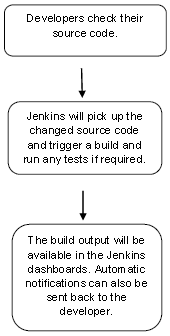
***JENKINS***

**Jenkins** – an open source automation server which enables developers around the world to reliably build, test, and deploy their software.

Why Jenkins?

Jenkins is a software that allows **continuous integration**. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.



Along with Jenkins, sometimes, one might also see the association of **Hudson**. Hudson is a very popular open-source Java-based continuous integration tool developed by Sun Micro systems which was later acquired by Oracle. After the acquisition of Sun by Oracle, a fork was created from the Hudson source code, which brought about the introduction of Jenkins.

What is Continuous Integration?

Continuous Integration is a development practice that requires developers to integrate code into a shared repository at regular intervals. This concept was meant to remove the problem of finding later occurrence of issues in the build lifecycle. Continuous integration requires the developers to have frequent builds. The common practice is that whenever a code commit occurs, a build should be triggered.

Starting Jenkins

Open the command prompt. From the command prompt, browse to the directory where the Jenkins.war file is present. Run the following command

D:\>Java –jar Jenkins.war

After the command is run, various tasks will run, one of which is the extraction of the war file which is done by an embedded web server called win stone.

D:\>Java –jar Jenkins.war

Running from: D:\Jenkins.war

Web root: $user.home/.Jenkins

Sep 29, 2015 4:10:46 PM winstone.Logger logInternal

INFO: Beginning extraction from war file

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

INFO: Jenkins is fully up and running

Accessing Jenkins

Once Jenkins is up and running, one can access Jenkins from the link − **http://localhost:8080**

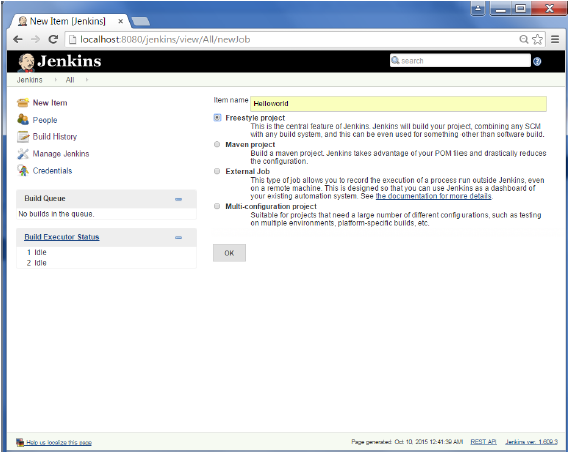
This link will bring up the Jenkins dashboard.

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

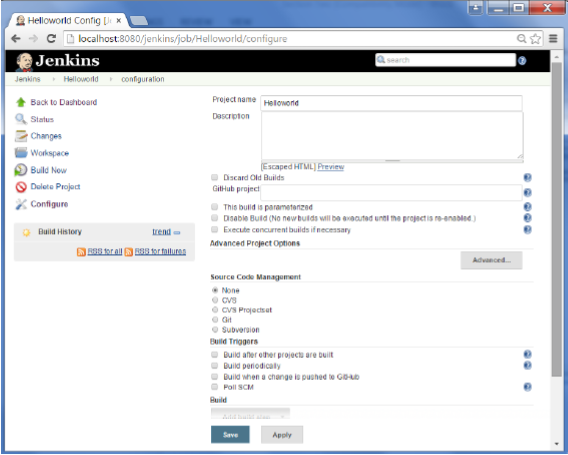
**Step 1** − Go to the Jenkins dashboard and Click on New Item



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

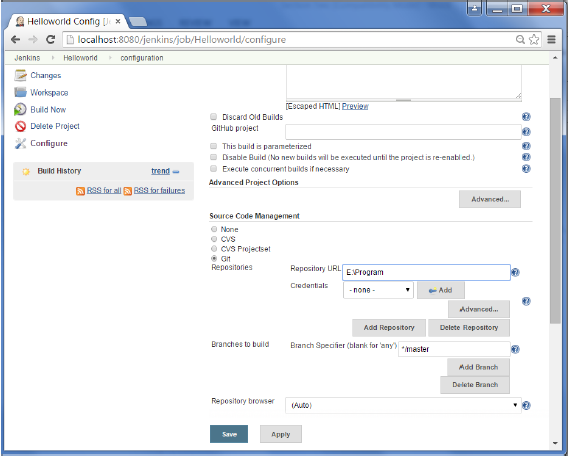


**Step 3** − The following screen will come up in which you can specify the details of the job.

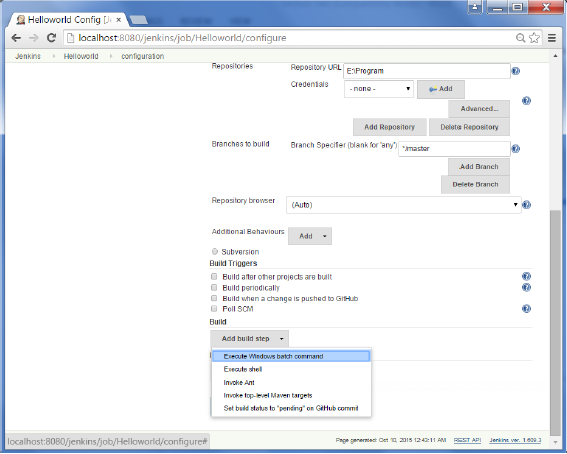


**Step 4** − We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a ‘HelloWorld.java’ file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

**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.



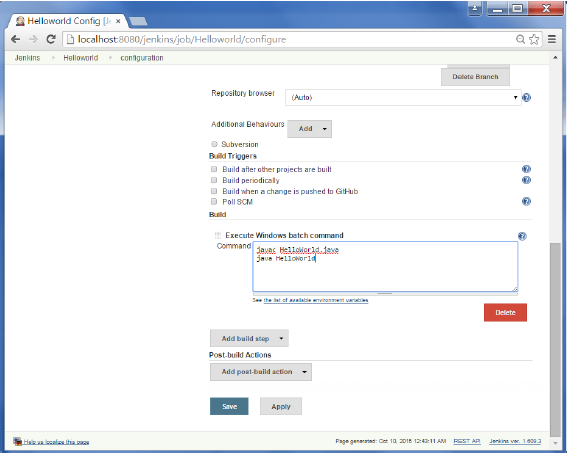
**Step 5** − Now go to the Build section and click on Add build step → Execute Windows batch command



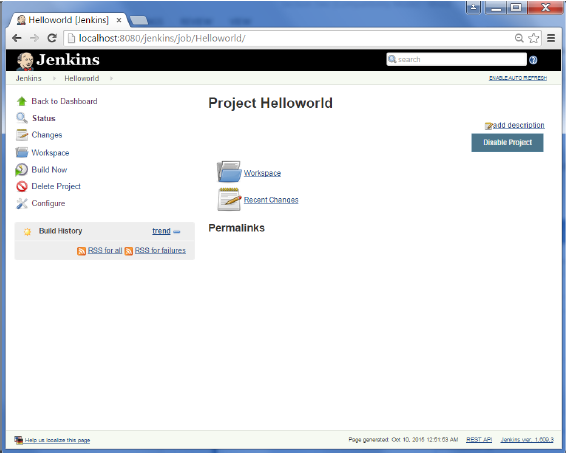
**Step 6** − In the command window, enter the following commands and then click on the Save button.

Javac Helloworld.java

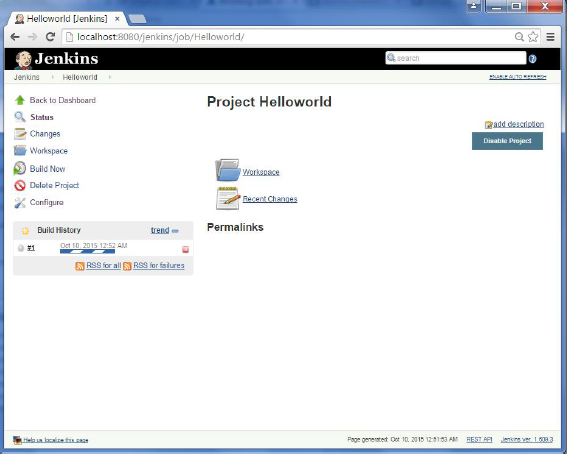
Java Helloworld



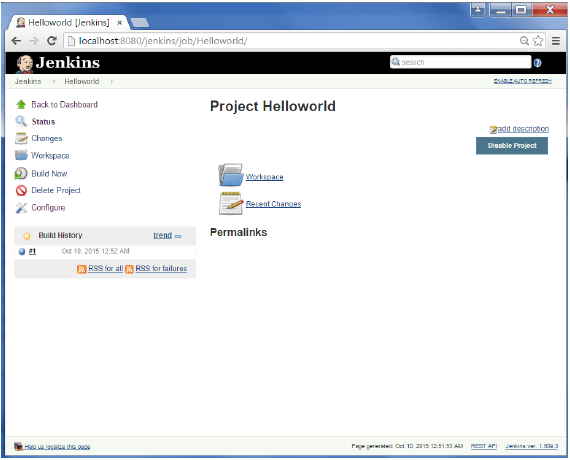
**Step 7** − Once saved, you can click on the Build Now option to see if you have successfully defined the job.



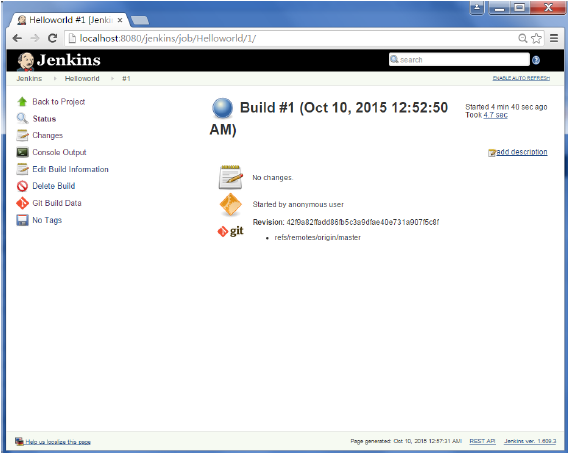
**Step 8** − Once the build is scheduled, it will run. The following Build history section shows that a build is in progress.



**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.

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](https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin" \t "_blank) it will give the list of Unit Testing plugins available.

The following are some of the basic activities you will carry out, some of which are best practices for Jenkins server maintenance

URL Options

The following commands when appended to the Jenkins instance URL will carry out the relevant actions on the Jenkins instance.

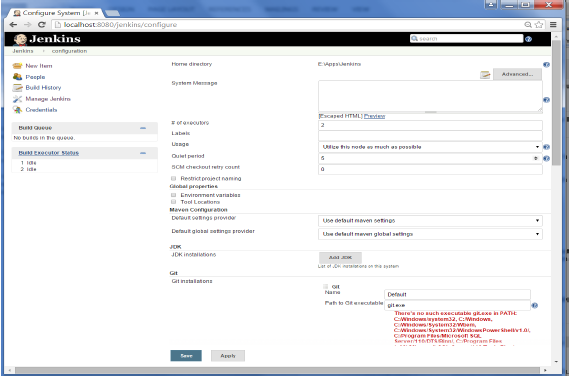
**http://localhost:8080/Jenkins/exit** − shutdown Jenkins

**http://localhost:8080/Jenkins/restart** − restart Jenkins

**http://localhost:8080/Jenkins/reload** − to reload the configuration

Backup Jenkins Home

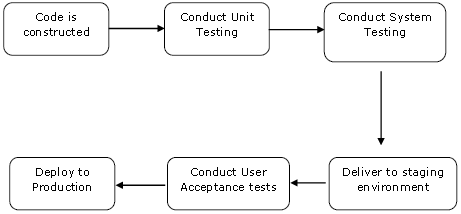
The Jenkins Home directory is nothing but the location on your drive where Jenkins stores all information for the jobs, builds etc. The location of your home directory can be seen when you click on Manage Jenkins → Configure system.



Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined and doing continuous builds, always ensure that Jenkins is setup on a drive that has enough hard disk space. If you hard disk runs out of space, then all builds on the Jenkins instance will start failing.

Another best practice is to write cron jobs or maintenance tasks that can carry out clean-up operations to avoid the disk where Jenkins is setup from becoming full.

Jenkins provides good support for providing continuous deployment and delivery. If you look at the flow of any software development through deployment, it will be as shown below.



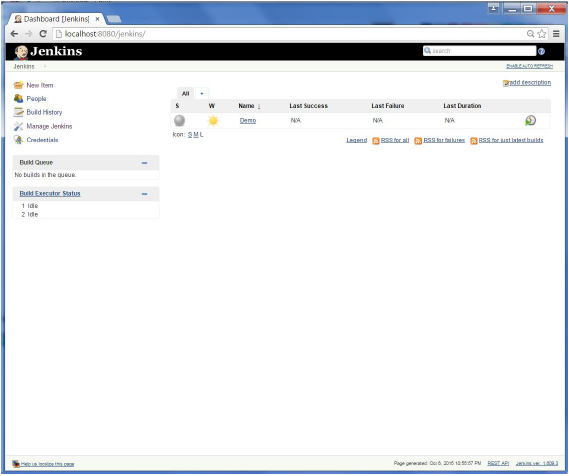
The main part of Continuous deployment is to ensure that the entire process which is shown above is automated. Jenkins achieves all of this via various plugins, one of them being the “Deploy to container Plugin” which was seen in the earlier lessons.



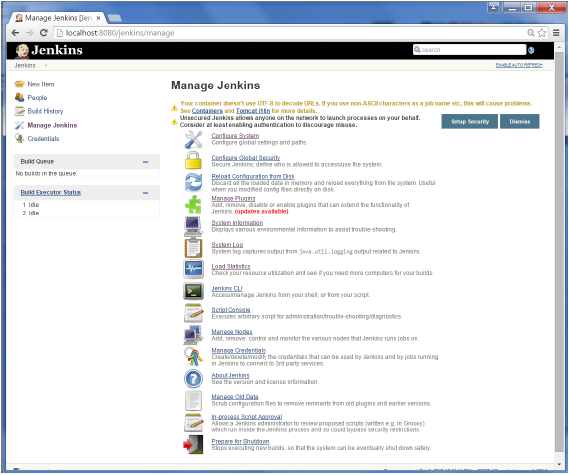
# **Jenkins - Management**

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

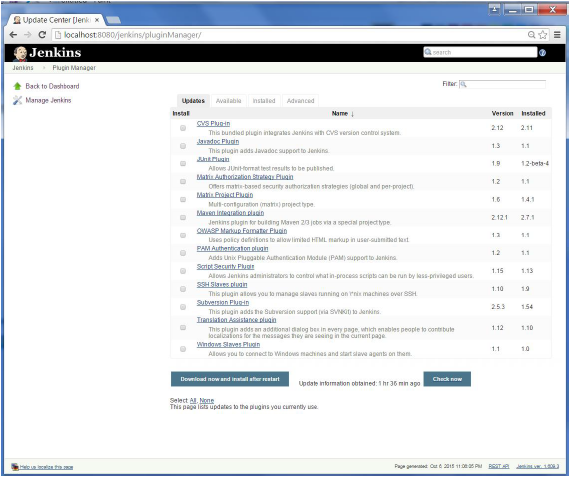
This is where one can manage paths to the various tools to use in builds, such as the JDKs, the versions of Ant and Maven, as well as security options, email servers, and other system-wide configuration details. When plugins are installed. Jenkins will add the required configuration fields dynamically 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 Jenkins’s builds directory. You don’t need to take Jenkins offline to do this—you can simply use the “Reload Configuration from Disk” option to reload the Jenkins system and build job configurations directly.

## Manage Plugin

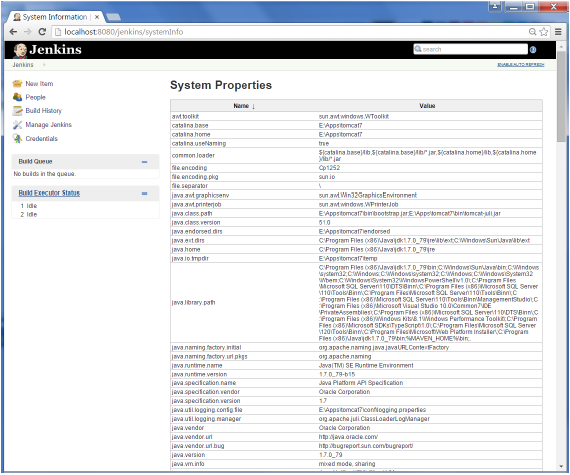
Here one can install a wide variety of third-party plugins right from different Source code management tools such as Git, Mercurial or ClearCase, to code quality and code coverage metrics reporting. Plugins can be installed, updated and removed through the Manage Plugins screen.



## 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.

# **Jenkins - Metrics & Trends**

There are various plugins which are available in Jenkins to showcase metrics for builds which are carried out over a period of time. These metrics are useful to understand your builds and how frequently they fail/pass over time. As an example, let’s look at the ‘Build History Metrics plugin’.

This plugin calculates the following metrics for all of the builds once installed

* Mean Time To Failure (MTTF)
* Mean Time To Recovery (MTTR)
* Standard Deviation of Build Times