Doc 08 Jenkins

Plz ensure:

Jenkins intsall —- Done

Tomcat setup — Done

Git setup — Done

Maven Setup — Done

Configuration –Done - JENKINS\_HOME" environment variable

Management –Done

Setup Build Jobs — Done

Install jdk 8 or JDK 17- pre requisite for jenkins

In the command prompt

Java -version

After installing Jenkins

In cmd plz check with the below command

D:\>Java -jar Jenkins.war

**Installation steps for Jenkins**

<https://www.simplilearn.com/tutorials/jenkins-tutorial/jenkins-installation-on-windows>

**For installing jenkins**

<https://www.jenkins.io/download/>

Default port for Jenkins [8080..or](http://8080..or) 8081 if tomcat is setup

Start → env→ click on environmental variables button

Set path → give opf the location of bin of Tomcat

In both admin and system variables

Set JENKINS\_HOME = give the path of Jenkins

Set it for both admin and system variables..

If error 👍

unable to access jarfile jenkins.war

Open Command Prompt as Administrator in Windows . Go to the directory where Jenkins is installed. and stop the Jenkins service first, using jenkins.exe stop

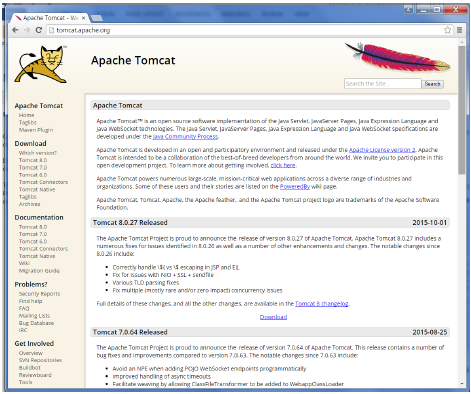
type the command to change the port using, java -jar jenkins.war

**Set up tomcat**

<https://www.tutorialspoint.com/jenkins/jenkins_tomcat_setup.htm>

## Download Tomcat

The official website for tomcat is [Tomcat](http://tomcat.apache.org/). If you click the given link, you can get the home page of the tomcat official website as shown below.



Browse to the link <https://tomcat.apache.org/download-70.cgi> to get the download for tomcat.

## Step 4: Jenkins and Tomcat Setup

Copy the Jenkis.war file which was downloaded from the previous section and copy it to the webapps folder in the tomcat folder.

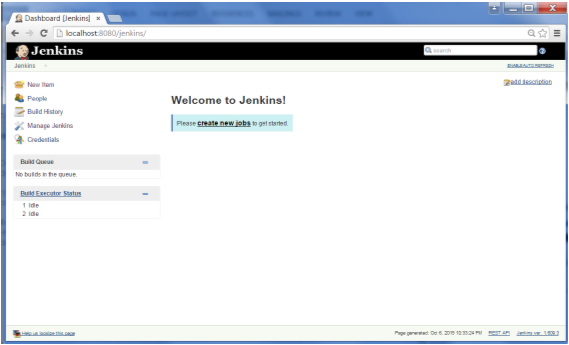
Now open the command prompt. From the command prompt, browse to the directory where the tomcat7 folder is location. Browse to the bin directory in this folder and run the start.bat file

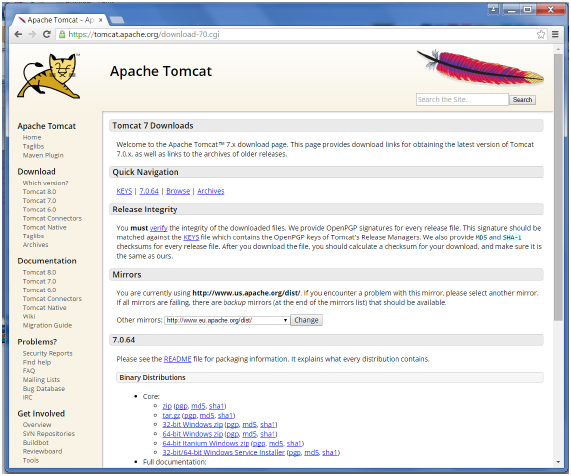
E:\Apps\tomcat7\bin>startup.bat

Once the processing is complete without major errors, the following line will come in the output of the command prompt.

INFO: Server startup in 1302 ms

Open the browser and go to the link − **http://localhost:8080/jenkins**. Jenkins will be up and running on tomcat.





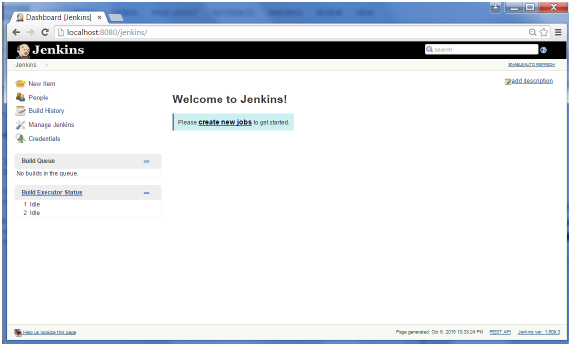
Go to the Binary Distributions section. Download the 32-bit Windows zip file.

Then unzip the contents of the downloaded zip file.

—--------------------------------------------------------------------------------------------

Git setup

For this exercise, you have to ensure that Internet connectivity is present from the machine on which Jenkins is installed. In your Jenkins Dashboard (Home screen), click the Manage Jenkins option on the left hand side.



In the next screen, click the Manage Plugins option.



In the next screen, click the Available tab. This tab will give a list of plugins which are available for downloading. In the Filter tab type Git plugin



The list will then be filtered. Check the Git Plugin option and click on the button Install without restart



The installation will then begin and the screen will be refreshed to show the status of the download.



Once all installations are complete, restart Jenkins by issue the following command in the browser. **http://localhost:8080/jenkins/restart**

After Jenkins is restarted, Git will be available as an option whilst configuring jobs. To verify, click on New Item in the menu options for Jenkins. Then enter a name for a job, in the following case, the name entered is Demo. Select Freestyle project as the item type. Click the Ok button.



In the next screen, if you browse to the Source code Management section, you will now see Git as an option.

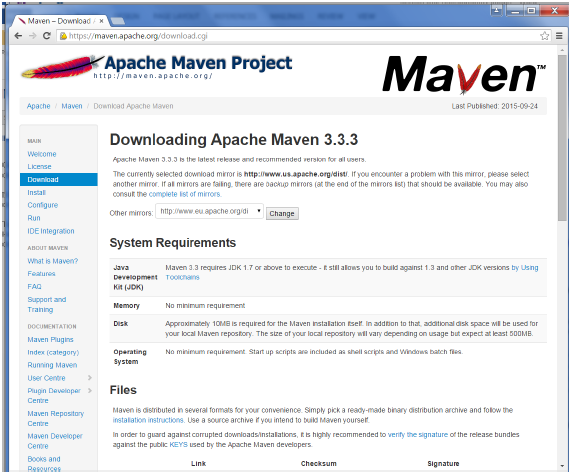


—-------------------------------------------------------------

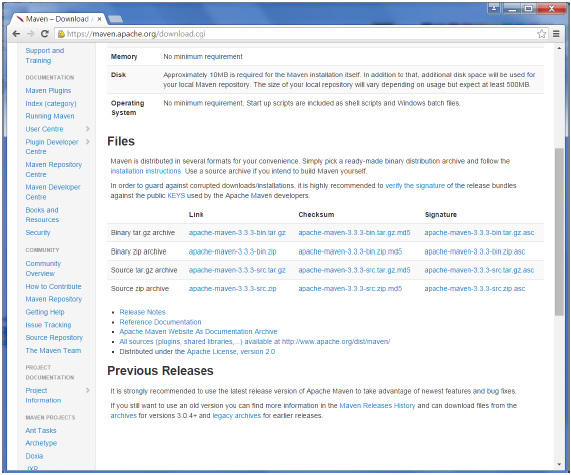
Maven Set up

## Step 1: Downloading and Setting Up Maven

The official website for maven is [Apache Maven](https://maven.apache.org/download.cgi). If you click the given link, you can get the home page of the maven official website as shown below.



While browsing to the site, go to the Files section and download the link to the Binary.zip file.

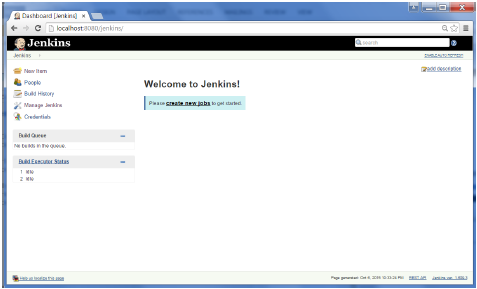


Once the file is downloaded, extract the files to the relevant application folder. For this purpose, the maven files will be placed in E:\Apps\apache-maven-3.3.3.

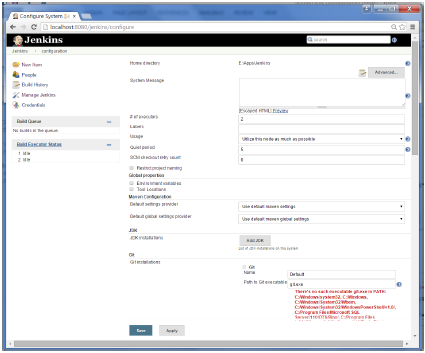
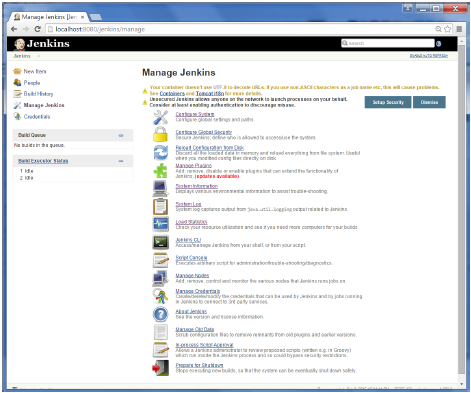
## 

## Step 2: Setting up Jenkins and Maven

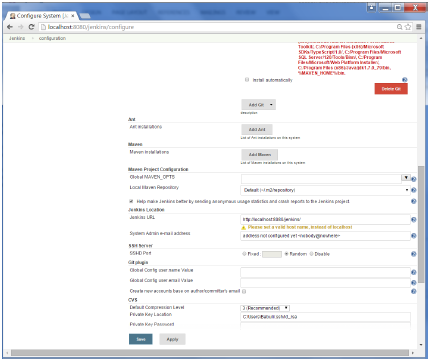
In the Jenkins dashboard (Home screen), click Manage Jenkins from the left-hand side menu.



Then, click on Configure System from the right hand side.



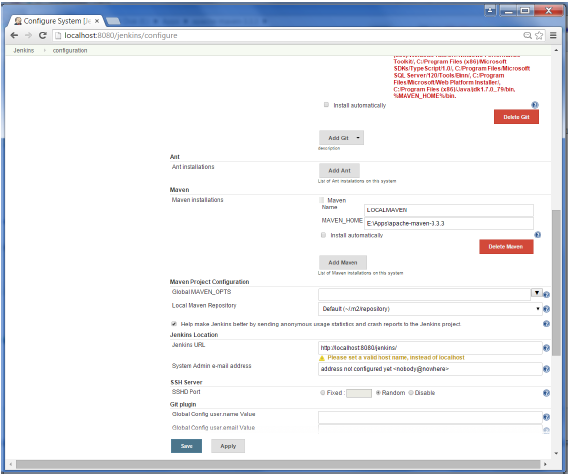
In the Configure system screen, scroll down till you see the Maven section and then click on the Add Maven button.



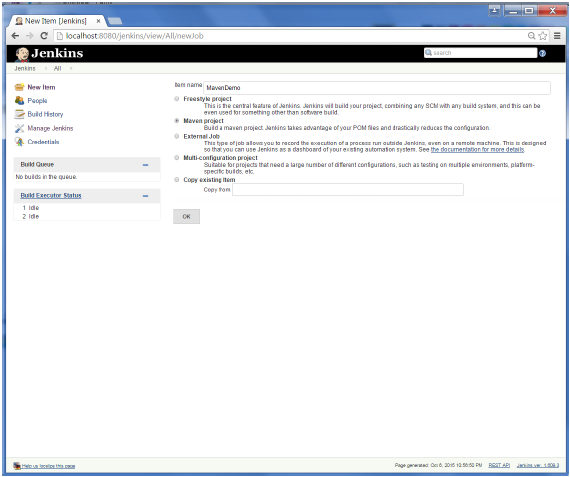
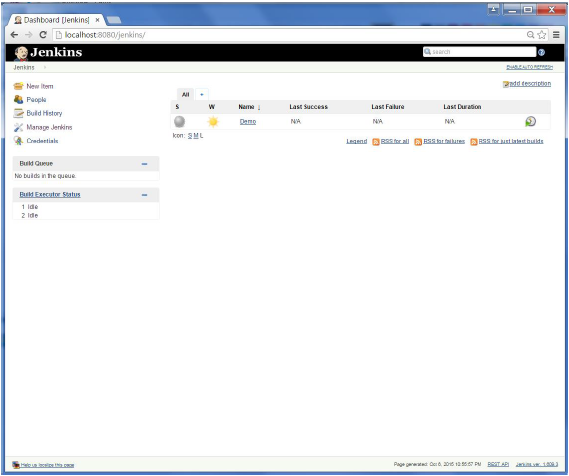
Uncheck the Install automatically option.

Add any name for the setting and the location of the MAVEN\_HOME.

Then, click on the Save button at the end of the screen.



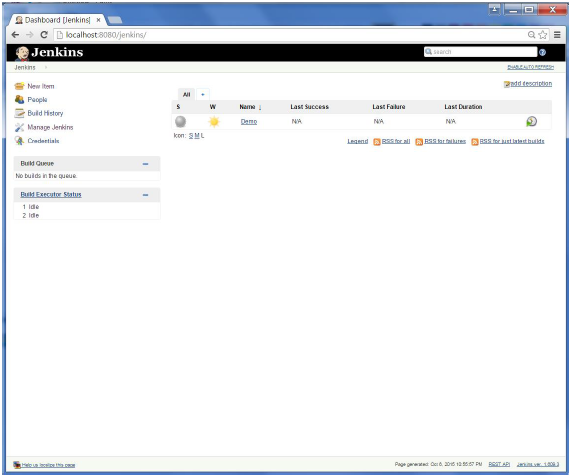
You can now create a job with the Maven project option. In the Jenkins dashboard, click the New Item option.



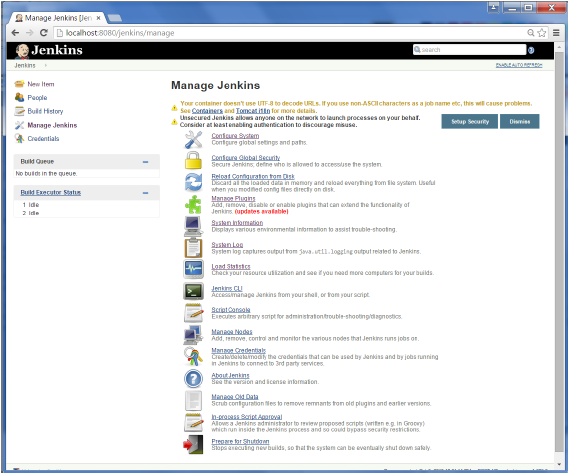
Jenkins - Configuration:

You probably would have seen a couple of times in the previous exercises wherein we had to configure options within Jenkins. The following shows the various configuration options in Jenkins.

So one can get the various configuration options for Jenkins by clicking the Manage Jenkins option from the left hand menu side.



You will then be presented with the following screen −



Click on Configure system. Discussed below are some of the Jenkins configuration settings which can be carried out.

## Jenkins Home Directory

Jenkins needs some disk space to perform builds and keep archives. One can check this location from the configuration screen of Jenkins. By default, this is set to ~/.jenkins, and this location will initially be stored within your user profile location. In a proper environment, you need to change this location to an adequate location to store all relevant builds and archives. Once can do this in the following ways

* Set "JENKINS\_HOME" environment variable to the new home directory before launching the servlet container.
* Set "JENKINS\_HOME" system property to the servlet container.
* Set JNDI environment entry "JENKINS\_HOME" to the new directory.

The following example will use the first option of setting the "JENKINS\_HOME" environment variable.

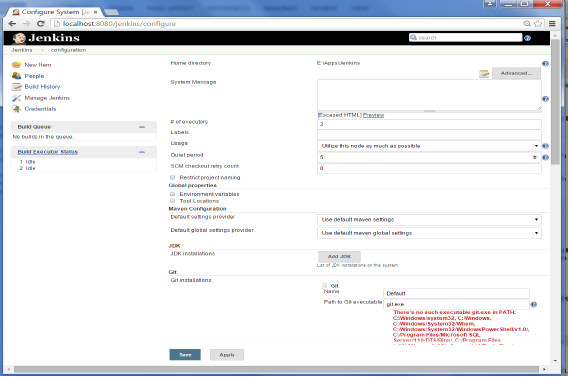
First create a new folder E:\Apps\Jenkins. Copy all the contents from the existing ~/.jenkins to this new directory.

Set the JENKINS\_HOME environment variable to point to the base directory location where Java is installed on your machine. For example,

| **OS** | **Output** |
| --- | --- |
| Windows | Set Environmental variable JENKINS\_HOME to youre the location you desire. As an example you can set it to E:\Apps\Jenkins |
| Linux | export JENKINS\_HOME =/usr/local/Jenkins or the location you desire. |

In the Jenkins dashboard, click Manage Jenkins from the left hand side menu. Then click on Configure System from the right hand side.

In the Home directory, you will now see the new directory which has been configured.



## # of executors

This refers to the total number of concurrent job executions that can take place on the Jenkins machine. This can be changed based on requirements. Sometimes the recommendation is to keep this number the same as the number of CPU on the machines for better performance.

## Environment Variables

This is used to add custom environment variables which will apply to all the jobs. These are key-value pairs and can be accessed and used in Builds wherever required.

## Jenkins URL

By default, the Jenkins URL points to localhost. If you have a domain name setup for your machine, set this to the domain name else overwrite localhost with IP of machine. This will help in setting up slaves and while sending out links using the email as you can directly access the Jenkins URL using the environment variable JENKINS\_URL which can be accessed as ${JENKINS\_URL}.

## Email Notification

In the email Notification area, you can configure the SMTP settings for sending out emails. This is required for Jenkins to connect to the SMTP mail server and send out emails to the recipient list.

—---------------------------------------------------------------------------------------------------------------------

JEnkins Home

—---------------------------------------------------------------------------------------------------------------------

## Jenkins Home Directory

Jenkins needs some disk space to perform builds and keep archives. One can check this location from the configuration screen of Jenkins. By default, this is set to ~/.jenkins, and this location will initially be stored within your user profile location. In a proper environment, you need to change this location to an adequate location to store all relevant builds and archives. Once can do this in the following ways

* Set "JENKINS\_HOME" environment variable to the new home directory before launching the servlet container.
* Set "JENKINS\_HOME" system property to the servlet container.
* Set JNDI environment entry "JENKINS\_HOME" to the new directory.

The following example will use the first option of setting the "JENKINS\_HOME" environment variable.

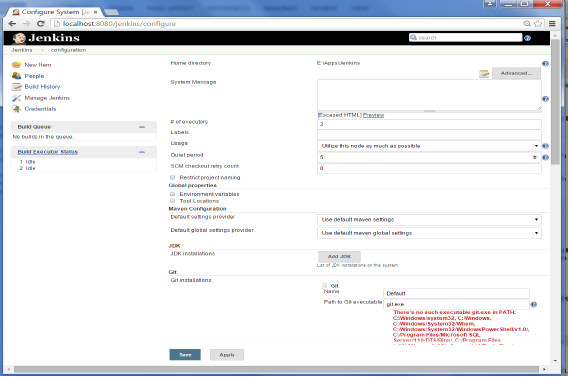
First create a new folder E:\Apps\Jenkins. Copy all the contents from the existing ~/.jenkins to this new directory.

Set the JENKINS\_HOME environment variable to point to the base directory location where Java is installed on your machine. For example,

| **OS** | **Output** |
| --- | --- |
| Windows | Set Environmental variable JENKINS\_HOME to youre the location you desire. As an example you can set it to E:\Apps\Jenkins |
| Linux | export JENKINS\_HOME =/usr/local/Jenkins or the location you desire. |

In the Jenkins dashboard, click Manage Jenkins from the left hand side menu. Then click on Configure System from the right hand side.

In the Home directory, you will now see the new directory which has been configured.



## # of executors

This refers to the total number of concurrent job executions that can take place on the Jenkins machine. This can be changed based on requirements. Sometimes the recommendation is to keep this number the same as the number of CPU on the machines for better performance.

## Environment Variables

This is used to add custom environment variables which will apply to all the jobs. These are key-value pairs and can be accessed and used in Builds wherever required.

## Jenkins URL

By default, the Jenkins URL points to localhost. If you have a domain name setup for your machine, set this to the domain name else overwrite localhost with IP of machine. This will help in setting up slaves and while sending out links using the email as you can directly access the Jenkins URL using the environment variable JENKINS\_URL which can be accessed as ${JENKINS\_URL}.

## Email Notification

In the email Notification area, you can configure the SMTP settings for sending out emails. This is required for Jenkins to connect to the SMTP mail server and send out emails to the recipient list.

—---------------------------------------------------------------------------------------------------------------------------

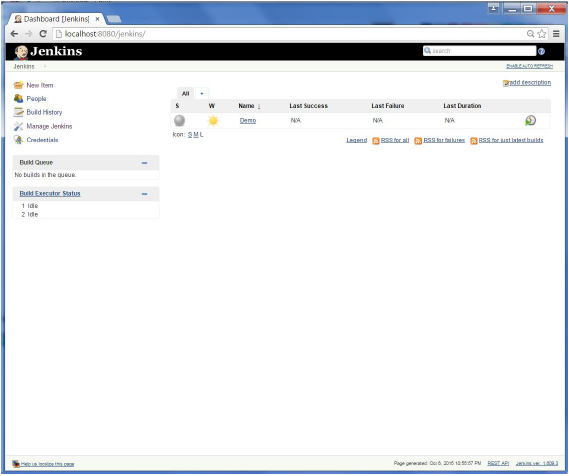
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

-------------------------------------------------------------------------------------------------------------------------------

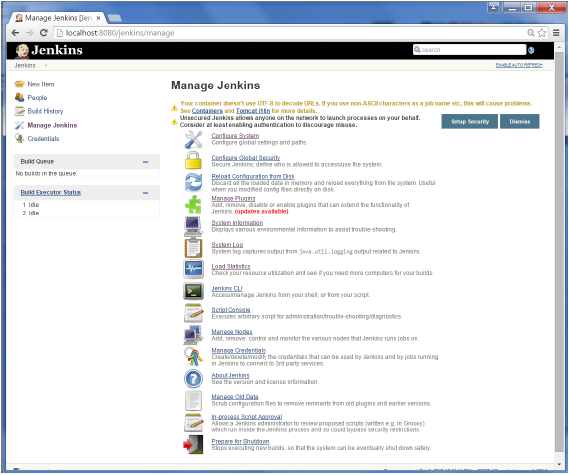
Jenkin Management

To manage Jenkins, click on the Manage Jenkins option from the left hand menu side.

So one can get the various configuration options for Jenkins by clicking the Manage Jenkins option from the left hand menu side.



You will then be presented with the following screen −



Some of the management options are as follows −

## Configure System

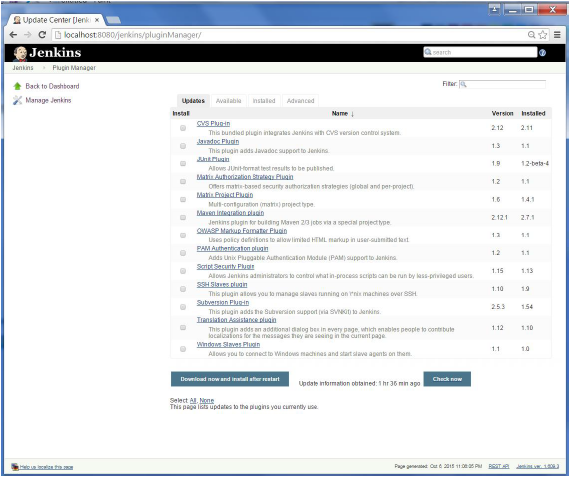
This is where one can manage paths to the various tools to use in builds, such as the JDKs, the versions of Ant and Maven, as well as security options, email servers, and other system-wide configuration details. When plugins are installed. Jenkins will add the required configuration fields dynamically after the plugins are installed.

## Reload Configuration from Disk

Jenkins stores all its system and build job configuration details as XML files which is stored in the Jenkins home directory. Here also all of the build history is stored. If you are migrating build jobs from one Jenkins instance to another, or archiving old build jobs, you will need to add or remove the corresponding build job directories to Jenkinss builds directory. You dont need to take Jenkins offline to do thisyou can simply use the Reload Configuration from Disk option to reload the Jenkins system and build job configurations directly.

## Manage Plugin

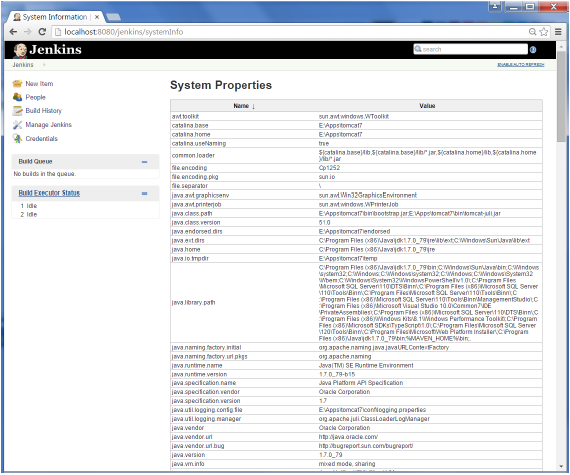
Here one can install a wide variety of third-party plugins right from different Source code management tools such as Git, Mercurial or ClearCase, to code quality and code coverage metrics reporting. Plugins can be installed, updated and removed through the Manage Plugins screen.



## System Information

This screen displays a list of all the current Java system properties and system environment variables. Here one can check exactly what version of Java Jenkins is running in, what user it is running under, and so forth.

The following screenshot shows some of the name-value information available in this section.



### System Log

The System Log screen is a convenient way to view the Jenkins log files in real time. Again, the main use of this screen is for troubleshooting.

### Load Statistics

This pages displays graphical data on how busy the Jenkins instance is in terms of the number of concurrent builds and the length of the build queue which gives an idea of how long your builds need to wait before being executed. These statistics can give a good idea of whether extra capacity or extra build nodes is required from an infrastructure perspective.

### Script Console

This screen lets you run Groovy scripts on the server. It is useful for advanced troubleshooting since it requires a strong knowledge of the internal Jenkins architecture.

### Manage nodes

Jenkins is capable of handling parallel and distributed builds. In this screen, you can configure how many builds you want. Jenkins runs simultaneously, and, if you are using distributed builds, set up build nodes. A build node is another machine that Jenkins can use to execute its builds.

### Prepare for Shutdown

If there is a need to shut down Jenkins, or the server Jenkins is running on, it is best not to do so when a build is being executed. To shut down Jenkins cleanly, you can use the Prepare for Shutdown link, which prevents any new builds from being started. Eventually, when all of the current builds have finished, one will be able to shut down Jenkins cleanly.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Now try to Setup Build a job

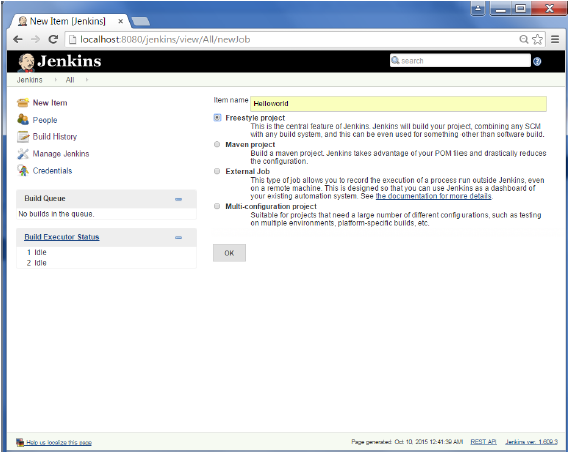
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

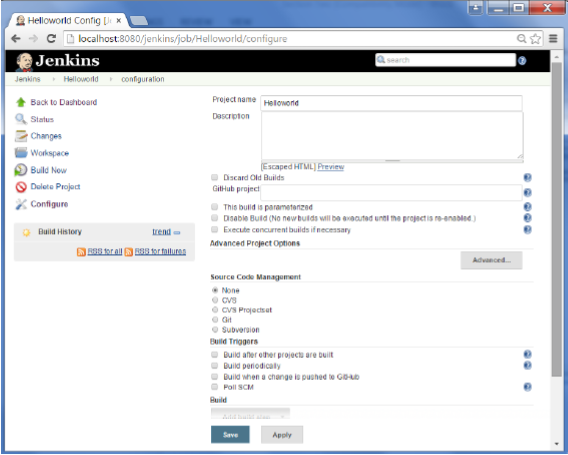
we will create a job in Jenkins which picks up a simple HelloWorld application, builds and runs the java program.

**Step 1** − Go to the Jenkins dashboard and Click on New Item



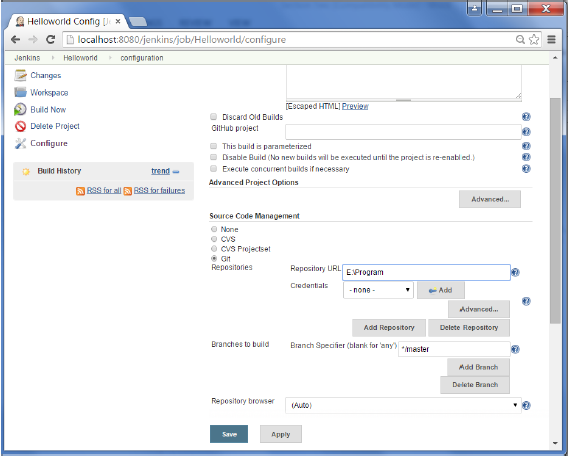
**Step 2** − In the next screen, enter the Item name, in this case we have named it Helloworld. Choose the Freestyle project option



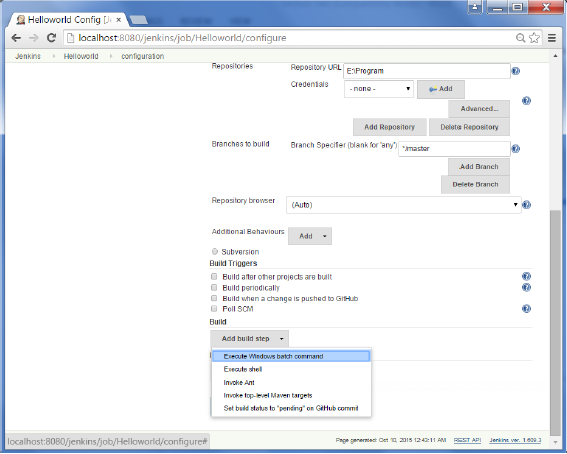
**Step 3** − The following screen will come up in which you can specify the details of the job.

**Step 4** − We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a HelloWorld.java file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

**Note** − If you repository if hosted on Github, you can also enter the url of that repository here. In addition to this, you would need to click on the Add button for the credentials to add a user name and password to the github repository so that the code can be picked up from the remote repository.



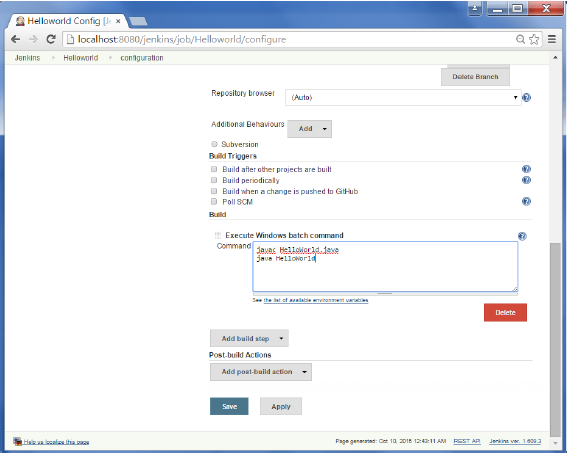
**Step 5** − Now go to the Build section and click on Add build step → Execute Windows batch command



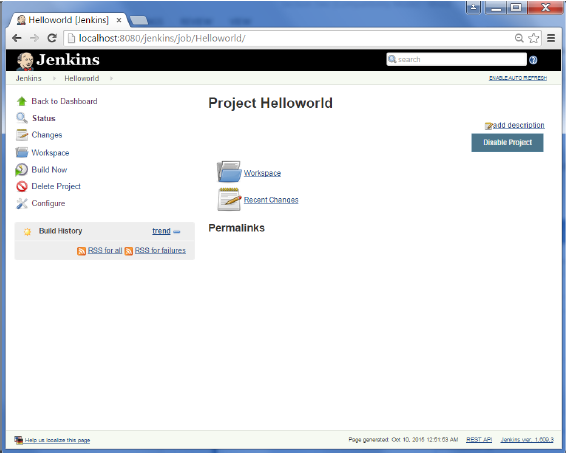
**Step 6** − In the command window, enter the following commands and then click on the Save button.

Javac HelloWorld.java

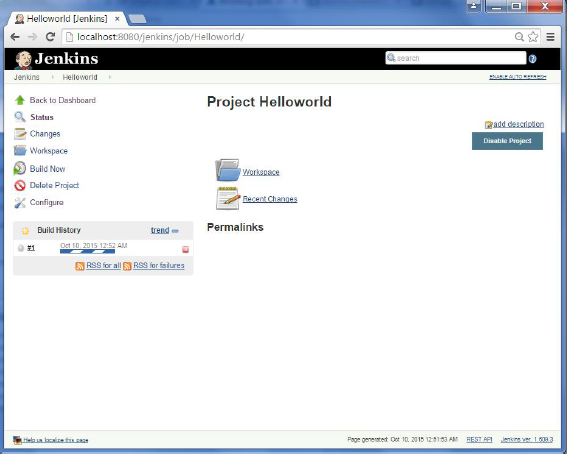
Java HelloWorld



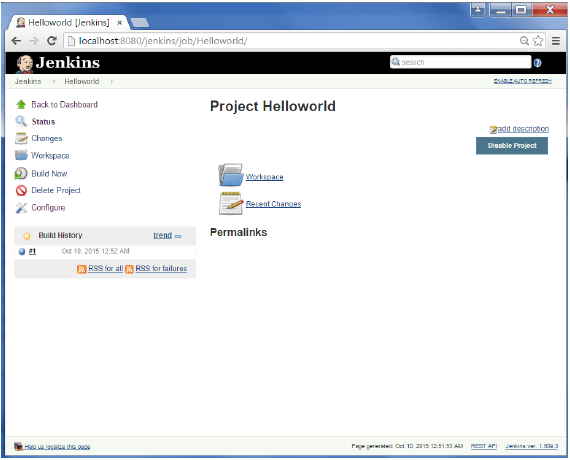
**Step 7** − Once saved, you can click on the Build Now option to see if you have successfully defined the job.



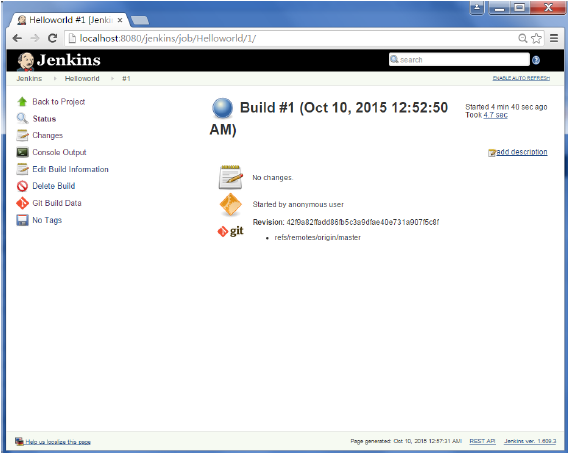
**Step 8** − Once the build is scheduled, it will run. The following Build history section shows that a build is in progress.



**Step 9** − Once the build is completed, a status of the build will show if the build was successful or not. In our case, the following build has been executed successfully. Click on the #1 in the Build history to bring up the details of the build.



**Step 10** − Click on the Console Output link to see the details of the build



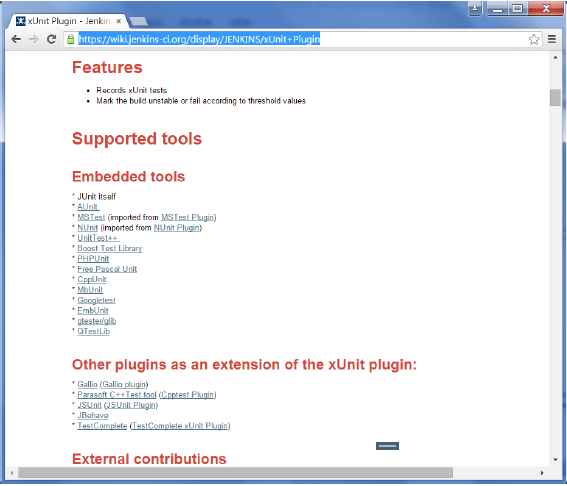
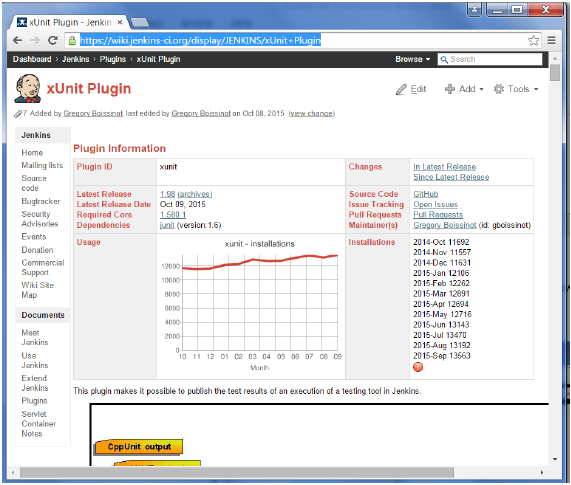
Apart from the steps shown above there are just so many ways to create a build job, the options available are many, which what makes Jenkins such a fantastic continuous deployment tool.

===============================================

UNIT Testing using Jemkins

===============================================

Jenkins provides an out of box functionality for Junit, and provides a host of plugins for unit testing for other technologies, an example being MSTest for .Net Unit tests. If you go to the link <https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin> it will give the list of Unit Testing plugins available.

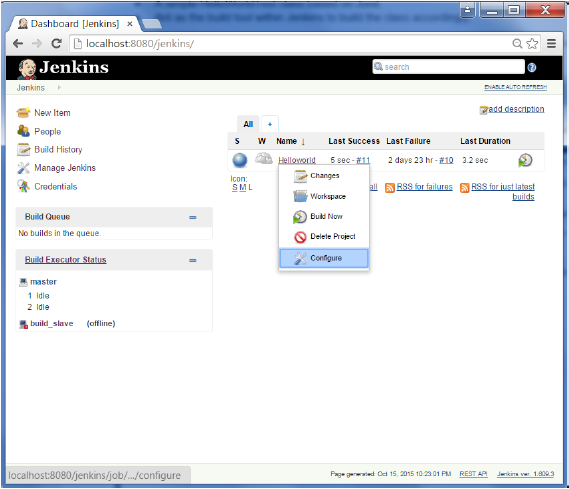


## Example of a Junit Test in Jenkins

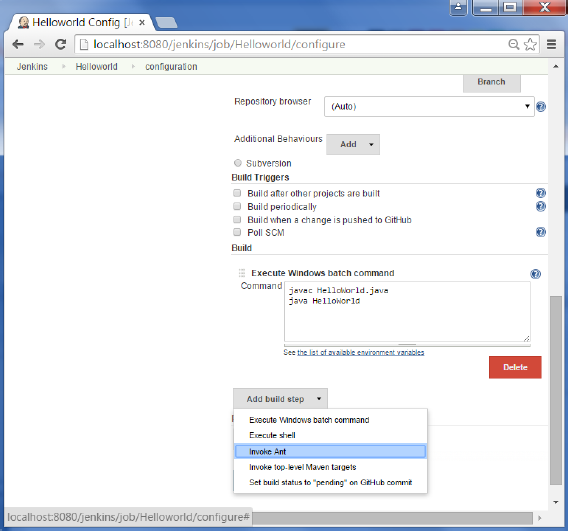
The following example will consider

* A simple HelloWorldTest class based on Junit.
* Ant as the build tool within Jenkins to build the class accordingly.

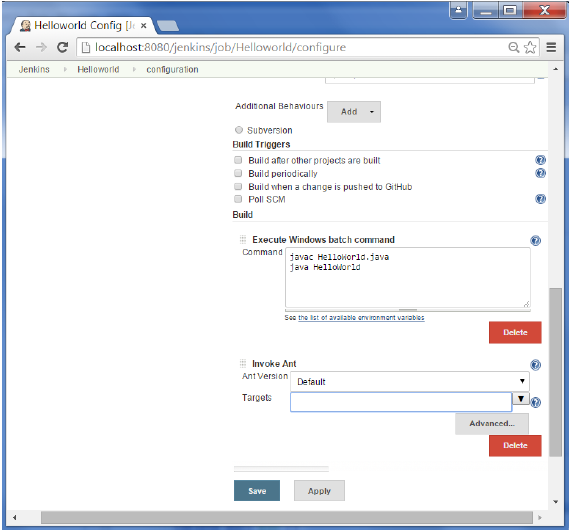
**Step 1** − Go to the Jenkins dashboard and Click on the existing HelloWorld project and choose the Configure option



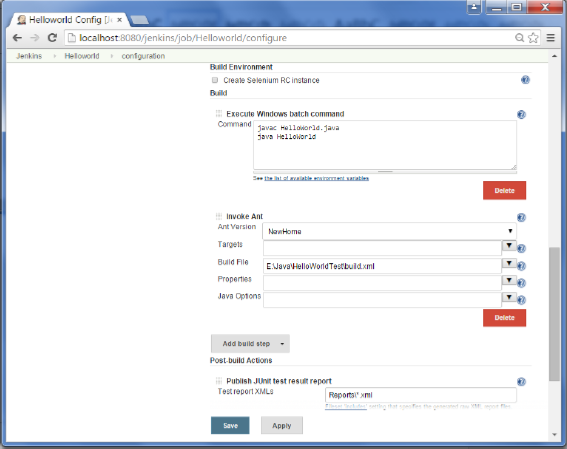
**Step 2** − Browse to the section to Add a Build step and choose the option to Invoke Ant.



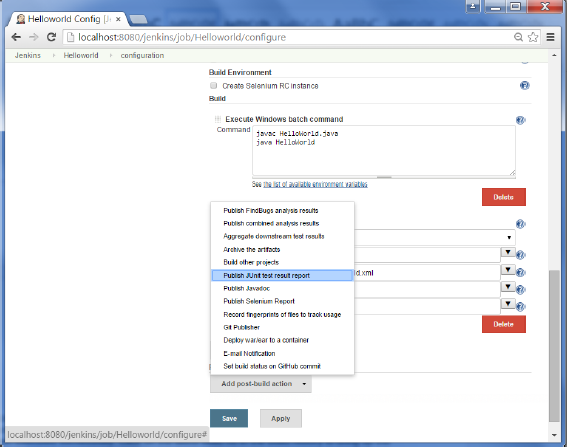
**Step 3** − Click on the Advanced button.



**Step 4** − In the build file section, enter the location of the build.xml file.

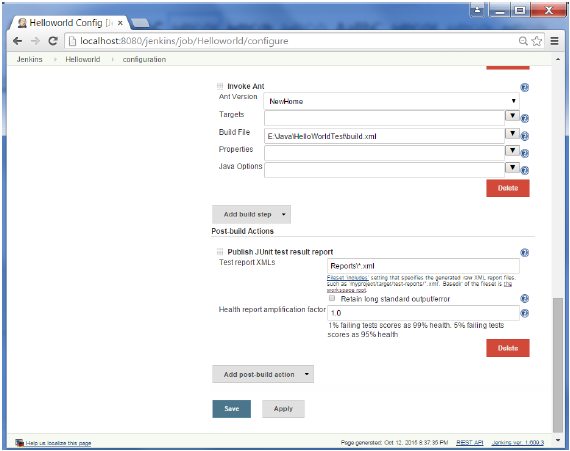


**Step 5** − Next click the option to Add post-build option and choose the option of Publish Junit test result report



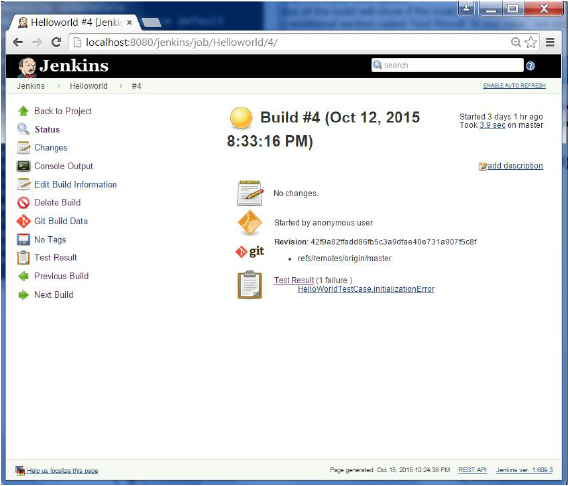
**Step 6** − In the Test reports XMLs, enter the location as shown below. Ensure that Reports is a folder which is created in the HelloWorld project workspace. The \*.xml basically tells Jenkins to pick up the result xml files which are produced by the running of the Junit test cases. These xml files which then be converted into reports which can be viewed later.

Once done, click the Save option at the end.

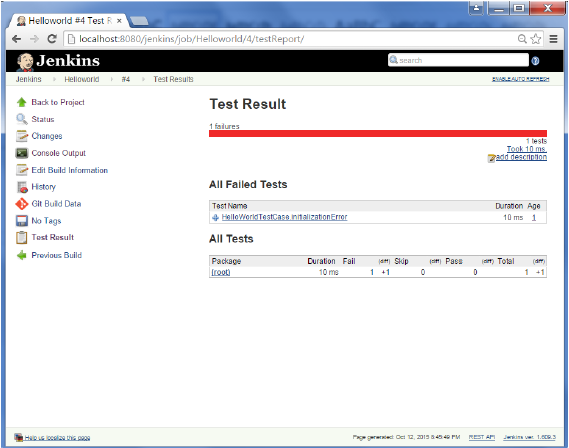


**Step 7** − Once saved, you can click on the Build Now option.

Once the build is completed, a status of the build will show if the build was successful or not. In the Build output information, you will now notice an additional section called Test Result. In our case, we entered a negative Test case so that the result would fail just as an example.



One can go to the Console output to see further information. But whats more interesting is that if you click on Test Result, you will now see a drill down of the Test results.



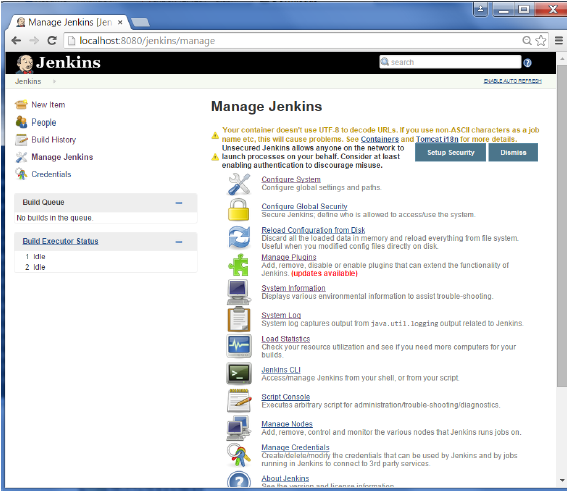
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Lets do Automated Testing

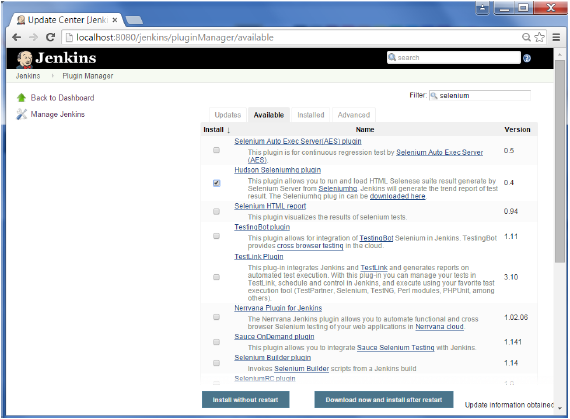
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

One of the basic principles of Continuous Integration is that a build should be verifiable. You have to be able to objectively determine whether a particular build is ready to proceed to the next stage of the build process, and the most convenient way to do this is to use automated tests. Without proper automated testing, you find yourself having to retain many build artifacts and test them by hand, which is hardly in the spirit of Continuous Integration. The following example shows how to use Selenium to run automated web tests.

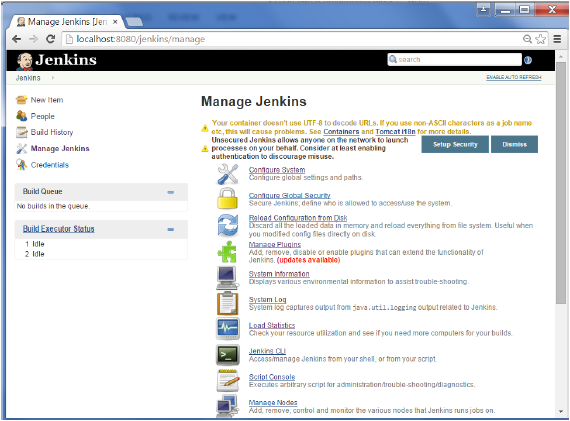
**Step 1** − Go to Manage Plugins.



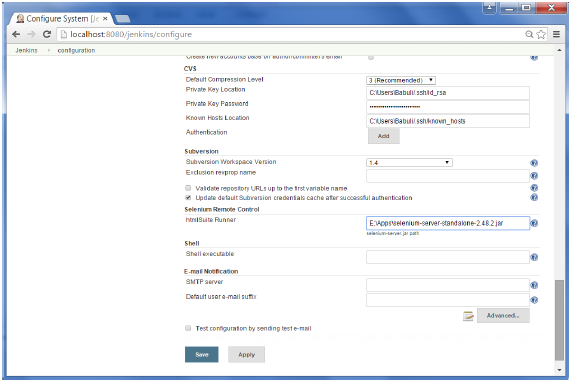
**Step 2** − Find the Hudson Selenium Plugin and choose to install. Restart the Jenkins instance.



**Step 3** − Go to Configure system.

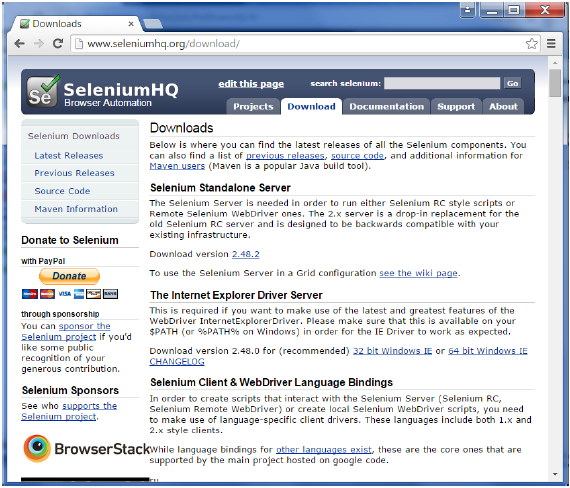


**Step 4** − Configure the selenium server jar and click on the Save button.

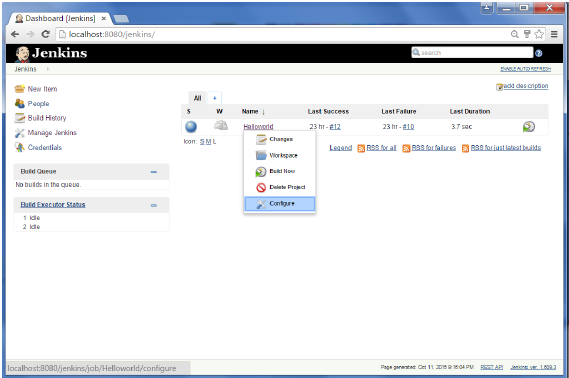


**Note** − The selenium jar file can be downloaded from the location [SeleniumHQ](http://www.seleniumhq.org/download/)

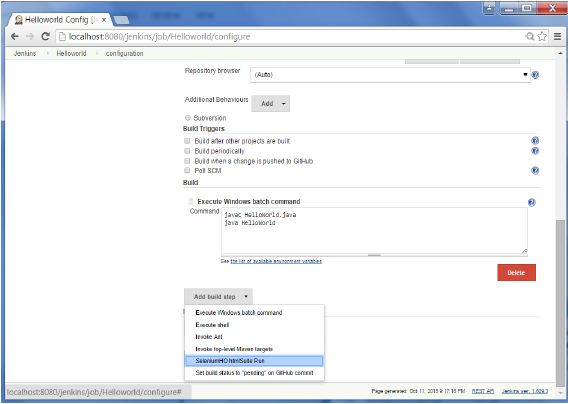
Click on the download for the Selenium standalone server.



**Step 5** − Go back to your dashboard and click on the Configure option for the HelloWorld project.



**Step 6** − Click on Add build step and choose the optin of SeleniumHQ htmlSuite Run



**Step 7** − Add the necessary details for the selenium test. Here the suiteFile is the TestSuite generated by using the Selenium IDE. Click on Save and execute a build. Now the post build will launch the selenium driver, and execute the html test.

