Devops driven Test Automation

Author : Sunidhi Aggarwal

Date : 07-OCT -2016

Contents

[DevOps Driven Test Automation 4](#_Toc464119787)

[I. Overview 4](#_Toc464119788)

[II. Tools and technology Used 4](#_Toc464119789)

[III. Prerequisites 4](#_Toc464119790)

[IV. Installation Steps 4](#_Toc464119791)

[V. Steps to Set Maven Path in eclipse 4](#_Toc464119792)

[VI. Create a Maven project. 5](#_Toc464119793)

[VII. Steps to Install Jenkins, Maven and Selenium Configuration for Testing 10](#_Toc464119794)

[VIII. Clonning Gitbash to GitHub and integrate it with Jenkins for Testing 14](#_Toc464119795)

[IX. References: 19](#_Toc464119796)

|  |  |  |
| --- | --- | --- |
| **Prepared/Modified by** | **Role** | **Date of Preparation** |
| Sunidhi Aggarwal | Software Engineer | 07th Oct 2016 |
| **Reviewed by** | **Role** | **Date of Review** |
| Sukhbir | Technical Specialist |  |
| **Approved by** | **Role** | **Date of Approval** |
| Amey Kadam | Senior Technical Architect |  |
| **Version Number** | **0.3** | 0.3 |

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#** | **Date** | **Version #** | **Section Changed** | **Details of changes made** | **Approved By** |
| 1 | 07/10/2016 | 0.1 | Template | Template |  |
| 2 | Sunidhi Aggarwal | 0.2 | Initial draft |  |  |
| 3 | Sukhbir | 0.3 | review |  |  |

# DevOps Driven Test Automation

## Overview

This document provides the guidelines how to setup the automated testing environment using open source tools.

## Tools and technology Used

1. GIT: - is a distributed revision control system and source code management system. It keeps record of each and every revision
2. Jenkins: - is a powerful tool that allows continuous integration and continuous delivery of project.
3. Maven:- (build management tool) has a XML file called pom.xml (Project Object Model), and that file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins.
4. Selenium: - (testing tool) web driver Automates browsers with interacting the browsers objects.

## Prerequisites

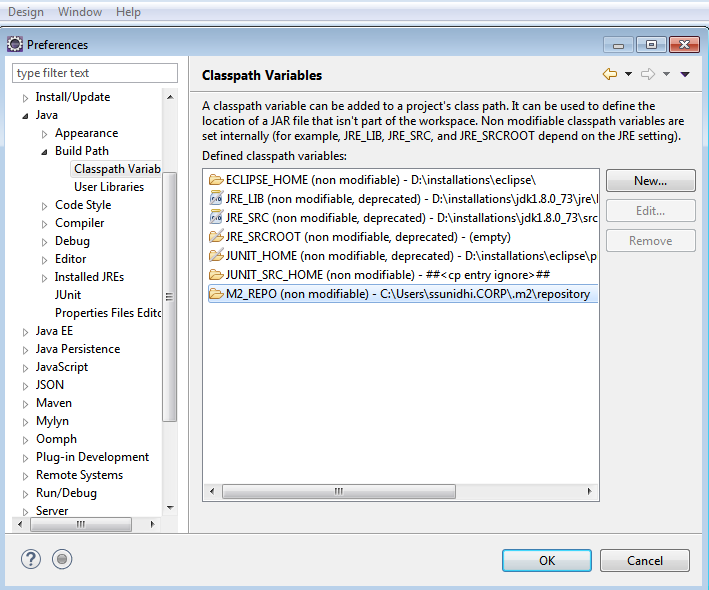
1. Eclipse IDE (Neon)
2. JDK-1.8
3. apache-maven-3.3.9
4. Jenkins-2.7.3
5. Selenium
6. JUnit
7. TestNG
8. Git

## Installation Steps

1. Download Eclipse (https://eclipse.org/downloads/) and install it.
2. Download JDK-1.8 (http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html) and Set its path in environment variables.
3. Download apache-maven-3.3.9 (https://maven.apache.org/download.cgi) and Set its path in environment variable.

## Steps to Set Maven Path in eclipse

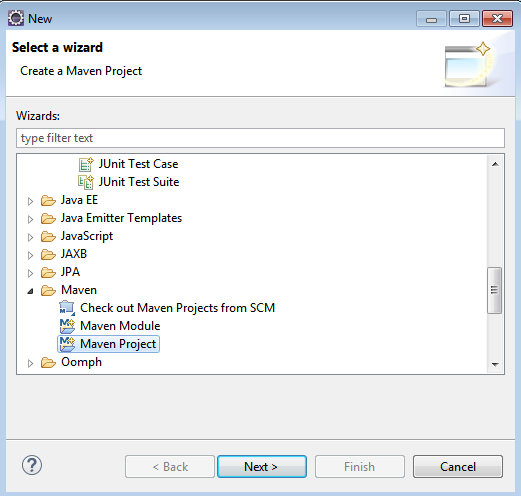
1. Eclipse IDE, menu bar.
2. Select Window > Preferences.
3. Select Java > Build Path > Classpath Variables
4. Click on the new button > defined a new M2\_REPO variable and point it to your local Maven repository as shown in the screenshot:



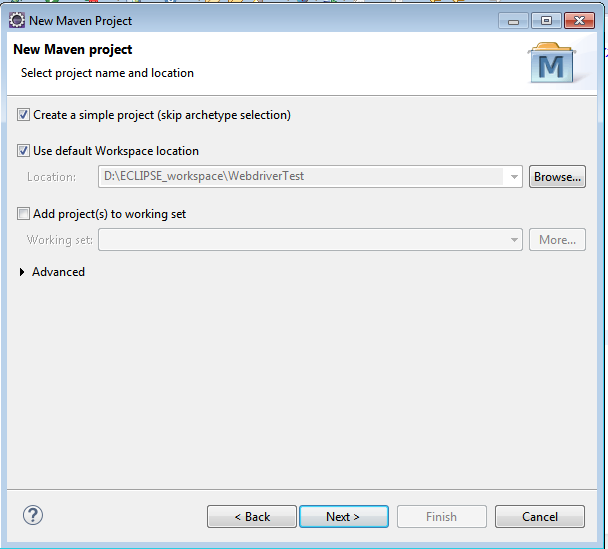
1. Click OK button.

## Create a Maven project.

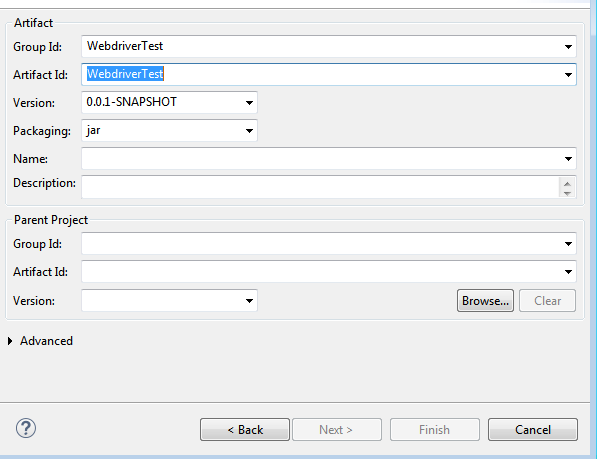
1. In Eclipse IDE, create a new project by selecting File | New | Other from Eclipse menu.
2. On the New dialog, select Maven | Maven Project and click Next.



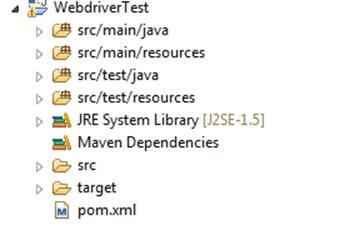
1. On the New Maven Project dialog select the Create a simple project and click Next.



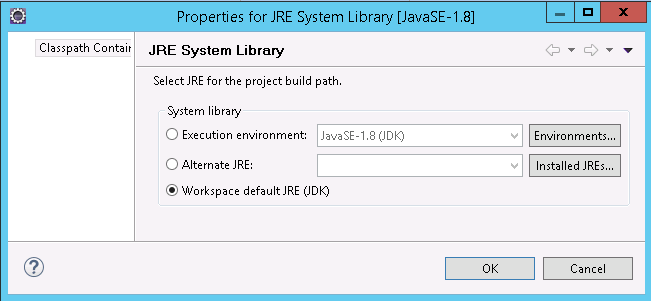
1. Enter WebdriverTest in Group Id: and Artifact Id: and click finish



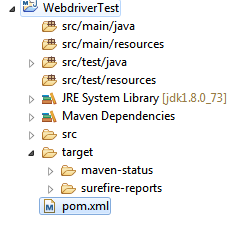
1. Eclipse will create WebdriverTest with following structure:

[](http://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin5.jpg)

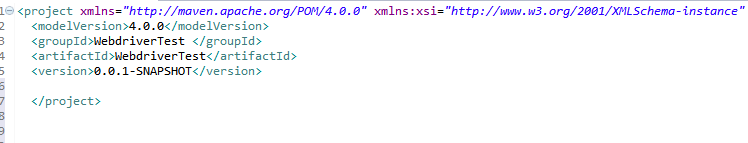
1. Right click on JRE System Library and select the Properties option from the menu and select workspace default JRE (JDK).

****

1. Select pom.xml from Project Explorer.



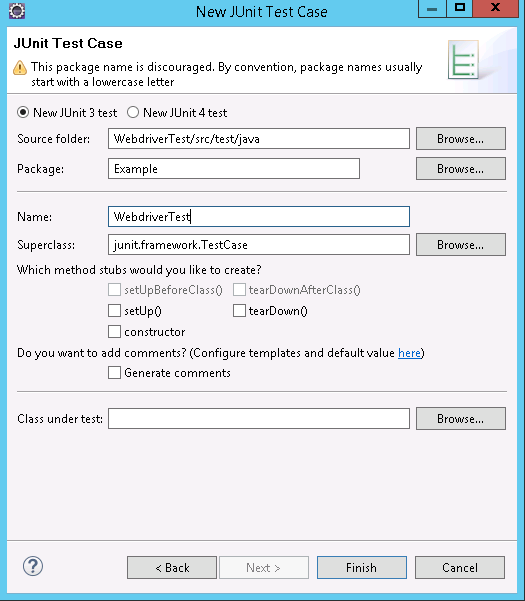
pom.xml file will Open in Editor section.



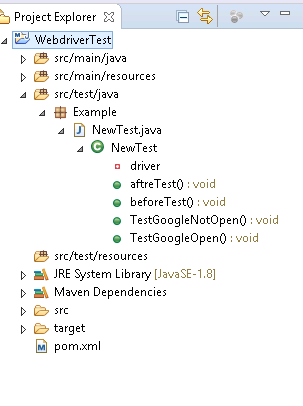
1. Add the Selenium, TestNG, chrome Driver and JUnit dependencies to pom.xml in the <project> node:



1. Create a New JUnit Class. Enter Package name as "example" and "NewTest" in the Name: textbox and click on the Finish button as shown in the following screenshot:



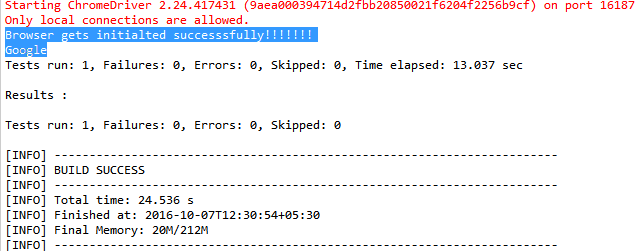
1. Eclipse will create the NewTest class as shown in the following screenshot:



1. Add the following code to the NewTest class:



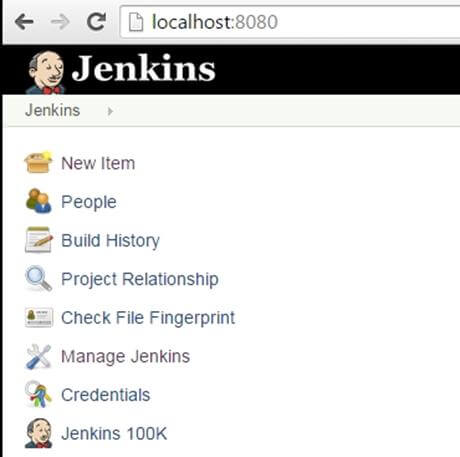
1. To run th**e** tests in the Maven lifecycle, Right-click on the WebdriverTest and select **Run As** | **Maven test**. Maven will execute test from the project.



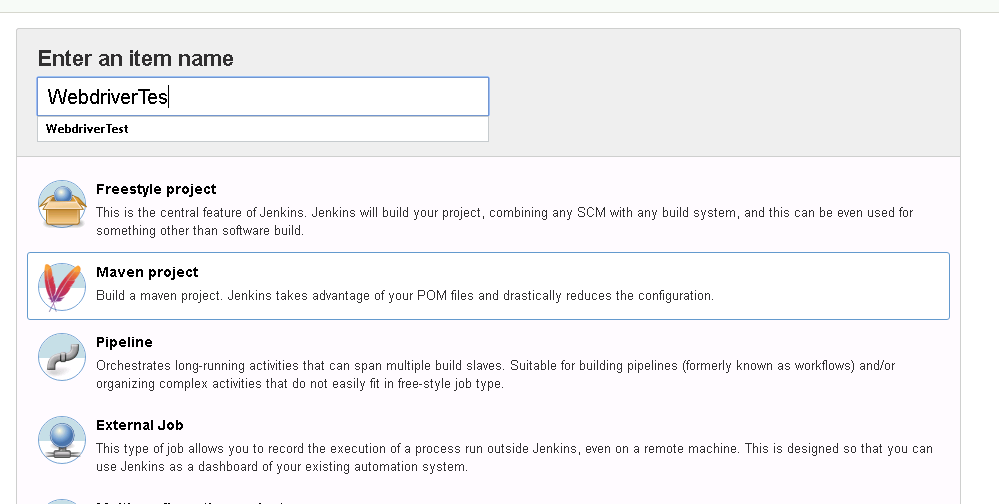
Make sure that build finished successfully

## Steps to Install Jenkins, Maven and Selenium Configuration for Testing

1. Go to <http://jenkins-ci.org/>and download correct package for your OS. Install Jenkins.
2. Once installation is done, navigate to the Jenkins Dashboard (http://localhost:8080 by default) in the browser window.

[](http://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin21.jpg)

1. Click on the **New Item** link to create a new job.
2. Select the Maven project radio button.

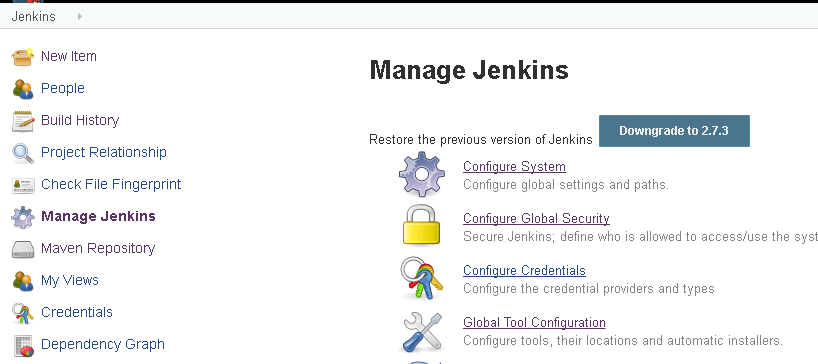


Using the Build a **Maven Project** option, Jenkins supports building and testing Maven projects.

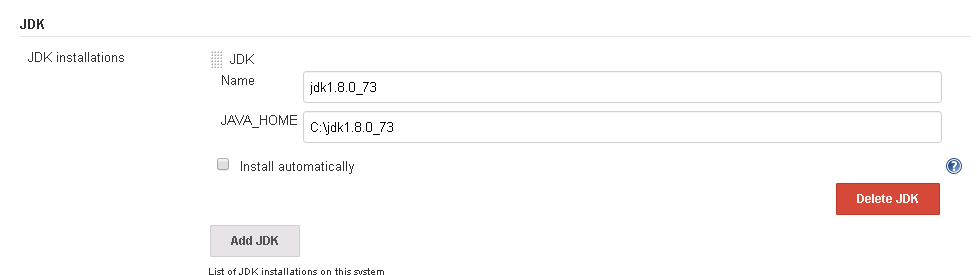
1. Click on OK button. A new job with name "WebdriverTest" is created in Jenkins Dashboard.

[Maven & Jenkins with Selenium: Complete Tutorial](http://cdn.guru99.com/images/5-2015/050115_1023_MavenJenkin24.jpg)

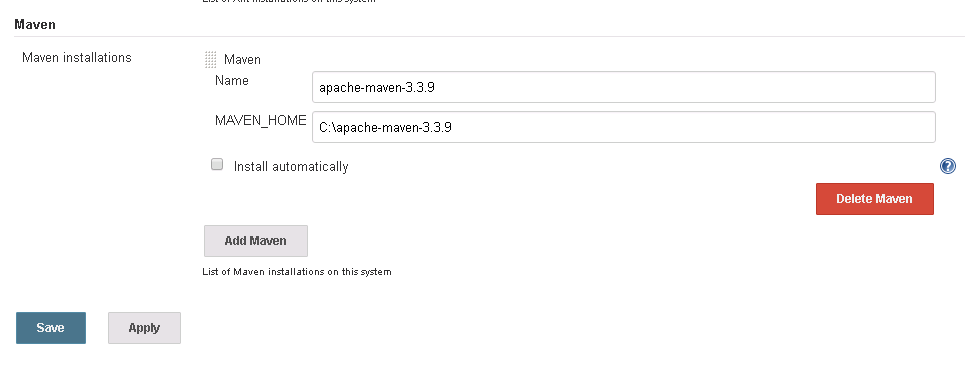
1. Go to **Manage Jenkins** => **Global Tool Configuration**.



1. Click on JDK installations and configure JDK.



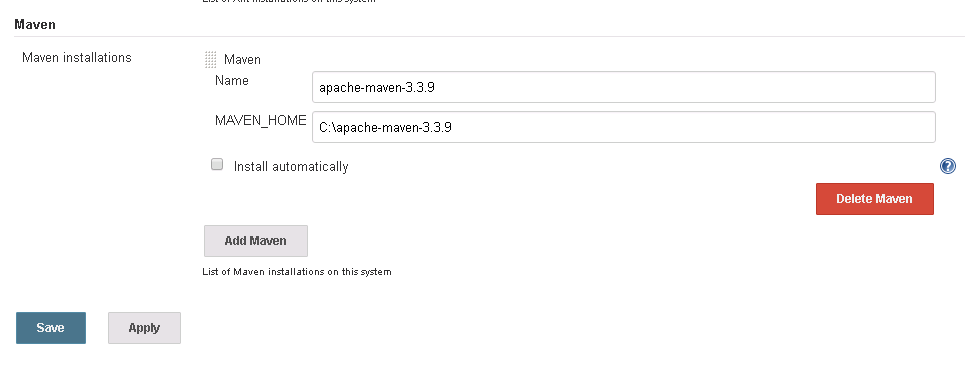
1. Click on Add Maven and configure it.



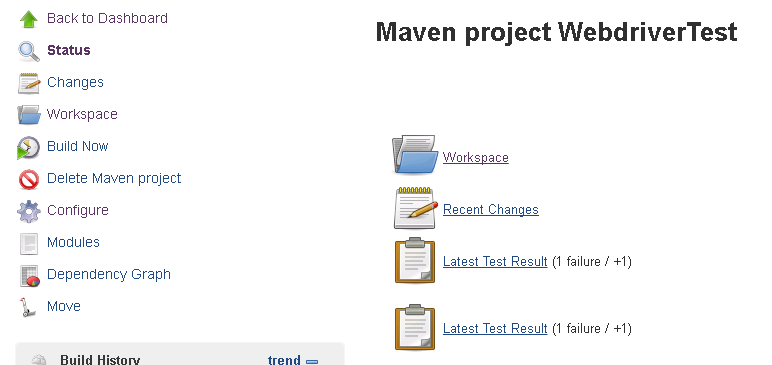
1. Go to the **Build** section of new job.

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

Goal ‘**test’** will run the test case.

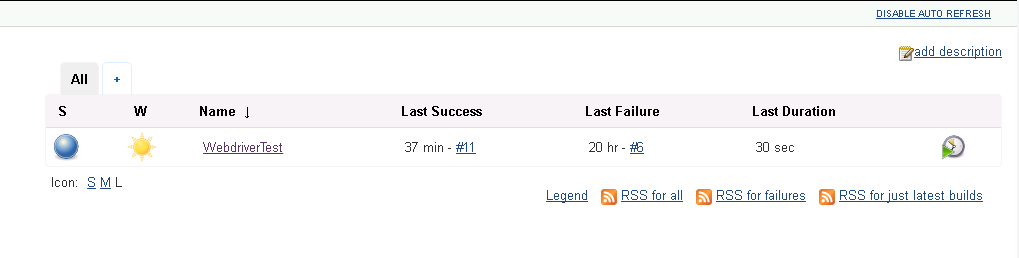


1. Click on Apply button below the page.
2. On the WebdriverTest project page, click on the Build Now link.

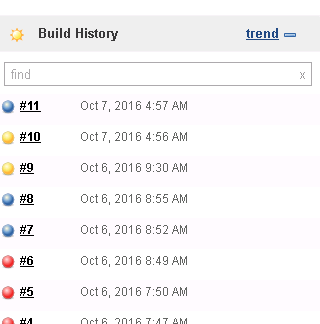


Maven will build the project. It will execute the test cases

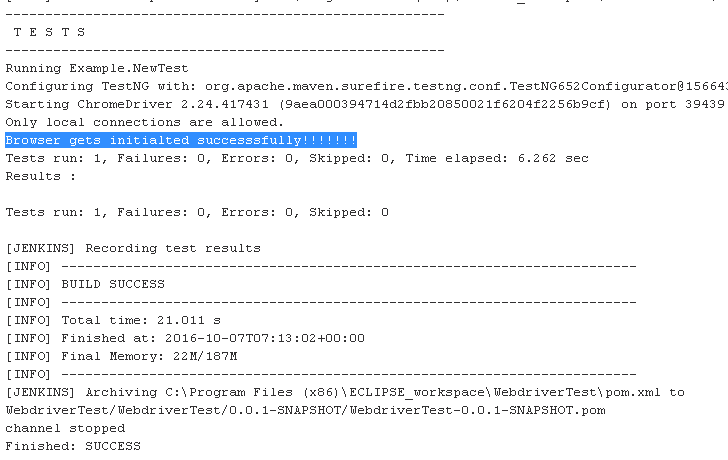
1. once the build process is completed, in Jenkins Dashboard click on the **WebdriverTest** project



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

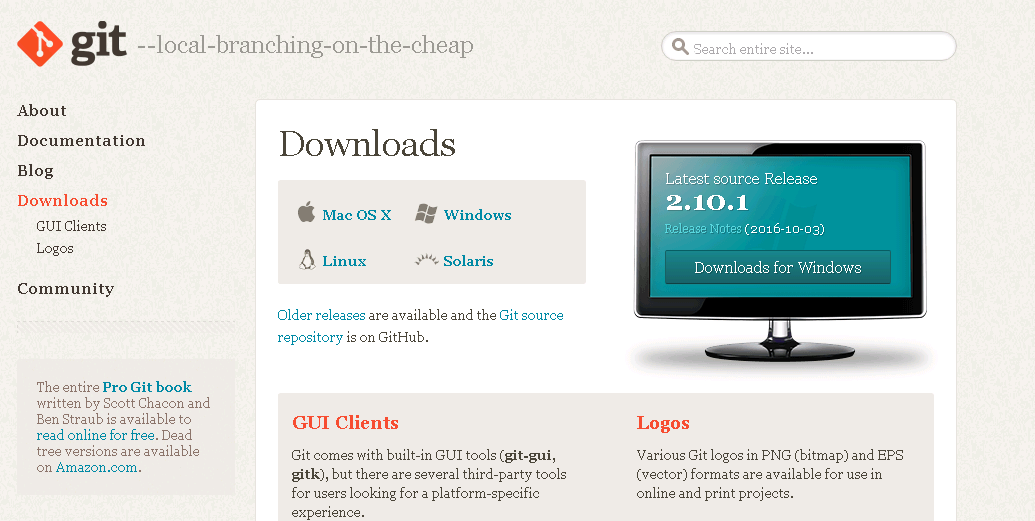


1. Select specific build and you will see the current status by clicking on "**console output**"

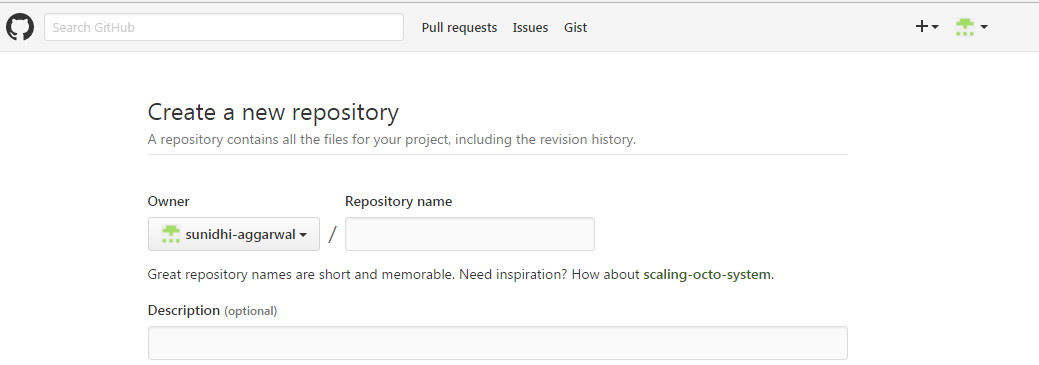


## Clonning Gitbash to GitHub and integrate it with Jenkins for Testing

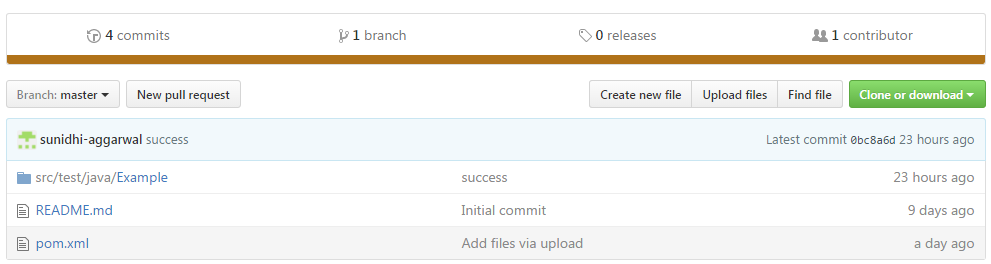
1. Create a new Account on Github using the provided link https://github.com/ .
2. Download GitBash according to your requirement. You can use the provided link <https://git-scm.com/downloads>.



1. Create a new repository on Github.



1. Upload a pom.xml of in repository by clicking on choose your files.



5. Install GitBash on your machine.

6. Open the GitBash prompt on your machine and clone it to Github through Commands.

Here are the following commands: -

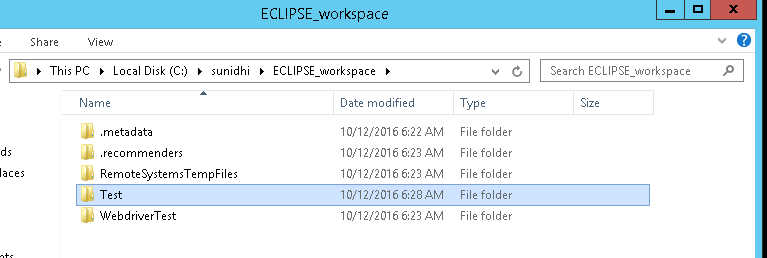
(i) Firstly, move to the directory where your project is created.

**$cd c:**

(ii) Move to the folder where your eclipse workspace is created.

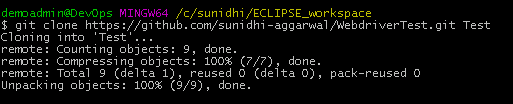
**$cd <<foldername>>/ ECLIPSE\_workspace**

(iii) Create a new folder in Eclipse\_workspace named it as “Test”.

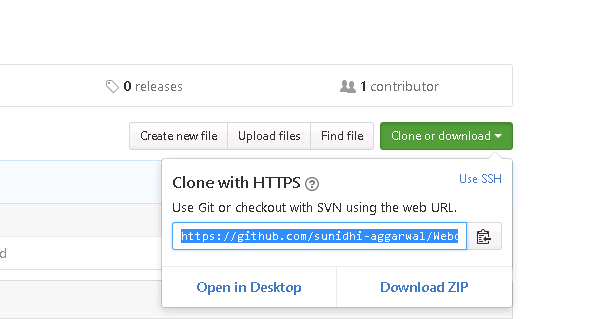


(iv) Now, clone the project created “WebDriverTest” in folder “Test” using the following command:

**$ git clone https://github.com/sunidhi-aggarwal/WebdriverTest.git Test**



In the above command, provide the path of your github repository along with the name of your newly created folder



You can get the path by click on green colored “**Clone or download**” tab option.

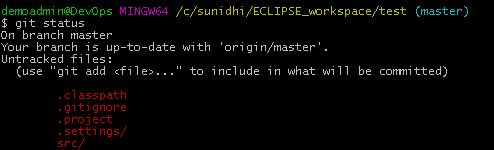
(v) On the gitbash command prompt, change the directory to test folder

**$cd Test**

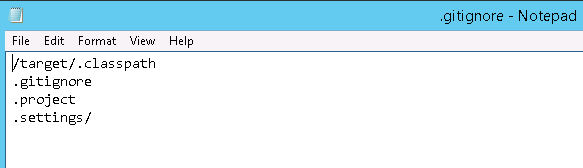
(vi) Use following command to check Git status:

**$cd git status**

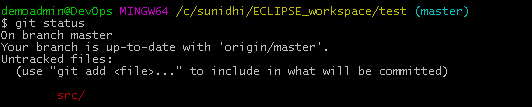
If there are many folders or files occur in test folder than we have to ignore all because we need only src folder.



(vi) Edit the .gitignore file in the Test folder and put all files and folder names in its remaining **src** folder.



(vii) Again check **git status** to verify the **src** folder

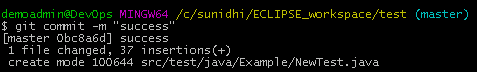


(vii) Add all files from src using

**$ git add –all**

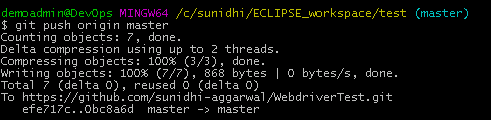
(viii) Commit all changes using,

**$ git commit -m "success"**

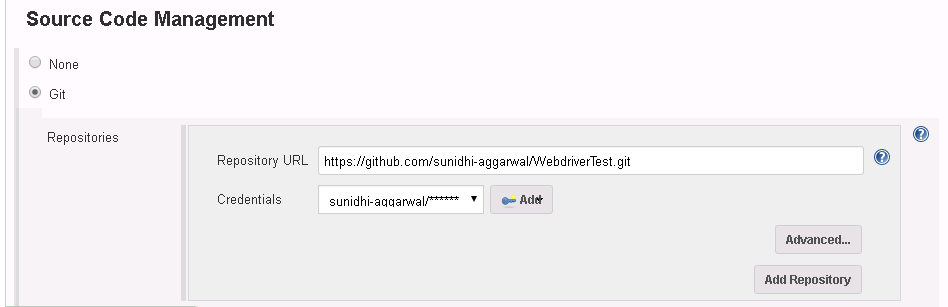


(ix) Push Project to Github using

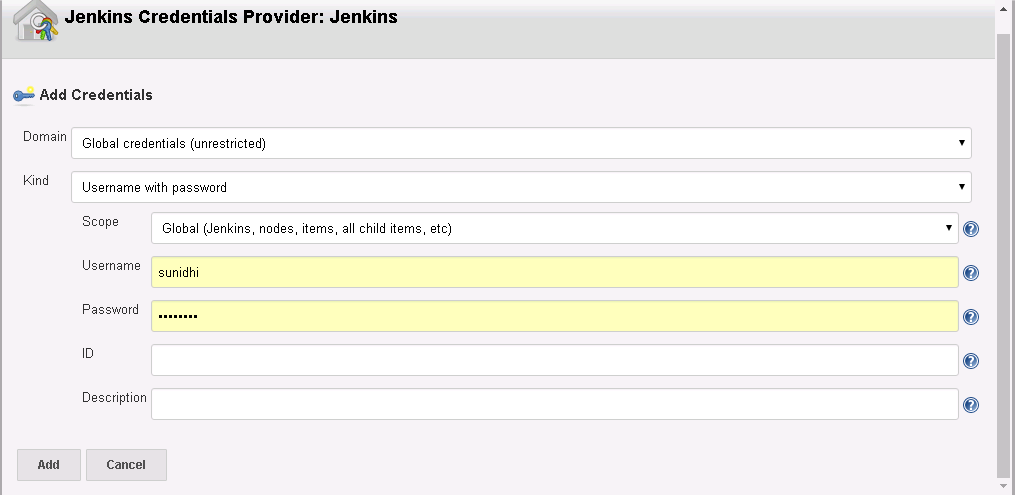
**$ git push origin master**



1. Now open Jenkins and create a new item in it.
2. Click on Configure.
3. Go to source Code Management-> Select Git.
4. Enter the URL of Your Github Repository.



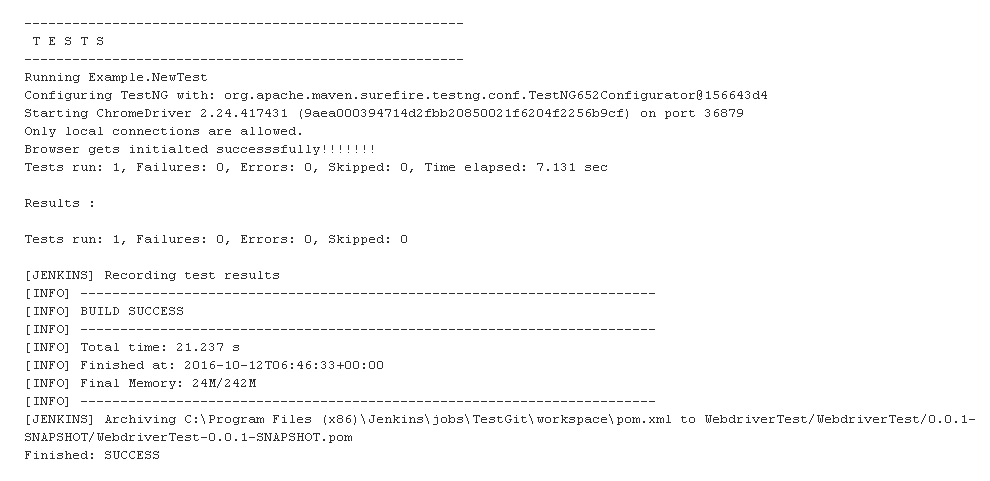
1. Also add the GitHub Credentials.



1. Enter the goal as “Test” to test the test cases.



1. Finally, Click on save and then apply it.
2. Click on **Build now** and check its output in console output window.



## References:

1) <http://www.guru99.com/maven-jenkins-with-selenium-complete-tutorial.html>

2) <https://www.tutorialspoint.com/jenkins/jenkins_maven_setup.htm>

3) <http://www.guru99.com/selenium-tutorial.html>

4) <http://www.tutorialspoint.com/maven/>