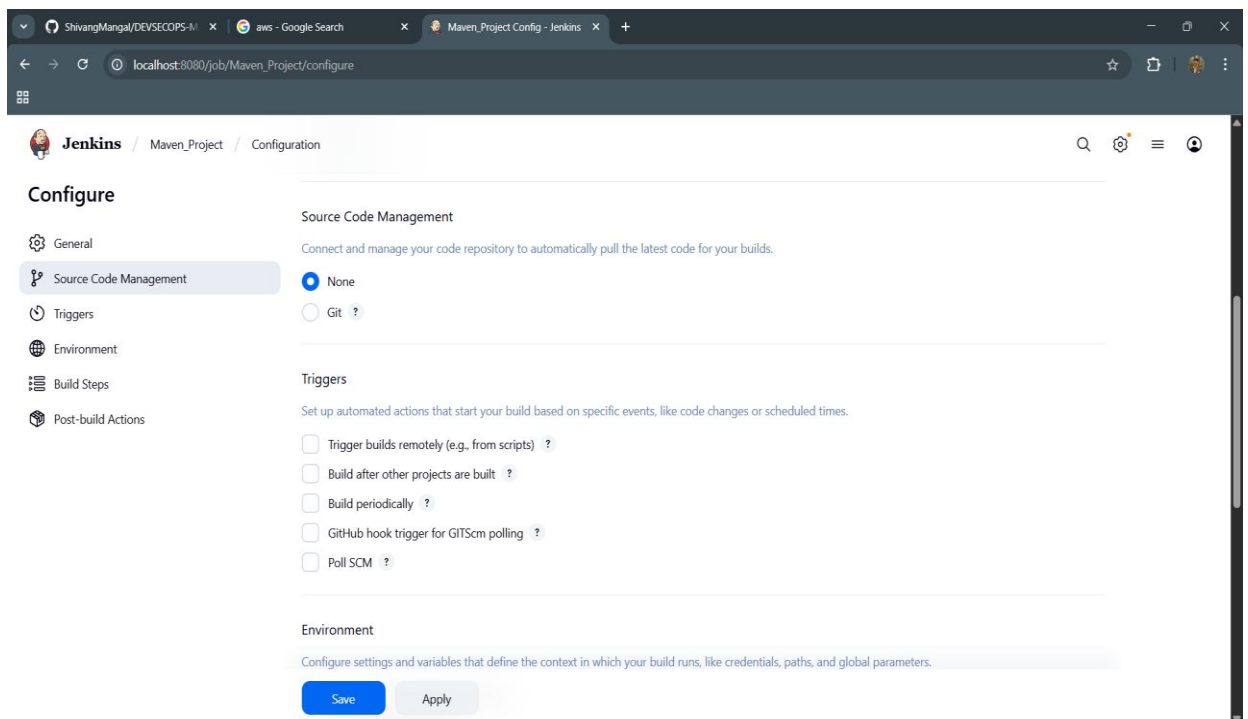
4.1 Click on **New Item** in the Jenkins Dashboard



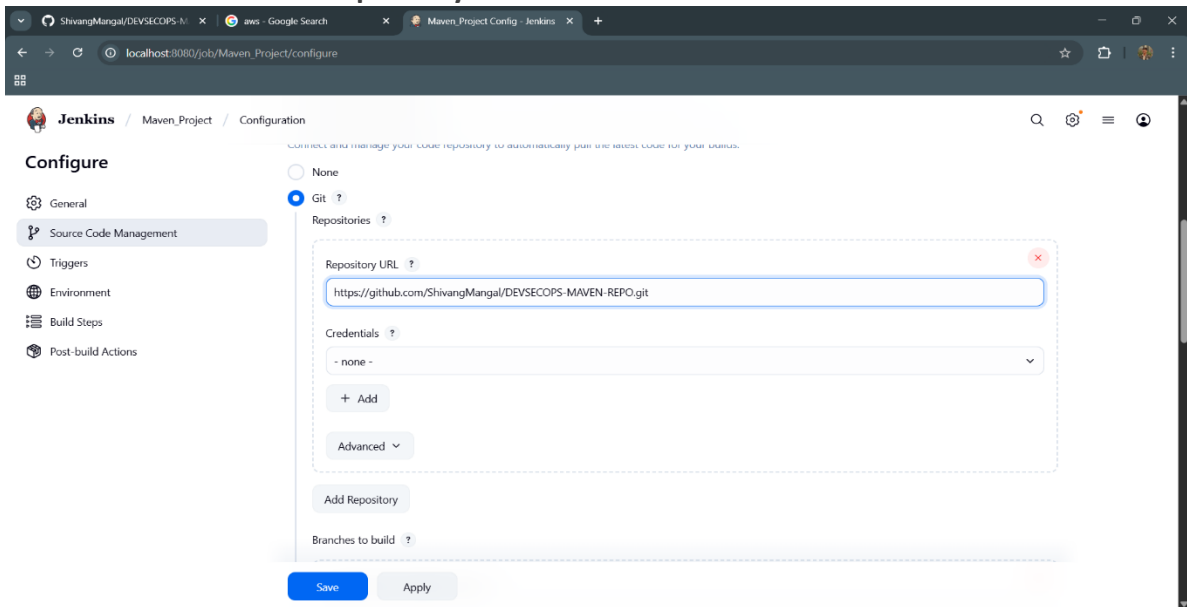
4.2 Enter a name for the project, select **Freestyle project** as the build job type, and click on the **OK** button as shown in the screenshot below:



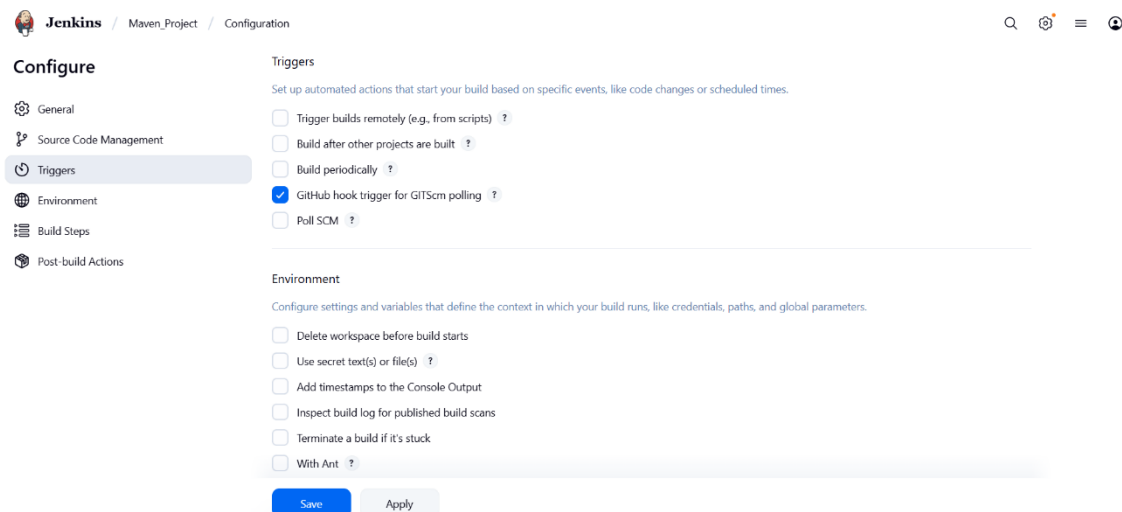
4.3 Click on **Source Code Management**



4.4 Select **Git** and enter the **Repository URL**



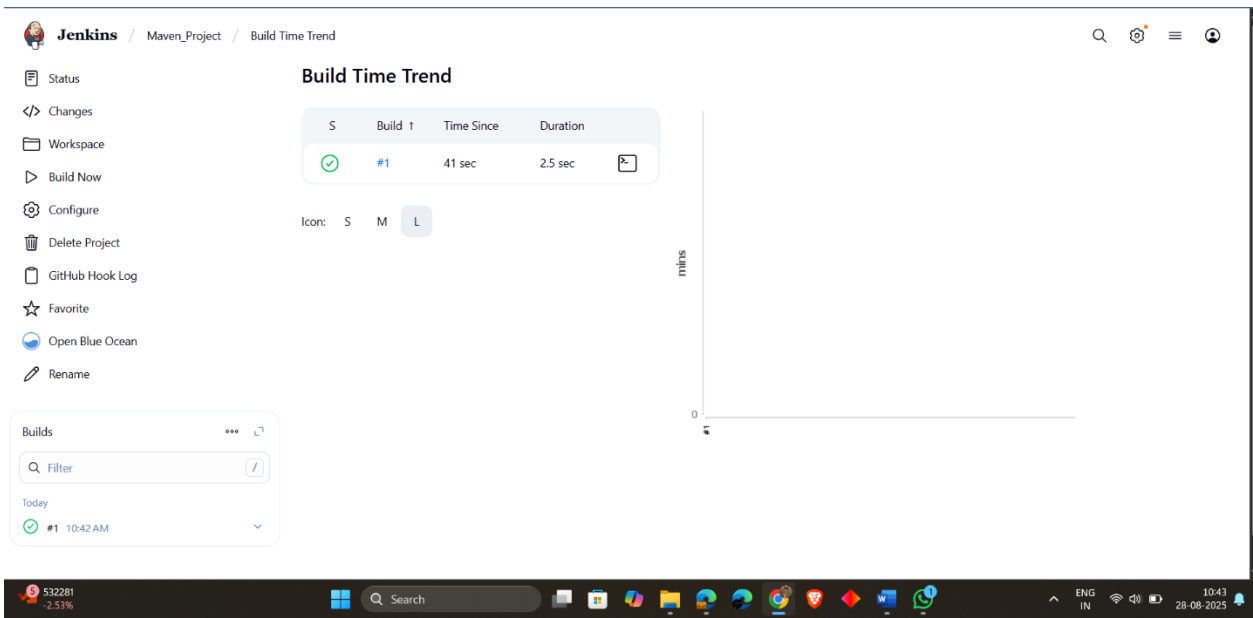
4.5 Click on **Build Triggers**, select the required option as shown in the screenshot below, and then click on **Save**



4.6 Click on **Build Now** to view the build results



4.7 Click on **trend** in the **Build History** as shown in the screenshot below:



4.8 Click on **Status** to view the build logs

The screenshot shows the Jenkins 'Maven_Project' Status page. The left sidebar is the same as in the previous screenshot. The main content area is titled 'Maven_Project' and has a green checkmark icon. Below the title is a section for 'Permalinks' with a list of links: 'Last build (#1), 1 min 17 sec ago', 'Last stable build (#1), 1 min 17 sec ago', 'Last successful build (#1), 1 min 17 sec ago', and 'Last completed build (#1), 1 min 17 sec ago'. At the bottom left, there is a 'Builds' section with a filter input and a list of builds for 'Today', showing build #1 at 10:42 AM. The bottom of the image shows a Windows taskbar with various application icons and system tray information.

By following these steps, you have successfully installed the Maven plugin in Jenkins, making it easier to automate Maven-based build tasks within the Jenkins environment for smoother integration and workflow automation.