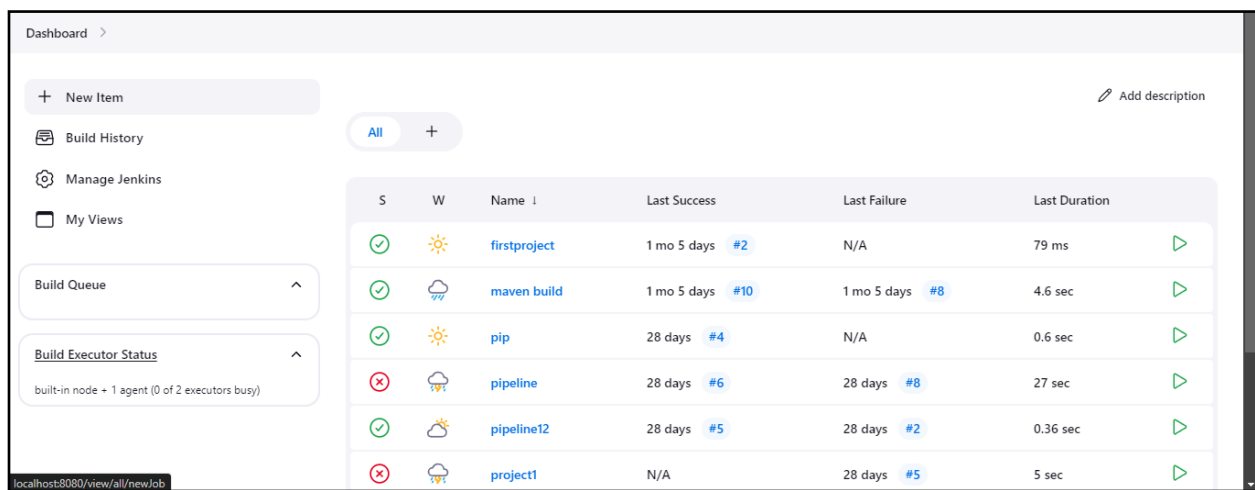


Experiment No. 7

Procedure :

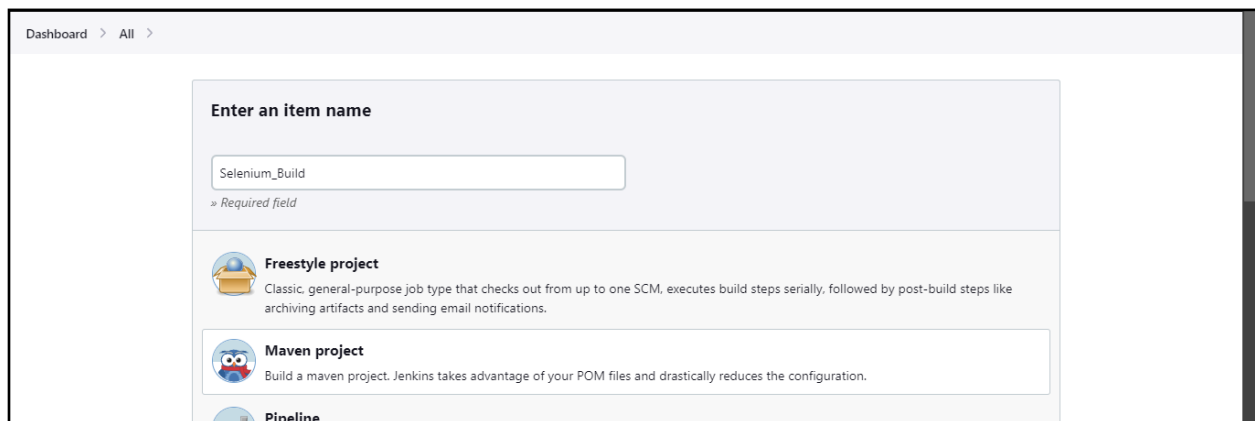
Step 1 : Visit Jenkins dashboard.



The screenshot shows the Jenkins Dashboard with a sidebar on the left containing links to 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. The main area displays a table of jobs with columns for status (S), weather icon (W), name, last success, last failure, and last duration. The jobs listed are 'firstproject', 'maven build', 'pip', 'pipeline', 'pipeline12', and 'project1'. The 'pipeline' job is marked as failed with a red 'X' icon.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀️	firstproject	1 mo 5 days #2	N/A	79 ms
✓	☁️	maven build	1 mo 5 days #10	1 mo 5 days #8	4.6 sec
✓	☀️	pip	28 days #4	N/A	0.6 sec
✗	☁️	pipeline	28 days #6	28 days #8	27 sec
✓	☀️	pipeline12	28 days #5	28 days #2	0.36 sec
✗	☁️	project1	N/A	28 days #5	5 sec

Step 2 : Create new item, enter item name and select maven project.



The screenshot shows the 'Enter an item name' form in Jenkins. The text 'Selenium_Build' is entered in the input field. Below the input field, there are two radio button options: 'Freestyle project' and 'Maven project'. The 'Maven project' option is selected. The description for 'Maven project' is 'Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.'

Enter an item name

Selenium_Build

» Required field

☒ **Freestyle project**
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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

☐ **Pipeline**

Step 3 : In project configuration, enter GitHub repository.

The screenshot shows the Jenkins Configuration page for the 'Selenium_Build' project. The 'Source Code Management' tab is selected in the left sidebar. Under 'Source Code Management', the 'Git' option is chosen. The 'Repositories' section is expanded, showing a 'Repository URL' field with the value 'https://github.com/jglick/simple-maven-project-with-tests.git'. The 'Credentials' dropdown is set to '- none -'. There is an '+ Add' button and an 'Advanced' dropdown menu. At the bottom, there are 'Save' and 'Apply' buttons.

Dashboard > Selenium_Build > Configuration

Configure

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

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

Credentials ?

+ Add

Advanced

Save Apply

Step 4 : enter pom.xml and goals as test.

The screenshot shows the Jenkins Configuration page for the 'Selenium_Build' project, specifically the 'Build' section. The 'Pre Steps' tab is selected in the left sidebar. Under 'Build', the 'Root POM' field contains 'pom.xml'. The 'Goals and options' field contains 'test'. There is an 'Advanced' dropdown menu at the bottom.

Dashboard > Selenium_Build > Configuration

Add pre-build step

Configure

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

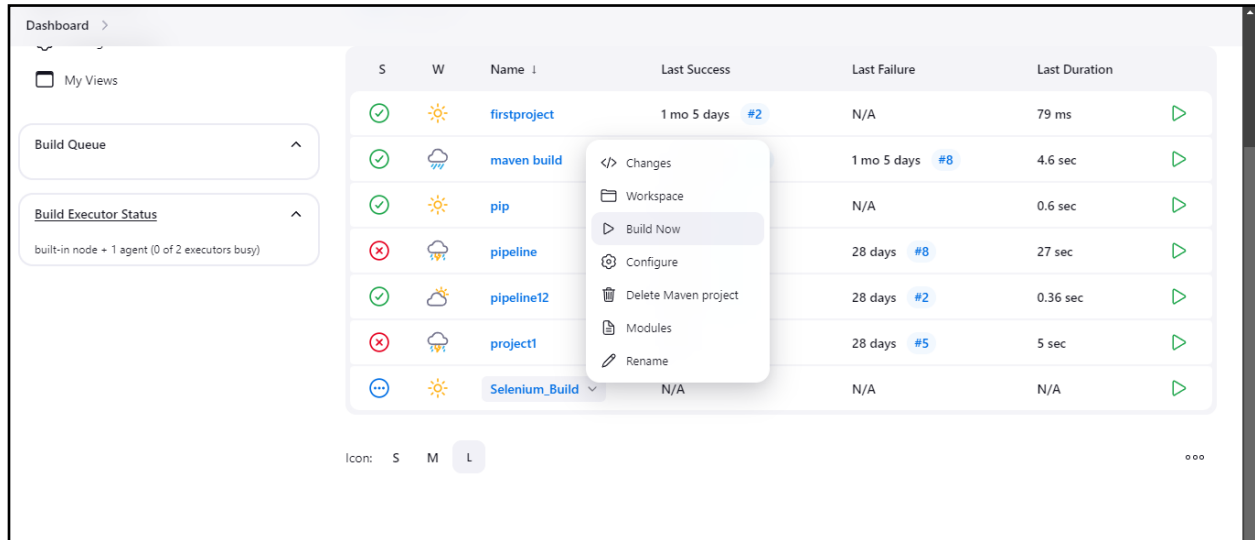
Build

Root POM ?

Goals and options ?

Advanced

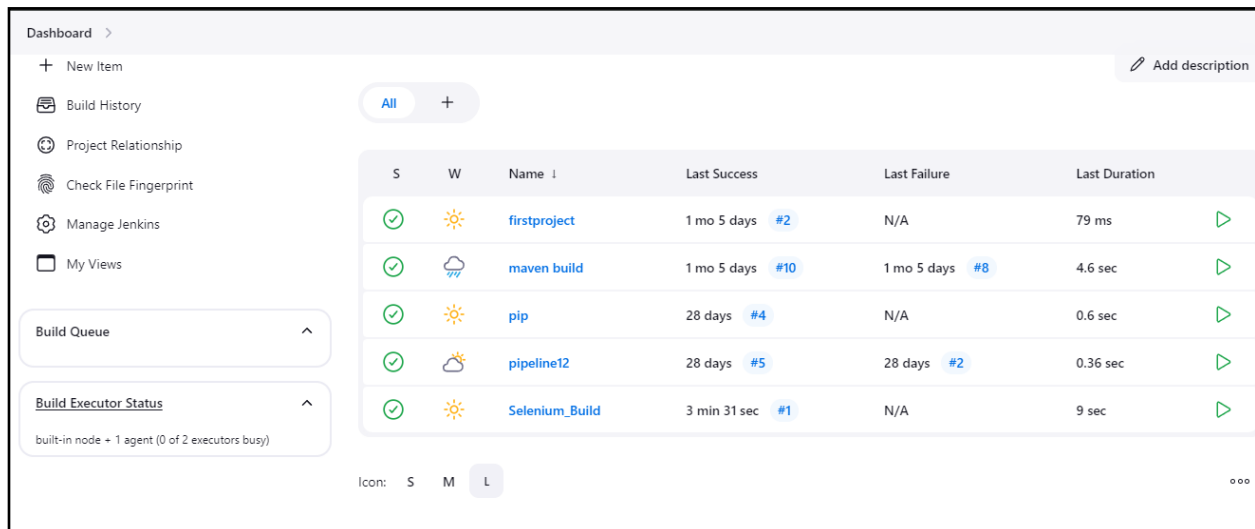
Step 5 : Build the project.



The screenshot shows the Jenkins Dashboard with a table of builds. A context menu is open over the 'firstproject' build, showing options like 'Changes', 'Workspace', 'Build Now', 'Configure', 'Delete Maven project', 'Modules', and 'Rename'. The 'Build Now' option is highlighted.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	firstproject	1 mo 5 days #2	N/A	79 ms
✓	☁	maven build	1 mo 5 days #8	N/A	4.6 sec
✓	☀	pip	N/A	N/A	0.6 sec
✗	☁	pipeline	28 days #8	27 sec	
✓	☀	pipeline12	28 days #2	0.36 sec	
✗	☁	project1	28 days #5	5 sec	
...	☀	Selenium_Build	N/A	N/A	N/A

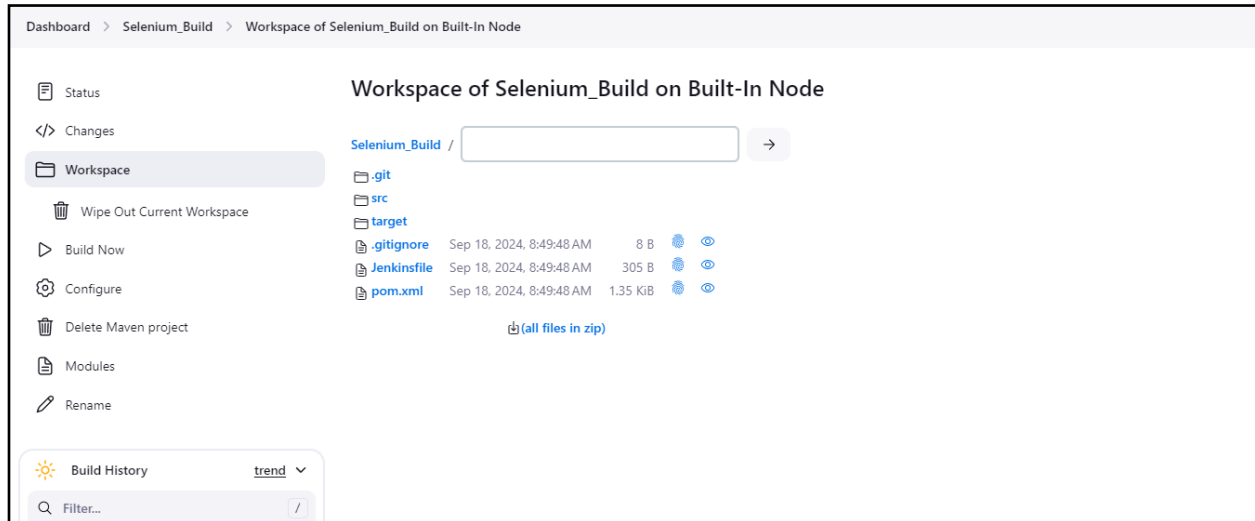
Step 6 : Build is successful.



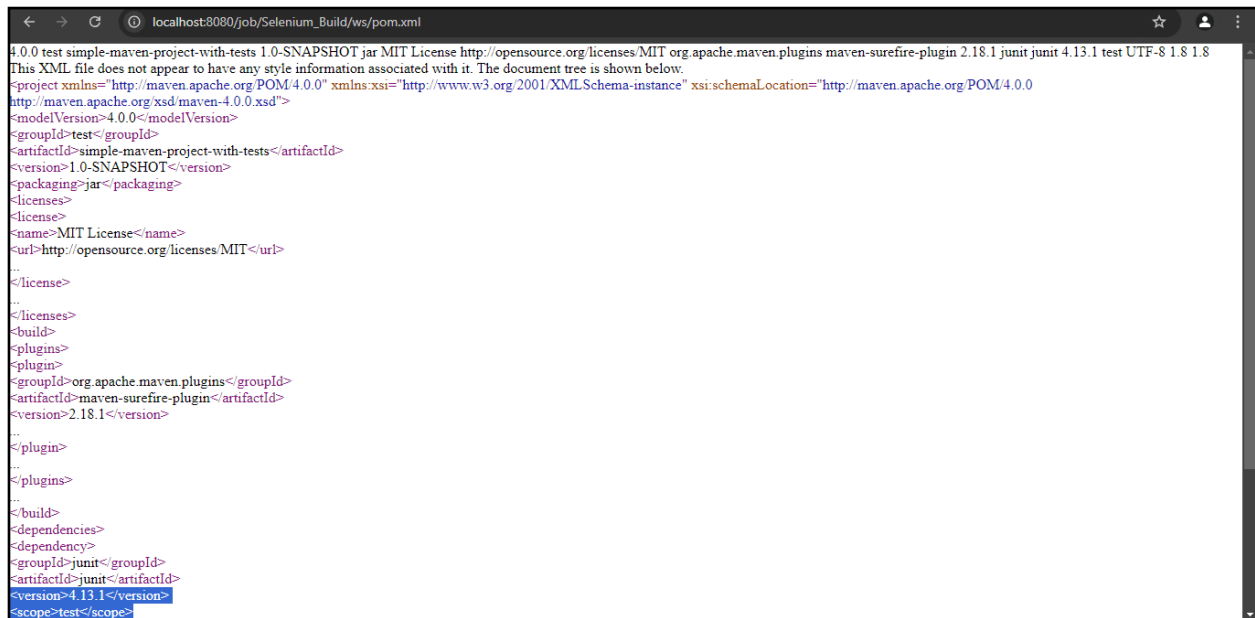
The screenshot shows the Jenkins Dashboard with the 'firstproject' build now marked as successful (green checkmark). The context menu is no longer visible. The 'Build Queue' and 'Build Executor Status' sections are also visible on the left.

S	W	Name	Last Success	Last Failure	Last Duration
✓	☀	firstproject	1 mo 5 days #2	N/A	79 ms
✓	☁	maven build	1 mo 5 days #10	1 mo 5 days #8	4.6 sec
✓	☀	pip	28 days #4	N/A	0.6 sec
✓	☀	pipeline12	28 days #5	28 days #2	0.36 sec
✓	☀	Selenium_Build	3 min 31 sec #1	N/A	9 sec

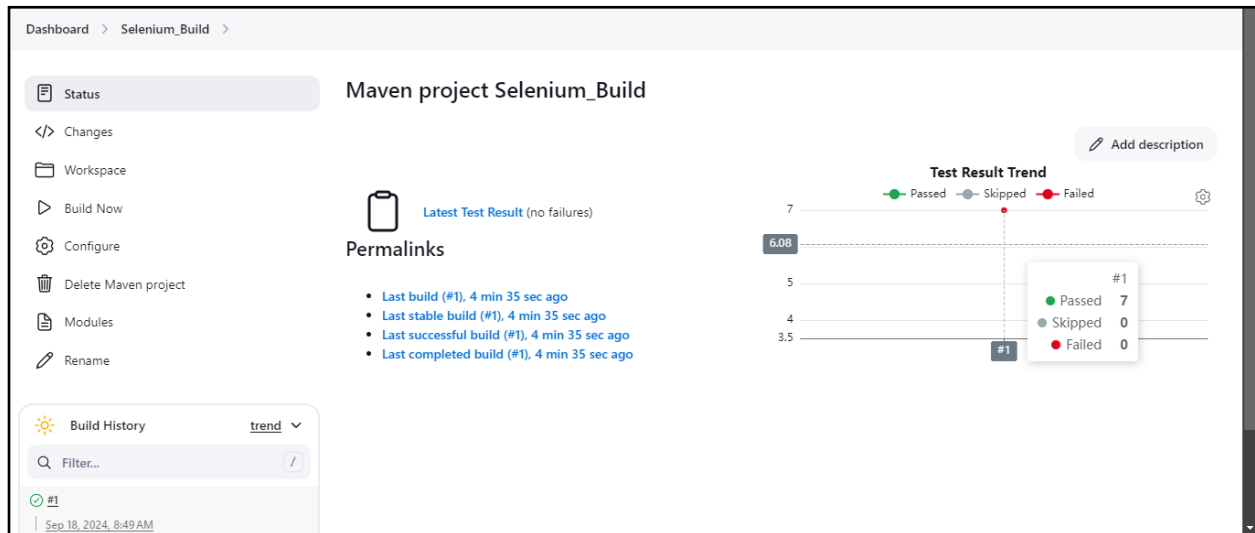
Step 7 : Workspace of the project.



Step 8 : JUnit version and scope can be seen in pom.xml



Step 9 : All test cases are passed.



Step 10 : Code that is executed.

The screenshot shows the GitHub web interface for the repository 'simple-maven-project-with-tests'. The file 'Base.java' is selected, showing its source code. The code is as follows:

```
24
25 package test;
26
27 import static org.junit.Assert.*;
28 import org.junit.internal.AssumptionViolatedException;
29
30 class Base {
31
32     protected void run() {
33         double r = Math.random();
34         if (r < 0.1) {
35             fail("oops");
36         } else if (r < 0.2) {
37             throw new AssumptionViolatedException("skipping");
38         }
39     }
40
41 }
```