**What is 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.

## Important Features of Jenkins

* Change Support: Jenkins generates the list of all changes done in repositories like SVN.
* Permanent links: Jenkins provides direct links to the latest build or failed build that can be used for easy communication
* Installation: Jenkins is easy to install either using direct installation file (exe) or war file to deploy using application server.
* Email integration: Jenkins can be configured to email the content of the status of the build.
* Easy Configuration: To configure various tasks on Jenkins is easy.
* TestNG test: Jenkins can be configured to run the automation test build on[Testng](https://www.guru99.com/all-about-testng-and-selenium.html)after each build of SVN.
* Multiple VMs: Jenkins can be configured to distribute the build on multiple machines.
* Project build: Jenkins documents the details of jar, version of jar and mapping of build and jar numbers.
* Plugins: 3rd party plugin can be configured in Jenkins to use features and additional functionality.

## 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](https://www.guru99.com/software-testing.html)a project in continuous integration.

## What is 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.

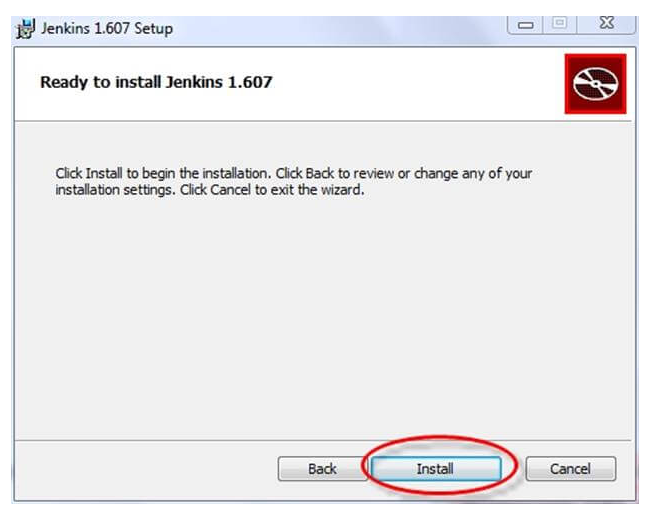
## Why Maven & Jenkins

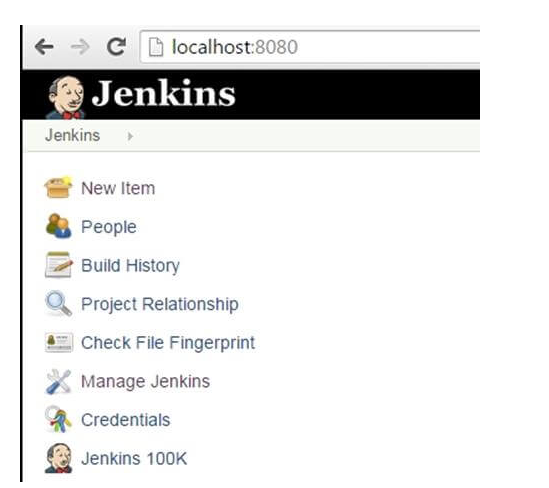
Selenium WebDriver is great for browser automation. But, when using it for testing and building a test framework, it feels underpowered. Integrating Maven with Selenium provides following benefitsApache Maven provides support for managing the full lifecycle of a test project.

* 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.

Jenkins:





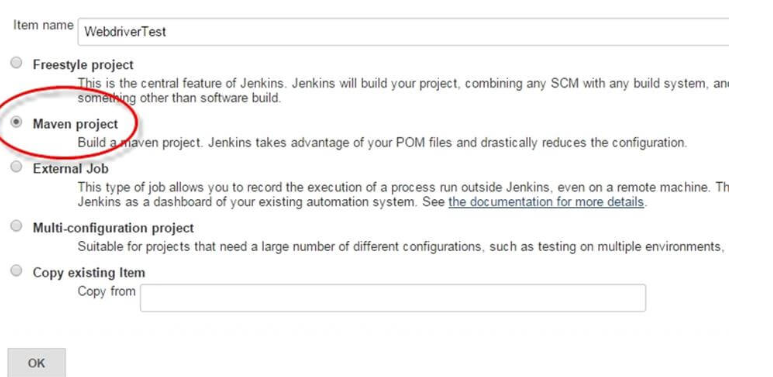


Once installation is done, navigate to the Jenkins Dashboard (http://localhost:8080 by default) in the browser window.

\*Click on the **New Item** link to create a CI job.



\*Select the Maven project radio button as shown in the following screenshot:

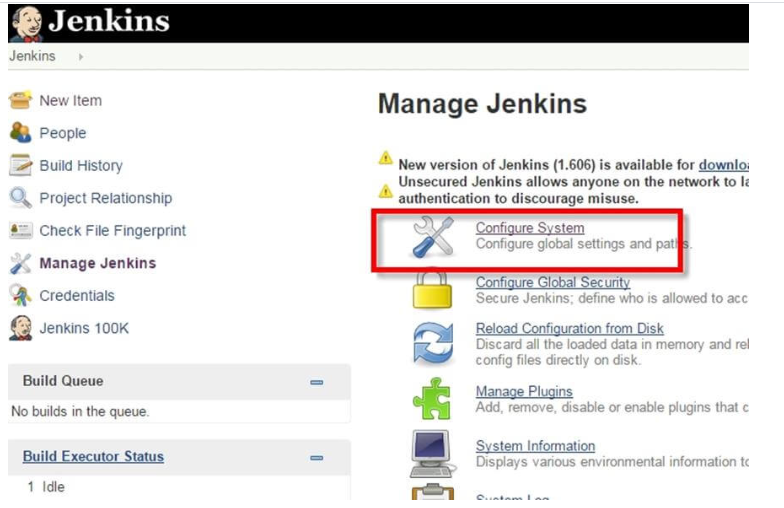


Using the Build a **Maven Project** option, Jenkins supports building and testing Maven projects.

\*Click on OK button. A new job with name "WebdriverTest" is created in Jenkins Dashboard.



\*Go to **Manage Jenkins** => **Configure System** as shown in the following screenshot.



\*Click on JDK installations and configure JDK as in the following screenshot:

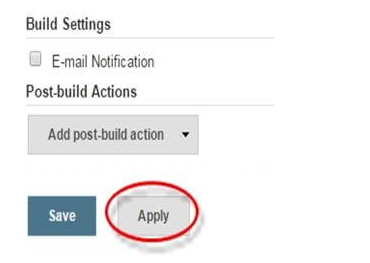


\*Go to the **Build** section of new job.

* In the **Root POM** textbox, enter full path to pom.xml
* In Goals and options section, enter "clean test"



\*Click on **Apply** button.



\*On the WebdriverTest project page, click on the **Build Now** link.



Maven will build the project. It will then have TestNG execute the test cases.

\*Once the build process is completed, in Jenkins Dashboard click on the **WebdriverTest** project



The WebdriverTest project page displays the build history and links to the results as shown in the following screenshot:

Once the build process is completed, in Jenkins Dashboard click on the **WebdriverTest** project



\*The WebdriverTest project page displays the build history and links to the results as shown in the following screenshot:

\*The WebdriverTest project page displays the build history and links to the results as shown in the following screenshot: