

## 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

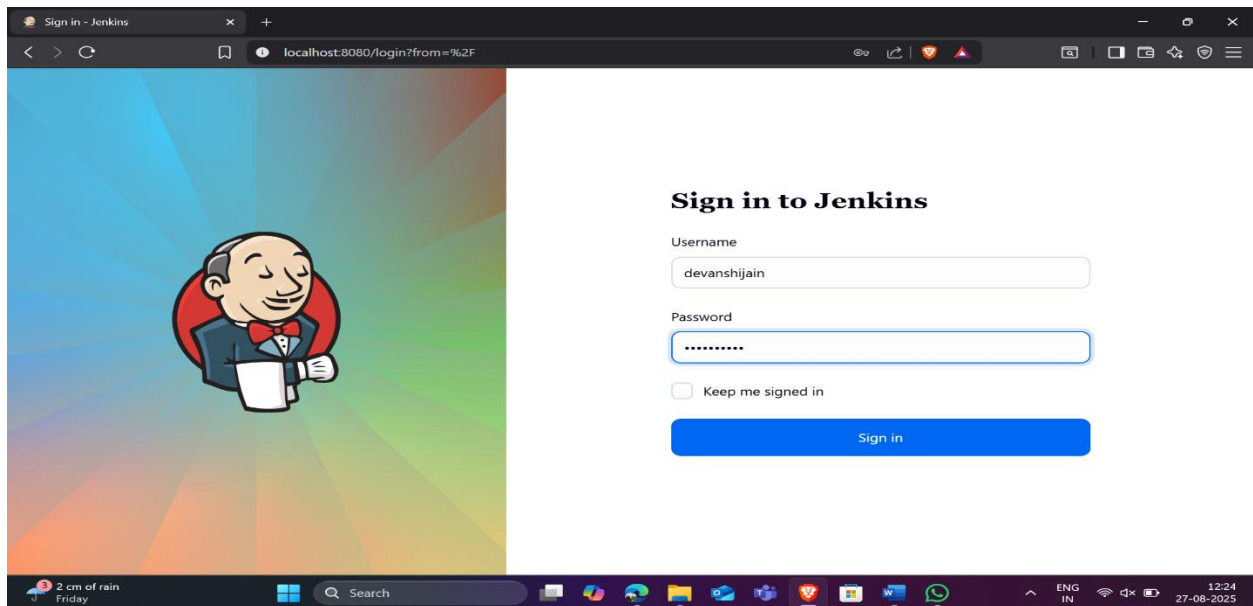
**Prerequisites:** None

Steps to be followed:

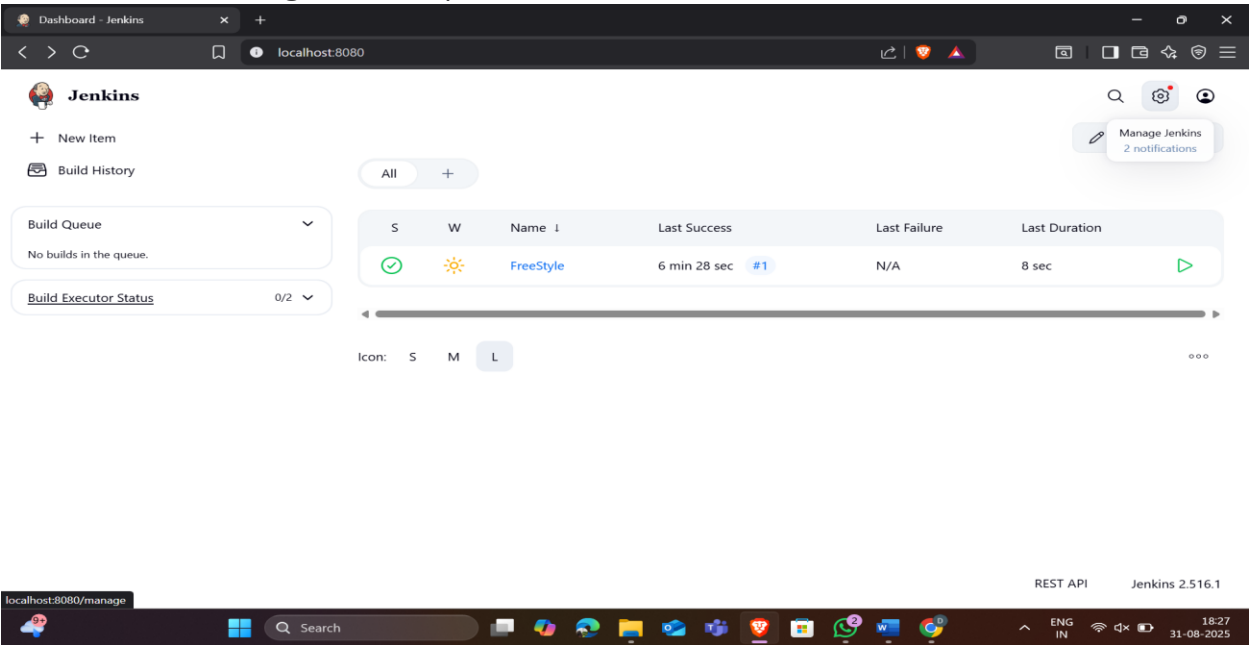
1. Install the Maven plugin
2. Set up Global Tool Configuration
3. Fork a sample repository
4. Integrate Maven with Jenkins

#### Step 1: Install the Maven plugin

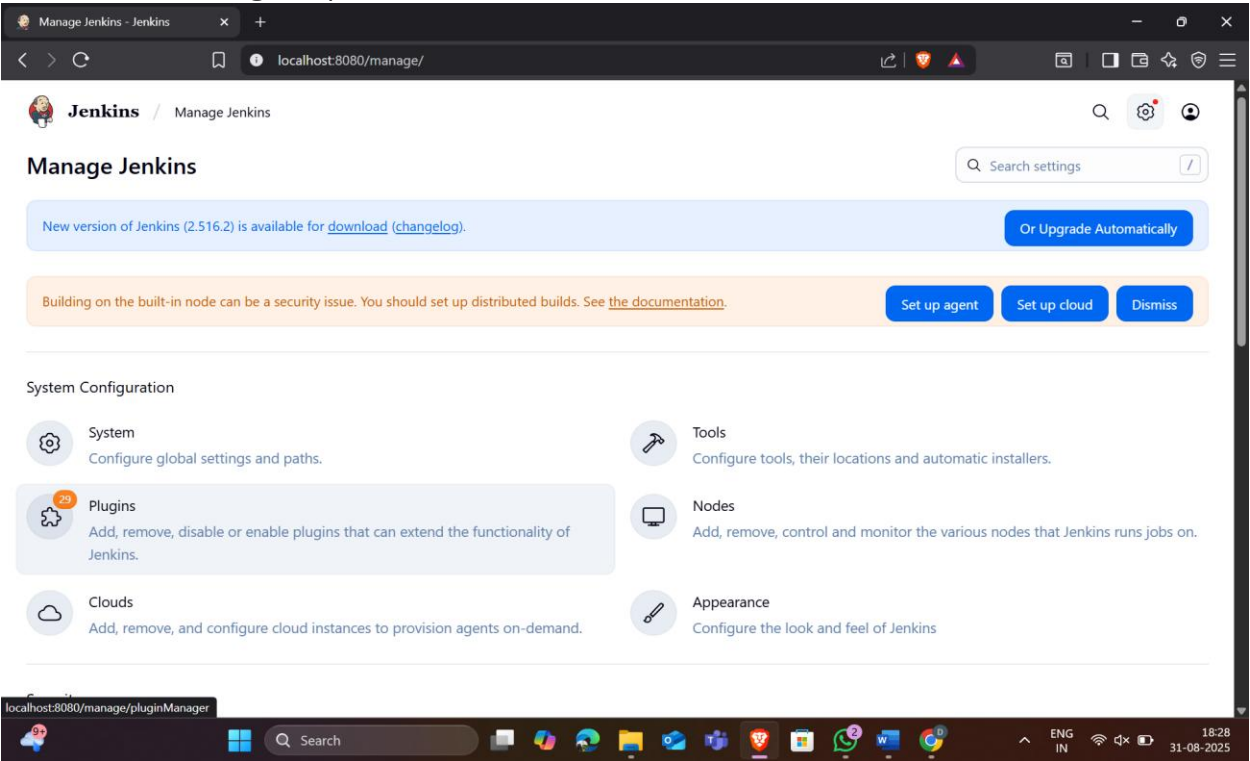
- 1.1 Open the browser, go to the Jenkins Dashboard by typing **localhost:8080** in your browser, provide the credentials, and click the **Sign in** button



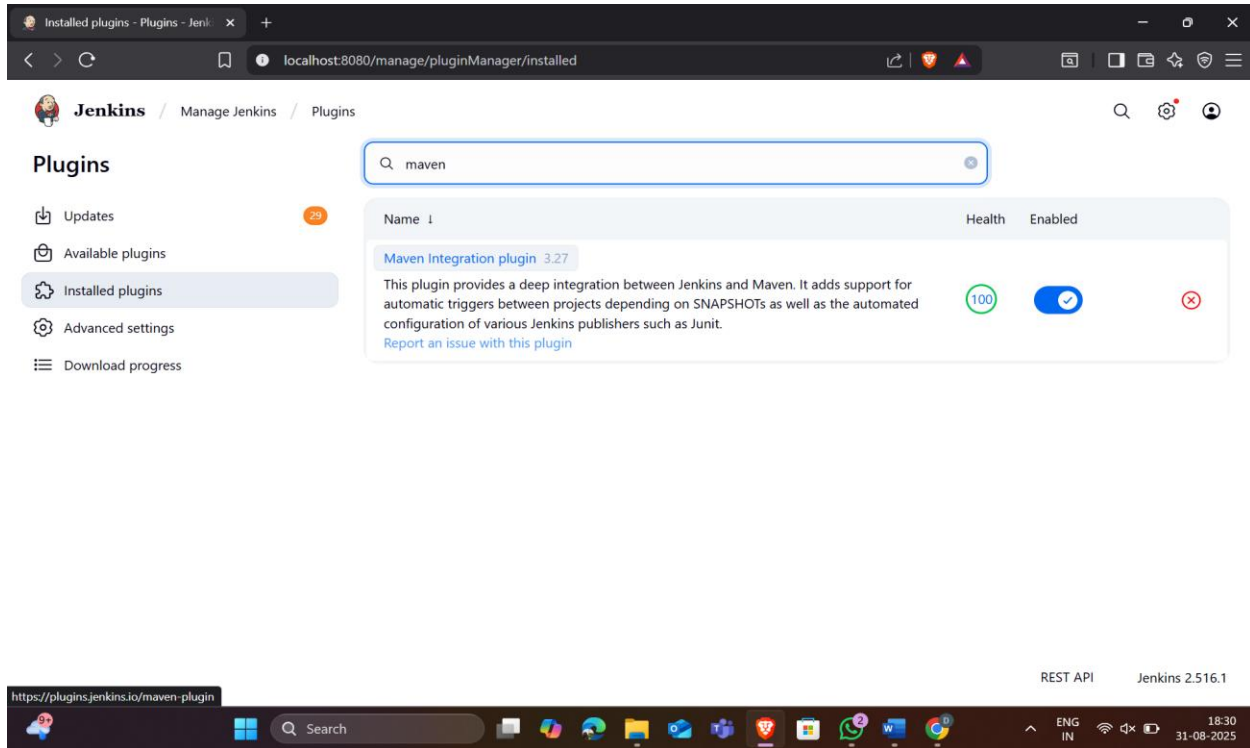
1.2 Click on the **Manage Jenkins** option as shown in the screenshot below:



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



The screenshot shows the Jenkins web interface. The browser address bar indicates the URL is `localhost:8080/manage/pluginManager/installed`. The Jenkins header shows the logo and navigation links for 'Manage Jenkins' and 'Plugins'. On the left sidebar, the 'Plugins' section is active, with sub-links for 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress'. The main content area displays a search bar with 'maven' entered. Below the search bar, a table lists installed plugins. The first entry is the 'Maven Integration plugin' version 3.27. The description states: 'This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTS as well as the automated configuration of various Jenkins publishers such as Junit.' The 'Health' column shows a green circle with '100', and the 'Enabled' column shows a blue toggle switch. A red 'X' icon is visible in the rightmost column. The footer of the Jenkins interface shows 'REST API' and 'Jenkins 2.516.1'. The Windows taskbar at the bottom shows the system clock as 18:30 on 31-08-2025.

**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.

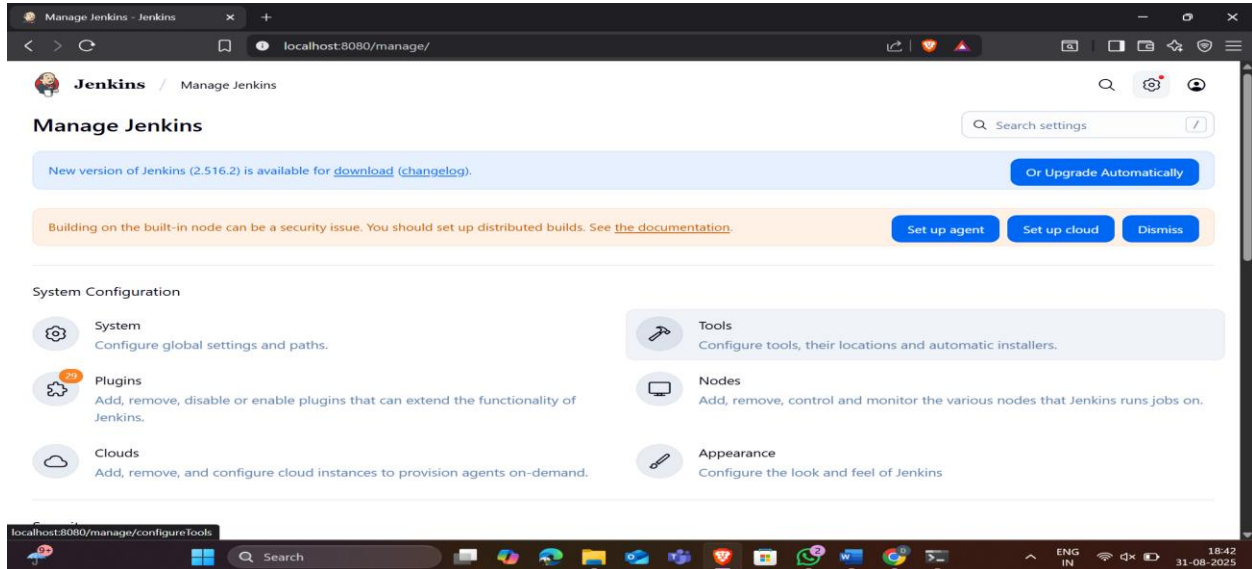
1.5 Use the following command to check the Maven version:

**mvn -version**

```
PS C:\Users\Devanshi> mvn --version
Apache Maven 3.9.9 (8e8579a9e76f7d015ee5ec7bfcdc97d260186937)
Maven home: C:\Users\Devanshi\Downloads\maven-mvnd-1.0.2-windows-amd64\maven-mvnd-1.0.2-windows-amd64\mvn
Java version: 21.0.7, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-21
Default locale: en_IN, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
PS C:\Users\Devanshi>
```

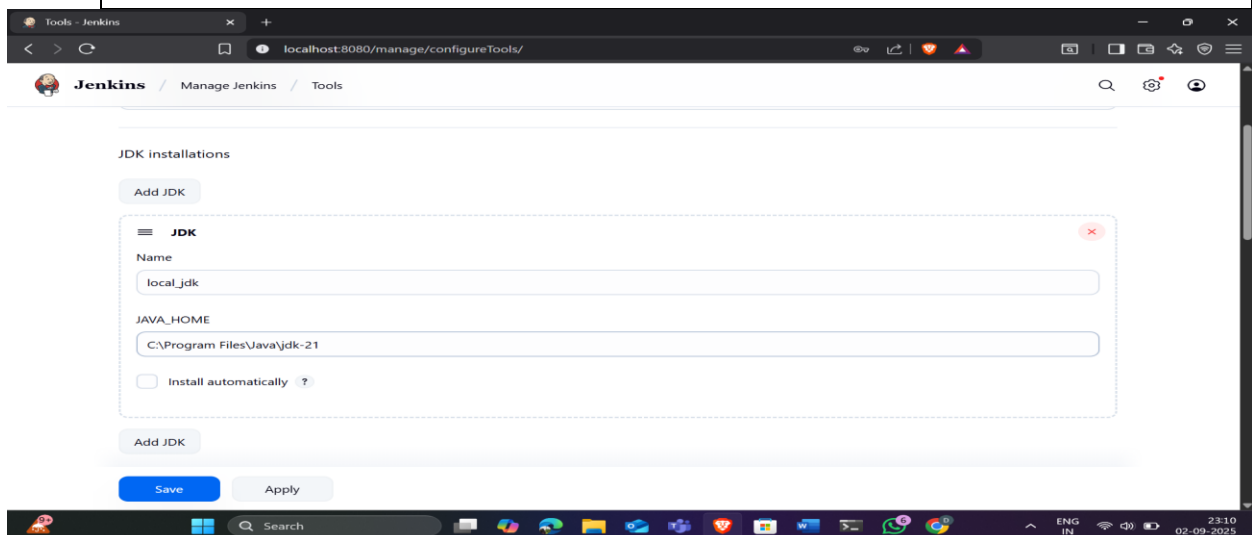
## Step 2: Set up Global Tool Configuration

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



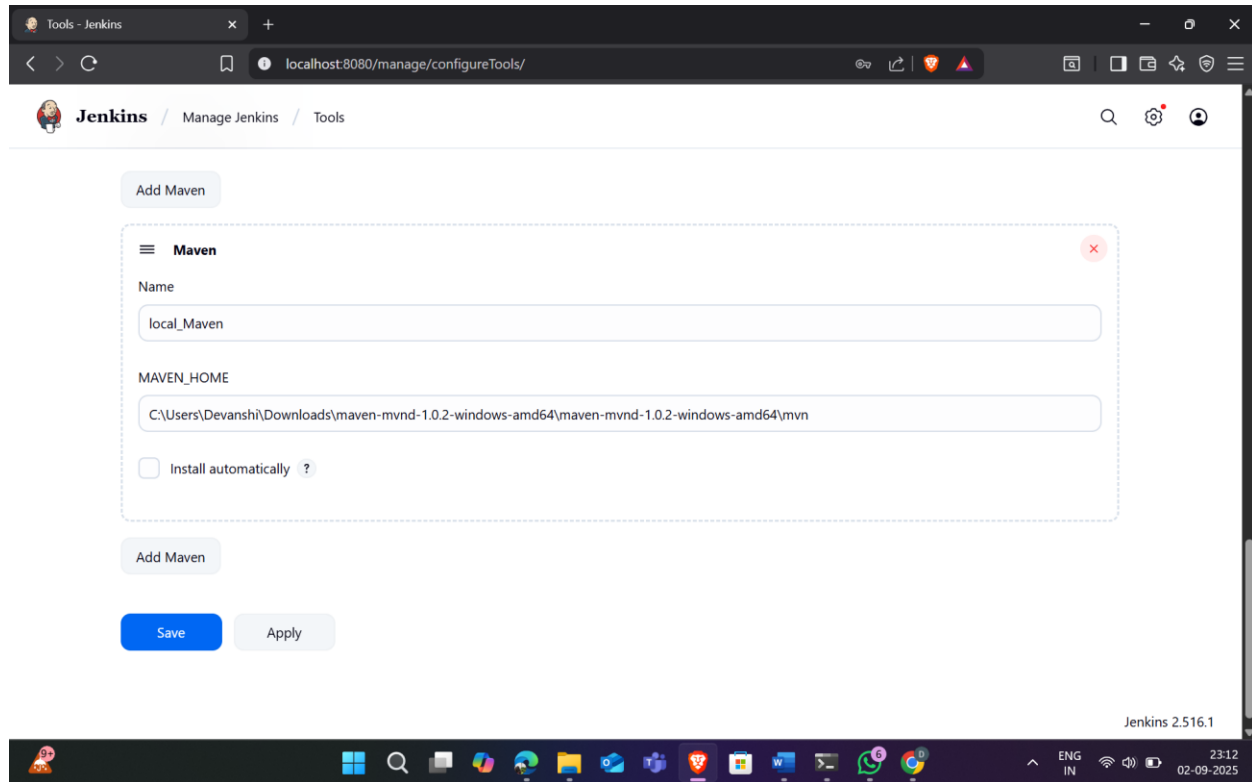
2.2 Click on **JDK installations** and provide the **Name** and **JAVA\_HOME** path

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



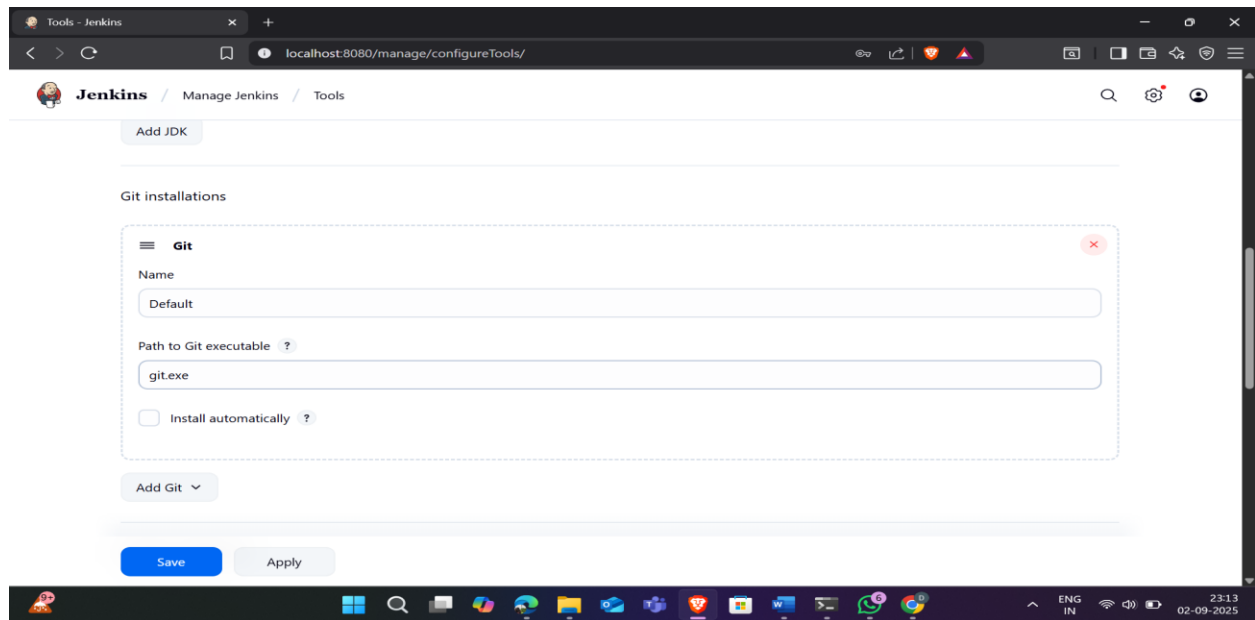
2.3 To configure Maven, click on the **Maven installations** button in the Maven section and enter a **Name** and **MAVEN\_HOME** path

**Note:** Set the **MAVEN\_HOME** environment variable to **/usr/share/maven**



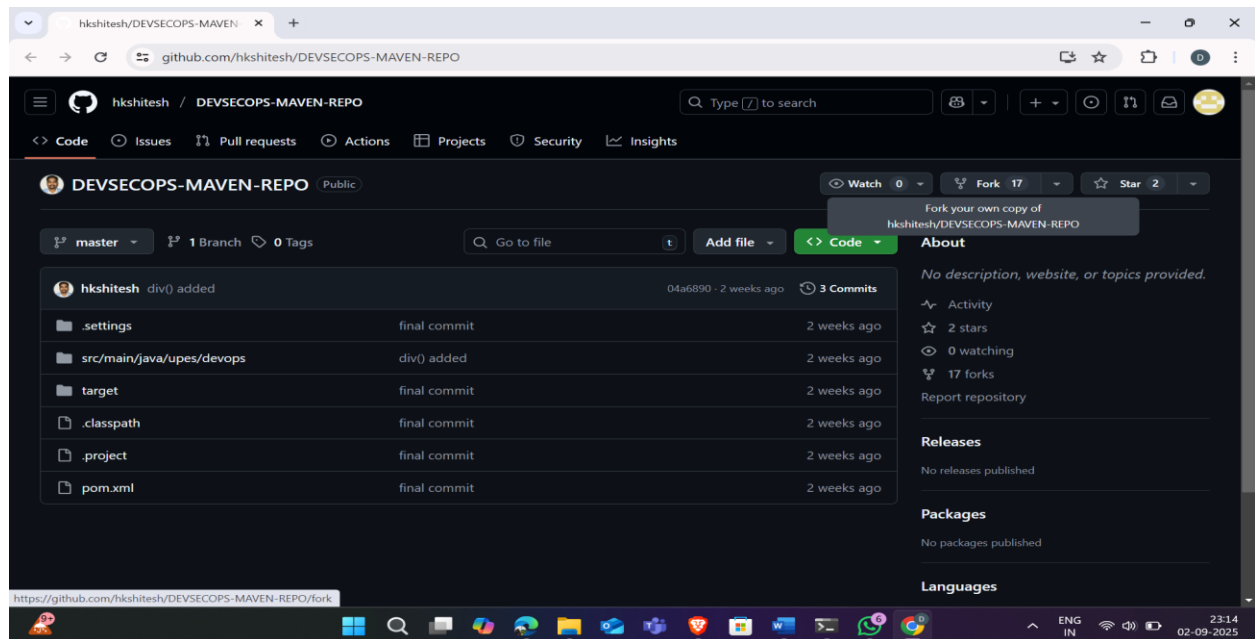
2.4 To configure Git, click on **Git installations** and add the **Name** and **Path to Git executable**

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



### 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

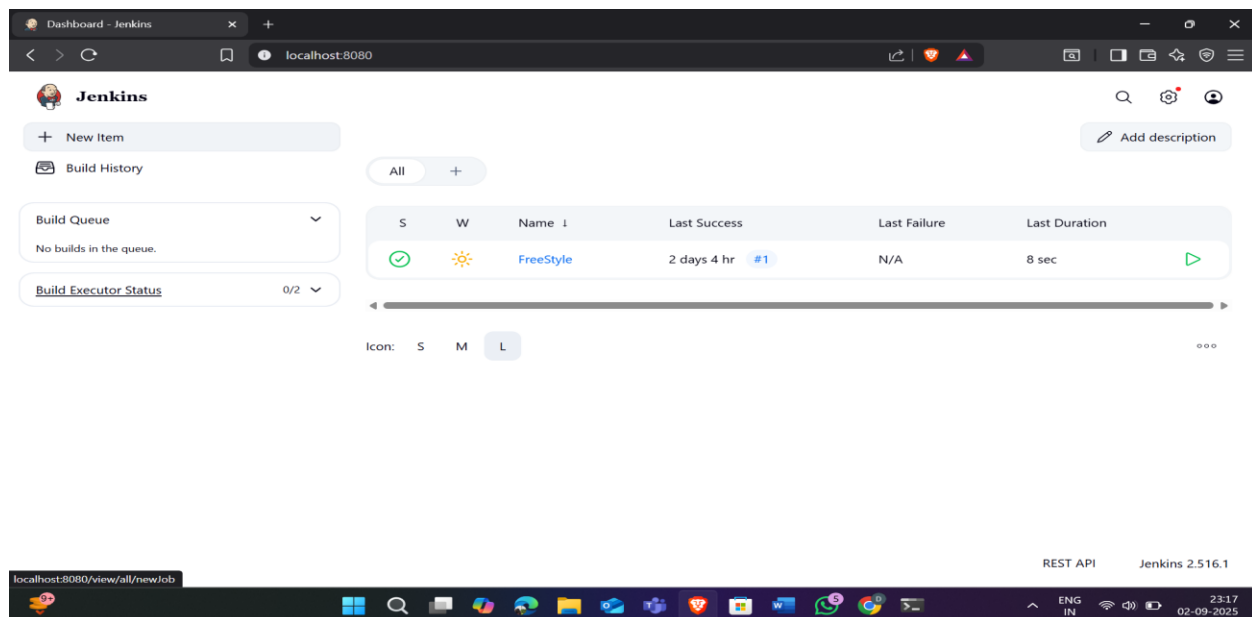
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

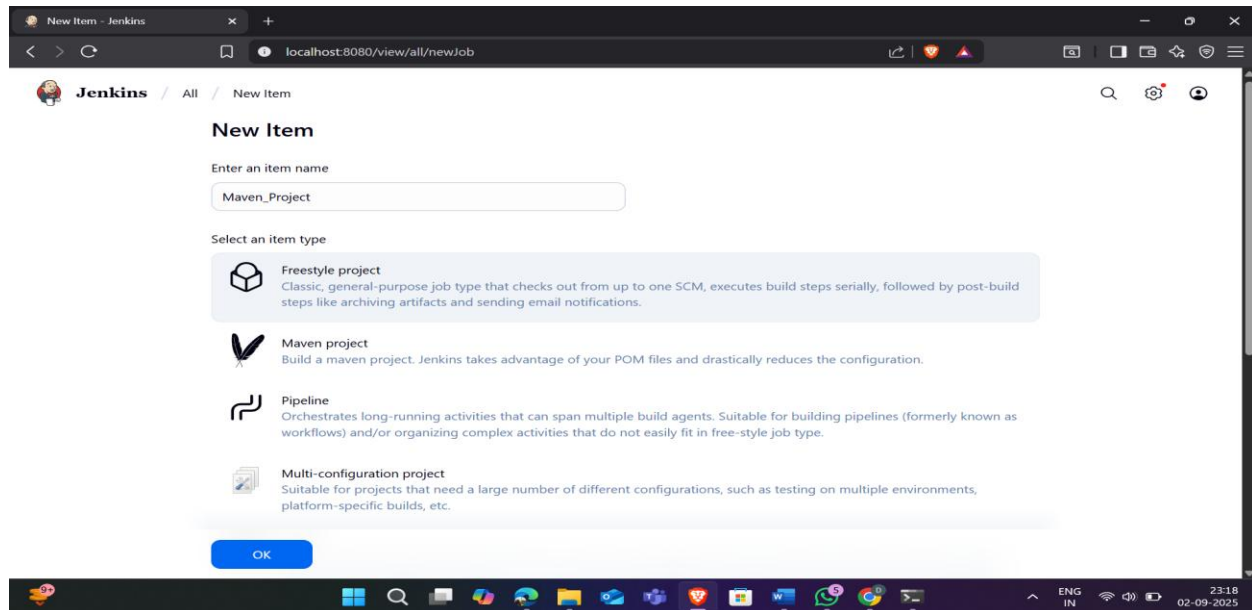
PS C:\Users\Devanshi> git clone https://github.com/Devanshii-git/DEVSECOPS-MAVEN-REPO.git
Cloning into 'DEVSECOPS-MAVEN-REPO'...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 32 (delta 4), reused 28 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (32/32), 5.17 KiB | 188.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.
PS C:\Users\Devanshi> |
```

## Step 4: Integrate Maven with Jenkins

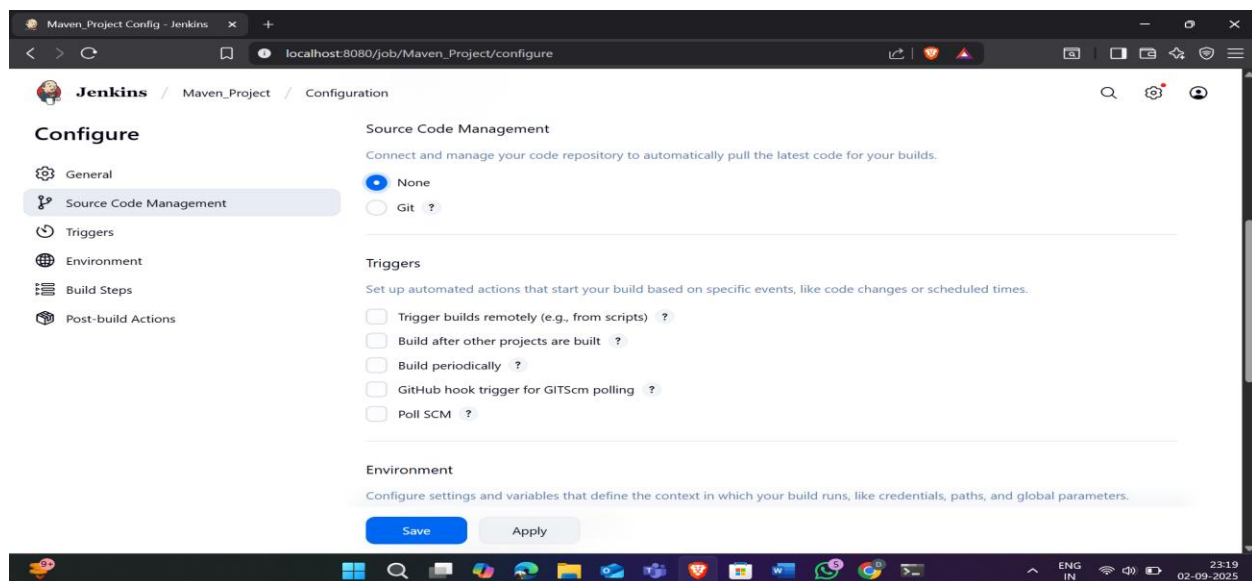
### 4.1 Click on **New Item** in the Jenkins Dashboard



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:

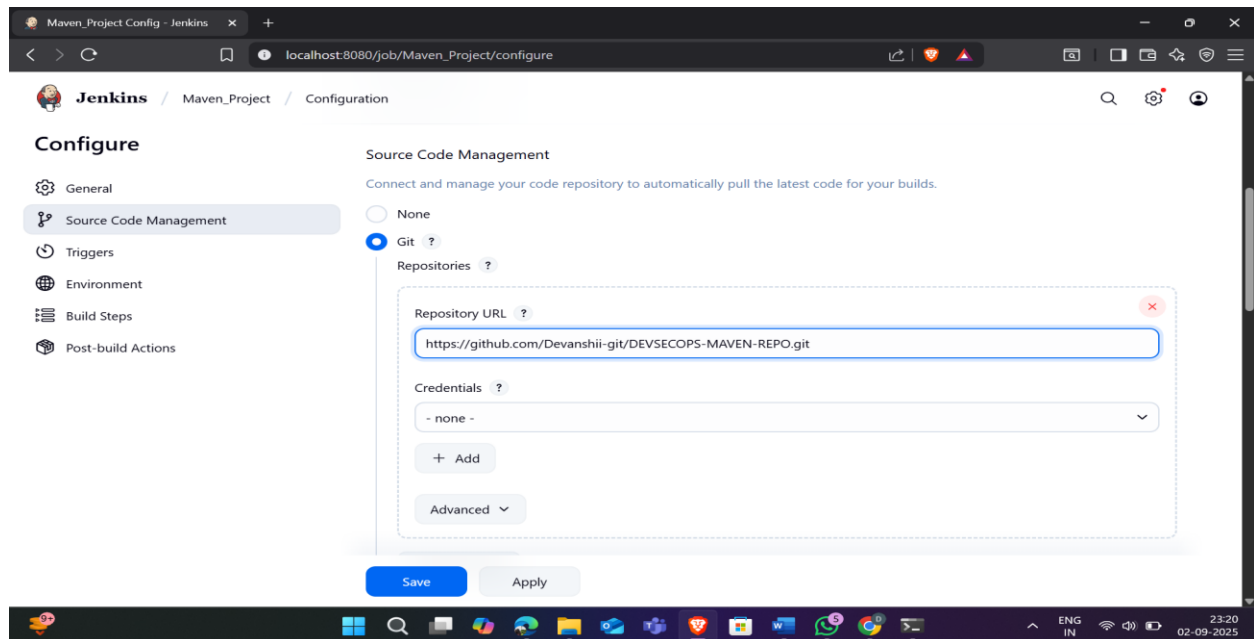


4.3 Click on **Source Code Management**

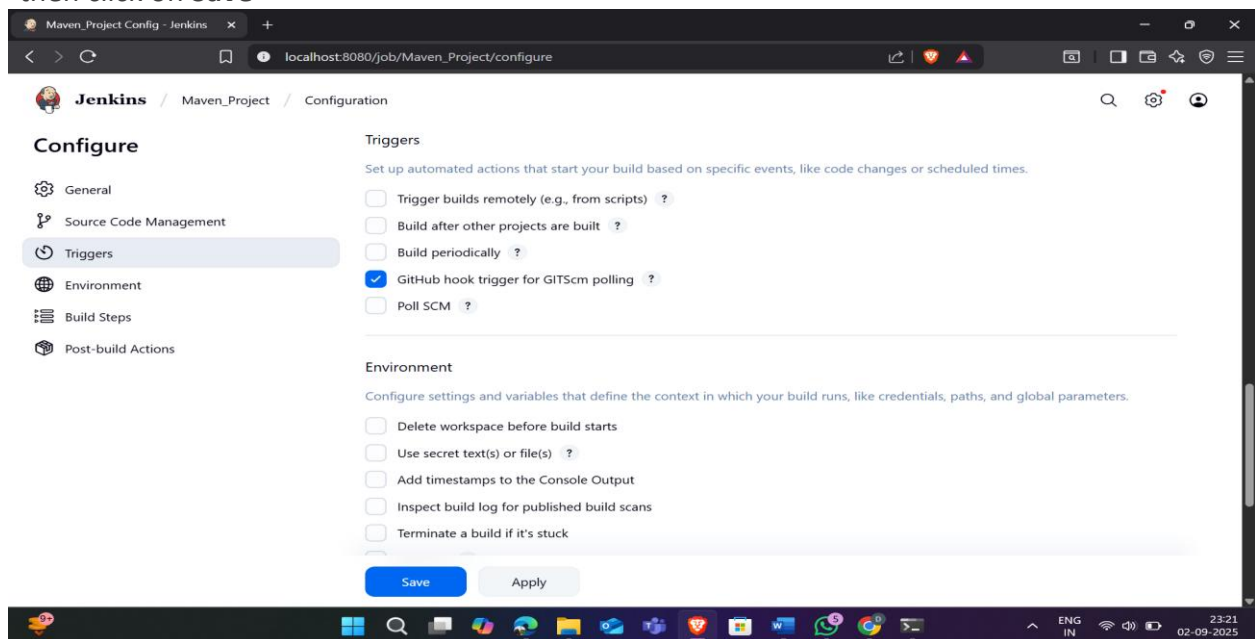




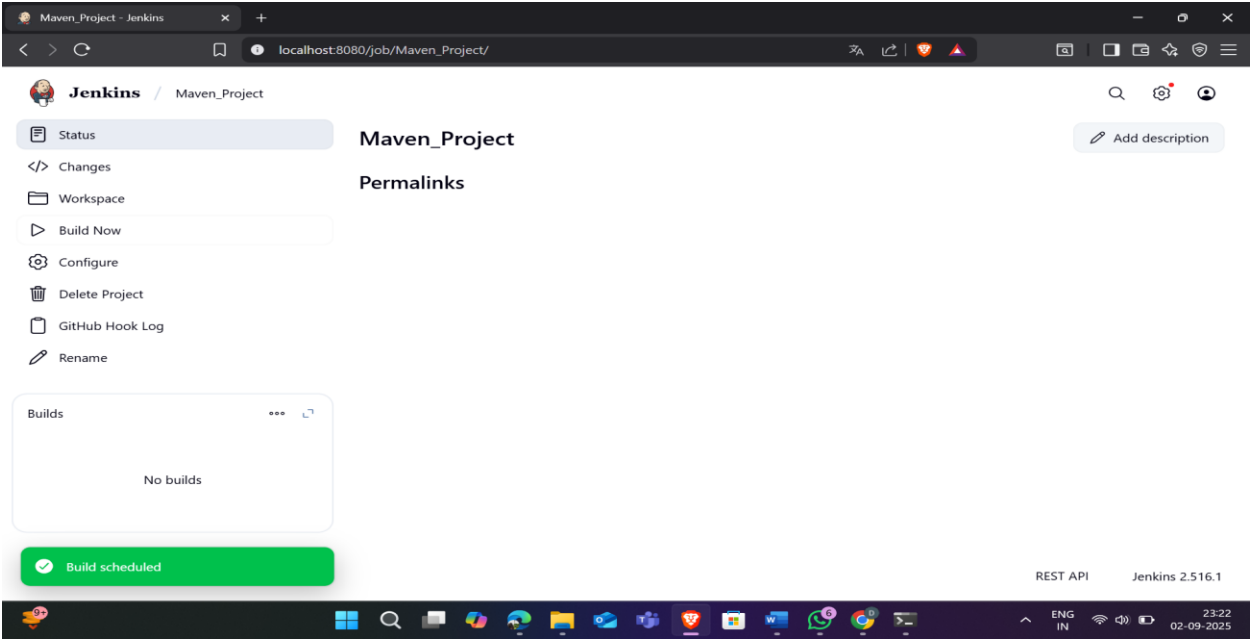
#### 4.4 Select **Git** and enter the **Repository URL**



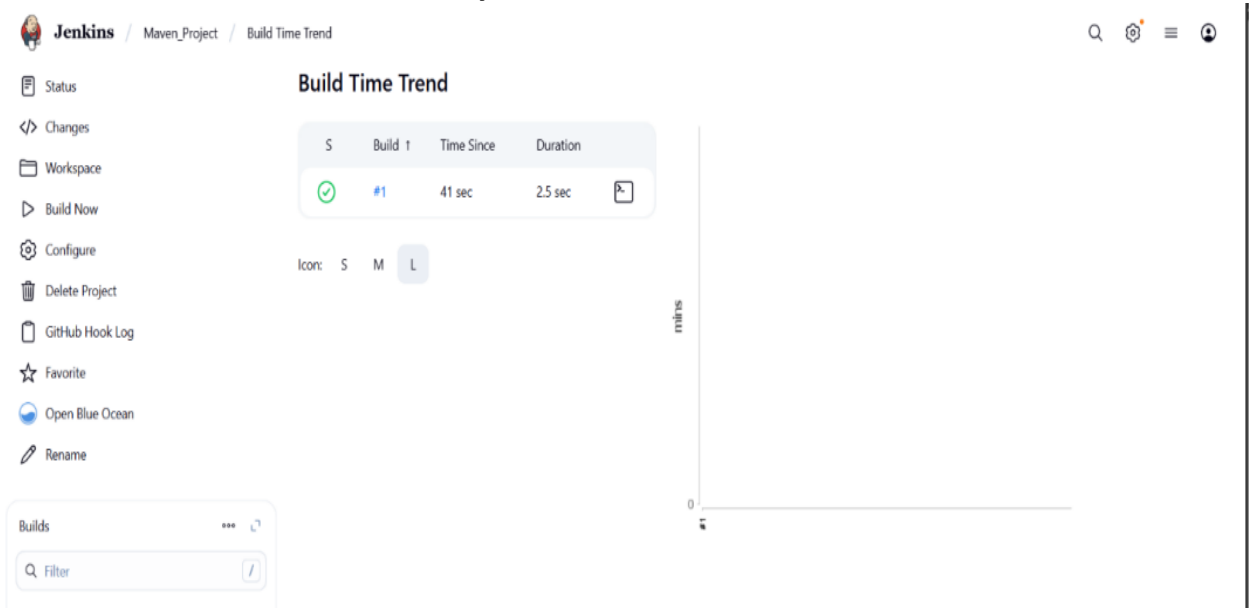
#### 4.5 Click on **Build Triggers**, select the required option as shown in the screenshot below, and then click on **Save**



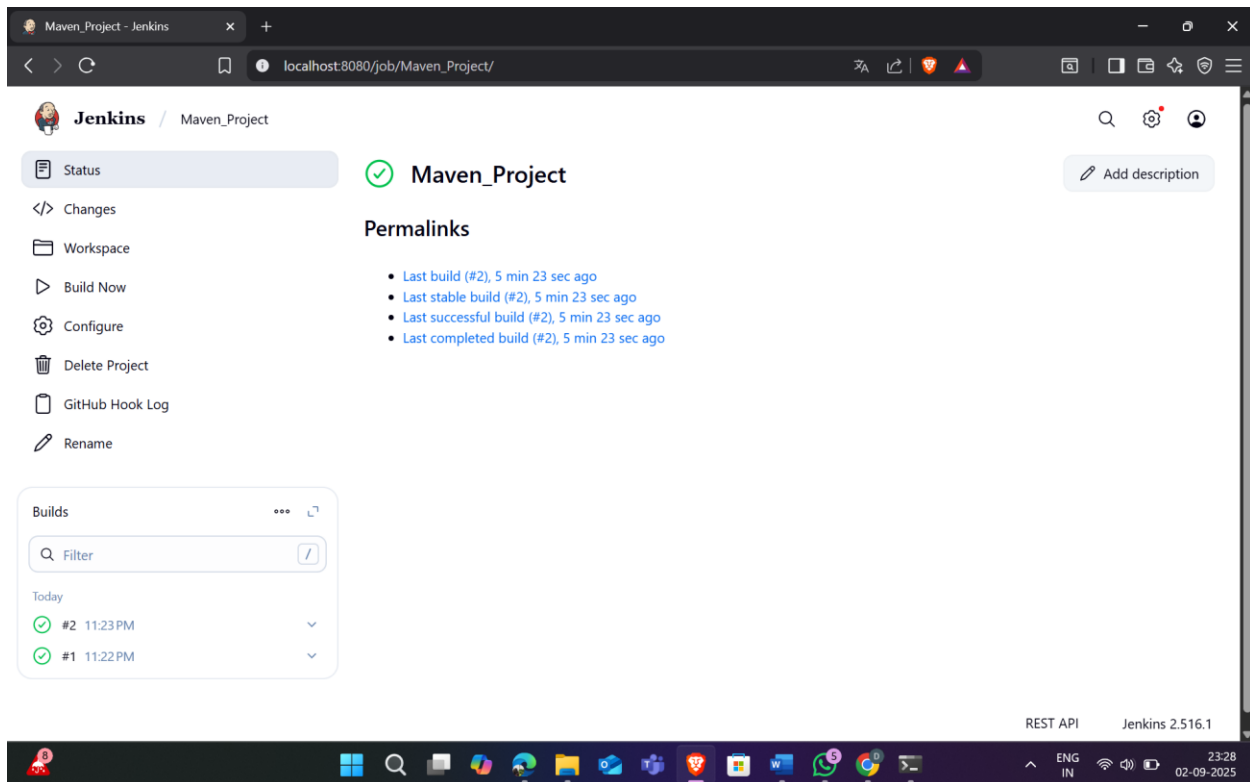
4.6 Click on **Build Now** to view the build results



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



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.