Lesson 5 Lesson-End Project

Building a Maven Project with Jenkins

Project Agenda: Use Jenkins to configure and set up a foundation for Maven builds with the help of GitHub

Description: As a DevOps engineer at a leading tech firm, you've been assigned to streamline the development process by building a Maven project using Jenkins. This initiative aims to centralize project management, automate build processes, and facilitate seamless collaboration among developers and stakeholders.

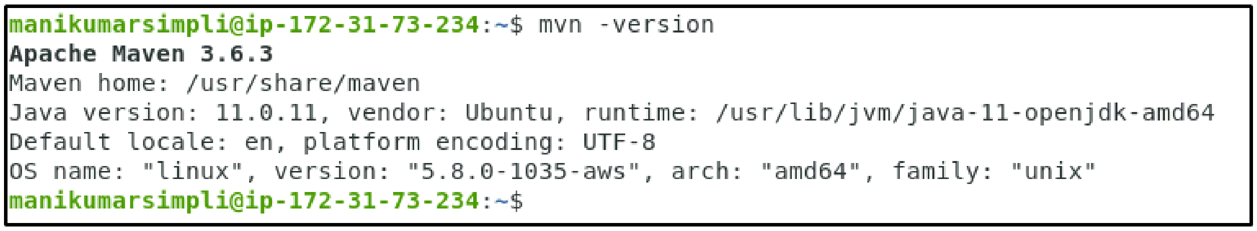
Tools required: Git, GitHub, and Jenkins

Expected Deliverables: Build a Maven project involving Maven goals and execute within the Jenkins environment.

Steps to be followed:
1. Install the Jenkins Maven Integration plugin
2. Configure Maven in Jenkins
3. Fork a sample repository
4. Create a freestyle project
5. Configure the build
6. Build and view the console output

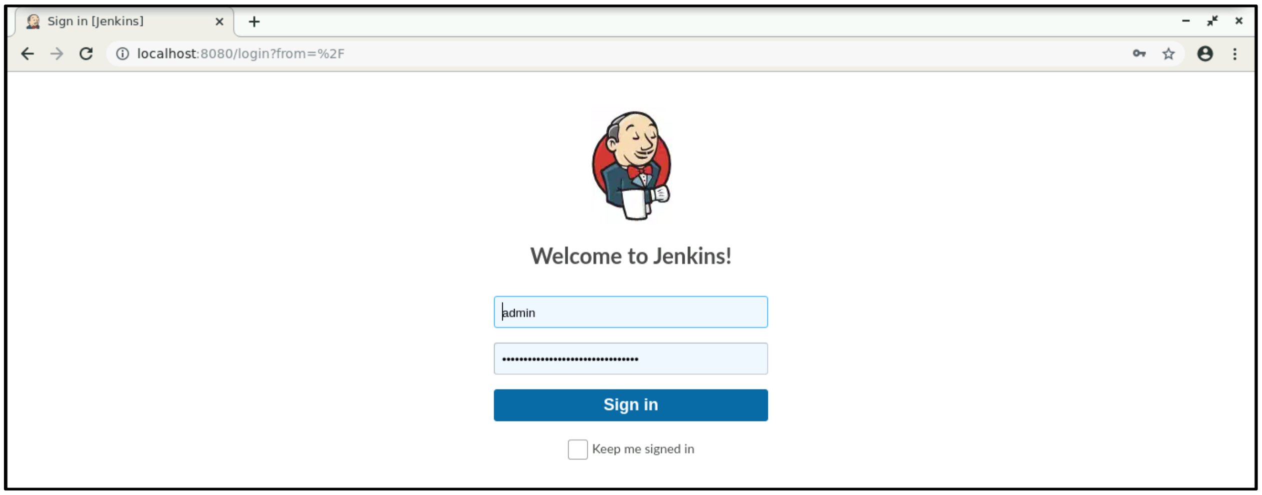
Step 1: Install the Jenkins Maven Integration plugin

1.1 Open the terminal, and if Maven is not installed, run **sudo apt-get install maven**, then verify the installation with **mvn -version**

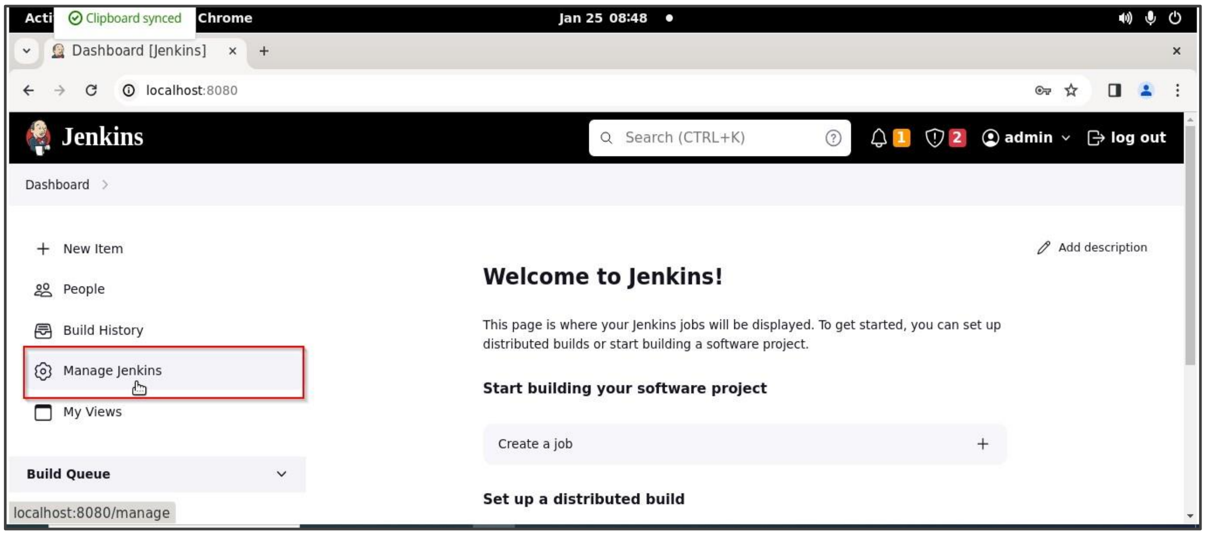


```
manikumarsimpli@ip-172-31-73-234:~$ mvn -version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.11, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "5.8.0-1035-aws", arch: "amd64", family: "unix"
manikumarsimpli@ip-172-31-73-234:~$
```

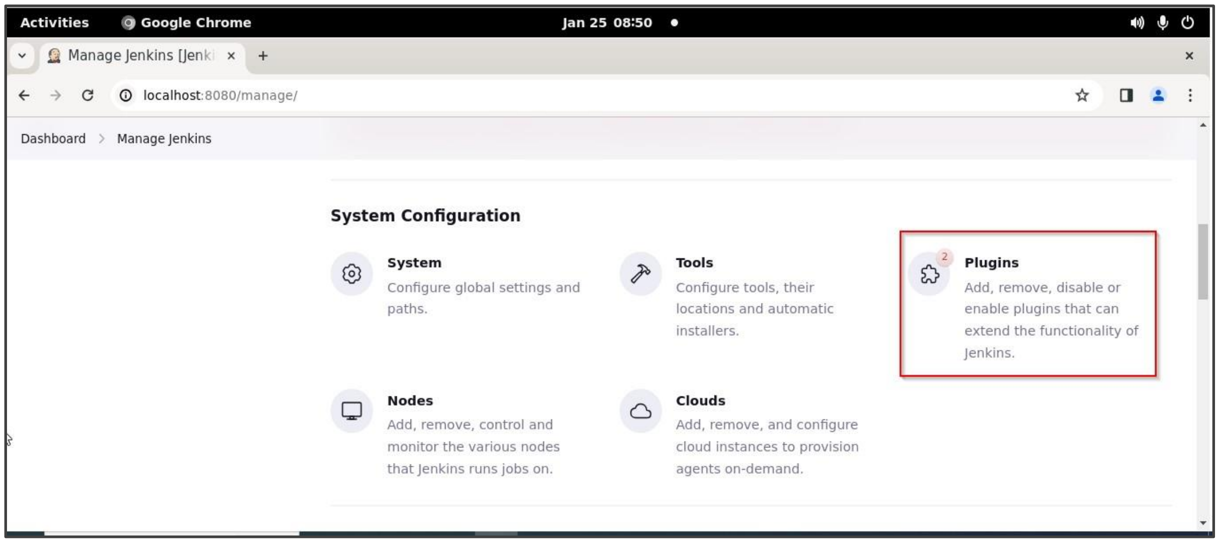
1.2 Launch your web browser and access Jenkins by entering **localhost:8080** in the address bar to sign in



1.3 Click on **Manage Jenkins** in the left panel of the Jenkins Dashboard


Dashboard [Jenkins]
localhost:8080
Jenkins
Search (CTRL+K)
admin
log out
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Build Queue
localhost:8080/manage
Add description
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build

1.4 Select **Plugins**



System Configuration

System
Configure global settings and paths.

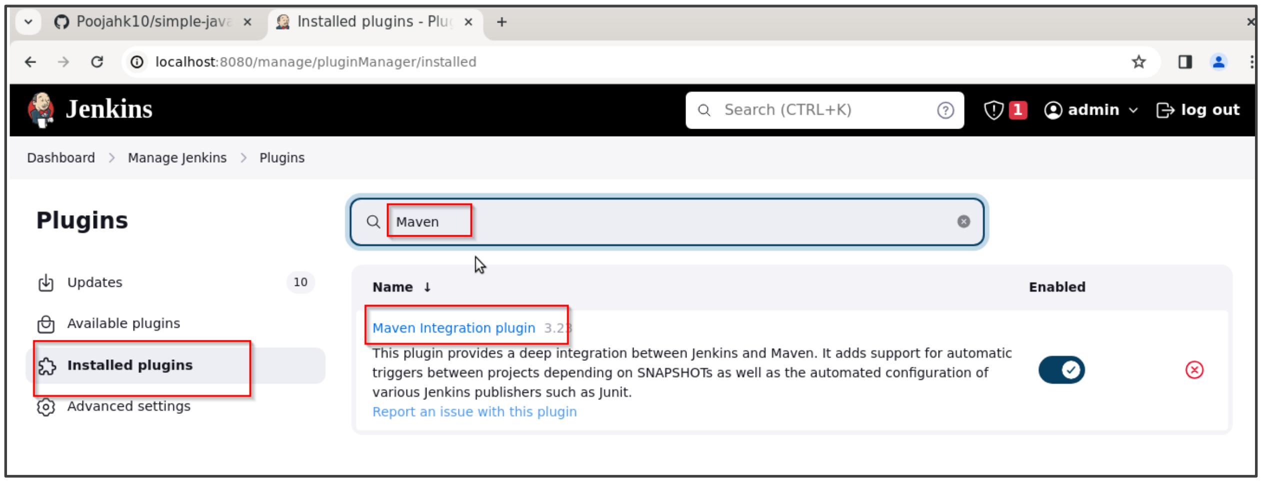
Tools
Configure tools, their locations and automatic installers.

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.

1.5 Click on **Installed plugins** to verify the installation of the **Maven Integration plugin**

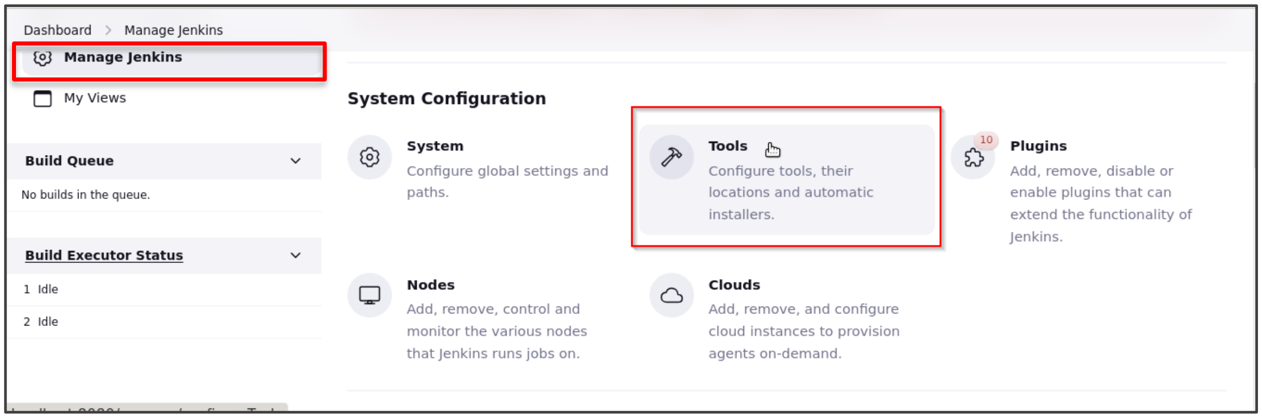


Maven Integration plugin 3.23
This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit.
Report an issue with this plugin

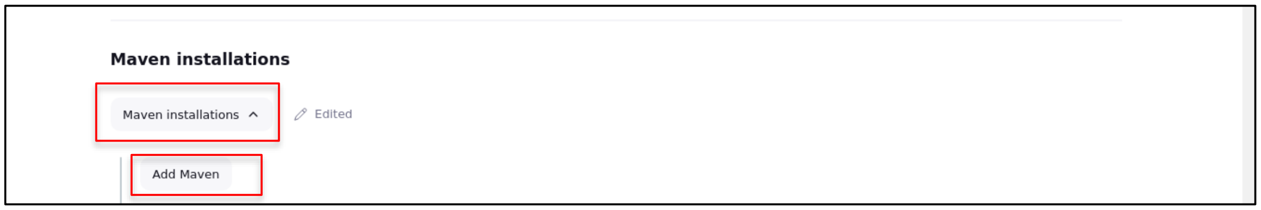
Note: Maven is already installed in your practice lab environment. If not, click on **Available plugins**, search for it, and install it.

Step 2: Configure Maven in Jenkins

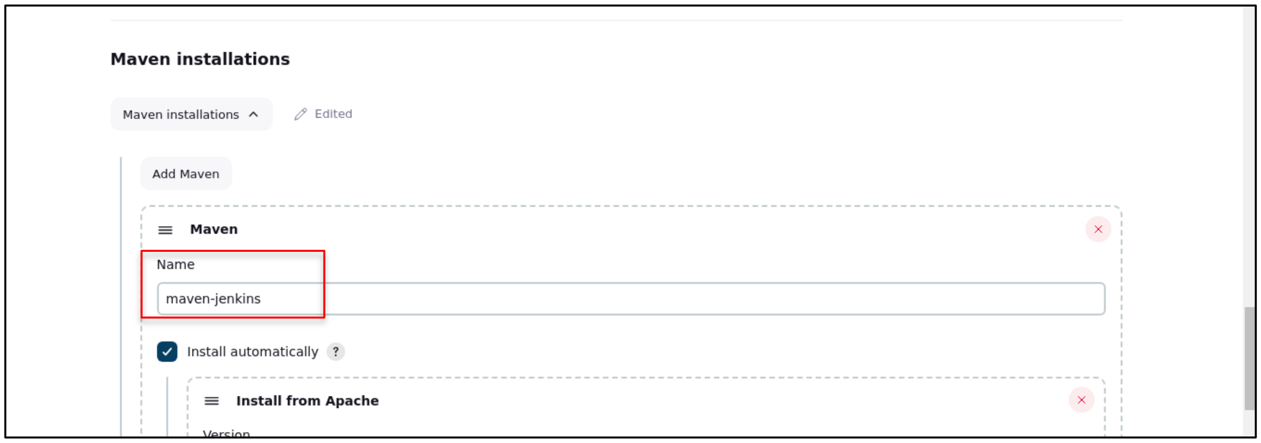
2.1 Go to the Jenkins Dashboard, click on **Manage Jenkins**, and then select **Tools** from the list of options


Dashboard > Manage Jenkins
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
System Configuration
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.

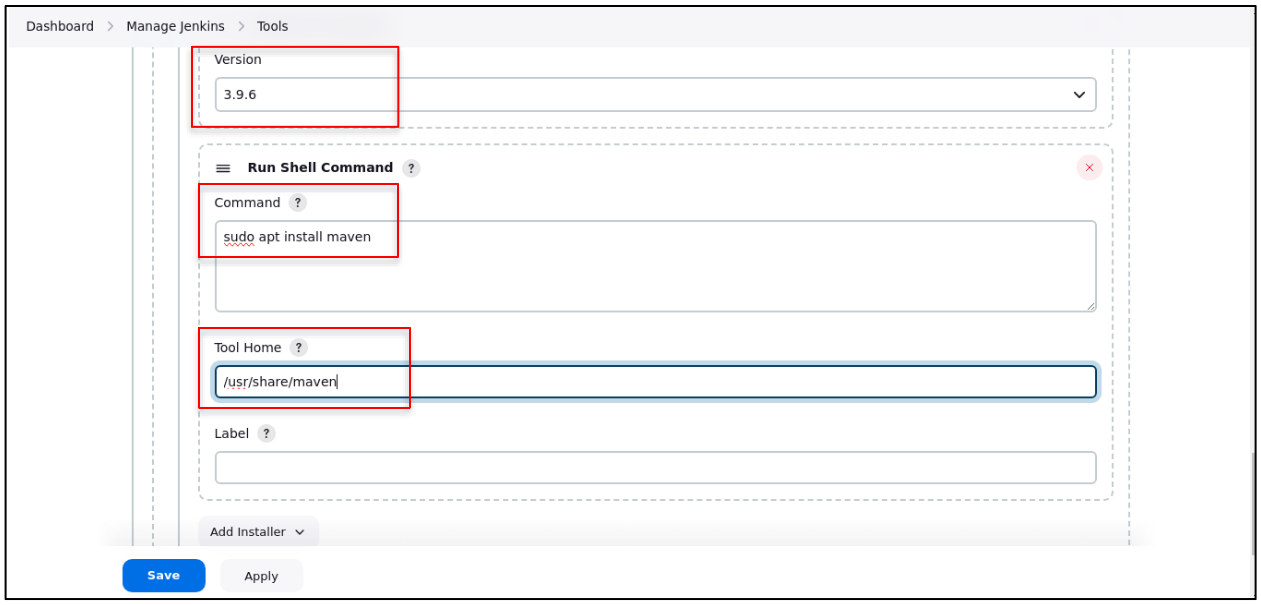
2.2 Scroll down to the Maven section, click on **Maven installations,** and then click on **Add Maven** to add a new installation



2.3 Provide an arbitrary name, specify the path to your Maven installation **/usr/share/maven**, and select the Maven version from the dropdown



Name: maven-jenkins


Dashboard > Manage Jenkins > Tools

Version: 3.9.6

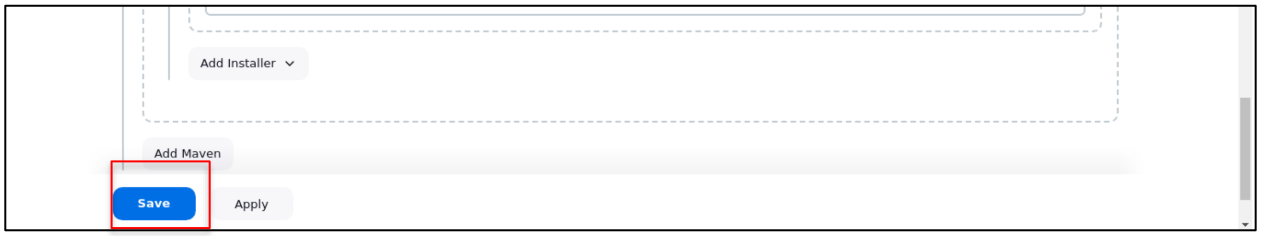
Run Shell Command

Command:
```
sudo apt install maven
```

Tool Home: /usr/share/maven

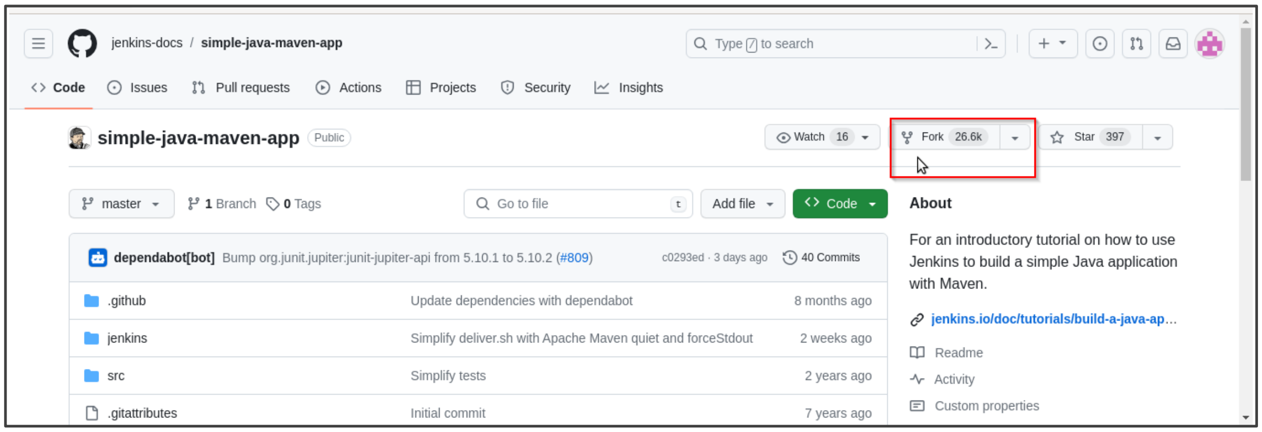
Label:

2.4 Click on **Save** to save the Maven installation configuration



Step 3: Fork a sample repository

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



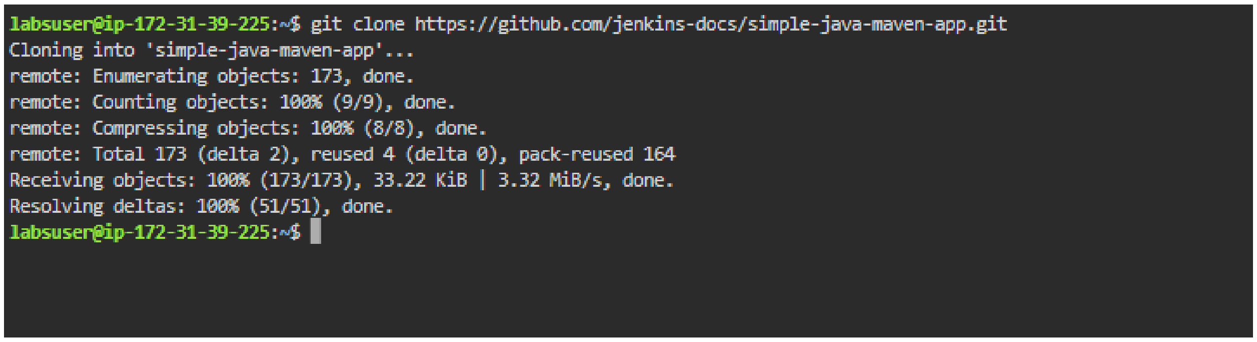
dependabot[bot] Bump org.junit.jupiter:junit-jupiter-api from 5.10.1 to 5.10.2 (#809)	c0293ed · 3 days ago	40 Commits
.github	Update dependencies with dependabot	8 months ago
jenkins	Simplify deliver.sh with Apache Maven quiet and forceStdout	2 weeks ago
src	Simplify tests	2 years ago
.gitattributes	Initial commit	7 years ago

For an introductory tutorial on how to use Jenkins to build a simple Java application with Maven.

jenkins.io/doc/tutorials/build-a-java-ap…

Readme
Activity
Custom properties

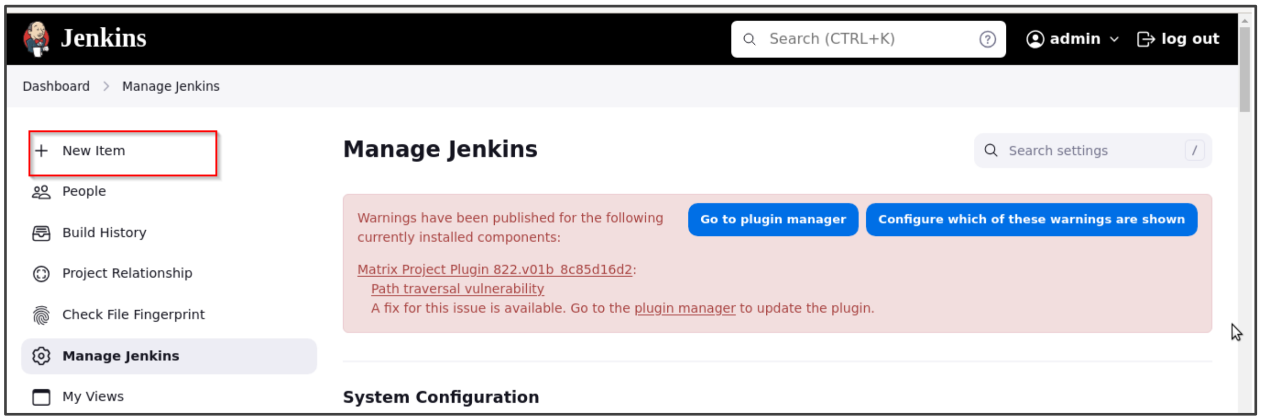
3.2 Run **git clone [Forked REPO URL]** in the terminal to clone the repository locally



```
labsuser@ip-172-31-39-225:~$ git clone https://github.com/jenkins-docs/simple-java-maven-app.git
Cloning into 'simple-java-maven-app'...
remote: Enumerating objects: 173, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 173 (delta 2), reused 4 (delta 0), pack-reused 164
Receiving objects: 100% (173/173), 33.22 KiB | 3.32 MiB/s, done.
Resolving deltas: 100% (51/51), done.
labsuser@ip-172-31-39-225:~$
```

Step 4: Create a freestyle project

4.1 On the Jenkins dashboard, click on **New Item**


Jenkins

Dashboard > Manage Jenkins

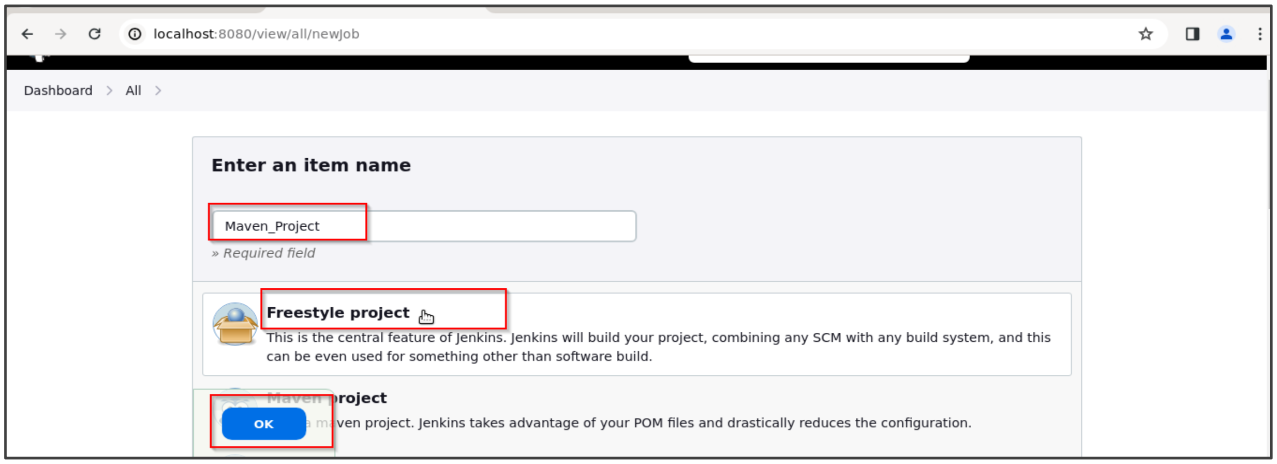
Manage Jenkins

Warnings have been published for the following currently installed components:

Matrix Project Plugin 822.v01b_8c85d16d2:
Path traversal vulnerability
A fix for this issue is available. Go to the plugin manager to update the plugin.

System Configuration

4.2 Enter an arbitrary name for your project (e.g., Maven_Project), and select **Freestyle project**, and click **OK** to create the project


localhost:8080/view/all/newJob

Dashboard > All >

Enter an item name

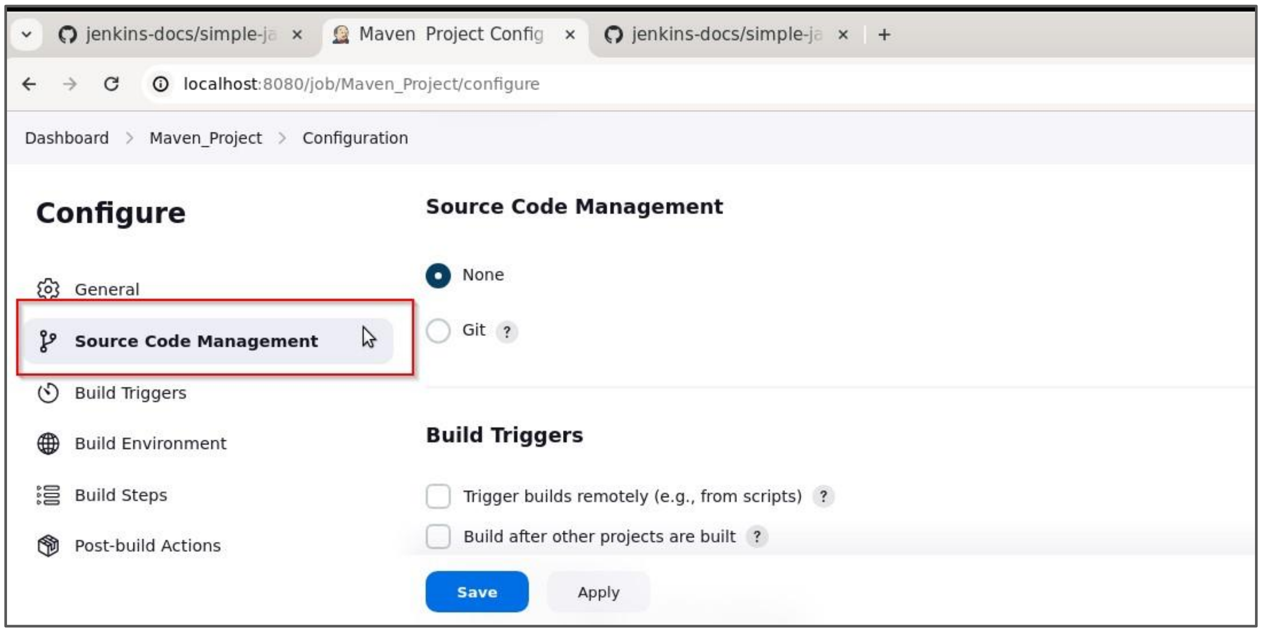
Maven_Project

» 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
... maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

4.3 Click on **Source Code Management**


jenkins-docs/simple-ja | Maven Project Config | jenkins-docs/simple-ja

localhost:8080/job/Maven_Project/configure

Dashboard > Maven_Project > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Source Code Management

[✓ None]

Git ?

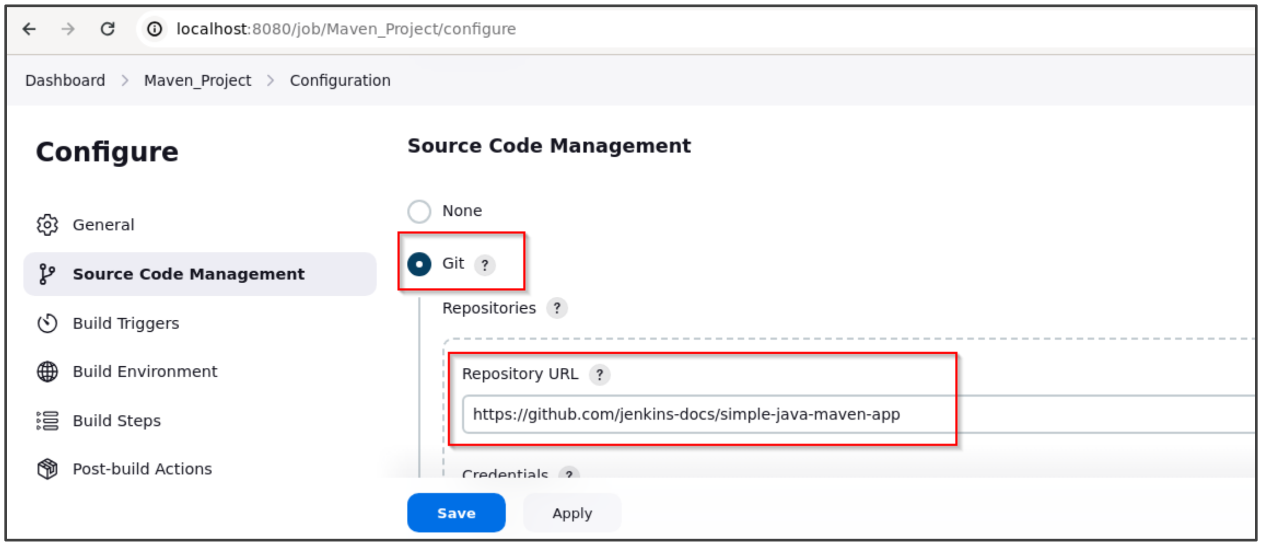
Build Triggers

Trigger builds remotely (e.g., from scripts) ?

Build after other projects are built ?

Save Apply

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


localhost:8080/job/Maven_Project/configure

Dashboard > Maven_Project > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Source Code Management

None

[✓ Git] ?

Repositories ?

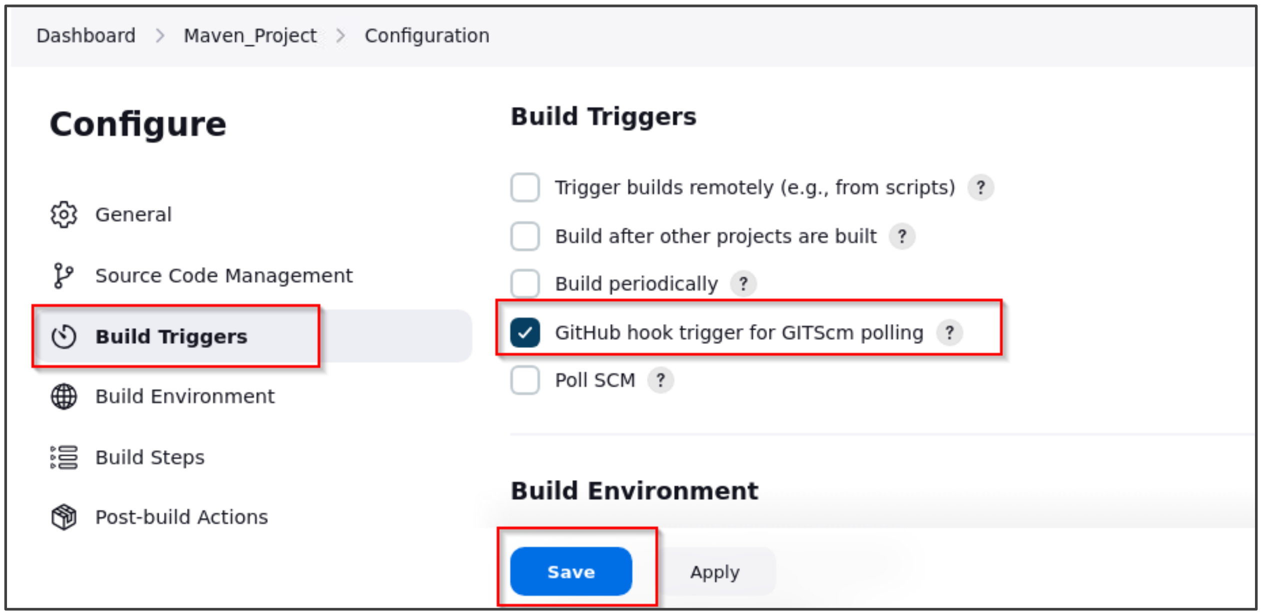
Repository URL ?

https://github.com/jenkins-docs/simple-java-maven-app

Credentials ?

Save Apply

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


Dashboard > Maven_Project > Configuration

Configure

General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions

Build Triggers

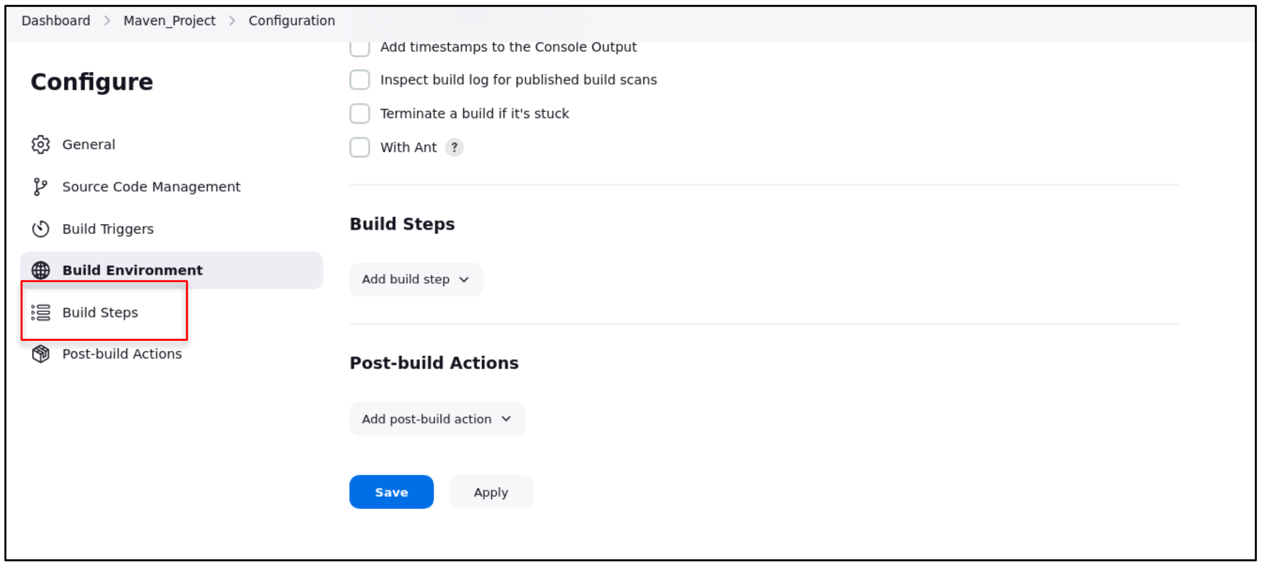
- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

Build Environment

Save Apply

Step 5: Configure the build

5.1 In the project configuration, select the **Build Steps** option


Dashboard > Maven_Project > Configuration

Configure

General
Source Code Management
Build Triggers
Build Environment
Build Steps
Post-build Actions

- [ ] Add timestamps to the Console Output
- [ ] Inspect build log for published build scans
- [ ] Terminate a build if it's stuck
- [ ] With Ant

Build Steps

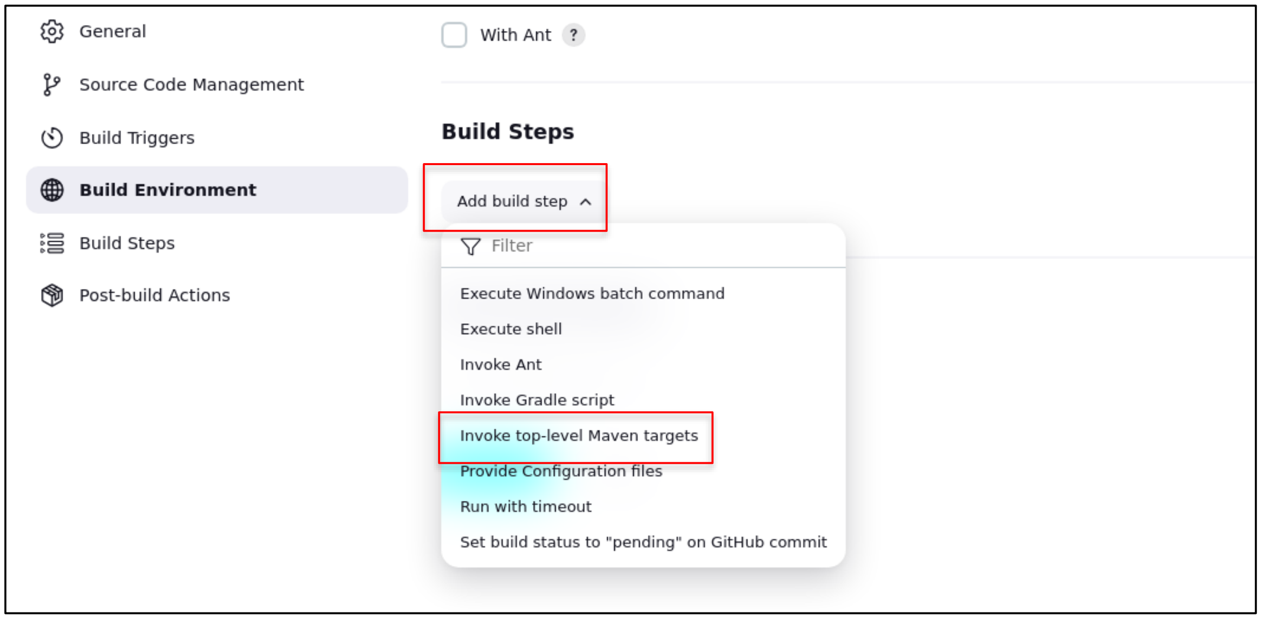
Add build step

Post-build Actions

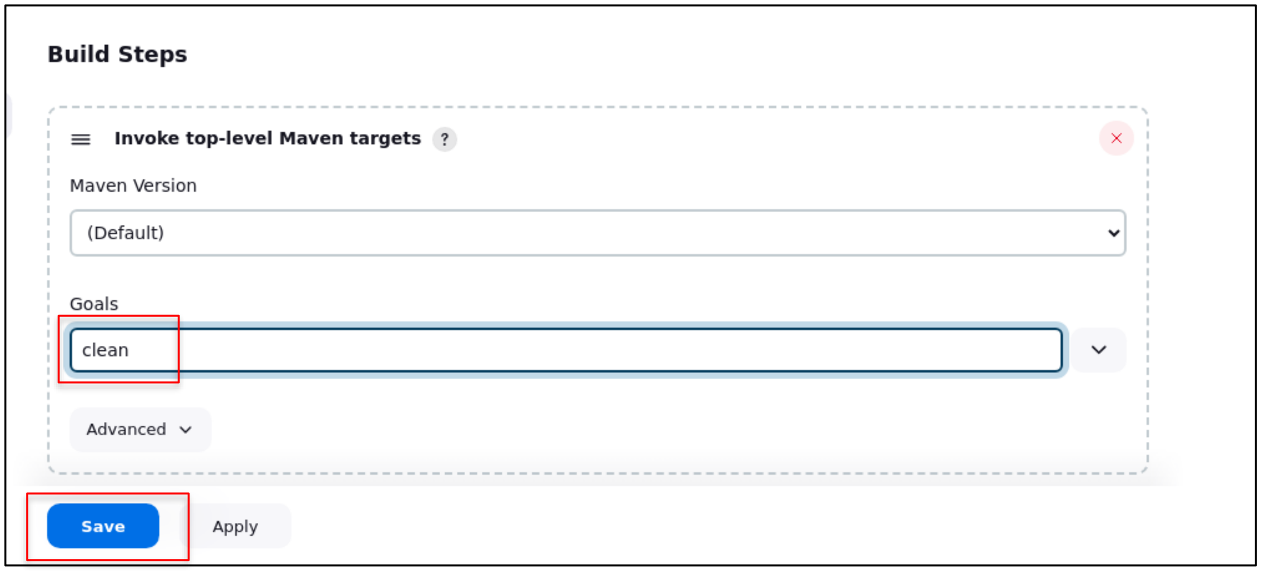
Add post-build action

Save Apply

5.2 Click on the **Add build step** and select **Invoke top-level Maven targets**

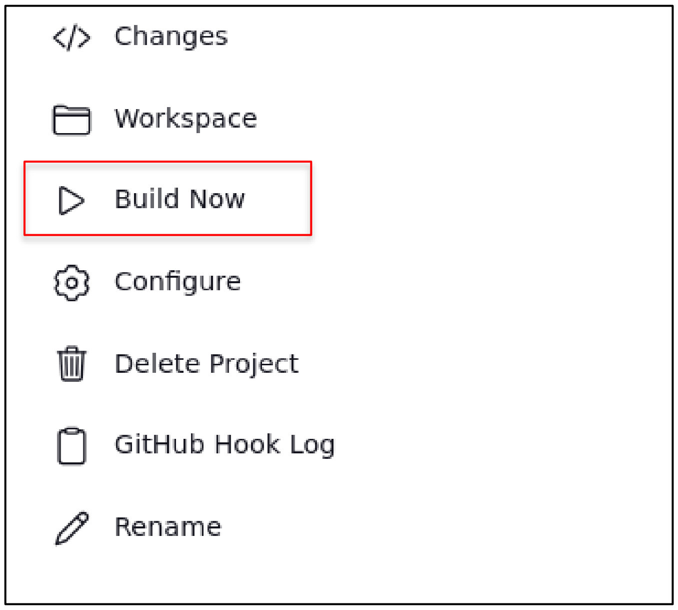


5.3 In the **Goals** field, enter **clean** to perform a Maven clean, scroll to the bottom of the project configuration page, and click **Save** to save your project configuration

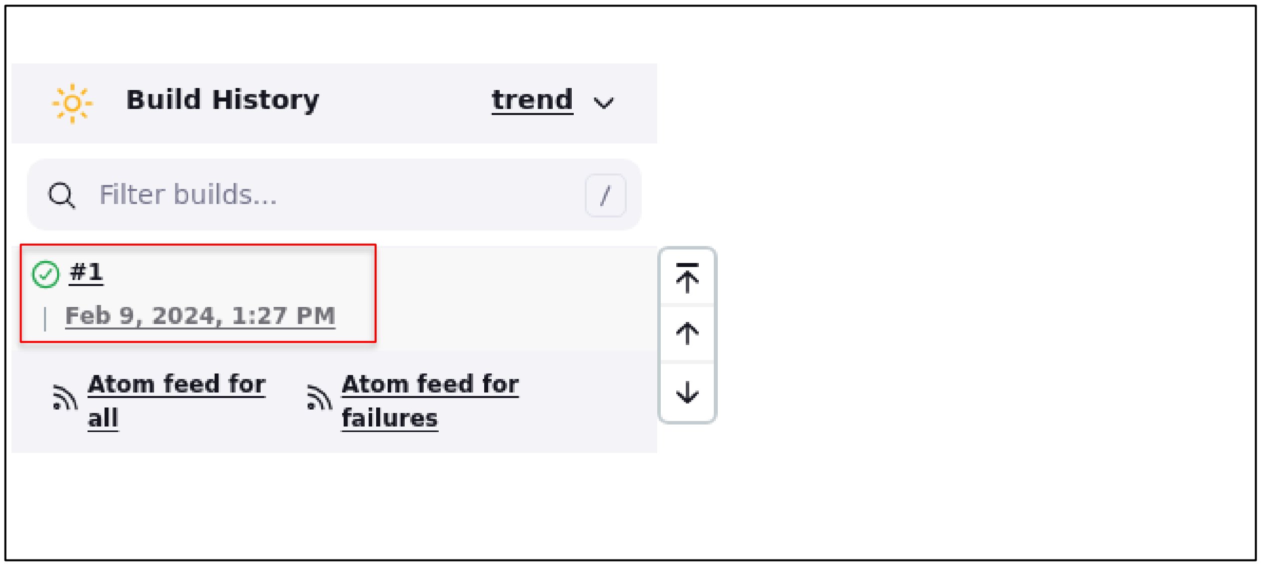


Step 6: Build and view console output

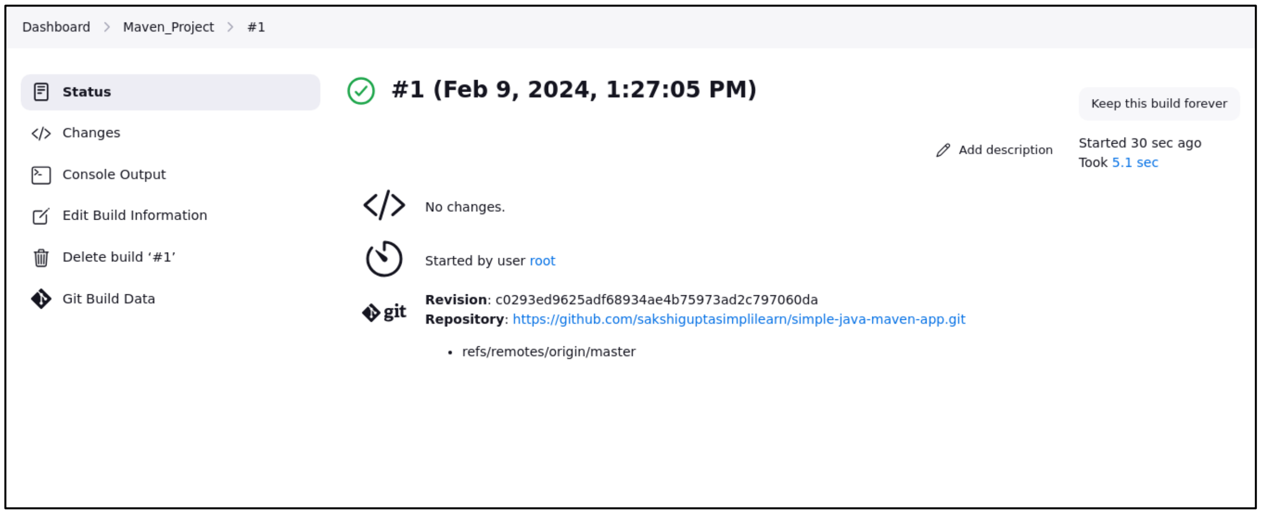
6.1 Click on **Build Now** on the left side of the project page to trigger a manual build



6.2 After triggering the build, monitor the progress



6.3 Click on the build number (e.g., #1) in the **Build History**



#1 (Feb 9, 2024, 1:27:05 PM)

Started 30 sec ago
Took 5.1 sec

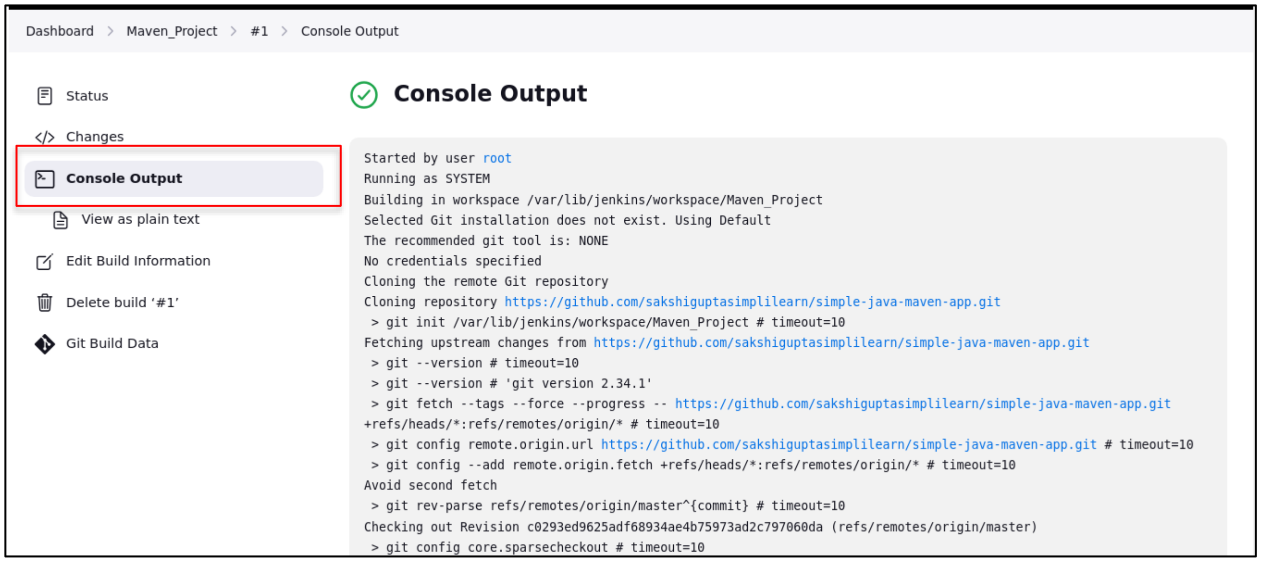
No changes.

Started by user root

Revision: c0293ed9625adf68934ae4b75973ad2c797060da
Repository: https://github.com/sakshiguptasimplilearn/simple-java-maven-app.git

- refs/remotes/origin/master

6.4 Select **Console Output** to view detailed information and logs of the Maven clean build process



```
Started by user root
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Maven_Project
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/sakshiguptasimplilearn/simple-java-maven-app.git
 > git init /var/lib/jenkins/workspace/Maven_Project # timeout=10
Fetching upstream changes from https://github.com/sakshiguptasimplilearn/simple-java-maven-app.git
 > git --version # timeout=10
 > git --version # 'git version 2.34.1'
 > git fetch --tags --force --progress -- https://github.com/sakshiguptasimplilearn/simple-java-maven-app.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/sakshiguptasimplilearn/simple-java-maven-app.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision c0293ed9625adf68934ae4b75973ad2c797060da (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
```

By following these steps, you have successfully used Jenkins to configure and set up a foundation for Maven builds with the help of GitHub.