Experiment no. 7

AIM : To Setup and Run Selenium Tests in Jenkins Using Maven.

Theory:

Jenkins:

Jenkins is the leading open-source continuous integration tool developed by Hudson lab. It is cross-platform and can be used on Windows, Linux, Mac OS and Solaris environments. Jenkins is written in Java. Jenkin's chief usage is to monitor any job which can be SVN checkout, cron or any application states. It fires pre-configured actions when a particular step occurs in jobs.

Maven:

Maven is a powerful project / build management tool, based on the concept of a POM (Project Object Model) that includes project information and configuration information for Maven such as construction directory, source directory, dependency, test source directory, Goals, plugins, etc.

Selenium WebDriver is great for browser automation. But, when using it for testing and building a test framework, it feels underpowered. Integrating Maven with Selenium provides following benefits

Apache Maven provides support for managing the full lifecycle of a test project.

- Maven is used to define project structure, dependencies, build, and test management.
- Using pom.xml(Maven) you can configure dependencies needed for building testing and running code.
- Maven automatically downloads the necessary files from the repository while building the project.

Why Jenkins and Selenium?

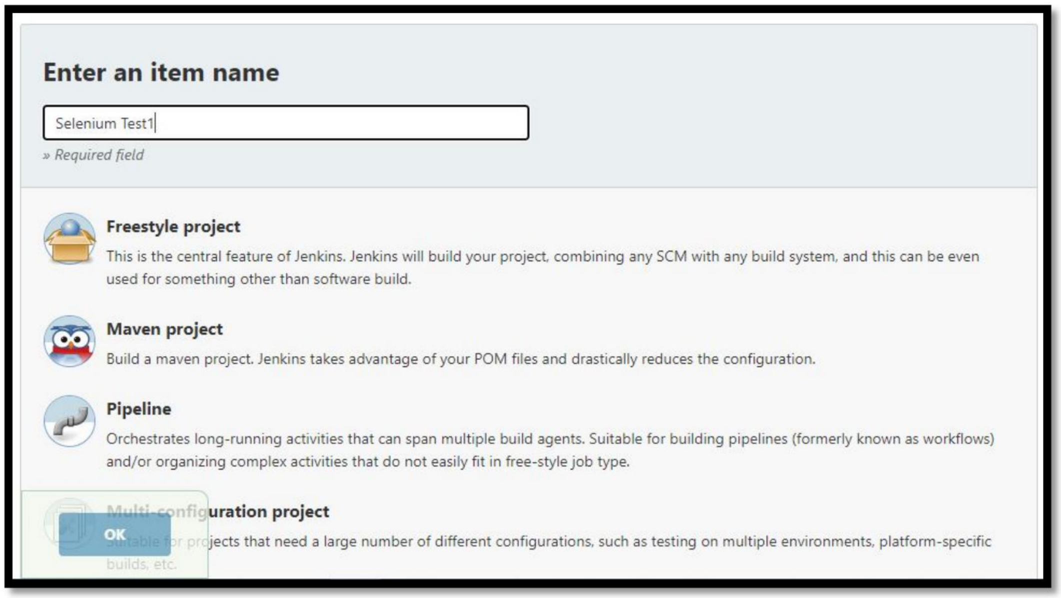
Running Selenium tests in Jenkins allows you to run your tests every time your software changes and deploy the software to a new environment when the tests pass.

- Jenkins can schedule your tests to run at specific time.
- You can save the execution history and Test Reports.
- Jenkins supports Maven for building and Testing a project in continuous integration.

Advantages of Using Maven and Jenkins with selenium:

- Jenkins provides a way to do smoke testing for every time the code changes and deployed to a new environment. It will make sure that the code is running properly.
- You cans schedule your test cases with Jenkins so that if regression suite takes almost 6-7 hours to run then you can have nightly build run so that by the time you reach office it will be done.
- Jenkins server will act as a common server for client as well as technical people to logon to it and see all test reports and test execution history.
- Maven in turn reduces dependency on hard coding of jars.
- Maven can make the build process very easy.
- Also, at some time if you need to update the jars with a specific number, you don't need to go to Build path and add that particular jar. You can just change the number of version in pom.xml and it will be done.
- In a team where people are distributed across geographical locations it is easy to share artefact id and version id to clone the project rather than sharing on a common drive.

To setup and run selenium tests in Jenkins using maven:



Enter an item name

Selenium Test1

» 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

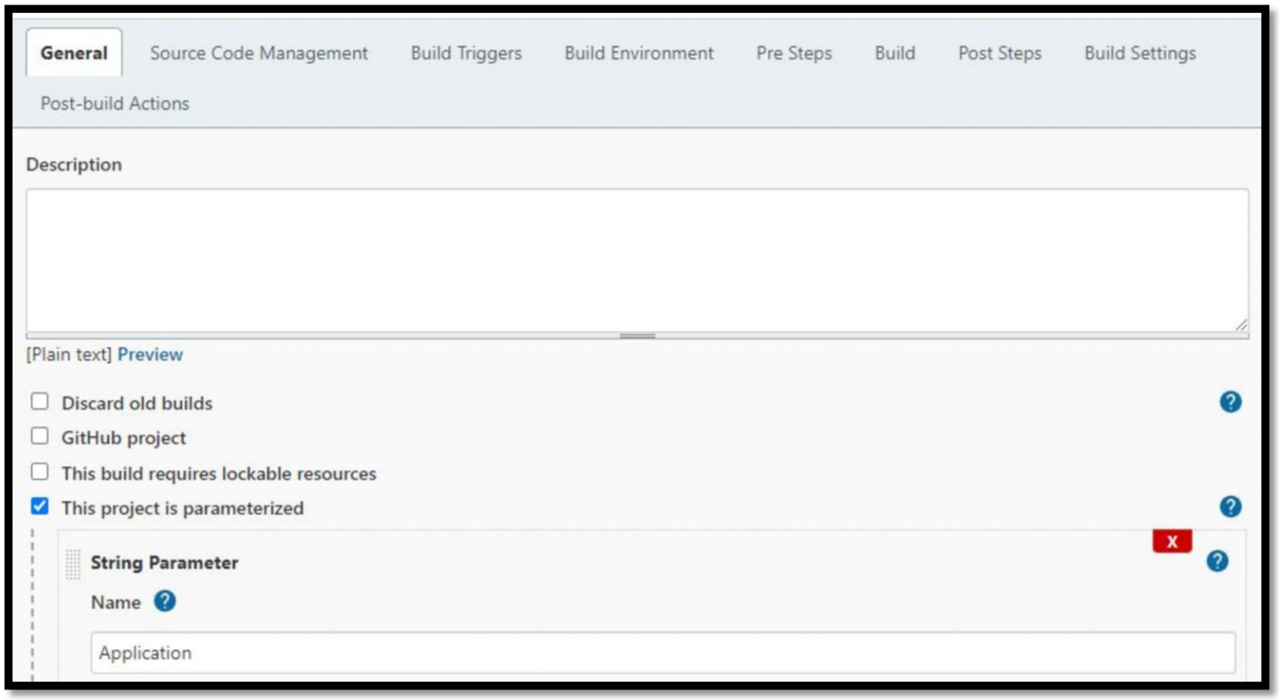
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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.

Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Description

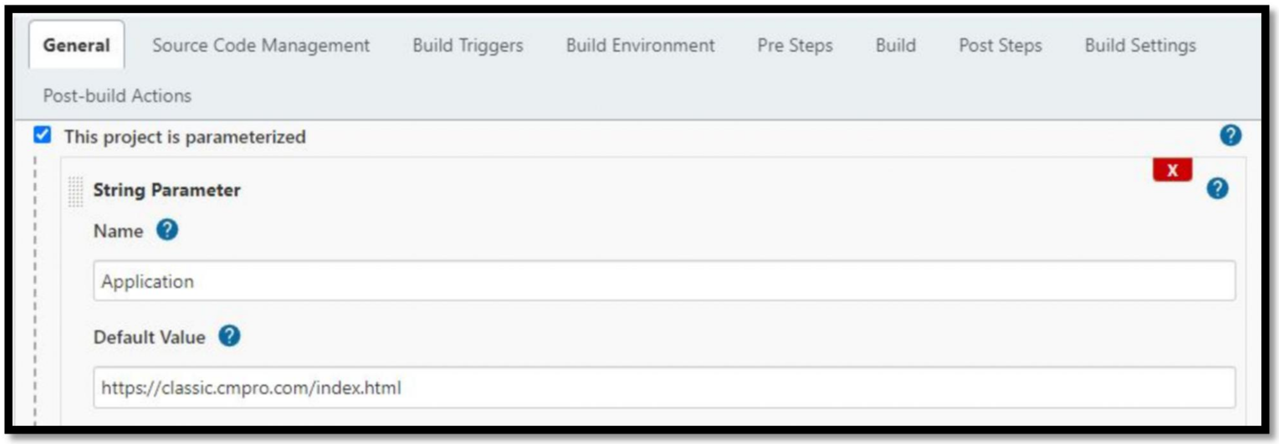
[Plain text] Preview

- [ ] Discard old builds
- [ ] GitHub project
- [ ] This build requires lockable resources
- [x] This project is parameterized

String Parameter

Name

Application

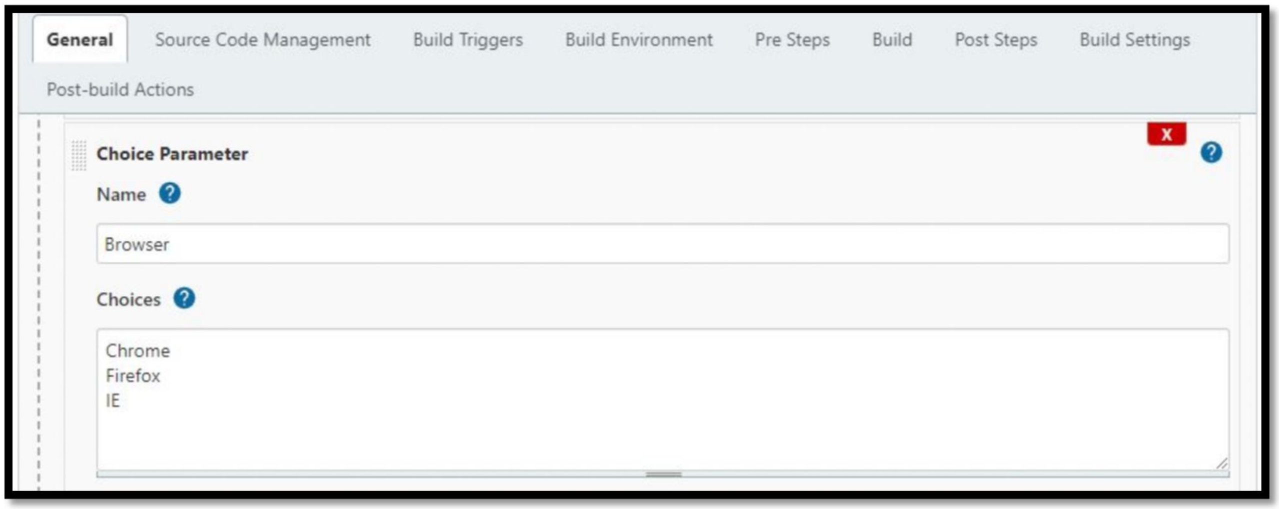


This project is parameterized [✓]

String Parameter

Name: Application

Default Value: https://classic.cmpro.com/index.html

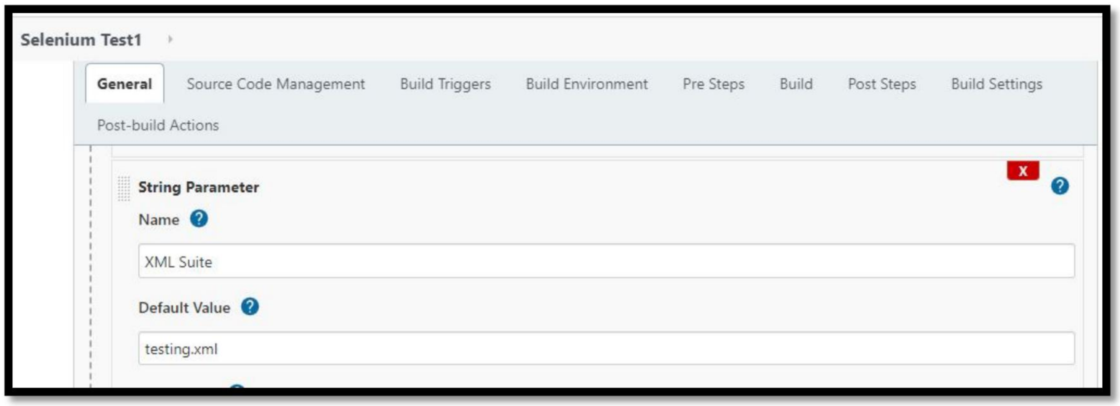


Choice Parameter

Name: Browser

Choices:

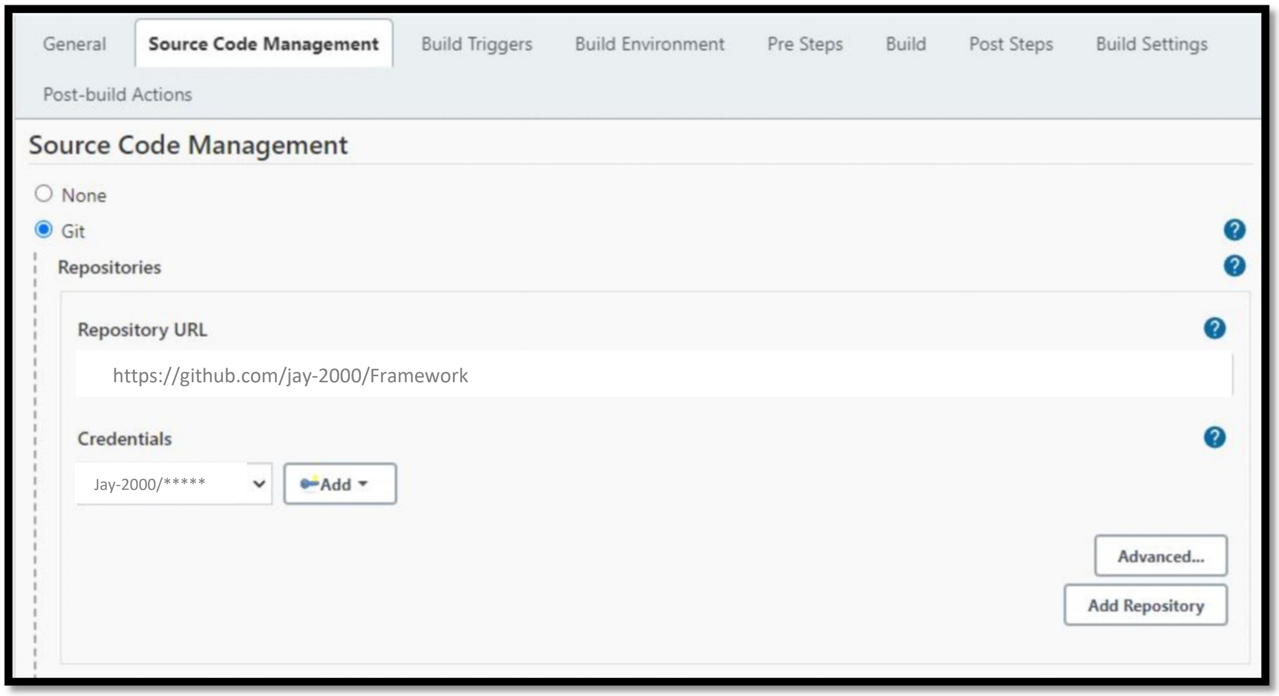
```
Chrome
Firefox
IE
```



String Parameter

Name: XML Suite

Default Value: testing.xml



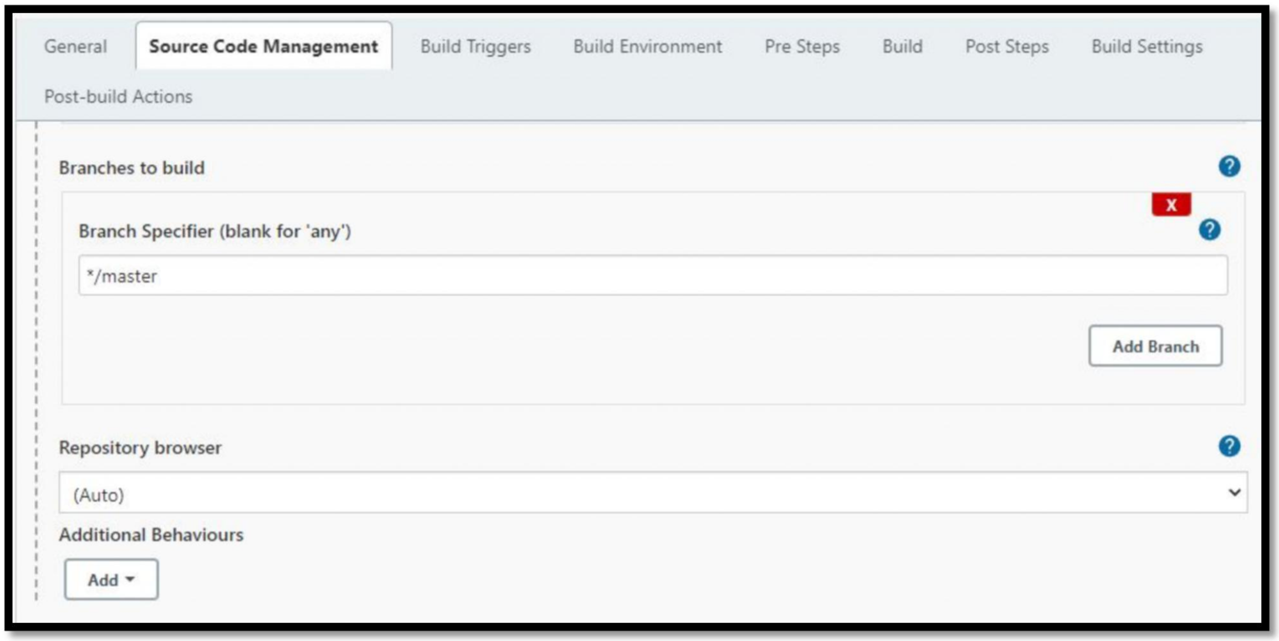
Source Code Management

None
[✓ Git]

Repositories

Repository URL: https://github.com/jay-2000/Framework

Credentials: Jay-2000/*****

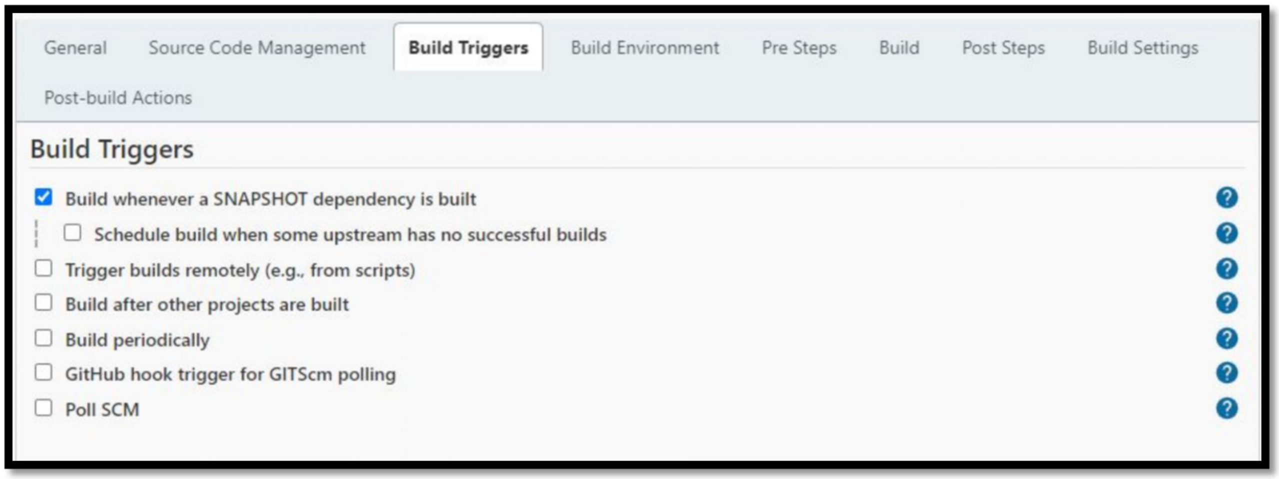


Branches to build

Branch Specifier (blank for 'any'): */master

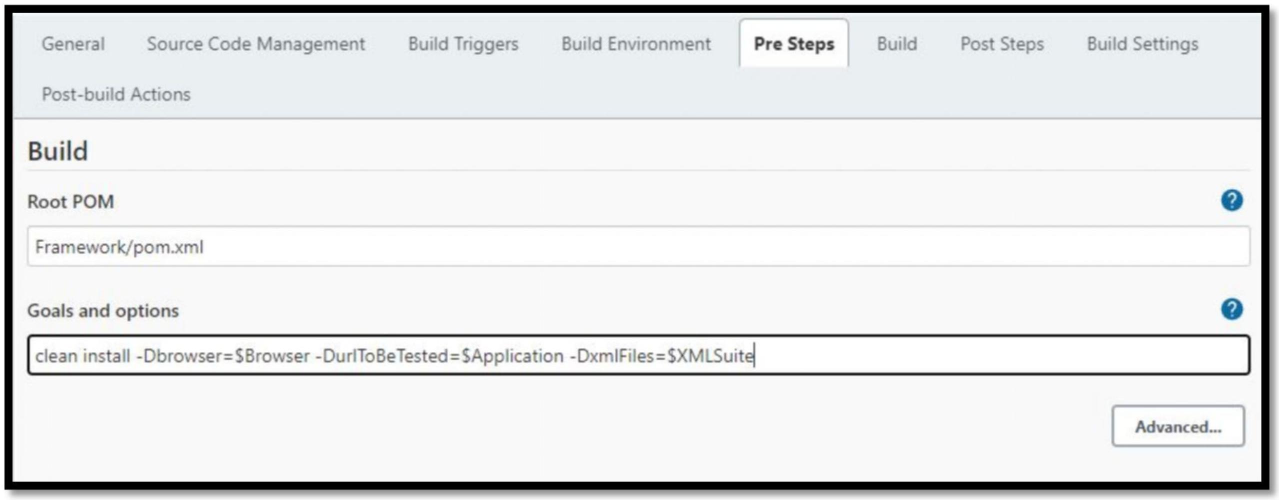
Repository browser: (Auto)

Additional Behaviours


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

Build Triggers

- [✓] Build whenever a SNAPSHOT dependency is built
  - [ ] Schedule build when some upstream has no successful builds
- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [ ] Poll SCM


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

Build

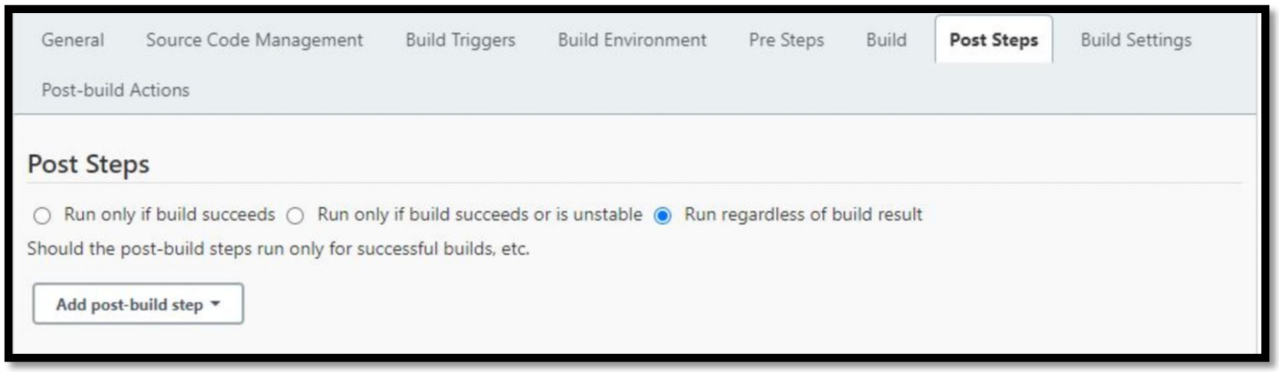
Root POM

Framework/pom.xml

Goals and options

```
clean install -Dbrowser=$Browser -DurlToBeTested=$Application -DxmlFiles=$XMLSuite
```

Advanced...

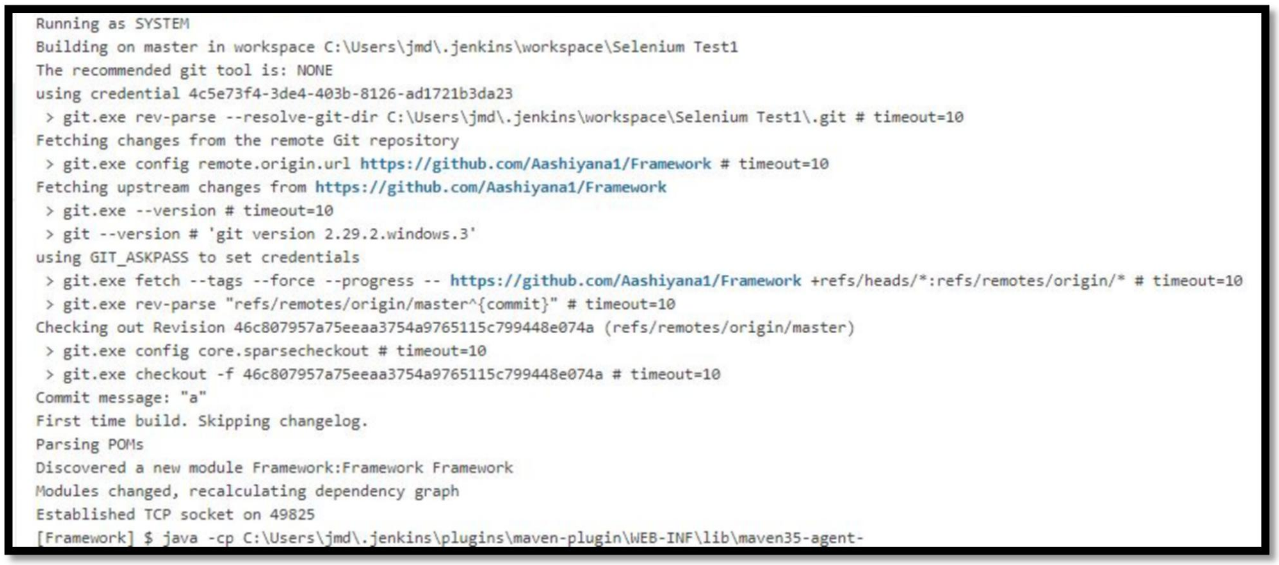

General | Source Code Management | Build Triggers | Build Environment | Pre Steps | Build | **Post Steps** | Build Settings | Post-build Actions

Post Steps

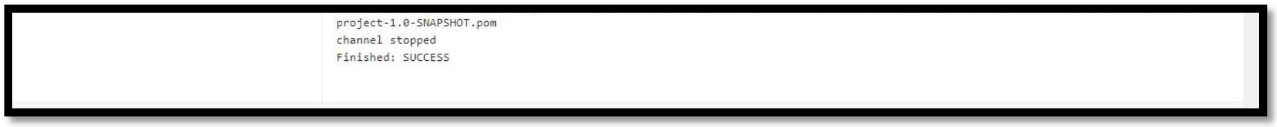
Run only if build succeeds [✓ Run regardless of build result] Run only if build succeeds or is unstable

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

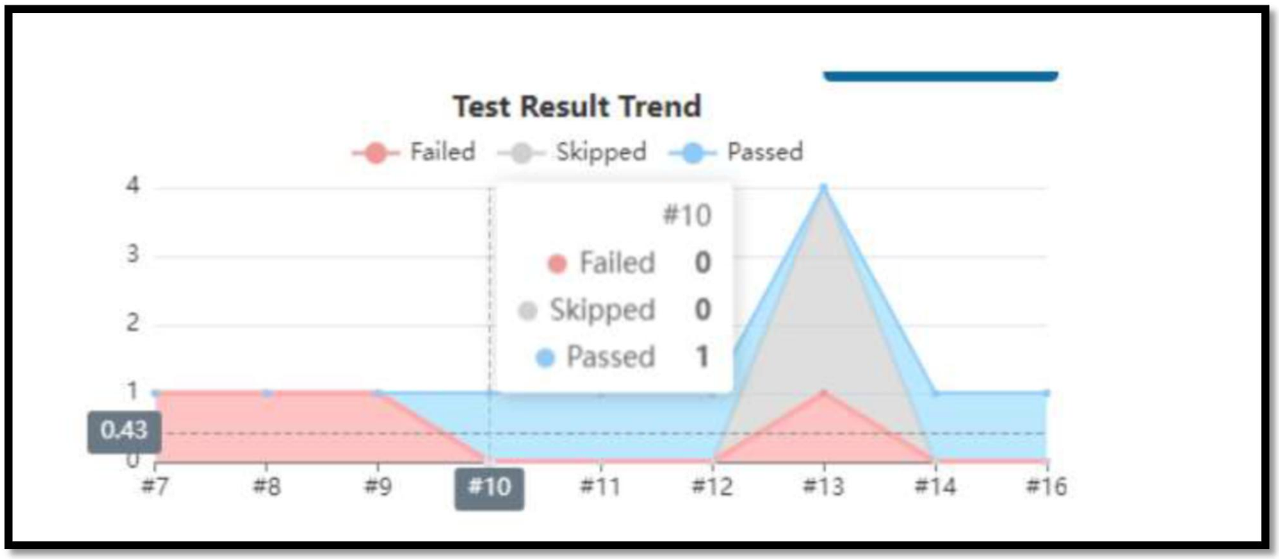
Add post-build step



```
Running as SYSTEM
Building on master in workspace C:\Users\jmd\.jenkins\workspace\Selenium Test1
The recommended git tool is: NONE
using credential 4c5e73f4-3de4-403b-8126-ad1721b3da23
 > git.exe rev-parse --resolve-git-dir C:\Users\jmd\.jenkins\workspace\Selenium Test1\.git # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/Aashiyana1/Framework # timeout=10
Fetching upstream changes from https://github.com/Aashiyana1/Framework
 > git.exe --version # timeout=10
 > git --version # 'git version 2.29.2.windows.3'
using GIT_ASKPASS to set credentials
 > git.exe fetch --tags --force --progress -- https://github.com/Aashiyana1/Framework +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 46c807957a75eeaa3754a9765115c799448e074a (refs/remotes/origin/master)
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 46c807957a75eeaa3754a9765115c799448e074a # timeout=10
Commit message: "a"
First time build. Skipping changelog.
Parsing POMs
Discovered a new module Framework:Framework Framework
Modules changed, recalculating dependency graph
Established TCP socket on 49825
[Framework] $ java -cp C:\Users\jmd\.jenkins\plugins\maven-plugin\WEB-INF\lib\maven35-agent-
```



```
project-1.0-SNAPSHOT.pom
channel stopped
Finished: SUCCESS
```



Conclusion:

Thus, we successfully learn and perform about Selenium Tests in Jenkins using Maven.