

# Jenkins - Quick Guide

Advertisements



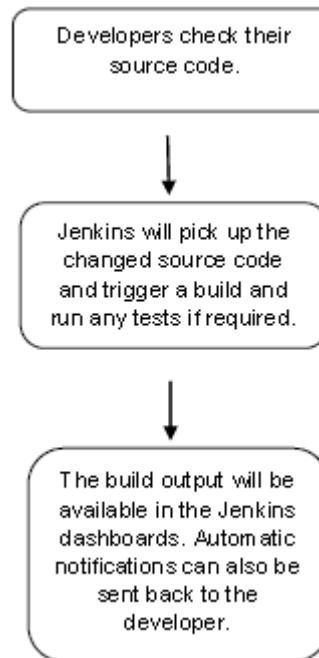
« Previous Page

Next Page »

## Jenkins - Overview

### 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 Microsystems 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.

## System Requirements

JDK	JDK 1.5 or above
Memory	2 GB RAM (recommended)
Disk Space	No minimum requirement. Note that since all builds will be stored on the Jenkins machines, it has to be ensured that sufficient disk space is available for build storage.
Operating System Version	Jenkins can be installed on Windows, Ubuntu/Debian, Red Hat/Fedora/CentOS, Mac OS X, openSUSE, FReeBSD, OpenBSD, Gentoo.
Java Container	The WAR file can be run in any container that supports Servlet 2.4/JSP 2.0 or later.(An example is Tomcat 5).

## Jenkins - Installation

### Download Jenkins

The official website for Jenkins is [Jenkins](https://jenkins.io/). If you click the given link, you can get the home page of the Jenkins official website as shown below.



By default, the latest release and the Long-Term support release will be available for download. The past releases are also available for download. Click the Long-Term Support Release tab in the download section.



Click the link "Older but stable version" to download the Jenkins war file.

## 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 webserver called winstome.

```
D:\>Java -jar Jenkins.war
Running from: D:\jenkins.war
Webroot: $user.home/ .jenkins
Sep 29, 2015 4:10:46 PM winstome.Logger logInternal
INFO: Beginning extraction from war file
```

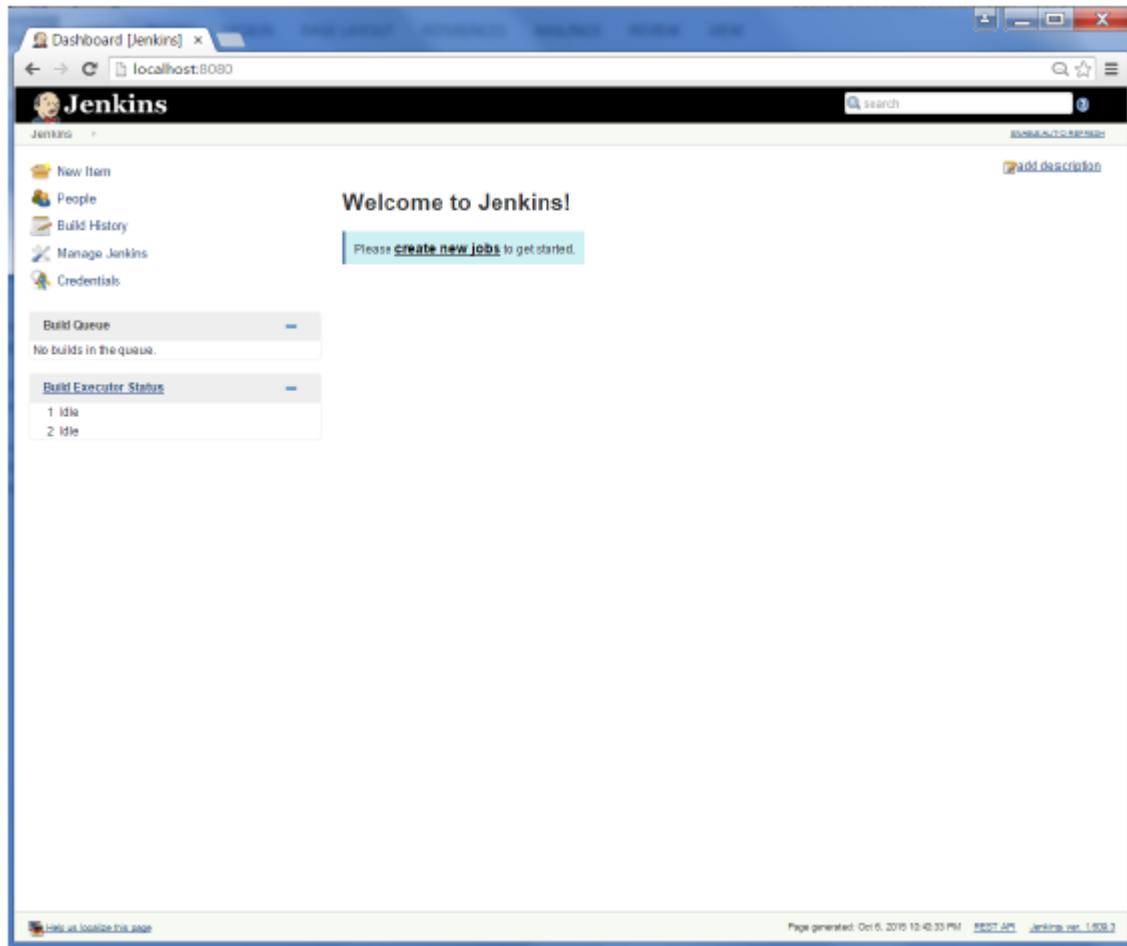
Once the processing is complete without major errors, the following line will come in the output of the command prompt.

```
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.



## Jenkins – Tomcat Setup

The following prerequisites must be met for Jenkins Tomcat setup.

### Step 1: Verifying Java Installation

To verify Java installation, open the console and execute the following java command.

OS	Task	Command
Windows	Open command console	\>java -version
Linux	Open command terminal	\$java -version

If Java has been installed properly on your system, then you should get one of the following outputs, depending on the platform you are working on.

OS	Output

Windows	Java version "1.7.0_60" Java (TM) SE Run Time Environment (build 1.7.0_60-b19) Java Hotspot (TM) 64-bit Server VM (build 24.60-b09, mixed mode)
Linux	java version "1.7.0_25" Open JDK Runtime Environment (rhel-2.3.10.4.el6_4-x86_64) Open JDK 64-Bit Server VM (build 23.7-b01, mixed mode)

We assume the readers of this tutorial have Java 1.7.0\_60 installed on their system before proceeding for this tutorial.

In case you do not have Java JDK, you can download it from the link Oracle

## Step 2: Verifying Java Installation

Set the JAVA\_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 JAVA_HOME to C:\ProgramFiles\java\jdk1.7.0_60
Linux	export JAVA_HOME=/usr/local/java-current

Append the full path of the Java compiler location to the System Path.

OS	Output
Windows	Append the String; C:\Program Files\Java\jdk1.7.0_60\bin to the end of the system variable PATH.
Linux	export PATH=\$PATH:\$JAVA_HOME/bin/

Verify the command java-version from command prompt as explained above.

## Step 3: 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.

The screenshot shows the Apache Tomcat website at [tomcat.apache.org](https://tomcat.apache.org). The main navigation bar includes links for Home, Taglibs, Maven Plugin, Download, Documentation, Problems?, and Get Involved. The 'Download' section is highlighted. It contains links for 'Which version?', Tomcat 8.0, Tomcat 7.0, Tomcat 6.0, Tomcat Connectors, Tomcat Native, Taglibs, Archives, and Documentation. Below this, there's a summary of 'Tomcat 8.0.27 Released' (2015-10-01) and 'Tomcat 7.0.64 Released' (2015-08-25), each with a 'Download' button.

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

The screenshot shows the 'Tomcat 7 Downloads' page at <https://tomcat.apache.org/download-70.cgi>. The left sidebar mirrors the main website's navigation. The main content area has sections for 'Tomcat 7 Downloads', 'Quick Navigation' (with links to KEYS, 7.0.64, Browse, and Archives), 'Release Integrity' (warning about verifying file integrity), 'Mirrors' (listing a primary mirror at <http://www.us.apache.org/dist/>), and '7.0.64' (linking to the README file). The 'Binary Distributions' section is expanded, listing various download options including 'Core' (zip, tar.gz, 32-bit Windows zip, 64-bit Windows zip, 64-bit Itanium Windows zip, 32-bit/64-bit Windows Service Installer), and 'Full documentation'.

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

Then unzip the contents of the downloaded zip file.

## 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.

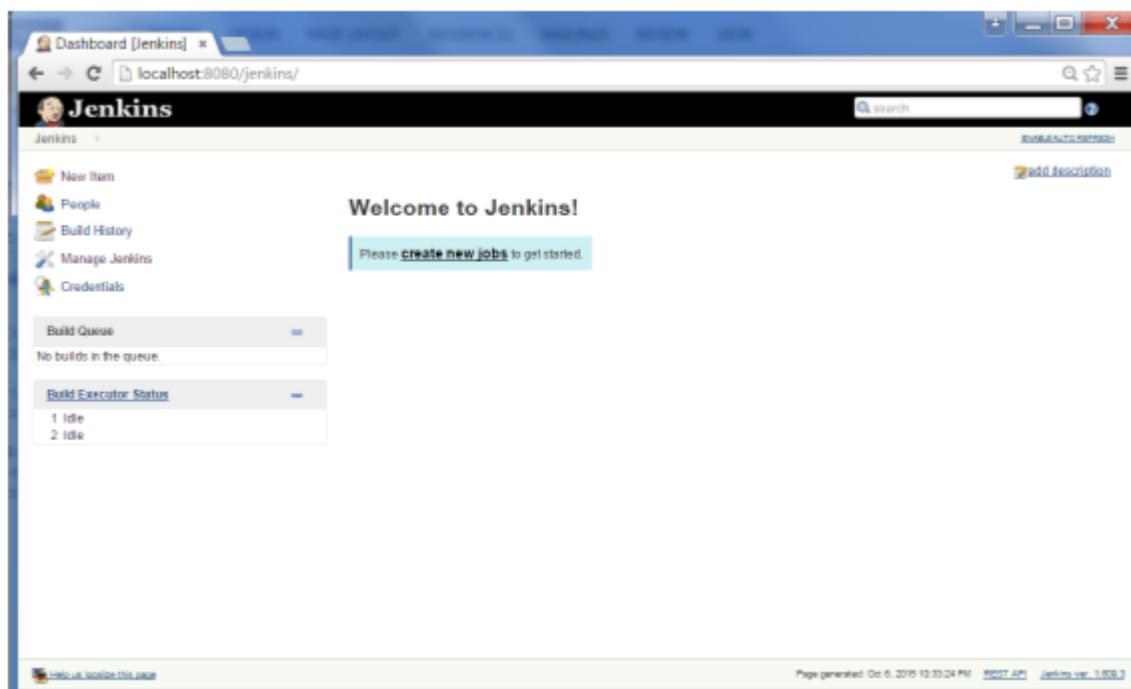
Now open the command prompt. From the command prompt, browse to the directory where the tomcat7 folder is location. Browse to the bin directory in this folder and run the start.bat file

```
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.

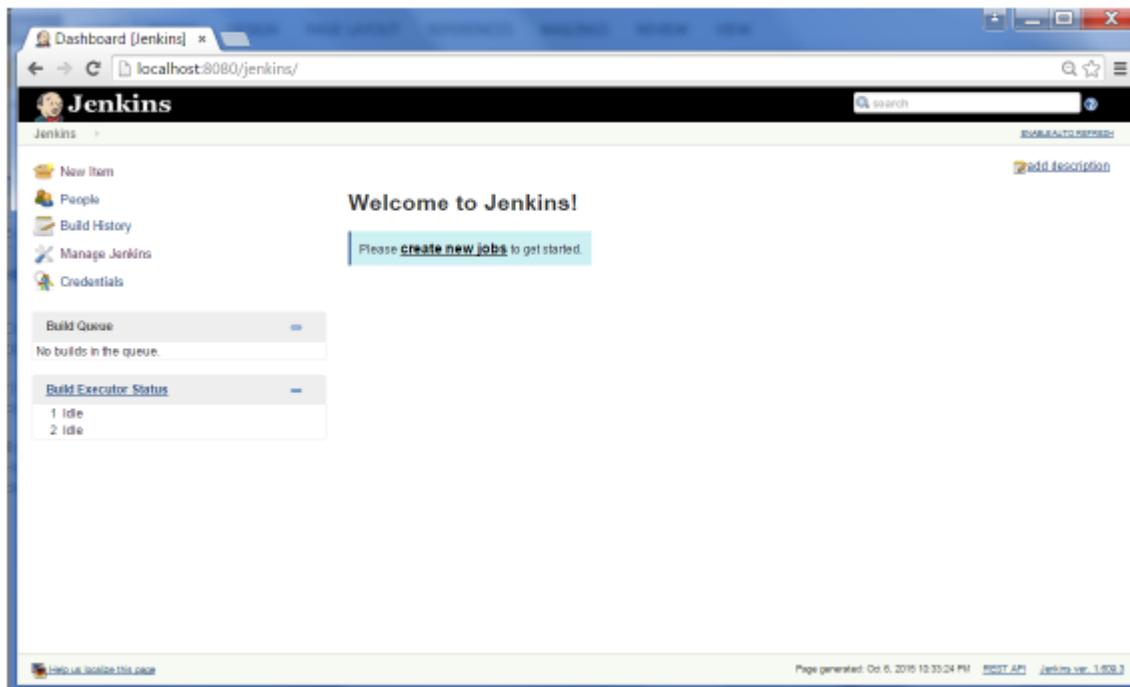
```
INFO: Server startup in 1302 ms
```

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

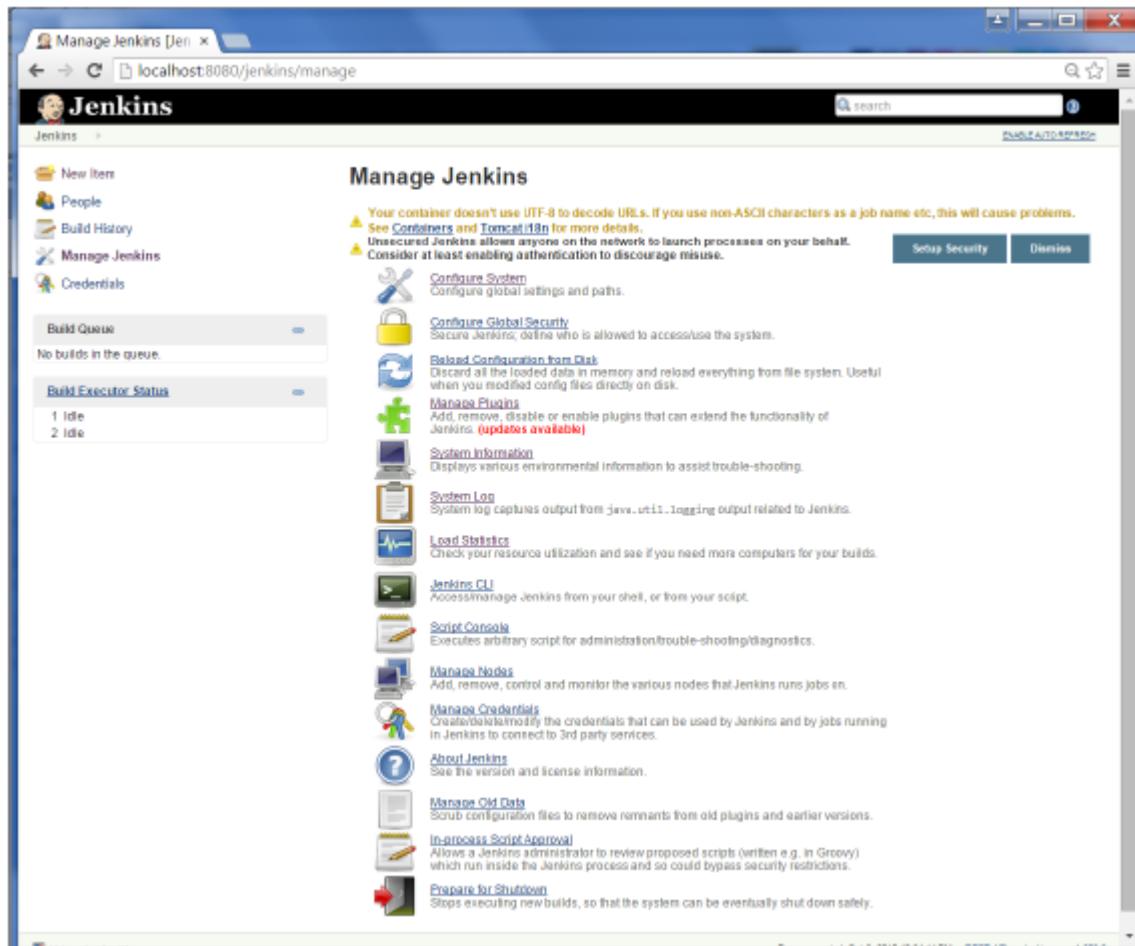


## Jenkins - Git Setup

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'

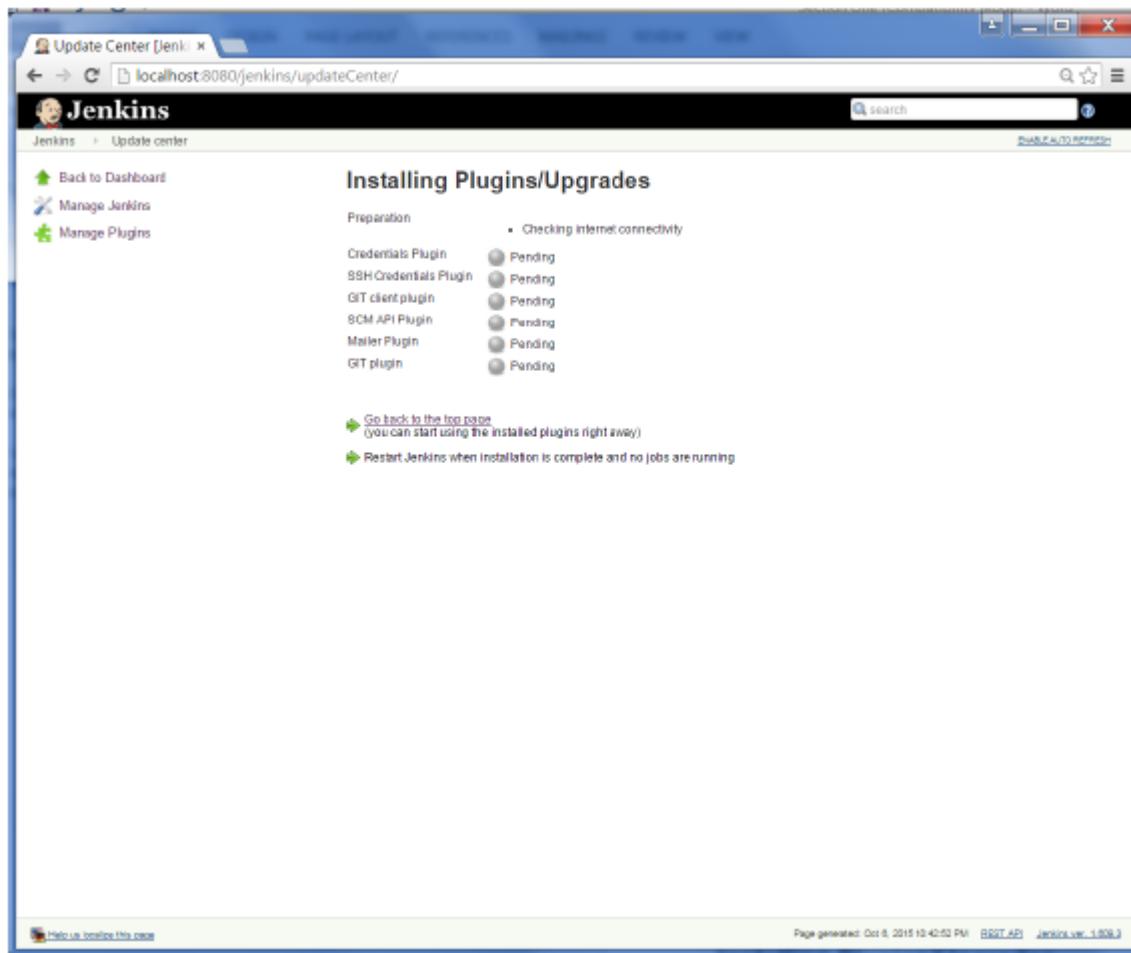
The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected, and a search bar at the top right contains the text 'Filter: Git plugin'. Below the search bar, there is a table with columns: Name, Version, and several descriptive text blocks. One row, 'GIT plugin', has its checkbox checked. At the bottom of the table are four buttons: 'Install without restart', 'Download now and install after restart', 'Update information obtained: 1 hr 0 min ago', and 'Check now'.

Name	Version
<a href="#">Git Parameter Plug-in</a>	0.4.0
<a href="#">UserContent In Git plugin</a>	1.4
<a href="#">Alternative build chooser</a>	1.1
<a href="#">Team Concert Git Plugin</a>	1.0.10
<a href="#">Tracking Git Plugin</a>	1.0
<a href="#">GIT plugin</a>	2.4.0

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

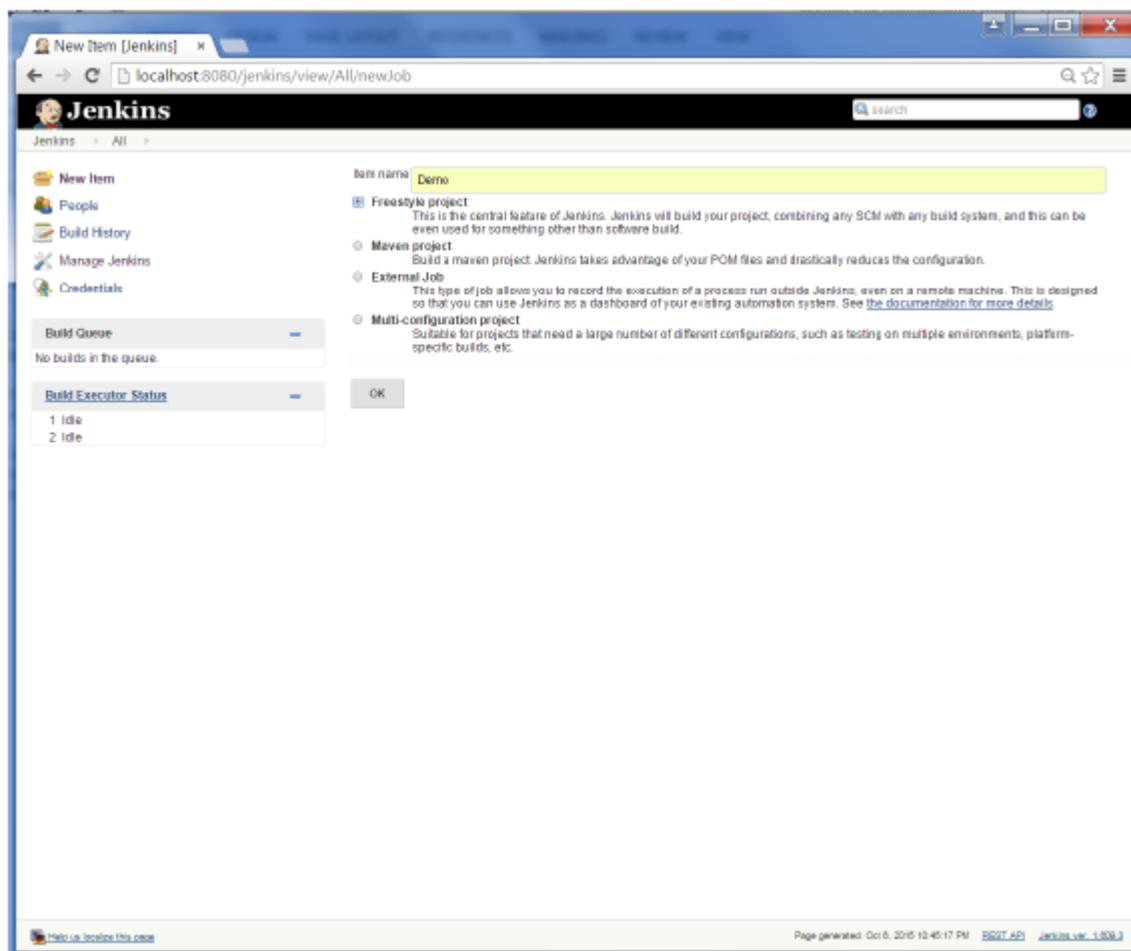
This screenshot is identical to the one above, showing the Jenkins Plugin Manager with the 'Available' tab selected and the 'Git plugin' filter applied. The 'GIT plugin' row has its checkbox checked. The same set of buttons at the bottom are present.

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

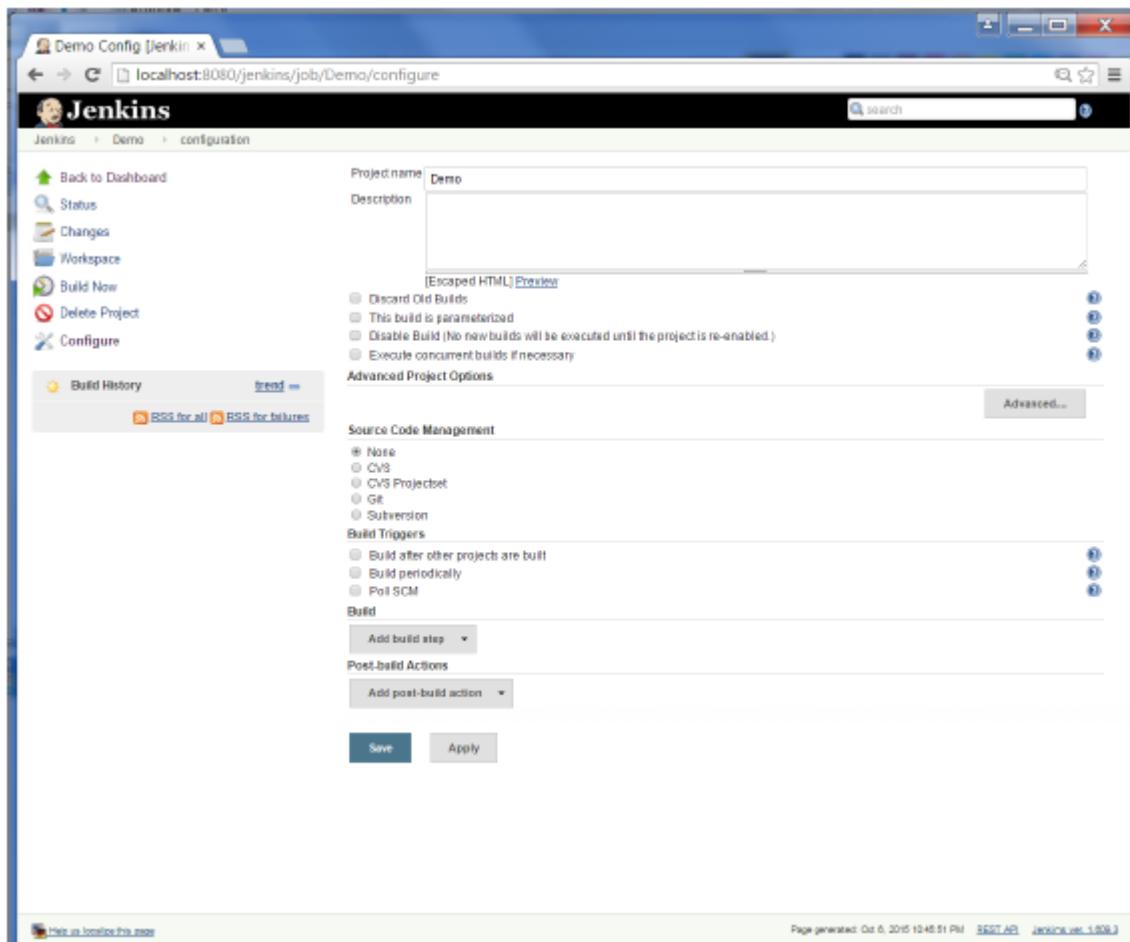


Once all installations are complete, restart Jenkins by issue the following command in the browser. **<http://localhost:8080/jenkins/restart>**

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.



## Jenkins – Maven Setup

### Step 1: Downloading and Setting Up Maven

The official website for maven is Apache Maven <http://maven.apache.org>. If you click the given link, you can get the home page of the maven official website as shown below.

The screenshot shows a web browser window titled "Maven - Download" displaying the Apache Maven Project download page at <https://maven.apache.org/download.cgi>. The page features the Apache logo and the word "Maven" in large red letters. A sidebar on the left contains links for MAIN, Welcome, License, Download (which is highlighted), Install, Configure, Run, IDE Integration, ABOUT MAVEN, What is Maven?, Features, FAQ, Support and Training, DOCUMENTATION, Maven Plugins, Index (category), Running Maven, User Centre >, Plugin Developer Centre, Maven Repository Centre, Maven Developer Centre, Books and Resources. The main content area has a heading "Downloading Apache Maven 3.3.3". It states that Apache Maven 3.3.3 is the latest release and recommended version for all users. It includes a note about mirrors and a dropdown menu for "Other mirrors" set to <http://www.eu.apache.org/dist/> with a "Change" button. Below this is a section titled "System Requirements" with tables for Java Development Kit (JDK), Memory, Disk, and Operating System. At the bottom is a "Files" section with a table for Maven distributions, showing columns for Link, Checksum, and Signature.

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

The screenshot shows the Apache Maven download page. On the left, there's a sidebar with links like Support and Training, Documentation, Maven Plugins, Index (category), Running Maven, User Centre, Plugin Developer Centre, Maven Repository Centre, Maven Developer Centre, Books and Resources, Security, Community Overview, How to Contribute, Maven Repository, Getting Help, Issue Tracking, Source Repository, The Maven Team, Project Documentation, Project Information, Maven Projects, Ant Tasks, Archetype, Doxia, and XPP. The main content area has sections for Memory (No minimum requirement), Disk (Approximately 10MB required for the Maven installation itself), and Operating System (No minimum requirement). Below this is a 'Files' section with a table showing download links, checksums, and signatures for Binary tar.gz, Binary zip, Source tar.gz, and Source zip archive formats. At the bottom of the 'Files' section is a bulleted list of distribution details. The 'Previous Releases' section below it encourages using the latest version and provides links for older versions.

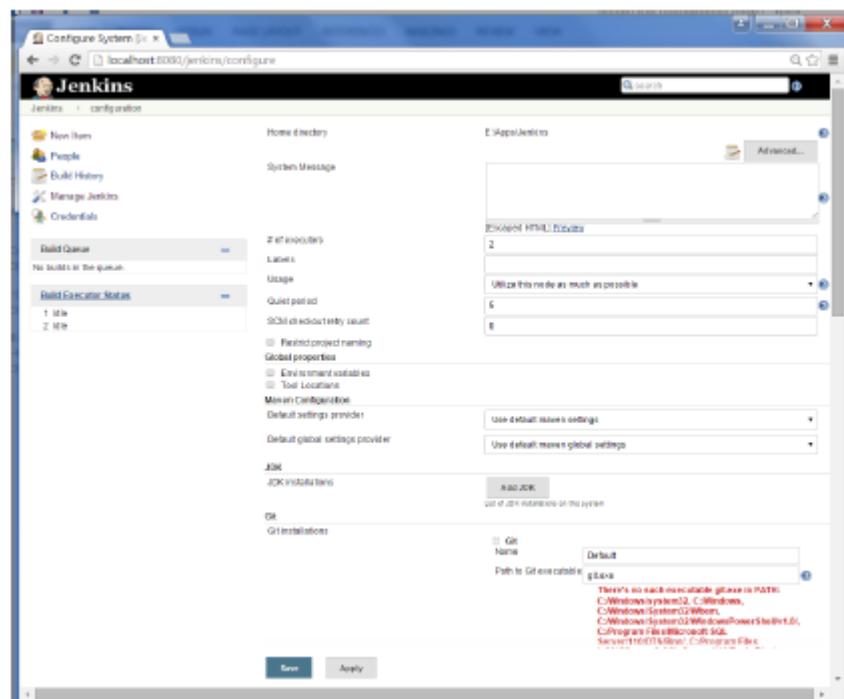
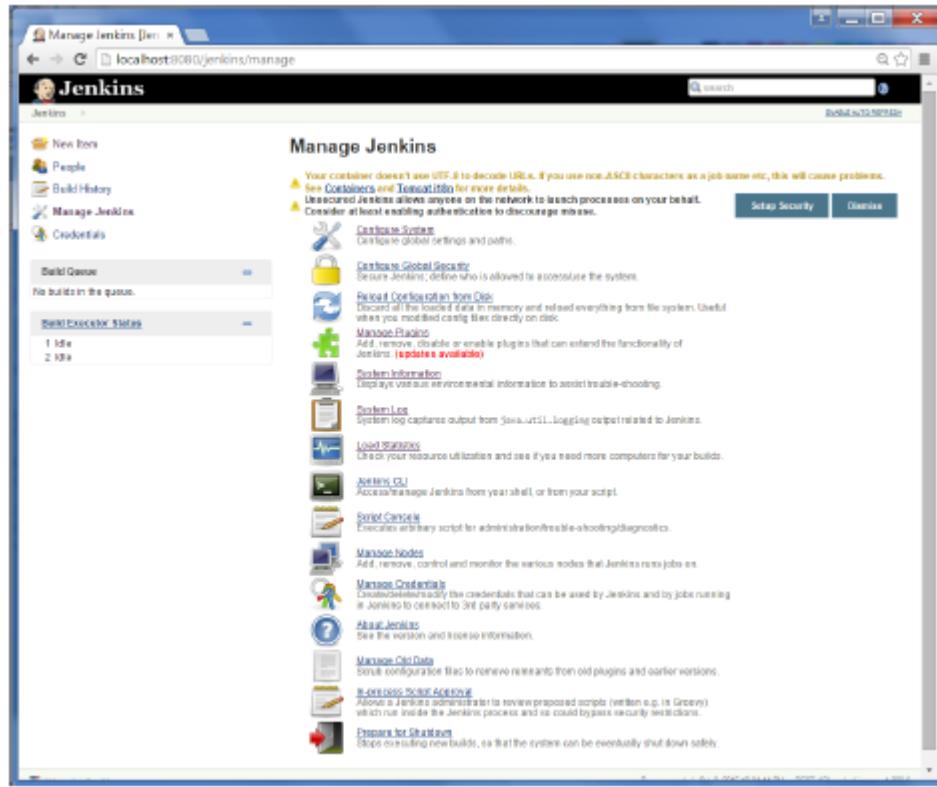
Once the file is downloaded, extract the files to the relevant application folder. For this purpose, the maven files will be placed in E:\Apps\apache-maven-3.3.3.

## Step 2: Setting up Jenkins and Maven

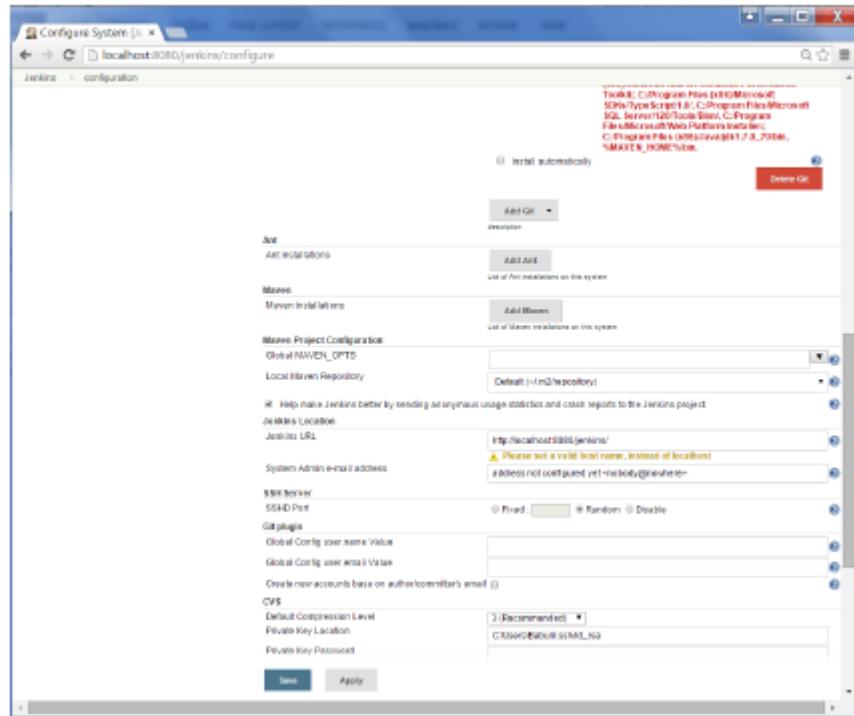
In the Jenkins dashboard (Home screen), click Manage Jenkins from the left-hand side menu.

The screenshot shows the Jenkins dashboard. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. The main area displays a 'Welcome to Jenkins!' message with a button to 'create new jobs'. Below this are sections for 'Build Queue' (No items in the queue) and 'Build Executor Status' (1 idle, 2 busy). At the bottom, there are links for Help and Feedback, Page generated, and Jenkins ver.

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



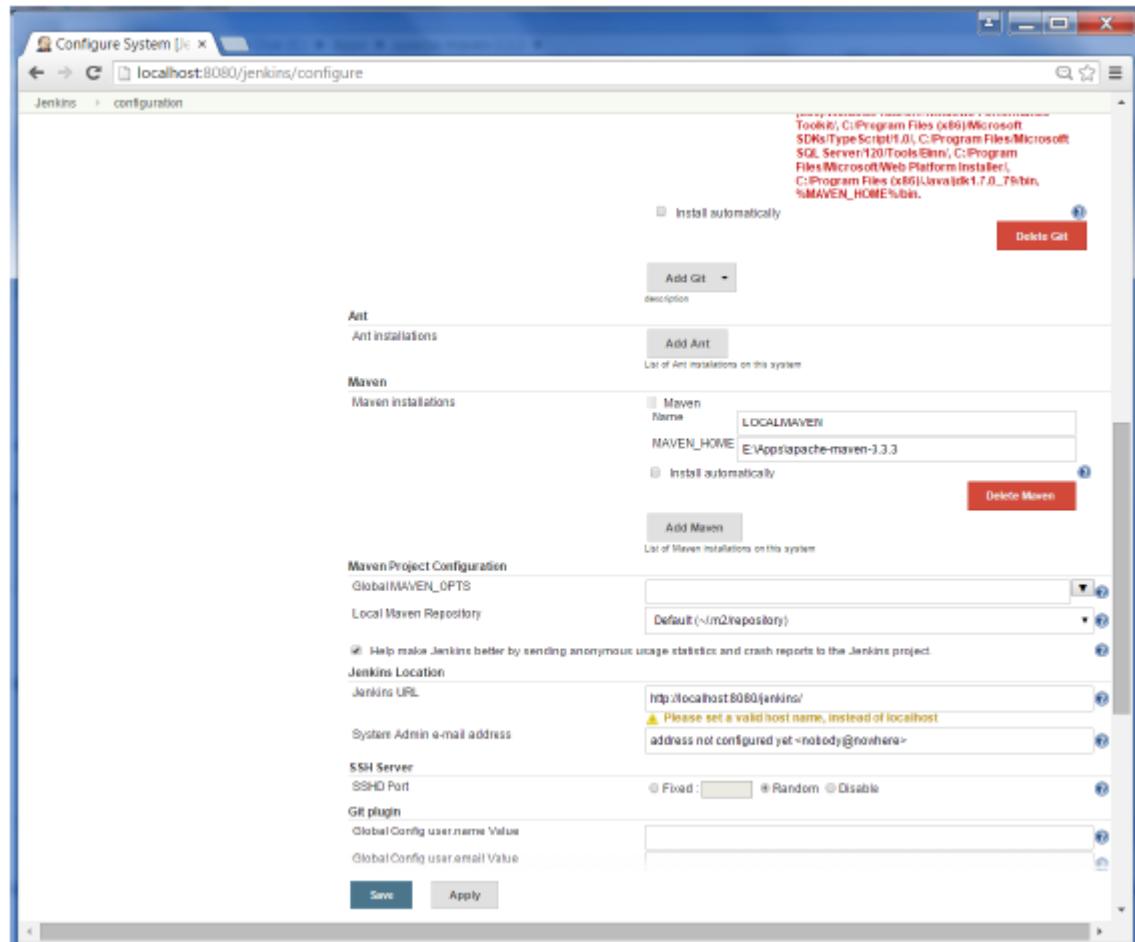
In the Configure system screen, scroll down till you see the Maven section and then click on the 'Add Maven' button.



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.



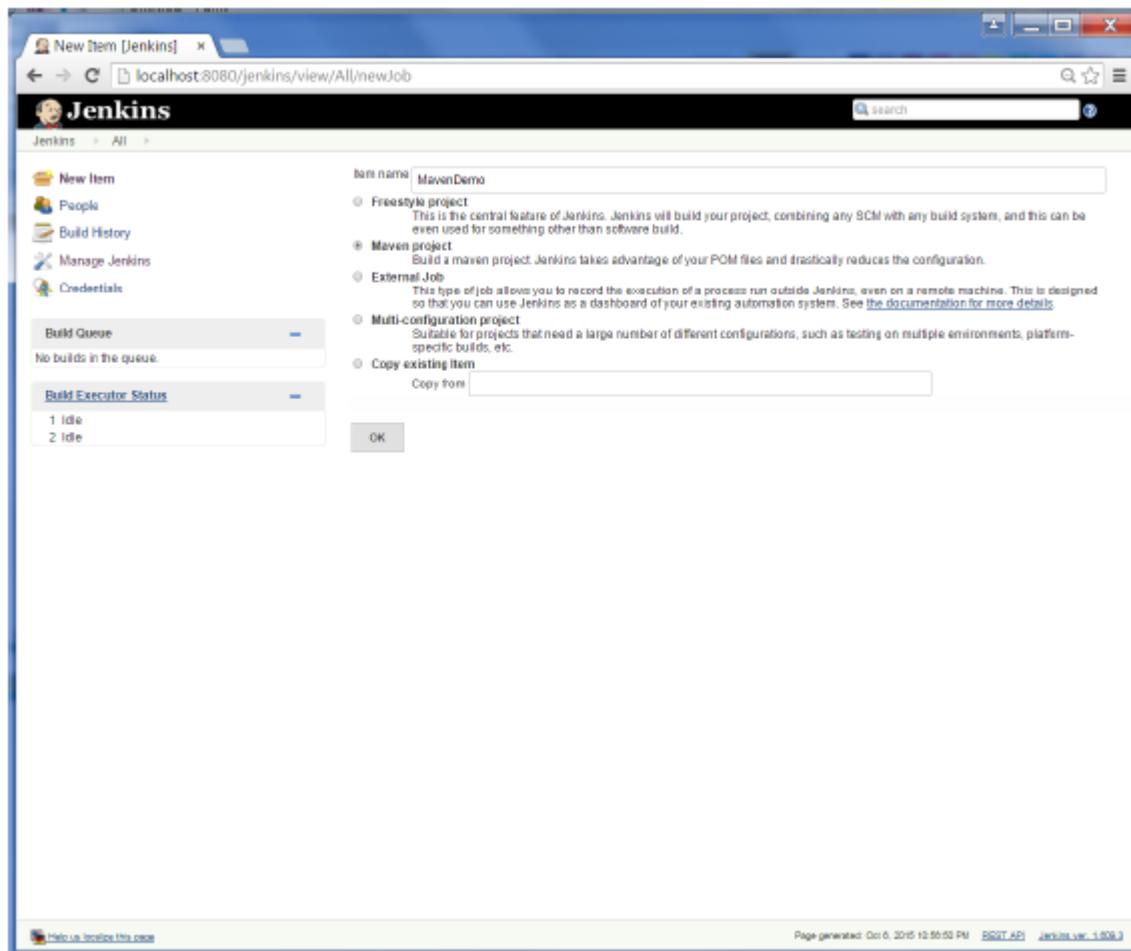
You can now create a job with the 'Maven project' option. In the Jenkins dashboard, click the New Item option.

The screenshot shows the Jenkins dashboard at the URL [localhost:8080/jenkins/](http://localhost:8080/jenkins/). The left sidebar contains links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. The main area displays a table of jobs. One job, 'Demo', is listed with the status 'N/A' for both 'Last Success' and 'Last Failure', and 'N/A' for 'Last Duration'. A legend below the table indicates RSS feeds for all builds, failed builds, and the last build. At the bottom, there are links for 'Help Us Improve this page' and 'Page generated: Oct 6, 2015 12:55:57 PM'.

S	W	Name	Last Success	Last Failure	Last Duration
icon: ☰	icon: ☀	Demo	N/A	N/A	N/A

Legend: [RSS for all](#) [RSS for failures](#) [RSS for last build](#)

Page generated: Oct 6, 2015 12:55:57 PM [API](#) Jenkins ver. 1.60.3



## 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.

The screenshot shows the Jenkins Dashboard at [localhost:8080/jenkins/](http://localhost:8080/jenkins/). The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. The main area displays a table for the 'Demo' job, which is currently Idle. A legend at the bottom right indicates RSS feeds for all builds, failed builds, and the latest build.

S	W	Name	Last Success	Last Failure	Last Duration
Idle	Idle	Demo	N/A	N/A	N/A

Build Queue: No builds in the queue.

Build Executor Status: 1 Idle, 2 Idle.

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.

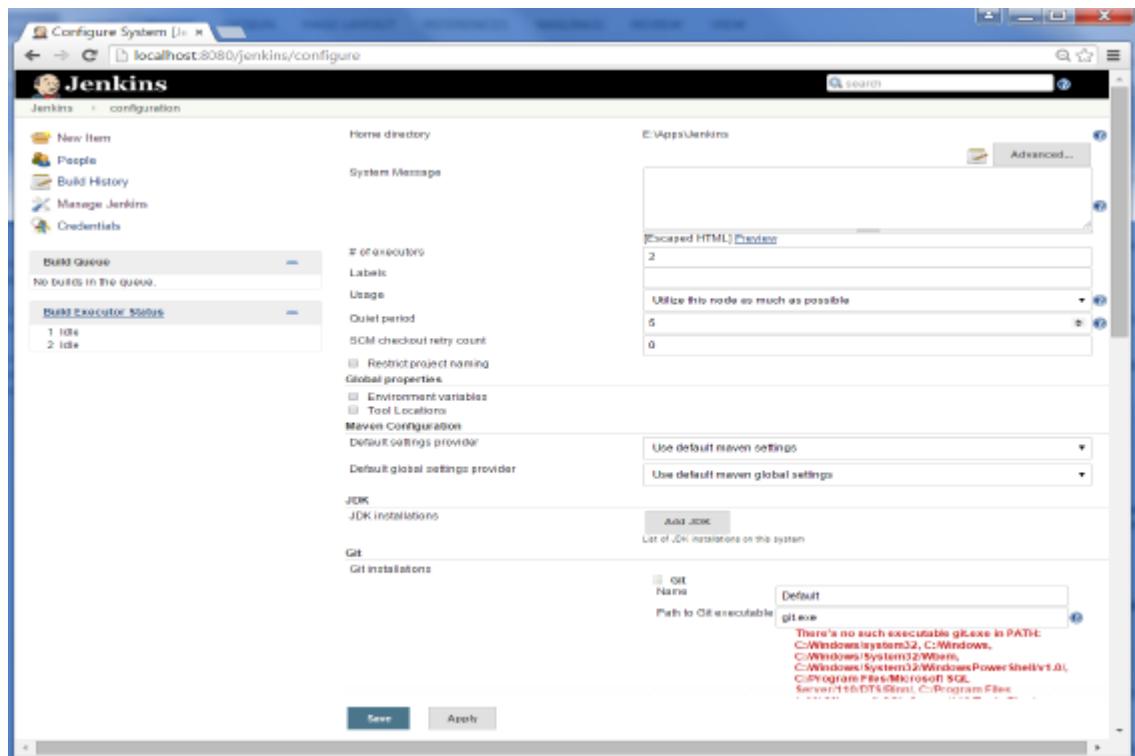
First create a new folder `E:\Apps\Jenkins`. Copy all the contents from the existing `~/.jenkins` to this new directory.

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 you're 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}.

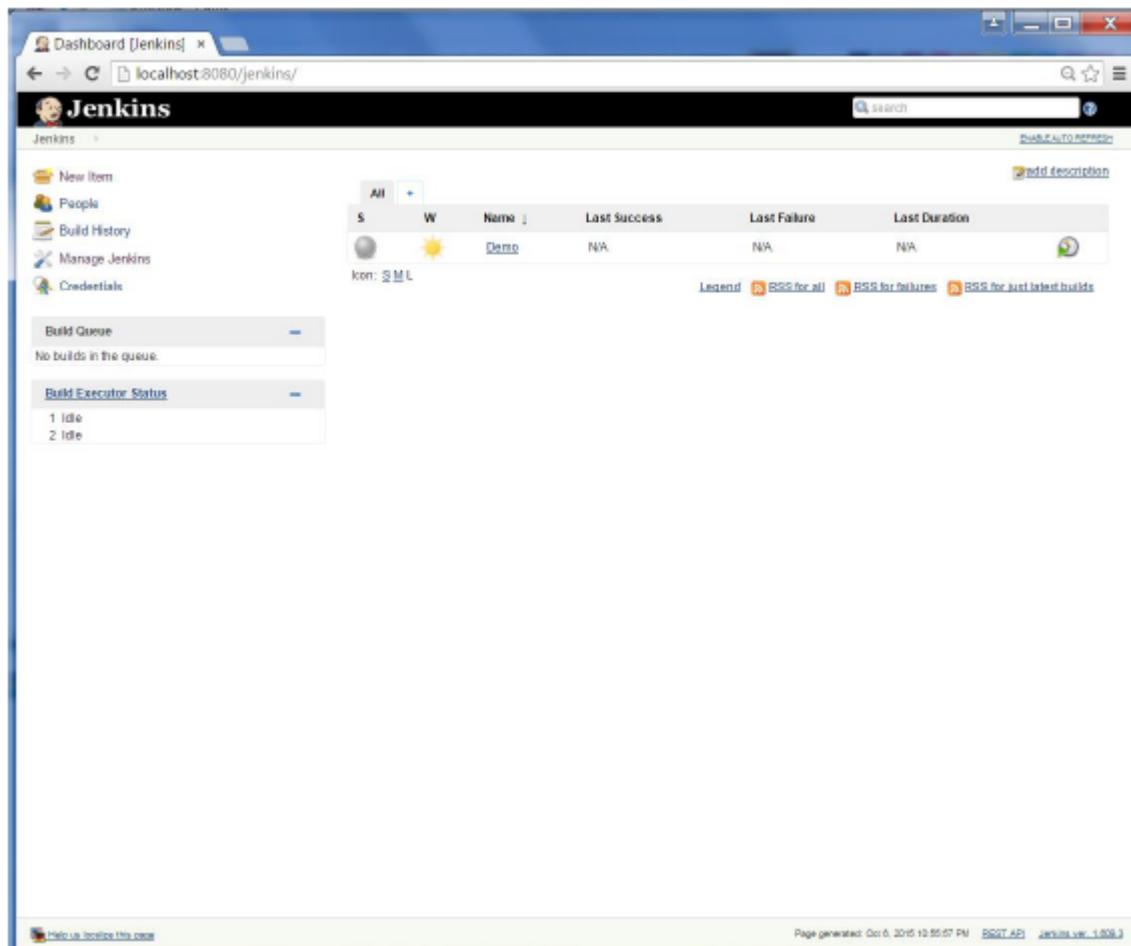
## Email Notification

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 - 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.

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Jenkins]". The address bar shows "localhost:8080/jenkins/pluginManager/". The main header has tabs: "Jenkins", "Plugin Manager", "Back to Dashboard", and "Manage Jenkins". Below the tabs is a search bar and a "Filter" button. The main content area has tabs: "Updates" (selected), "Available", "Installed", and "Advanced". The "Updates" tab displays a list of plugins:

Name	Version	Installed
<a href="#">CVS Plug-In</a>	2.12	2.11
<a href="#">Javadoc Plugin</a>	1.3	1.1
<a href="#">JUnit Plugin</a>	1.9	1.2-beta-4
<a href="#">Matrix Authorization Strategy Plugin</a>	1.2	1.1
<a href="#">Matrix Project Plugin</a>	1.6	1.4.1
<a href="#">Maven Integration plugin</a>	2.12.1	2.7.1
<a href="#">OWASP Markup Formatter Plugin</a>	1.3	1.1
<a href="#">PAM Authentication plugin</a>	1.2	1.1
<a href="#">Script Security Plugin</a>	1.15	1.13
<a href="#">SSH Slaves plugin</a>	1.10	1.9
<a href="#">Subversion Plug-in</a>	2.5.3	1.54
<a href="#">Translation Assistance plugin</a>	1.12	1.10
<a href="#">Windows Slaves Plugin</a>	1.1	1.0

Below the table are two buttons: "Download now and install after restart" and "Check now". A status message says "Update information obtained: 1 hr 36 min ago". A note below says "Select All, None. This page lists updates to the plugins you currently use." At the bottom are links for "Help us localize this page", "Page generated: Oct 6 2015 11:08:25 PM", "E2DT-NB", and "Jenkins ver. 1.603".

## 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.

The screenshot shows the Jenkins System Information page at [localhost:8080/jenkins/systemInfo](http://localhost:8080/jenkins/systemInfo). The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. Under Build Queue, it says "No builds in the queue." Under Build Executor Status, there are two entries: "1 Idle" and "2 Idle". The main content area is titled "System Properties" and displays a table of system properties with their names and values.

Name	Value
awt.toolkit	sun.awt.windows.WToolkit
catalina.base	E:\Appstromcat7
catalina.home	E:\Appstromcat7
catalina.useNaming	true
common.loader	\$[catalina.base]lib;\$[catalina.base]lib\*;jar;\$[catalina.home]lib;\$[catalina.home]\lib\*;jar
file.encoding	Cp1252
file.encoding.pkg	sun.ja
file.separator	\
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.path	E:\Appstromcat7\bin\bootstrap.jar;E:\Appstromcat7\bin\lombok-all.jar
java.class.version	51.0
java.endorsed.dirs	E:\Appstromcat7\endorsed
java.ext.dirs	C:\Program Files (x86)\Java\jdk1.7.0_79\jre\lib\ext;C:\Windows\Sun\Java\lib\ext
java.home	C:\Program Files (x86)\Java\jdk1.7.0_79\jre
java.io.tmpdir	E:\Appstromcat7\temp
java.library.path	C:\Program Files (x86)\Java\jdk1.7.0_79\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\10.0\Tools\Binn\;C:\Program Files\Microsoft SQL Server\10.0\Tools\Binn\ManagementStudio\;C:\Program Files\Microsoft SQL Server\10.0\Tools\Binn\T-SQL\Scripting\Tools\Binn\;C:\Program Files\Microsoft Visual Studio 10.0\Common7\IDE\PrivateAssemblies\;C:\Program Files (x86)\Microsoft SQL Server\11.0\Tools\Binn\;C:\Program Files (x86)\Windows Performance Toolkit\;C:\Program Files (x86)\Microsoft SDKs\TypeScript\1.0\;C:\Program Files\Microsoft SQL Server\12.0\Tools\Binn\;C:\Program Files\Microsoft\Web Platform\Install\;C:\Program Files (x86)\Java\jdk1.7.0_79\bin;%MAVEN_HOME%\bin;
java.naming.factory.initial	org.apache.naming.java.javaURLContextFactory
java.naming.factory.url.pkgs	org.apache.naming
java.runtime.name	Java(TM) SE Runtime Environment
java.runtime.version	1.7_0_79-b15
java.specification.name	Java Platform API Specification
java.specification.vendor	Oracle Corporation
java.specification.version	1.7
java.util.logging.config.file	E:\Appstromcat7\conf\logging.properties
java.util.logging.manager	org.apache.juli.ClassLoaderLogManager
java.vendor	Oracle Corporation
java.vendor.url	http://java.oracle.com/
java.vendor.url_bug	http://bugreport.sun.com/bugreport/
java.version	1.7_0_79
java.vm.info	mixed mode, sharing

## 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 page 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.

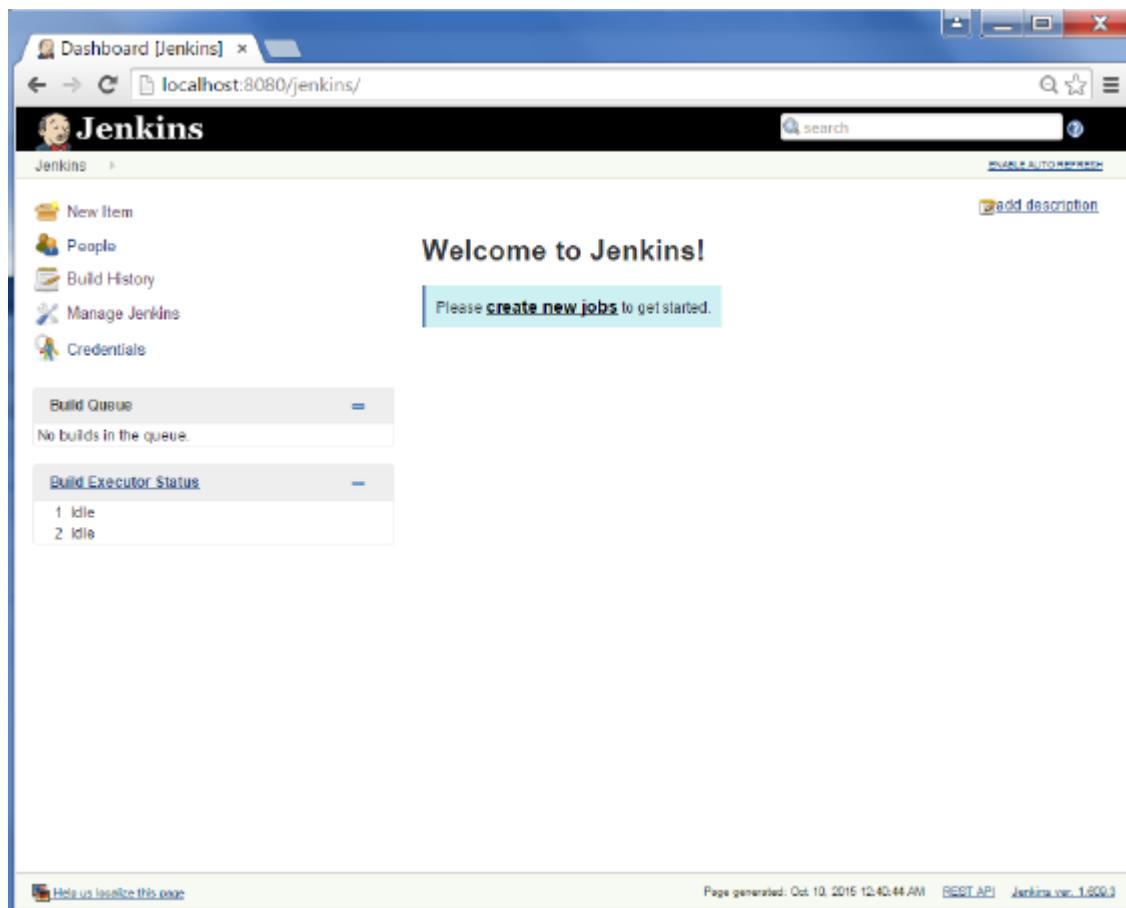
## Prepare for Shutdown

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.

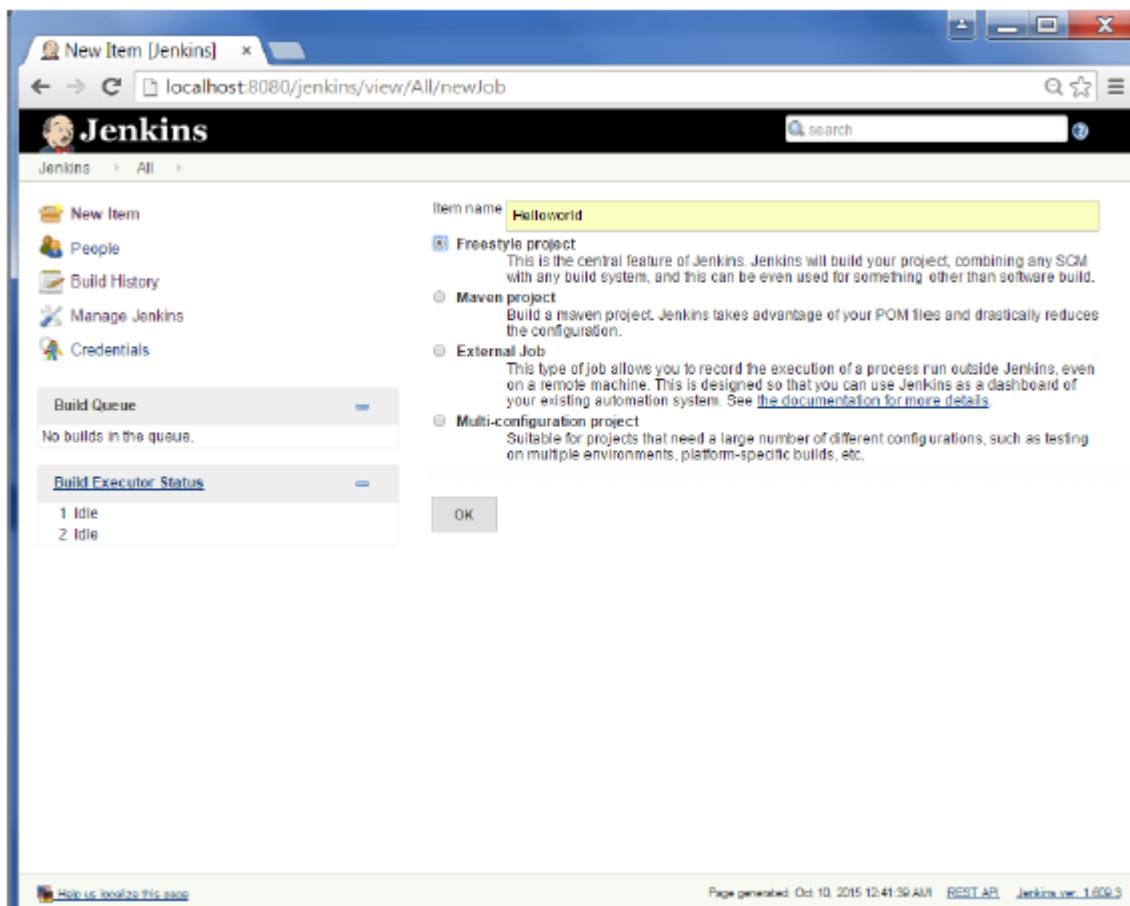
# Jenkins - Setup Build Jobs

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

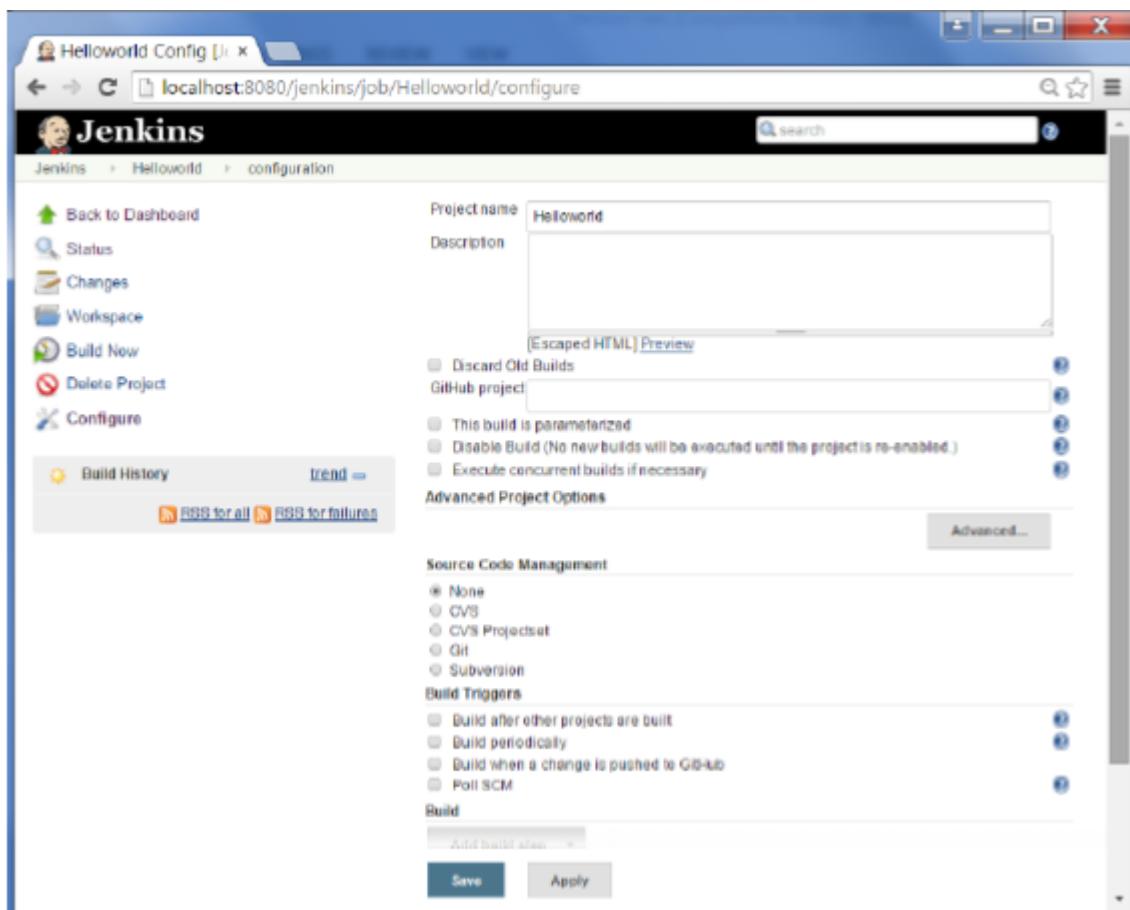
**Step 1** – Go to the Jenkins dashboard and Click on New Item



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

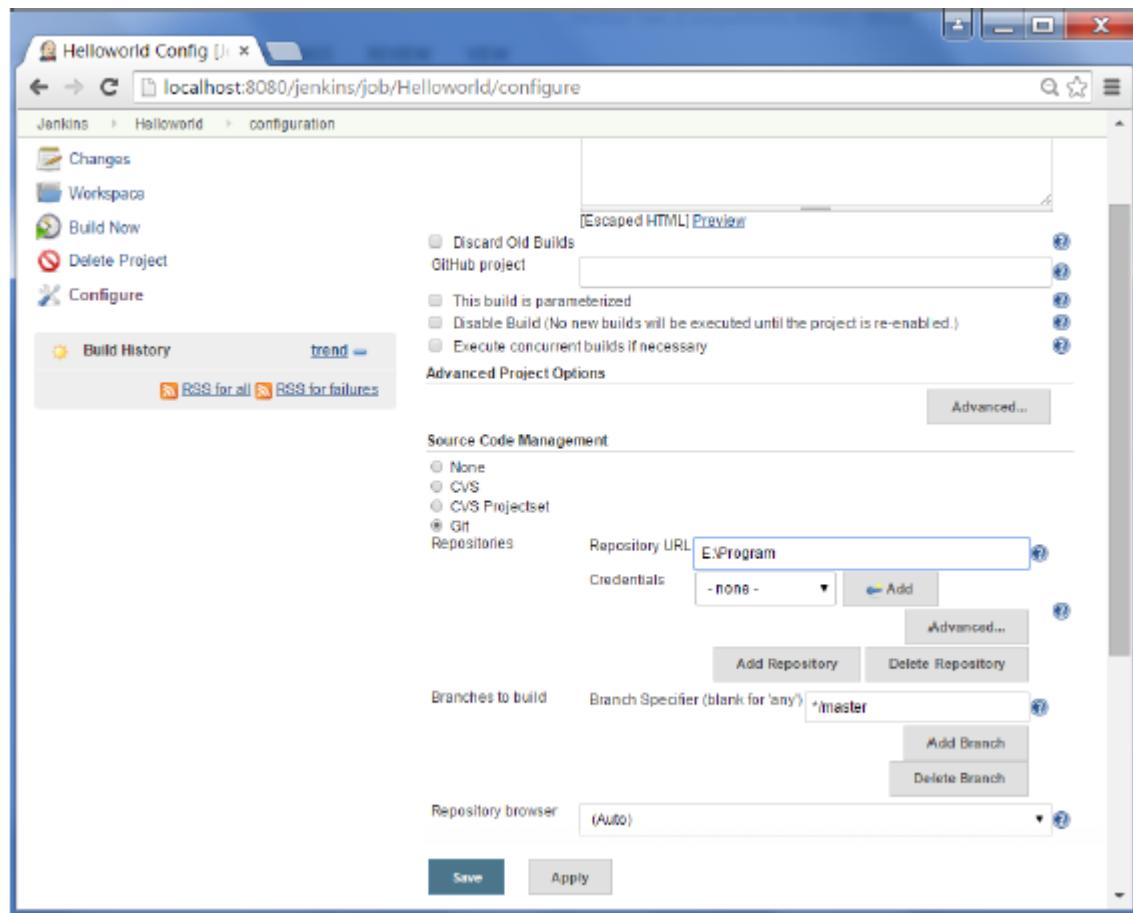


**Step 3** – The following screen will come up in which you can specify the details of the job.

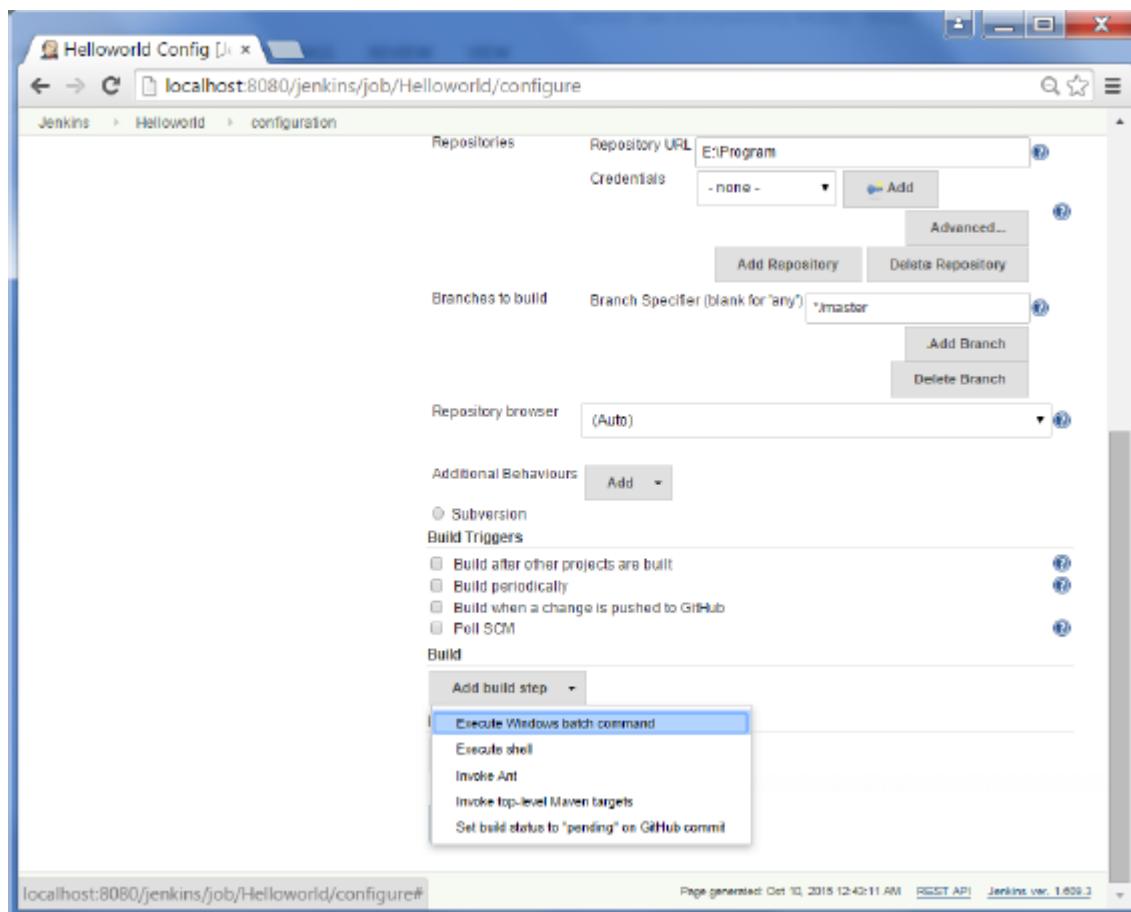


**Step 4** – We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a 'HelloWorld.java' file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

**Note** – If your repository is 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.

```
Java HelloWorld.java  
Java HelloWorld
```

The screenshot shows the Jenkins configuration interface for the 'Helloworld' job. In the 'Build' section, there is a step titled 'Execute Windows batch command' with the command 'java HelloWorld.java'. Below the command input field is a link 'See the list of available environment variables'. At the bottom of the configuration page are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins project page for 'Project Helloworld'. On the left sidebar, there are links for Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main area displays the project name 'Project Helloworld' and two links: 'Workspace' and 'Recent Changes'. There are also buttons for 'Add description' and 'Disable Project'. At the bottom, there is a 'Build History' section with 'RSS for all' and 'RSS for failures' links.

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

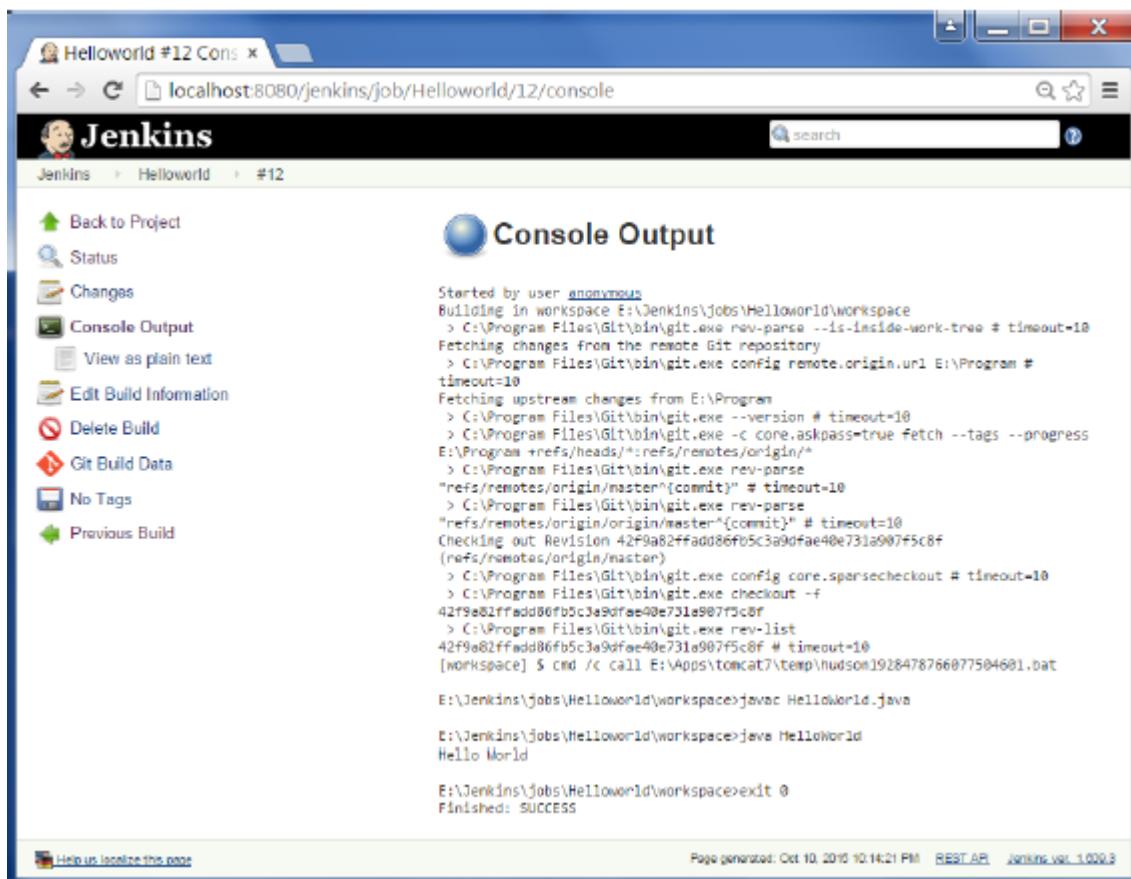
The screenshot shows the Jenkins interface for the 'Helloworld' project. The title bar says 'Helloworld [Jenkins]'. The URL in the address bar is 'localhost:8080/jenkins/job/Helloworld/'. The main content area is titled 'Project Helloworld'. On the left, there's a sidebar with links: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. On the right, there are buttons for 'Add description' and 'Disable Project'. Below the sidebar, there's a 'Build History' section showing one build (#1) from 'Oct 10, 2015 12:52 AM'. It includes links for 'RSS for all' and 'RSS for failures'. At the bottom, there's a footer with links for 'Help us localize this page', 'Page generated: Oct 10, 2015 12:51:53 AM', 'REST API', and 'Jenkins ver. 1.600.3'.

**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.

The screenshot shows the Jenkins interface for the 'Helloworld' project. At the top, there's a navigation bar with links like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. To the right, there's a search bar and a 'Disable Project' button. The main title is 'Project Helloworld'. Below the title, there are two sections: 'Workspace' (with a folder icon) and 'Recent Changes' (with a document icon). A 'Permalinks' section follows, containing a table with one row showing a build history entry from Oct 10, 2015, at 12:52 AM. At the bottom, there are links for 'RSS for all' and 'RSS for failures'.

## Step 10 – Click on the Console Output link to see the details of the build

The screenshot shows the Jenkins interface for the first build of the 'Helloworld' project. The title is 'Build #1 (Oct 10, 2015 12:52:50 AM)'. It includes information about the build start time ('Started 4 min 40 sec ago') and duration ('Took 4.7 sec'). There are several status icons: a blue circle for the build itself, a document icon for 'No changes.', a yellow diamond for 'Started by anonymous user', and a red diamond with 'git' for 'Revision: 42f9a82ffadd86fb5c3a9d9e40e731a907f5c8f'. Below these, it says '• ref/remotes/origin/master'. On the left, a sidebar lists options: 'Back to Project', 'Status', 'Changes', 'Console Output' (which is selected and highlighted in blue), 'Edit Build Information', 'Delete Build', 'Git Build Data', and 'No Tags'. At the bottom, there's a footer with links for 'Help us localize this page', 'Page generated: Oct 10, 2015 12:51:53 AM', 'REST API', and 'Jenkins ver. 1.602.3'.



The screenshot shows the Jenkins interface for a build named "Helloworld #12". The left sidebar lists various build-related links: Back to Project, Status, Changes, Console Output (which is currently selected), View as plain text, Edit Build Information, Delete Build, Git Build Data, No Tags, and Previous Build. The main content area is titled "Console Output" and displays the command-line log for the build. The log shows the execution of a Git fetch, a Java compilation (javac HelloWorld.java), and a Java execution (java HelloWorld). The build completed successfully.

```

Started by user anonymous
Building in workspace E:\Jenkins\jobs\Helloworld\workspace
> C:\Program Files\Git\bin\git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url E:\Program #
timeout=10
Fetching upstream changes from E:\Program
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> C:\Program Files\Git\bin\git.exe -c core.askpass=true fetch --tags --progress
E:\Program +refs/heads/*:refs/remotes/origin/*
> C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/master^{commit}" # timeout=10
> C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/origin^{commit}" # timeout=10
Checking out Revision 42f9a82ffad086f05c3a9dfeae40e731a907f5c8f
(refs/remotes/origin/master)
> C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -f
42f9a82ffad086f05c3a9dfeae40e731a907f5c8f
> C:\Program Files\Git\bin\git.exe rev-list
42f9a82ffad086f05c3a9dfeae40e731a907f5c8f # timeout=10
[workspace] $ cmd /c call E:\Apps\tomcat7\temp\hudson1928478766077504601.bat

E:\Jenkins\jobs\Helloworld\workspace>javac HelloWorld.java
E:\Jenkins\jobs\Helloworld\workspace>java HelloWorld
Hello World

E:\Jenkins\jobs\Helloworld\workspace>exit 0
Finished: SUCCESS

```

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 - Unit Testing

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.

**xUnit Plugin - Jenkins**

<https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin>

Dashboard > Jenkins > Plugins > xUnit Plugin

Browse Search

**xUnit Plugin**

Added by [Gregory Boissinot](#), last edited by [Gregory Boissinot](#) on Oct 08, 2015 (view change)

**Jenkins**

- Home
- Mailing lists
- Source code
- Bugtracker
- Security Advisories
- Events
- Donation
- Commercial Support
- Wiki Site Map
- Documents**
- Meet Jenkins
- Use Jenkins
- Extend Jenkins
- Plugins
- Servlet Container Notes

**Plugin Information**

Plugin ID	xunit	Changes	In Latest Release Since Latest Release
Latest Release	<a href="#">1.98 (archives)</a>		
Latest Release Date	Oct 09, 2015		
Required Core Dependencies	<a href="#">JUnit</a> (version: 1.6)		
Usage		Installations	<a href="#">2014-Oct 11692</a> <a href="#">2014-Nov 11557</a> <a href="#">2014-Dec 11631</a> <a href="#">2015-Jan 12105</a> <a href="#">2015-Feb 12262</a> <a href="#">2015-Mar 12891</a> <a href="#">2015-Apr 12894</a> <a href="#">2015-May 12716</a> <a href="#">2015-Jun 13143</a> <a href="#">2015-Jul 13470</a> <a href="#">2015-Aug 13192</a> <a href="#">2015-Sep 13663</a>

This plugin makes it possible to publish the test results of an execution of a testing tool in Jenkins.

**CppUnit output**

**xUnit Plugin - Jenkins**

<https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin>

## Features

- Records xUnit tests
- Mark the build unstable or fail according to threshold values

## Supported tools

### Embedded tools

- \* JUnit itself
- \* [JUnit](#)
- \* [MSUnit](#) (imported from [MSTest Plugin](#))
- \* [NUnit](#) (imported from [NUnit Plugin](#))
- \* [UnitTest++](#)
- \* [Boost Test Library](#)
- \* [PHPUnit](#)
- \* [Free Pascal Unit](#)
- \* [CppUnit](#)
- \* [MbUnit](#)
- \* [GoogleTest](#)
- \* [EmblUnit](#)
- \* [gtest/glib](#)
- \* [QTestLib](#)

### Other plugins as an extension of the xUnit plugin:

- \* [Gallio](#) ([Gallio plugin](#))
- \* [Parasoft C++Test tool](#) ([CppUnit Plugin](#))
- \* [JSUnit](#) ([JSUnit Plugin](#))
- \* [JBehave](#)
- \* [TestComplete](#) ([TestComplete xUnit Plugin](#))

### External contributions

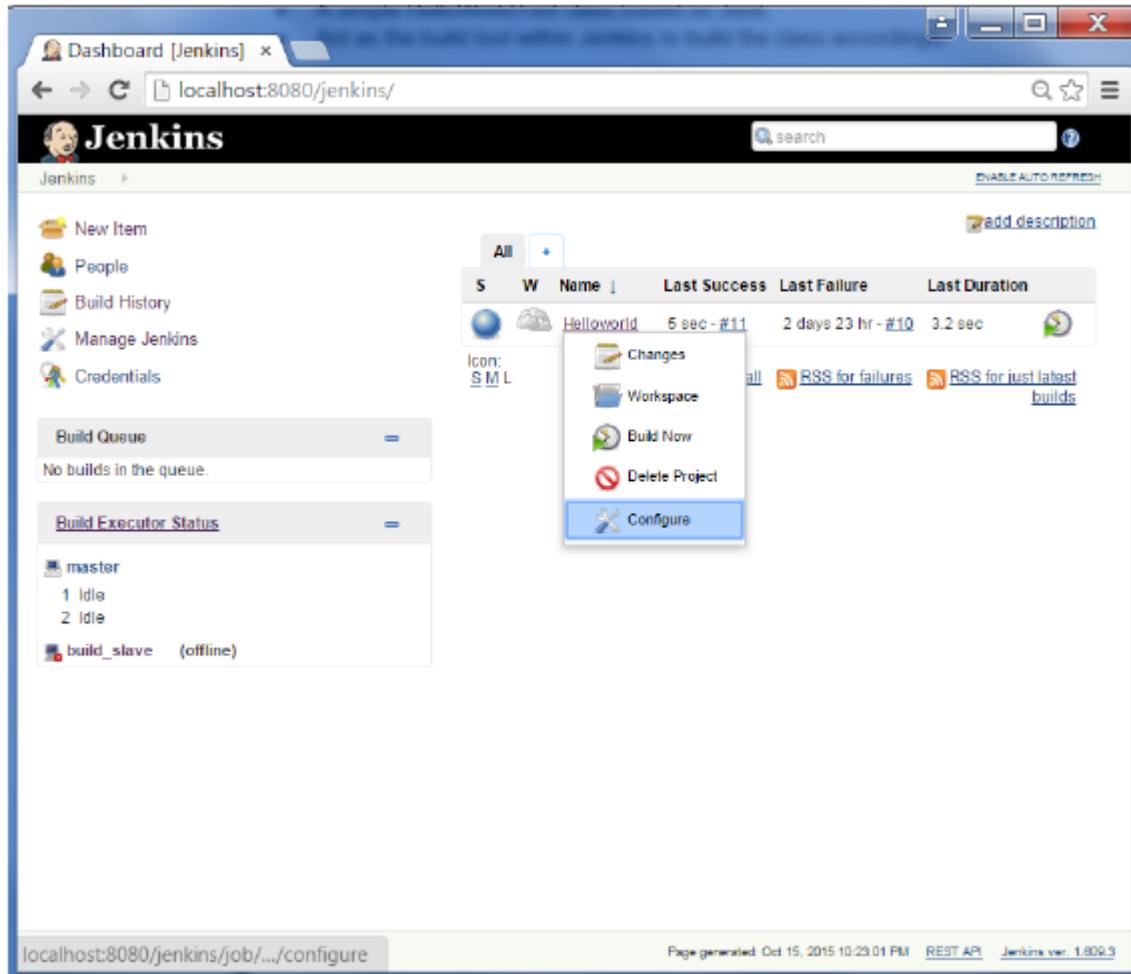
# Example of a Junit Test in Jenkins

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.

The screenshot shows the Jenkins configuration page for a job named "Helloworld". The URL is `localhost:8080/jenkins/job/Helloworld/configure`. The page has tabs for "Branch", "Repository browser" (set to "(Auto)", "Additional Behaviours" (with "Subversion" selected), "Build Triggers" (with options like "Build after other projects are built", "Build periodically", "Build when a change is pushed to GitHub", and "Poll SCM"), and "Build" (containing an "Execute Windows batch command" step with the command `javac HelloWorld.java` and `java HelloWorld`). A dropdown menu under "Add build step" shows options: "Execute Windows batch command", "Execute shell", "Invoke Ant" (which is highlighted in blue), "Invoke top-level Maven targets", and "Set build status to 'pending' on GitHub commit".

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

The screenshot shows the Jenkins configuration page for the 'Helloworld' job. At the top, there's a header bar with the title 'Helloworld Config [J]'. Below it is a breadcrumb navigation: 'Jenkins > Helloworld > configuration'. The main content area is titled 'Additional Behaviours' with an 'Add' button. A radio button for 'Subversion' is selected. Under 'Build Triggers', four options are listed: 'Build after other projects are built', 'Build periodically', 'Build when a change is pushed to GitHub', and 'Poll SCM'. In the 'Build' section, there are two entries. The first is 'Execute Windows batch command' with a command box containing 'javac HelloWorld.java' and 'java HelloWorld'. Below this is a link 'See the list of available environment variables'. To the right of the command box is a red 'Delete' button. The second entry is 'Invoke Ant' with an 'Ant Version' dropdown set to 'Default' and a 'Targets' dropdown. To the right of the targets dropdown is a red 'Delete' button. At the bottom of the configuration page are 'Save' and 'Apply' buttons.

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

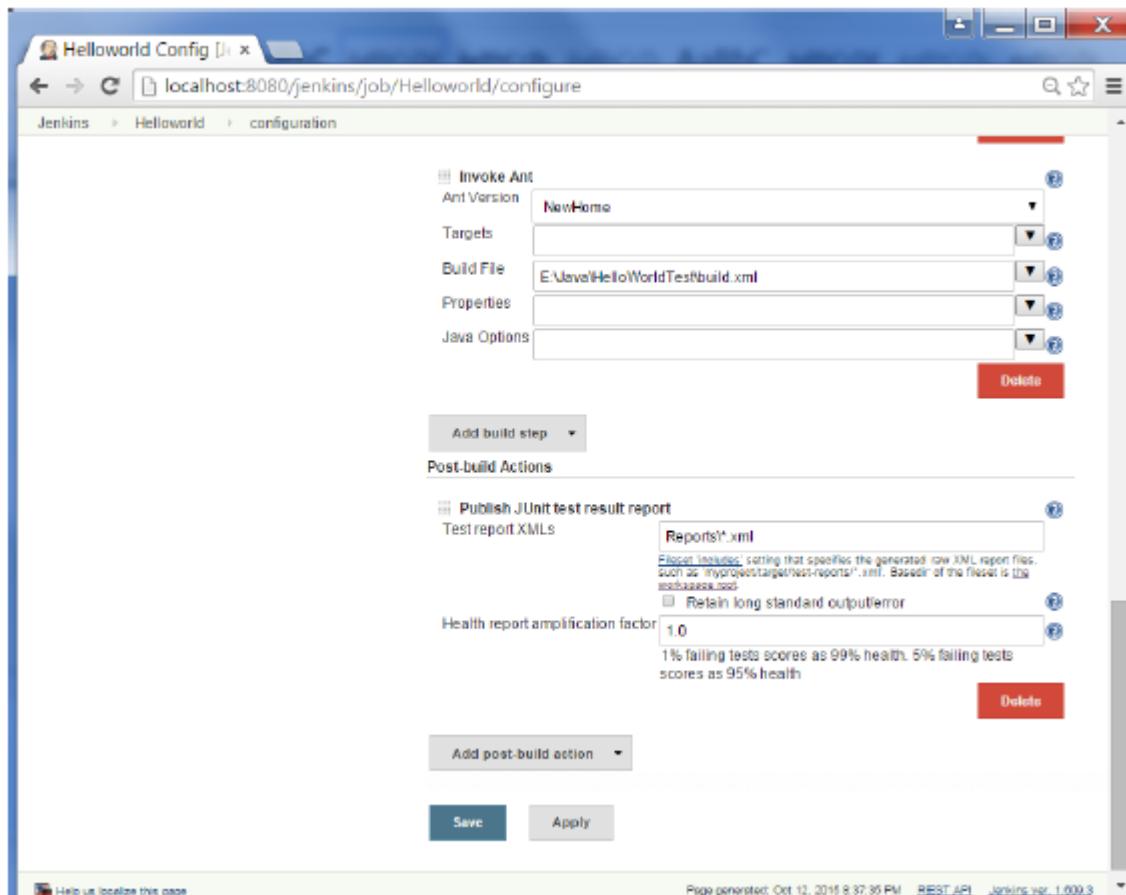
The screenshot shows the Jenkins configuration page for a job named "Helloworld". Under the "Build" section, there is a "Execute Windows batch command" step with the command "javac HelloWorld.java" and "java HelloWorld". Below it is an "Invoke Ant" step with "Ant Version" set to "NewHome", "Targets" set to "NewHome", "Build File" set to "E:\Java\HelloWorldTest\build.xml", and "Properties" and "Java Options" both empty. Under "Post-build Actions", there is a "Publish JUnit test result report" step with "Test report XMLs" set to "Reports\\*.xml". At the bottom are "Save" and "Apply" buttons.

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

The screenshot shows the Jenkins configuration page for the same "Helloworld" job. A dropdown menu has been opened under the "Post-build Actions" section, and the "Publish JUnit test result report" option is highlighted with a blue selection bar. Other options in the menu include "Publish FindBugs analysis results", "Publish combined analysis results", "Aggregate downstream test results", "Archive the artifacts", "Build other projects", "Publish Javadoc", "Publish Selenium Report", "Record fingerprints of files to track usage", "Git Publisher", "Deploy war/ear to a container", "E-mail Notification", and "Set build status on GitHub commit". At the bottom are "Save" and "Apply" buttons.

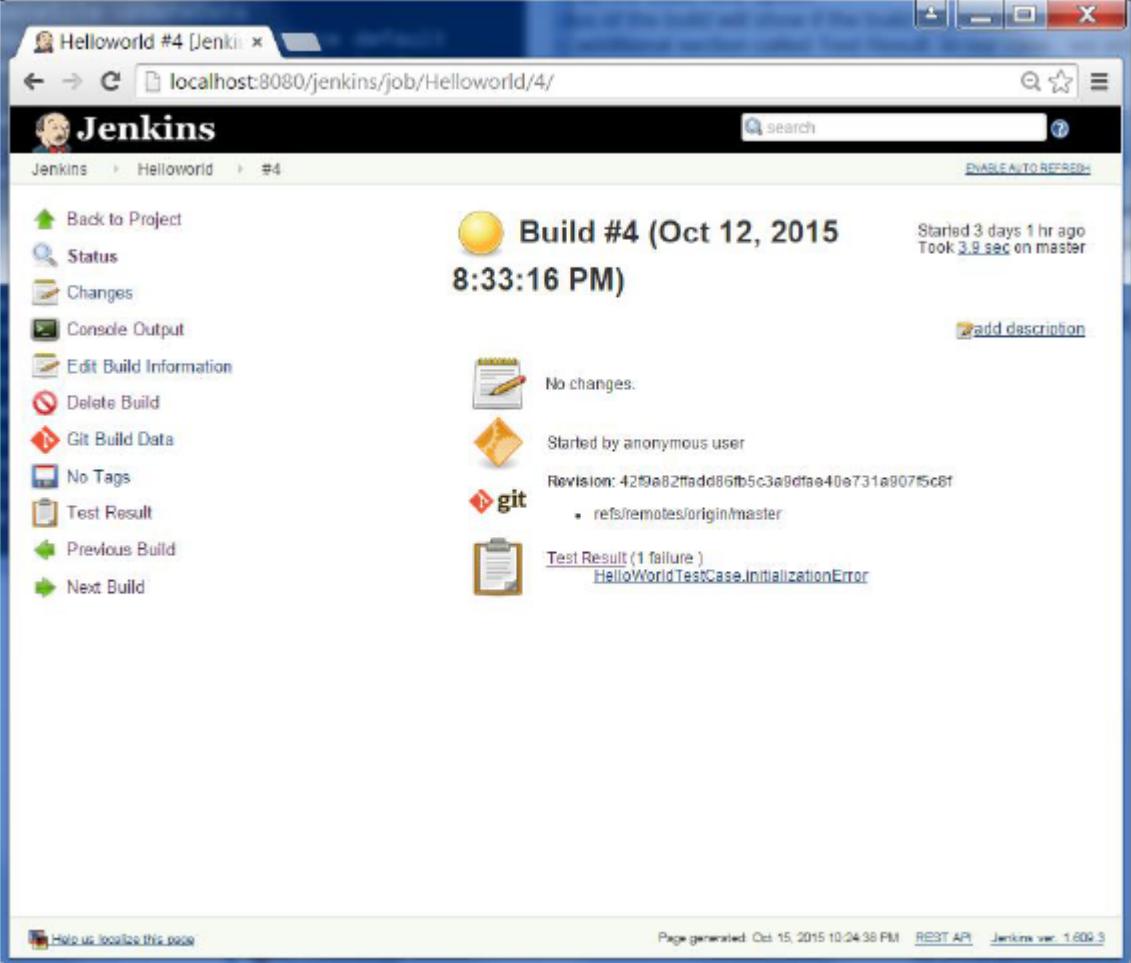
**Step 6** – In the Test reports XML's, enter the location as shown below. Ensure that Reports is a folder which is created in the HelloWorld project workspace. The “\*.xml” basically tells Jenkins to pick up the result xml files which are produced by the running of the Junit test cases. These xml files 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.



The screenshot shows the Jenkins interface for a build named "Helloworld #4". The main title is "Build #4 (Oct 12, 2015) 8:33:16 PM". It indicates the build started 3 days 1 hr ago and took 3.9 sec on master. A "No changes." message is shown. The build was started by an anonymous user from a git commit (revision: 42f9a82ffadd86fb5c3a0dfa040e731a807f5c8f, refs/remotes/origin/master). A test result section shows one failure: "HelloWorldTestCase.InitializationError". Navigation links on the left include Back to Project, Status, Changes, Console Output, Edit Build Information, Delete Build, Git Build Data, No Tags, Test Result, Previous Build, and Next Build.

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

The screenshot shows the Jenkins Test Result page for the 'Helloworld #4 Test R' job. The left sidebar contains links like Back to Project, Status, Changes, Console Output, Edit Build Information, History, Git Build Data, No Tags, Test Result, and Previous Build. The main content area is titled 'Test Result' and shows '1 failures'. It includes a table for 'All Failed Tests' with one entry: 'HelloWorldTestCase InitializationError' (Duration: 10 ms, Age: 1). Another table for 'All Tests' shows a single test in the 'root' package taking 10 ms, with 1 fail and 0 passes.

## Jenkins - 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.

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links for New Item, People, Build History, Manage Jenkins, and Credentials. Below that are two dropdown menus: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Manage Jenkins' and contains several configuration links:

- Configure System**: Configure global settings and paths.
- Configure Global Security**: Secure Jenkins; define who is allowed to access/use the system.
- Reload Configuration from Disk**: Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins. ([updates available](#))
- System Information**: Displays various environmental information to assist trouble-shooting.
- System Log**: System log captures output from `java.util.logging` output related to Jenkins.
- Load Statistics**: Check your resource utilization and see if you need more computers for your builds.
- Jenkins CLI**: Access/manage Jenkins from your shell, or from your script.
- Script Console**: Executes arbitrary script for administration/trouble-shooting/diagnostics.
- Manage Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Manage Credentials**: Create/delete/modify the credentials that can be used by Jenkins and by jobs running in Jenkins to connect to 3rd party services.
- About Jenkins**: See the version and license information.

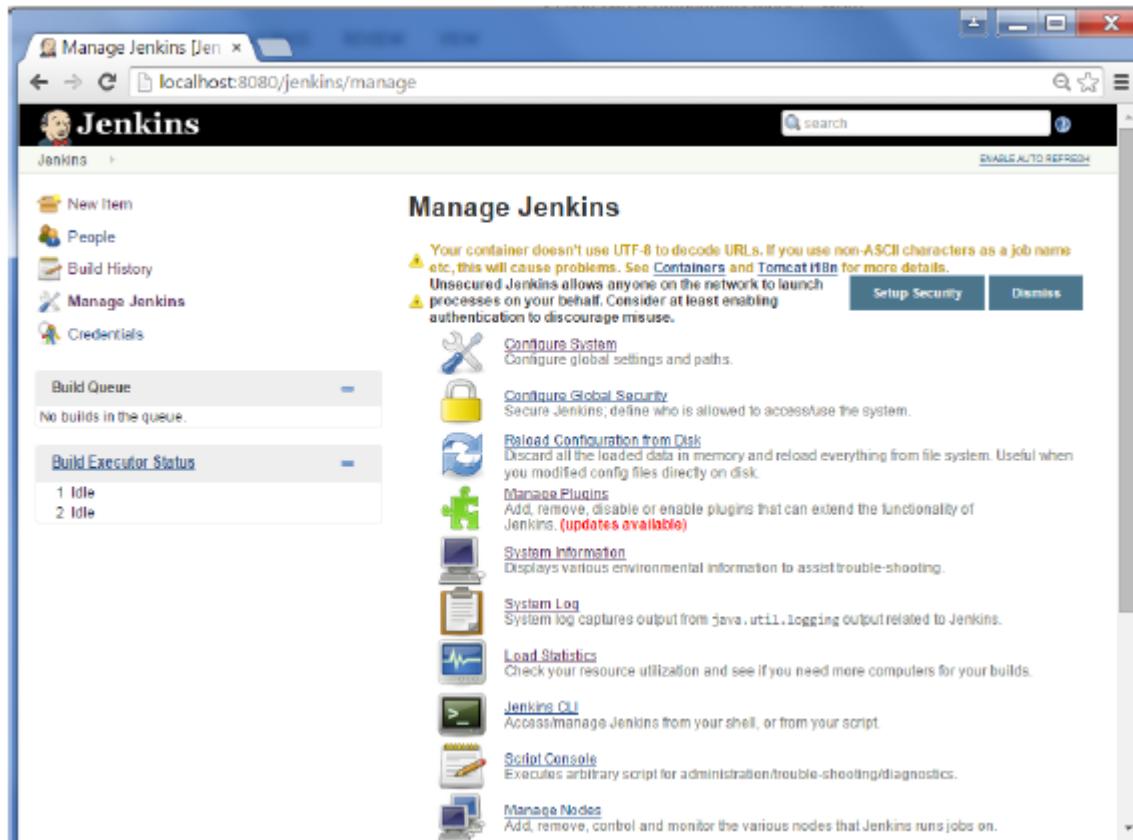
**Step 2** – Find the Hudson Selenium Plugin and choose to install. Restart the Jenkins instance.

The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected. A search bar at the top right is set to 'Filter: selenium'. The table below lists several available plugins:

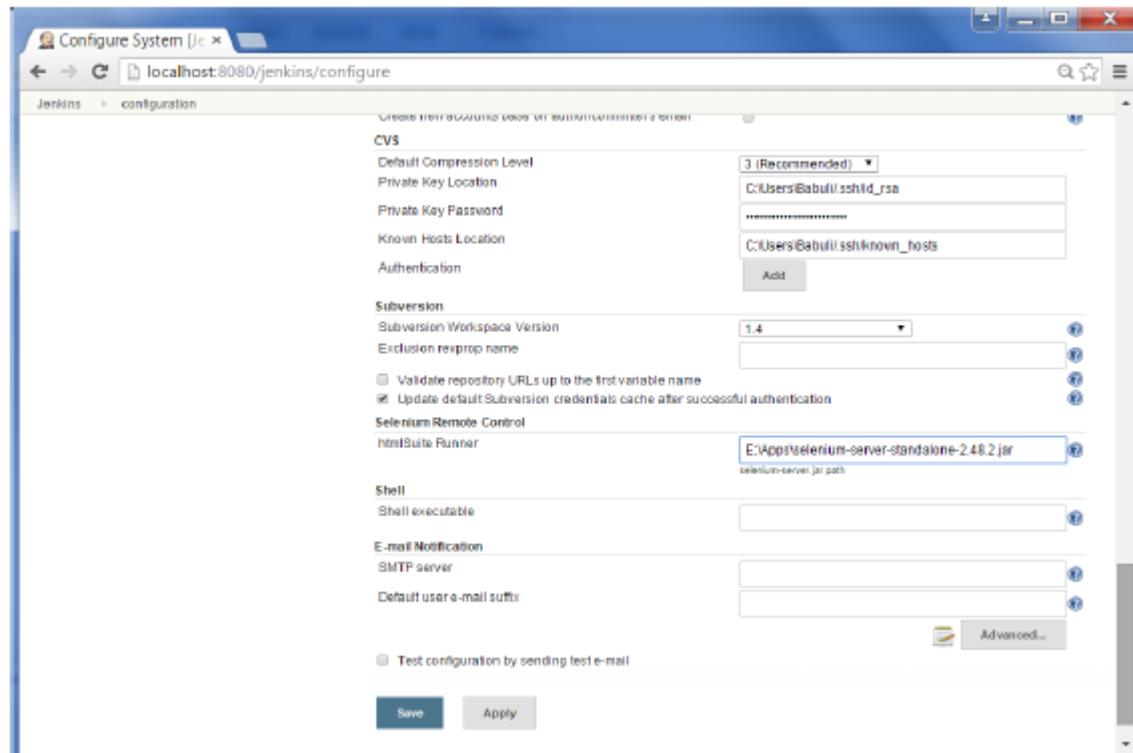
	Name	Version
<input checked="" type="checkbox"/>	Selenium Auto Exec Server(AES) plugin	0.5
<input checked="" type="checkbox"/>	Hudson Selenium plugin	0.4
<input checked="" type="checkbox"/>	Selenium HTML report	0.94
<input checked="" type="checkbox"/>	TestingBot plugin	1.11
<input checked="" type="checkbox"/>	TestLink Plugin	3.10
<input checked="" type="checkbox"/>	Nemvana Plugin for Jenkins	1.02.06
<input checked="" type="checkbox"/>	Source OnDemand plugin	1.141
<input checked="" type="checkbox"/>	Selenium Builder plugin	1.14
<input checked="" type="checkbox"/>	SeleniumRC plugin	1.14

At the bottom, there are three buttons: 'Install without restart', 'Download now and install after restart', and 'Update information obtained'.

### 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

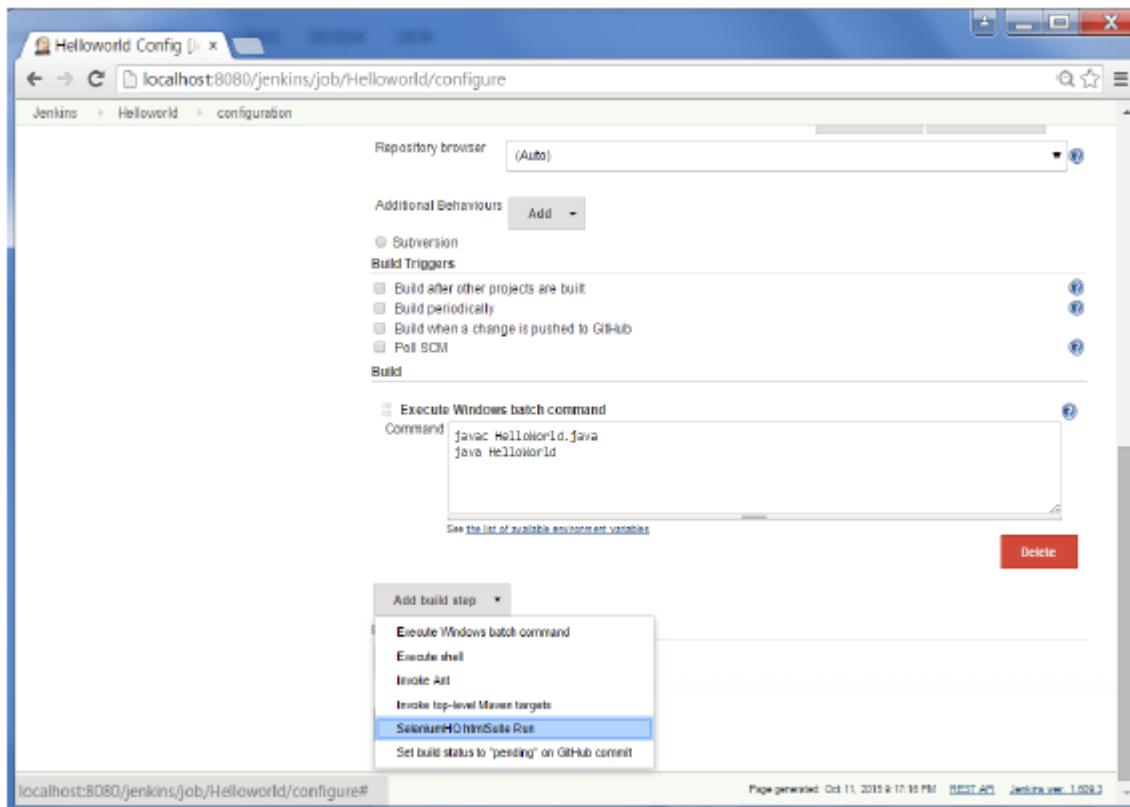
Click on the download for the Selenium standalone server.

The screenshot shows the SeleniumHQ website at [www.seleniumhq.org/download/](http://www.seleniumhq.org/download/). The 'Download' tab is selected. On the left, there's a sidebar with links like 'Selenium Downloads', 'Latest Releases', 'Previous Releases', 'Source Code', and 'Maven Information'. Below that is a 'Donate to Selenium' section with a 'Donate' button and payment method icons (PayPal, Credit Card, VISA, MasterCard). Further down is a 'through sponsorship' section with a link to 'selenium project'. To the right, under 'Downloads', there's a section for the 'Selenium Standalone Server'. It describes the server's purpose, provides a download link for version 2.48.2, and links to the wiki page for Grid configuration.

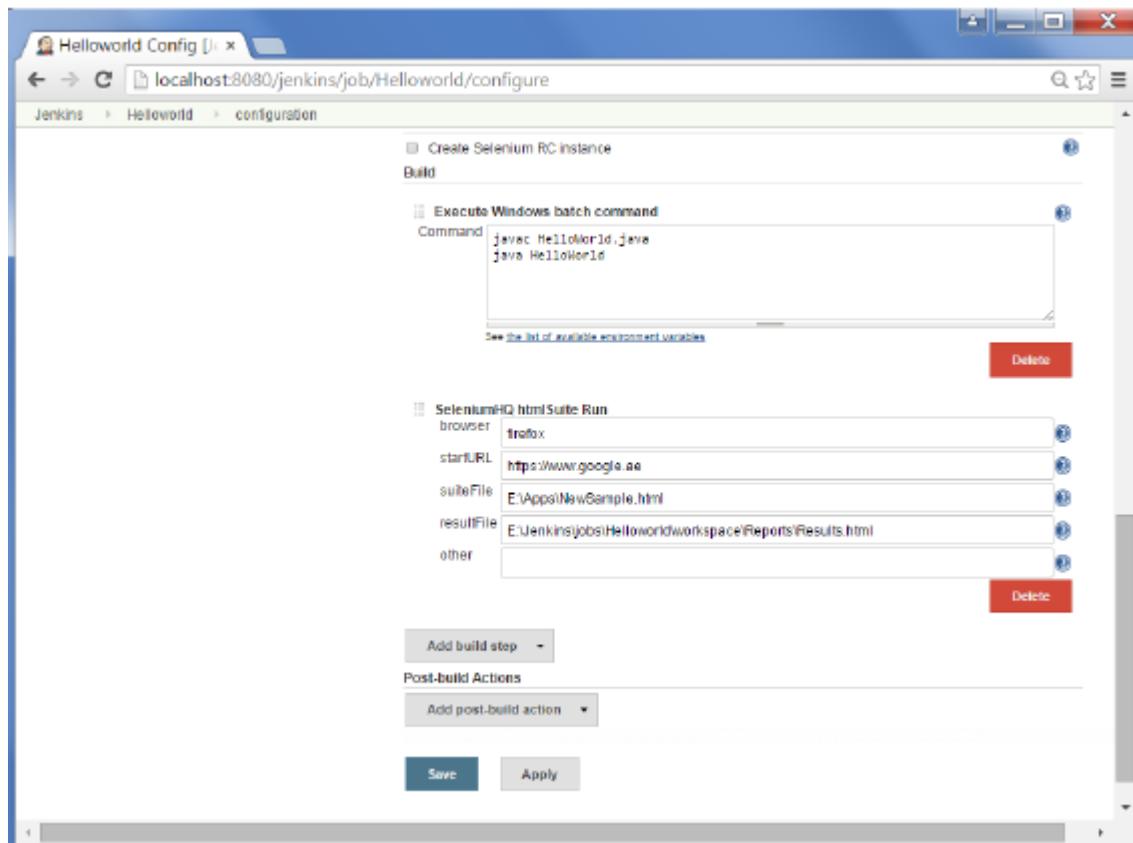
**Step 5** – Go back to your dashboard and click on the Configure option for the HelloWorld project.

The screenshot shows the Jenkins dashboard at [localhost:8080/jenkins/](http://localhost:8080/jenkins/). The 'Dashboard [Jenkins]' is visible in the address bar. The main area shows the 'Jenkins' logo and a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below that are 'Build Queue' and 'Build Executor Status' sections. In the center, there's a table of jobs. The 'HelloWorld' job is listed with its last success at '23 hr - #12', last failure at '23 hr - #10', and last duration at '3.7 sec'. A context menu is open over the 'HelloWorld' row, with the 'Configure' option highlighted. At the bottom of the screen, the URL 'localhost:8080/jenkins/job/HelloWorld/configure' is shown in the status bar.

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



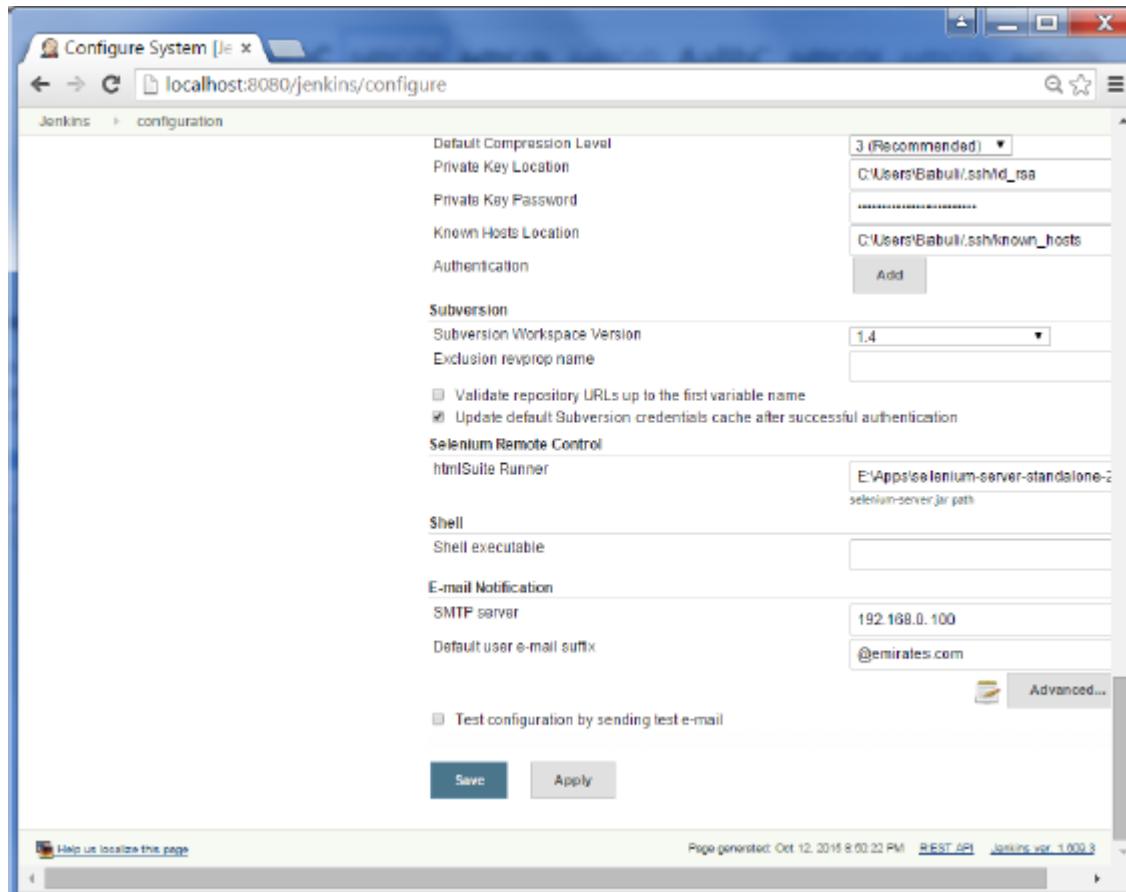
**Step 7** – Add the necessary details for the selenium test. Here the suiteFile is the TestSuite generated by using the Selenium IDE. Click on Save and execute a build. Now the post build will launch the selenium driver, and execute the html test.



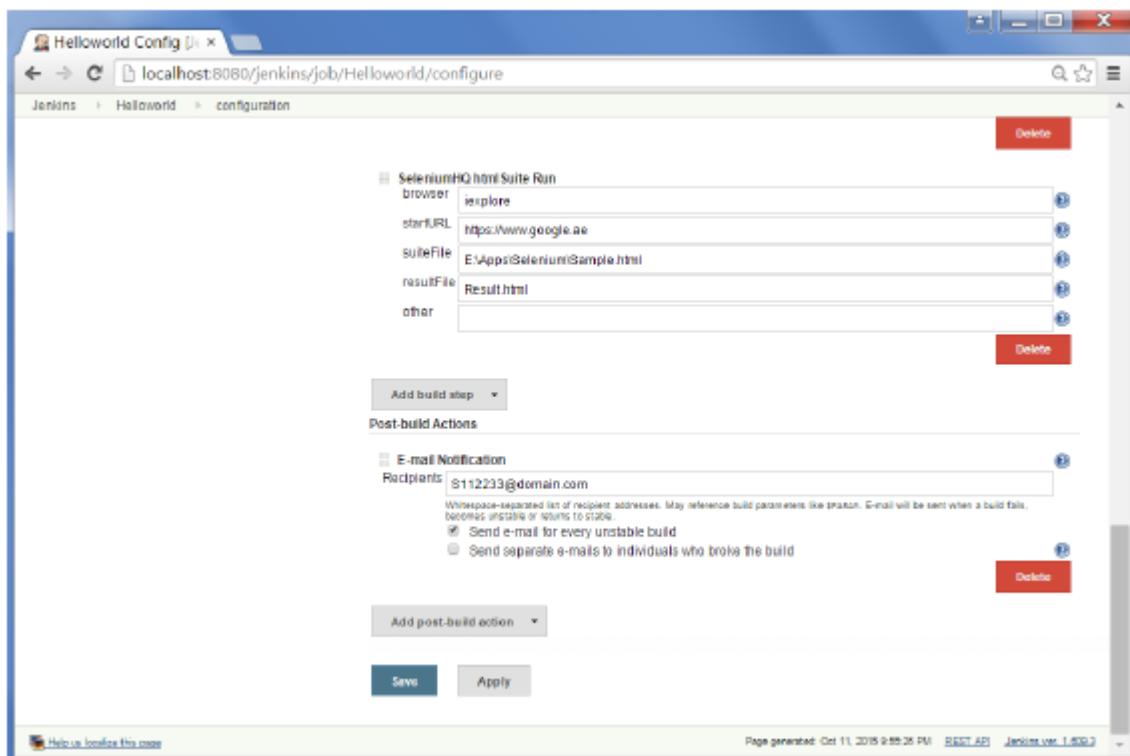
## Jenkins - Notification

Jenkins comes with an out of box facility to add an email notification for a build project.

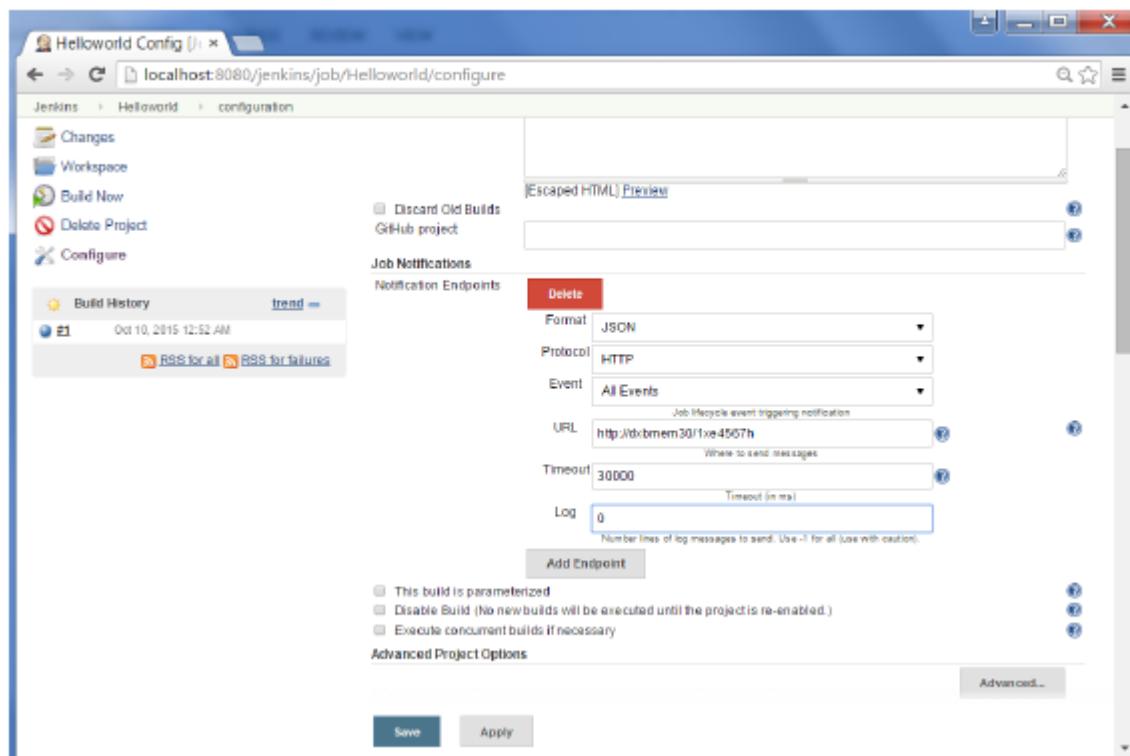
**Step 1** – Configuring an SMTP server. Goto Manage Jenkins → Configure System. Go to the E-mail notification section and enter the required SMTP server and user email-suffix details.



**Step 2** – Configure the recipients in the Jenkins project - When you configure any Jenkins build project, right at the end is the ability to add recipients who would get email notifications for unstable or broken builds. Then click on the Save button.



Apart from the default, there are also notification plugin's available in the market. An example is the notification plugin from Tikal Knowledge which allows sending Job Status notifications in JSON and XML formats. This plugin enables end-points to be configured as shown below.



Here are the details of each option –

**"Format"** – This is the notification payload format which can either be JSON or XML.

**"Protocol"** – protocol to use for sending notification messages, HTTP, TCP or UDP.

**"Event"** – The job events that trigger notifications: Job Started, Job Completed, Job Finalized or All Events (the default option).

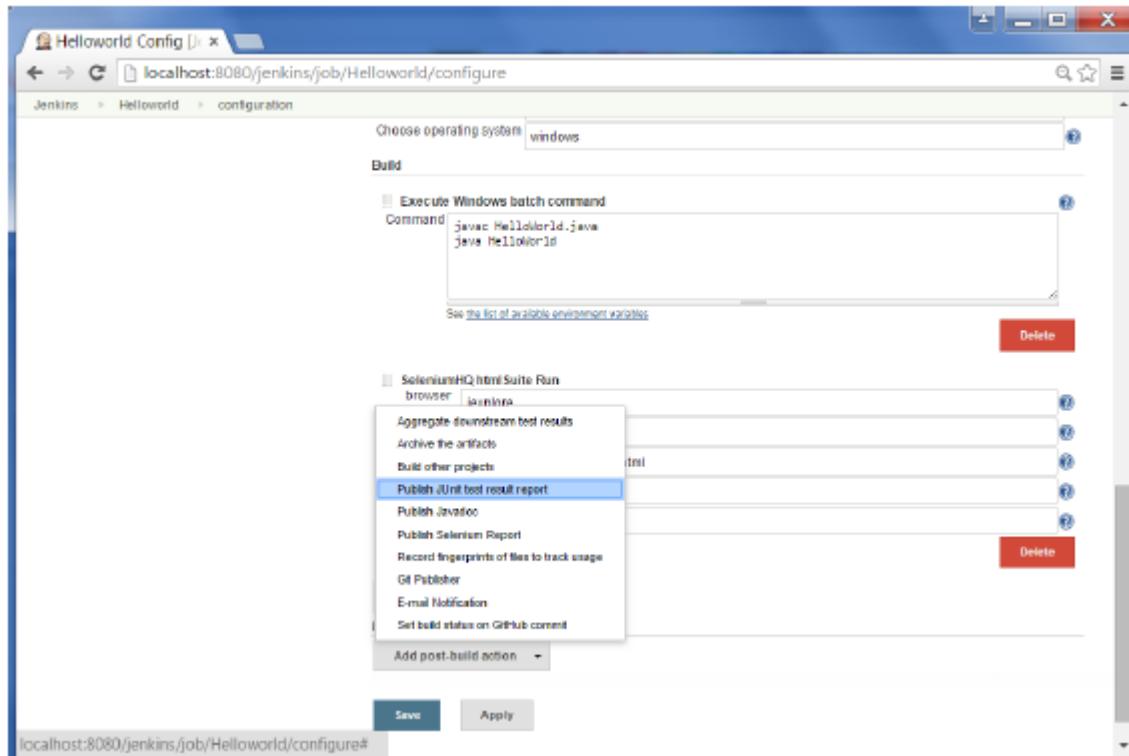
**"URL"** – URL to send notifications to. It takes the form of "http://host" for HTTP protocol, and "host:port" for TCP and UDP protocols.

**"Timeout"** – Timeout in milliseconds for sending notification request, 30 seconds by default.

## Jenkins - Reporting

As demonstrated in the earlier section, there are many reporting plugins available with the simplest one being the reports available for jUnit tests.

In the Post-build action for any job, you can define the reports to be created. After the builds are complete, the Test Results option will be available for further drill-down.



## Jenkins - Code Analysis

Jenkins has a host of Code Analysis plugin. The various plugins can be found at <https://wiki.jenkins-ci.org/display/JENKINS/Static+Code+Analysis+Plugins>

The screenshot shows the Jenkins Static Code Analysis Plugins page. On the left, there's a sidebar with links like Home, Mailing lists, Source code, Bugtracker, Security Advisories, Events, Donation, Commercial Support, and Wiki Site Map. Under 'Documents', it lists Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, Servlet Container, and Notes. The main content area has a title 'Static Code Analysis Plug-ins' with a sub-section for 'analysis-core'. It shows the plugin ID as 'analysis-core', the latest release as '1.74 (archives)' from 'Sep 07, 2015', and required core dependencies including 'ant', 'token-macro', 'maven-plugin', 'maven-project', and 'dashboard-view'. It also shows GitHub links for source code, issue tracking, and pull requests, and credits 'Ulli Hafner (id: drull)' as the maintainer. A 'Usage' section contains a line graph titled 'analysis-core - installations' showing the number of installations per month from October 2014 to September 2015. The installations count remains relatively stable around 25,000. Below the graph, a table lists the dates of installations. At the bottom of the page, a message says 'Waiting for wiki.jenkins-ci.org...'.

This plugin provides utilities for the static code analysis plugins. Jenkins can parse the results file from various Code Analysis tools such as CheckStyle, FindBugs, PMD etc. For each corresponding code analysis tool, a plugin in Jenkins needs to be installed.

Additionally the add-on plugin Static Analysis Collector is available that combines the individual results of these plugins into a single trend graph and view.

The plugins can provide information such as

- The total number of warnings in a job

- A showing of the new and fixed warnings of a build

- Trend Reports showing the number of warnings per build

- Overview of the found warnings per module, package, category, or type

- Detailed reports of the found warnings optionally filtered by severity (or new and fixed)

## Jenkins - Distributed Builds

Sometimes many build machines are required if there are instances wherein there are a larger and heavier projects which get built on a regular basis. And running all of these builds on a central machine may not be the best option. In such a scenario, one can configure other Jenkins machines to be slave machines to take the load off the master Jenkins server.

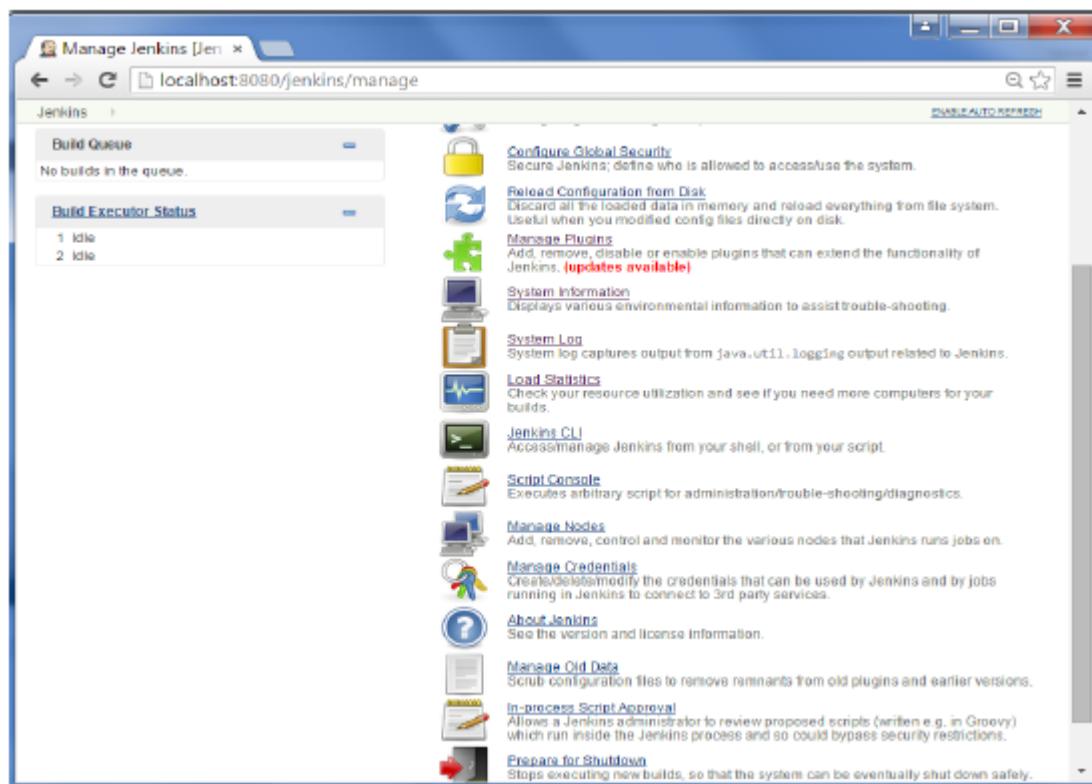
Sometimes you might also need several different environments to test your builds. In this case using a slave to represent each of your required environments is almost a must.

A slave is a computer that is set up to offload build projects from the master and once setup this distribution of tasks is fairly automatic. The exact delegation behavior depends on the configuration of each project; some projects may choose to "stick" to a particular machine for a build, while others may choose to roam freely between slaves.

Since each slave runs a separate program called a "slave agent" there is no need to install the full Jenkins (package or compiled binaries) on a slave. There are various ways to start slave agents, but in the end the slave agent and Jenkins master needs to establish a bi-directional communication link (for example a TCP/IP socket.) in order to operate.

To set up slaves/nodes in Jenkins follow the steps given below.

**Step 1** – Go to the Manage Jenkins section and scroll down to the section of Manage Nodes.



**Step 2** – Click on New Node

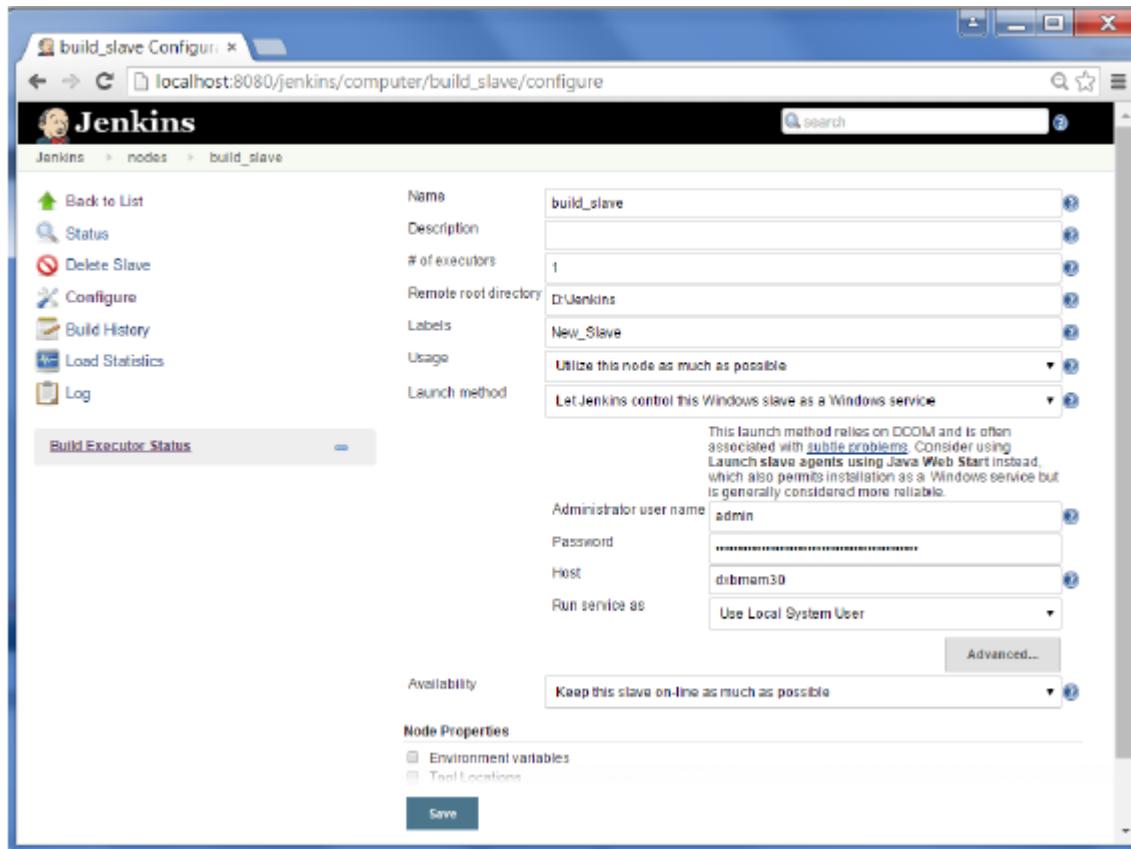
The screenshot shows the Jenkins 'Nodes' page. At the top, there's a header bar with the Jenkins logo and a search bar. Below the header, the main content area has a title 'Jenkins' and a breadcrumb navigation 'Jenkins > nodes'. On the left, there are several links: 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. In the center, there's a table titled 'Nodes' with one row. The row contains the following columns: S (Status icon), Name (master), Architecture (Windows 7 (x86)), Clock Difference (In sync), Free Disk Space (229.89 GB), Free Swap Space (12.13 GB), and Frequency (3 min 11 sec). Below the table, there's a link 'Data obtained' and a status summary: '3 min 12 sec' for disk space and '3 min 12 sec' for swap space. A 'Refresh status' button is located at the bottom right of the table. On the far left, there are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 12, 2015 9:22:44 PM', 'REST API', and 'Jenkins ver. 1.600.3'.

**Step 3** – Give a name for the node, choose the Dumb slave option and click on Ok.

The screenshot shows the 'New Node' configuration page. The URL in the address bar is 'localhost:8080/jenkins/computer/new'. The page title is 'Jenkins' with a breadcrumb 'Jenkins > nodes'. On the left, there are links: 'Back to Dashboard', 'Manage Jenkins', 'New Node' (which is selected and highlighted in blue), and 'Configure'. In the center, there's a form for creating a new node. The 'Node name' field is filled with 'build\_slave'. Below it, there's a radio button group with 'Dumb Slave' selected. A tooltip explains: 'Adds a plain, dumb slave to Jenkins. This is called "dumb" because Jenkins doesn't provide higher level of integration with these slaves, such as dynamic provisioning. Select this type if no other slave types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.' To the right of the form is an 'OK' button. On the far left, there's a collapsed 'Build Queue' section (No builds in the queue) and a 'Build Executor Status' section (1 Idle, 2 Idle). At the bottom, there are links for 'Help us localize this page', 'Page generated: Oct 12, 2015 9:25:11 PM', 'REST API', and 'Jenkins ver. 1.600.3'.

**Step 4** – Enter the details of the node slave machine. In the below example, we are considering the slave machine to be a windows machine, hence the option of "Let Jenkins

control this Windows slave as a Windows service" was chosen as the launch method. We also need to add the necessary details of the slave node such as the node name and the login credentials for the node machine. Click the Save button. The Labels for which the name is entered as "New\_Slave" is what can be used to configure jobs to use this slave machine.



Once the above steps are completed, the new node machine will initially be in an offline state, but will come online if all the settings in the previous screen were entered correctly. One can at any time make the node slave machine as offline if required.

The screenshot shows the Jenkins 'Nodes' page. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. Below this is a table titled 'Nodes' with columns for Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, and Free Temp S. It lists two nodes: 'build\_slave' (Windows 7 (x86)) and 'master' (Windows 7 (x86)). The 'master' node is marked as 'In sync' with 229.89 GB free disk space, 12.13 GB free swap space, and 229.8 free temp space. The 'build\_slave' node is marked as 'Data obtained' with 3 ms, 2 ms, and 1 ms respectively. A 'Refresh status' button is at the bottom right of the table. On the left, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (master: 1 idle, 2 idle; build\_slave: offline). At the bottom, there's a footer with links for 'Help us localize this page', 'Page generated: Oct 12, 2016 9:31:43 PM', 'REST API', and 'Jenkins ver. 1.600.3'.

## Jenkins - Automated Deployment

There are many plugins available which can be used to transfer the build files after a successful build to the respective application/web server. One example is the "Deploy to container Plugin". To use this follow the steps given below.

**Step 1** – Go to Manage Jenkins → Manage Plugins. Go to the Available section and find the plugin "Deploy to container Plugin" and install the plugin. Restart the Jenkins server.

The screenshot shows the Jenkins Plugin Manager interface. The title bar says 'Update Center [Jenk] x' and the address bar shows 'localhost:8080/jenkins/pluginManager/available'. The main content area is titled 'Plugin Manager' and has a table with columns 'Name' and 'Version'. A checkbox labeled 'Install' is checked next to the 'Deploy to container' plugin. The table lists several other plugins:

Install	Name	Version
<input type="checkbox"/>	<a href="#">Artifact Deployer Plug-in</a>	0.33
<input type="checkbox"/>	<a href="#">AWS Lambda Plugin</a>	0.3.1
<input type="checkbox"/>	<a href="#">AWS Elastic Beanstalk Deployment Plugin</a>	0.0.3
<input type="checkbox"/>	<a href="#">Capitomatic Plugin</a>	0.1.0
<input type="checkbox"/>	<a href="#">AWS CodeDeploy Plugin for Jenkins</a>	1.7
<input type="checkbox"/>	<a href="#">CRX Content Package Deployer Plugin</a>	1.3.2
<input checked="" type="checkbox"/>	<a href="#">Deploy to container Plugin</a>	1.10
<input type="checkbox"/>	<a href="#">Deploy to WebSphere container Plugin</a>	1.0
<input type="checkbox"/>	<a href="#">Xebialabs XL Deploy Plugin</a>	5.0.0

At the bottom of the page are three buttons: 'Install without restart', 'Download now and install after restart', and 'Update information'.

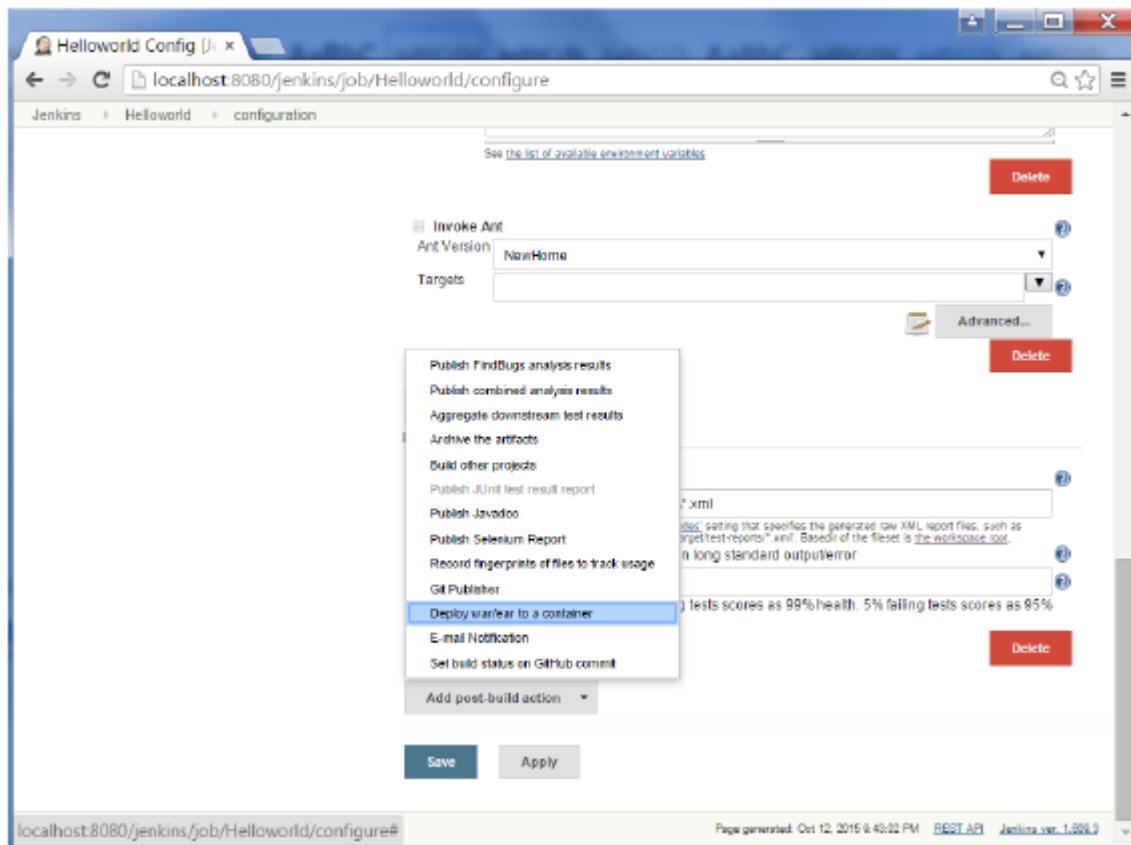
This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build.

Tomcat 4.x/5.x/6.x/7.x

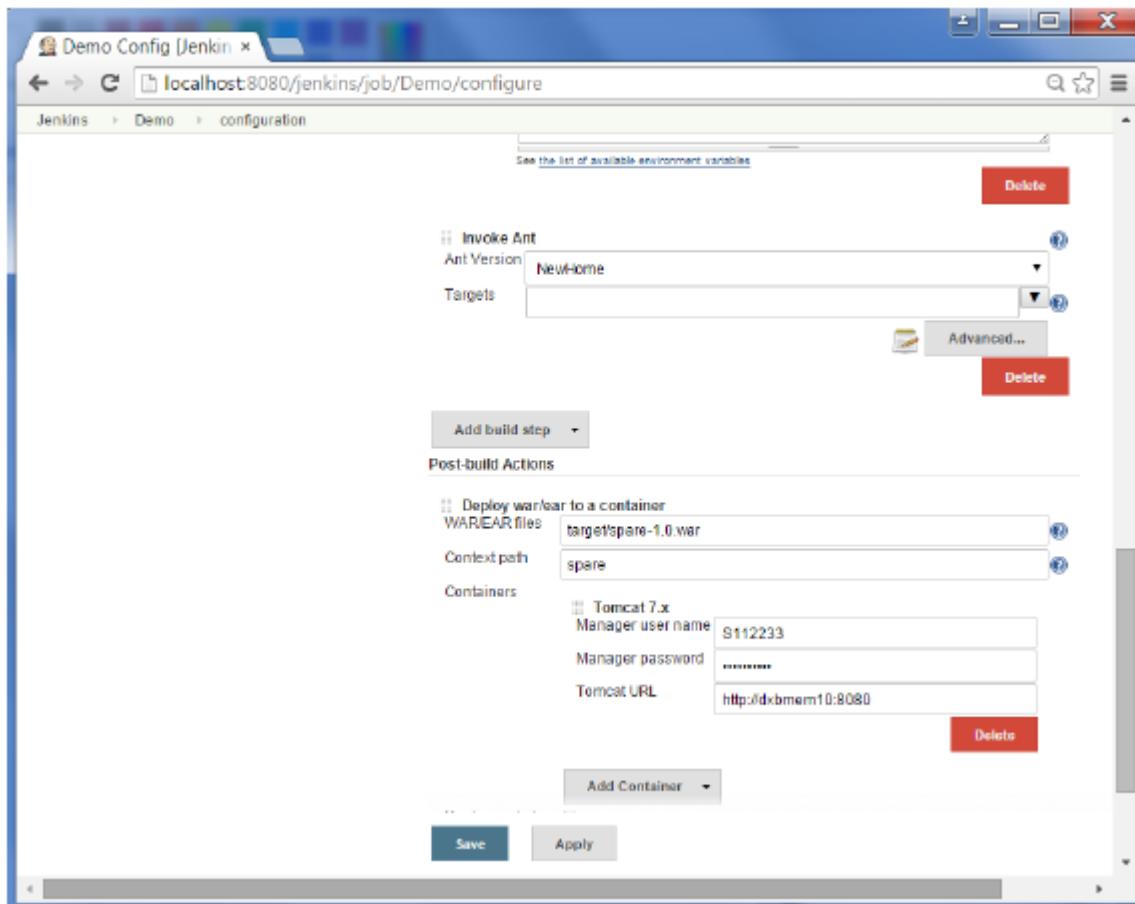
JBoss 3.x/4.x

Glassfish 2.x/3.x

**Step 2** – Go to your Build project and click the Configure option. Choose the option “Deploy war/ear to a container”



**Step 3** – In the Deploy war/ear to a container section, enter the required details of the server on which the files need to be deployed and click on the Save button. These steps will now ensure that the necessary files get deployed to the necessary container after a successful build.



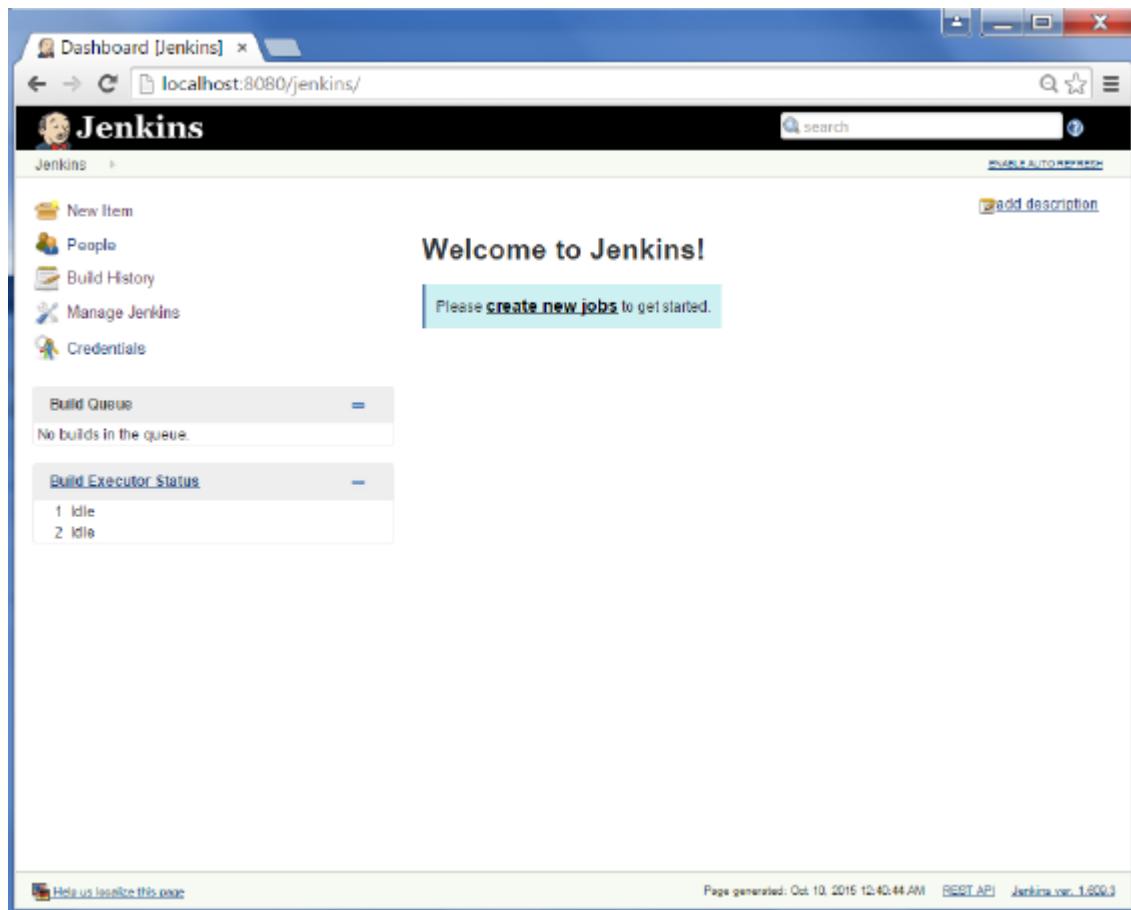
## Jenkins - Metrics & Trends

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

**Step 1** – Go to the Jenkins dashboard and click on Manage Jenkins



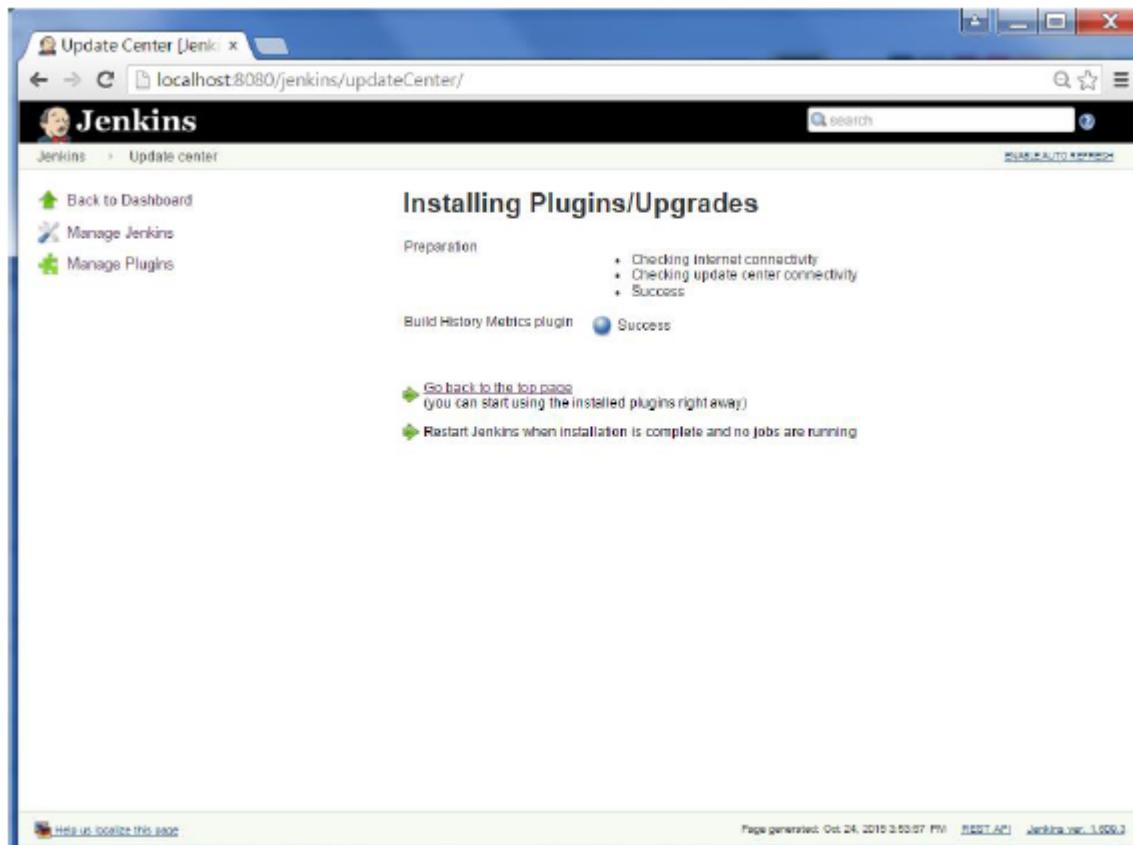
**Step 2** – Go to the Manage Plugins option.

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links for New Item, People, Build History, Manage Jenkins (which is selected), and Credentials. Below the sidebar are two collapsed sections: Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). The main content area is titled "Manage Jenkins". It contains several configuration links with icons: Configure System (Configure global settings and paths), Configure Global Security (Secure Jenkins; define who is allowed to access/use the system), Reload Configuration from Disk (Discard all the loaded data in memory and reload everything from file system), Manage Plugins (Add, remove, disable or enable plugins that can extend the functionality of Jenkins. **(updates available)**), System Information (Displays various environmental information to assist trouble-shooting), System Log (System log captures output from java.util.logging output related to Jenkins), Load Statistics (Check your resource utilization and see if you need more computers for your builds), Jenkins CLI (Access/manage Jenkins from your shell, or from your script), Script Console (Executes arbitrary script for administration/trouble-shooting/diagnostics), Manage Nodes (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), Manage Credentials (Create/delete/modify the credentials that can be used by Jenkins and by jobs running in Jenkins to connect to 3rd party services), and About Jenkins (Get the version and license information).

**Step 3** – Go to the Available tab and search for the plugin ‘Build History Metrics plugin’ and choose to ‘install without restart’.

The screenshot shows the Jenkins Plugin Manager interface. At the top, it says "Update Center [Jenkin...]" and "localhost:8080/jenkins/pluginManager/available". The main area has tabs for Updates, Available (which is selected), Installed, and Advanced. A search bar at the top right shows "Filter: build-history-metrics-plugin". Below the tabs is a table with columns: Name, Version, and Actions. One row is visible: "Build History Metrics plugin" (Version 1.2). Underneath the table are two buttons: "Install without restart" (highlighted in blue) and "Download now and install after restart". At the bottom, it says "Update information obtained: 2 mi".

**Step 4** – The following screen shows up to confirm successful installation of the plugin. Restart the Jenkins instance.



When you go to your Job page, you will see a table with the calculated metrics. Metric's are shown for the last 7 days, last 30 days and all time.

The screenshot shows the Jenkins Project Helloworld dashboard. On the left, there's a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main area has a title 'Project Helloworld'. Below it are two buttons: 'add description' and 'Disable Project'. There are two sections with icons: 'Workspace' (a folder icon) and 'Recent Changes' (a document icon). A large table displays performance metrics:

	Last 7 Days	0 ms
<b>MTTR</b>	Last 30 Days	23 hr
	All Time	23 hr
<b>MTTF</b>	Last 7 Days	0 ms
	Last 30 Days	2 days 4 hr
	All Time	2 days 4 hr
<b>Standard Deviation</b>	Last 7 Days	0 ms
	Last 30 Days	52 sec
	All Time	52 sec

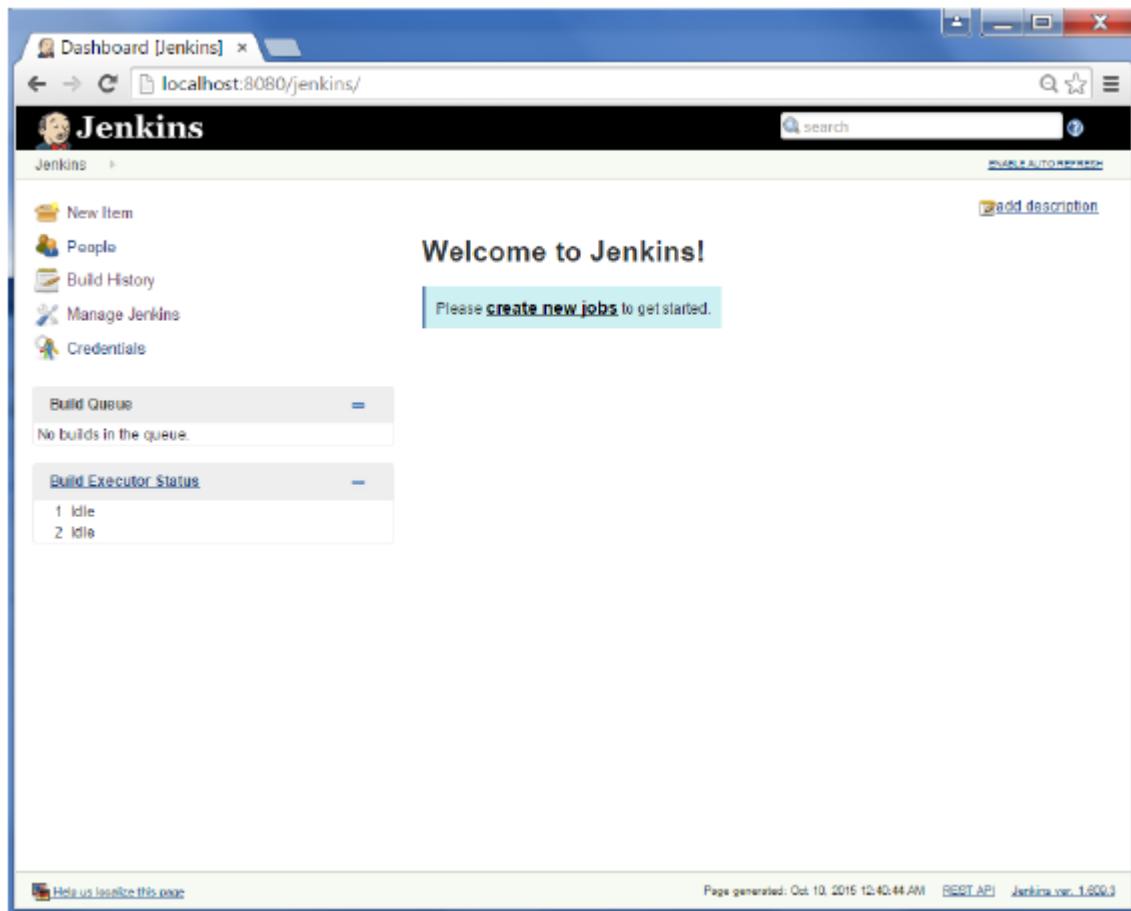
Below the table is a section titled 'Permalinks' with a bulleted list of links:

- Last build (#12), 5.5 sec ago
- Last stable build (#11), 8 days 17 hr ago
- Last successful build (#11), 8 days 17 hr ago
- Last failed build (#12), 5.5 sec ago
- Last unstable build (#4), 11 days ago
- Last unsuccessful build (#12), 5.5 sec ago

At the bottom, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 3:57:10 PM', 'REST API', and 'Jenkins ver. 1.600.3'.

To see overall trends in Jenkins, there are plugins available to gather information from within the builds and Jenkins and display them in a graphical format. One example of such a plugin is the 'Hudson global-build-stats plugin'. So let's go through the steps for this.

### Step 1 – Go to the Jenkins dashboard and click on Manage Jenkins



**Step 2** – Go to the Manage Plugins option

The screenshot shows the Jenkins 'Manage Jenkins' interface at [localhost:8080/jenkins/manage](http://localhost:8080/jenkins/manage). The left sidebar includes links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Under 'Build Queue', it says 'No builds in the queue.' Under 'Build Executor Status', there are 1 idle and 2 idle executors. The main content area is titled 'Manage Jenkins' and contains several configuration options:

- Configure System**: Configure global settings and paths.
- Configure Global Security**: Secure Jenkins; define who is allowed to access/use the system.
- Reload Configuration from Disk**: Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins. ([Updates available](#))
- System Information**: Displays various environmental information to assist trouble-shooting.
- System Log**: System log captures output from `java.util.logging` output related to Jenkins.
- Load Statistics**: Check your resource utilization and see if you need more computers for your builds.
- Jenkins CLI**: Access/manage Jenkins from your shell, or from your script.
- Script Console**: Executes arbitrary script for administration/trouble-shooting/diagnostics.
- Manage Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Manage Credentials**: Create/delete/modify the credentials that can be used by Jenkins and by jobs running in Jenkins to connect to 3rd party services.
- About Jenkins**: See this screen for basic information.

**Step 3** – Go to the Available tab and search for the plugin ‘Hudson global-build-stats plugin’ and choose to ‘install without restart’.

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [jenkins]" and the address bar says "localhost:8080/jenkins/pluginManager/available". The main area is titled "Plugin Manager" with tabs for "Updates", "Available" (which is selected), "Installed", and "Advanced". A search bar at the top right contains the text "global-build-stats". Below the tabs is a table with columns "Name", "Version", and "Description". One row in the table is highlighted for the "Hudson global-build-stats" plugin, which is version 1.3. The description states: "Global build stats plugin will allow to gather and display global build result statistics. It is a useful tool allowing to display global hudson build trend over time." There is a checkbox next to the plugin name, which is checked. At the bottom of the table are three buttons: "Install without restart" (in blue), "Download now and install after restart" (disabled and greyed out), and "Update information".

**Step 4** – The following screen shows up to confirm successful installation of the plugin. Restart the Jenkins instance.

The screenshot shows the Jenkins Update Center interface. The title bar says "Update Center [jenki]". The address bar shows "localhost:8080/jenkins/updateCenter/". The main header is "Jenkins" with a search bar and an "ENABLE AUTO REFRESH" link. On the left, there are links: "Back to Dashboard", "Manage Jenkins", and "Manage Plugins". The main content area is titled "Installing Plugins/Upgrades". It shows a "Preparation" section with a bulleted list: "Checking internet connectivity", "Checking update center connectivity", and "Success". Below that, it shows the "Hudson global-build-stats plugin" with a "Success" status indicator. At the bottom, there are two links: "Go back to the top page (you can start using the installed plugins right away)" and "Restart Jenkins when installation is complete and no jobs are running". The footer includes a "Help us localize this page" link, a timestamp "Page generated: Oct 24, 2015 4:01:04 PM", and "REST API Jenkins ver. 1.80.3".

To see the Global statistics, please follow the Step 5 through 8.

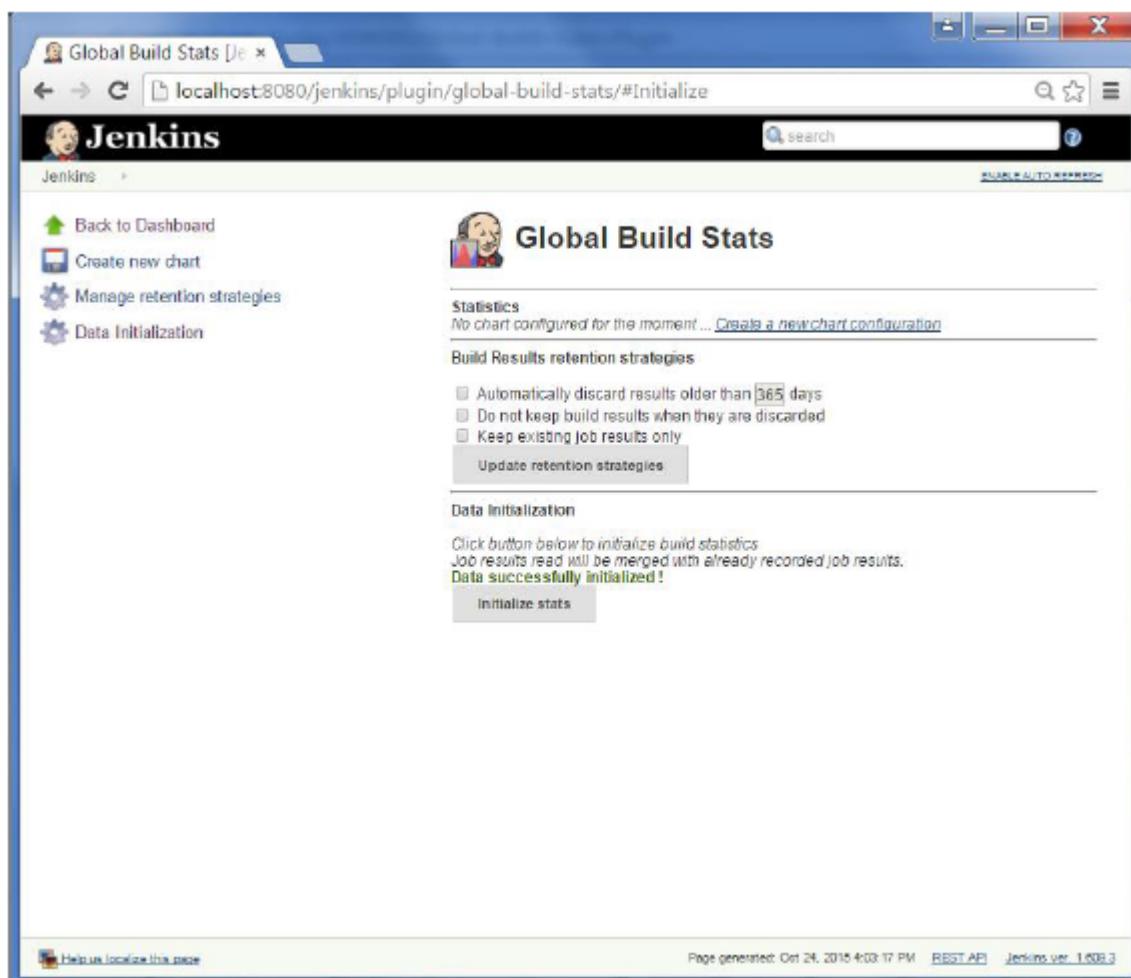
**Step 5** – Go to the Jenkins dashboard and click on Manage Jenkins. In the Manage Jenkins screen, scroll down and now you will now see an option called ‘Global Build Stats’. Click on this link.



**Step 6** – Click on the button 'Initialize stats'. What this does is that it gathers all the existing records for builds which have already been carried out and charts can be created based on these results.

The screenshot shows the Jenkins Global Build Stats plugin interface. At the top, there's a navigation bar with links to 'Back to Dashboard', 'Create new chart', 'Manage retention strategies', and 'Data Initialization'. The main title is 'Global Build Stats' with a small icon of a person holding a chart. Below the title, there's a section titled 'Statistics' which says 'No chart configured for the moment ... [Create a new chart configuration](#)'. Underneath is a section for 'Build Results retention strategies' with three checkboxes: 'Automatically discard results older than 30 days', 'Do not keep build results when they are discarded', and 'Keep existing job results only'. A 'Update retention strategies' button is below these checkboxes. Another section titled 'Data initialization' contains the text 'Click button below to initialize build statistics. Job results read will be merged with already recorded job results.' followed by a 'Initialize stats' button. At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 4:03:17 PM', 'REST API', and 'Jenkins ver. 1.809.3'.

**Step 7** – Once the data has been initialized, it's time to create a new chart. Click on the 'Create new chart' link.



**Step 8** – A pop-up will come to enter relevant information for the new chart details. Enter the following mandatory information

Title – Any title information, for this example is given as 'Demo'

Chart Width – 800

Chart Height – 600

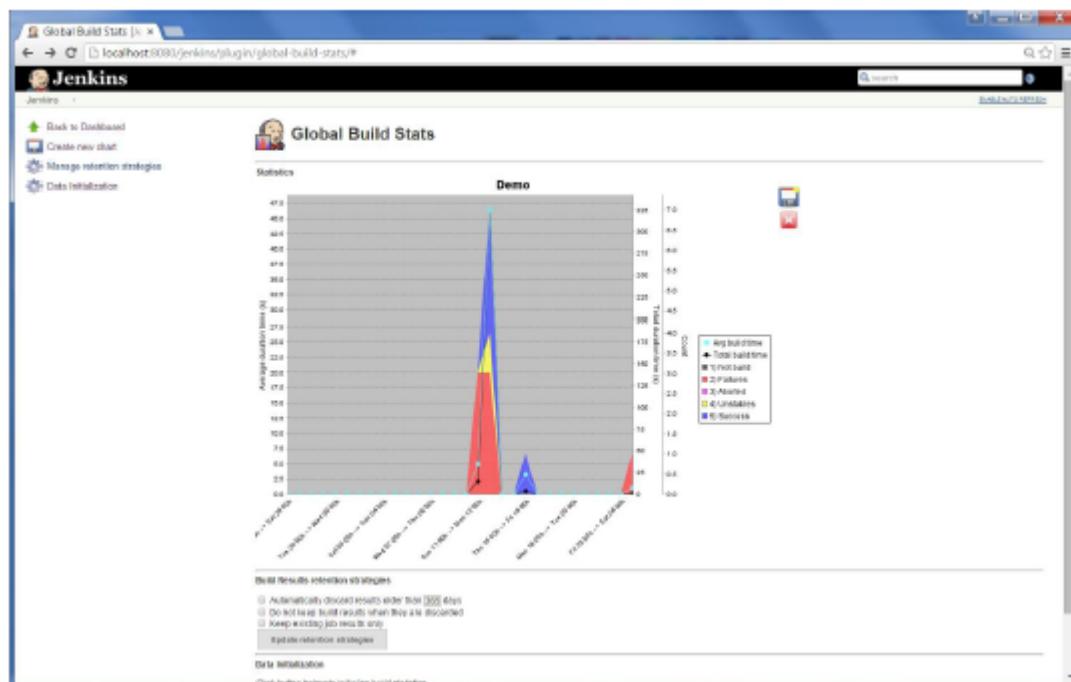
Chart time scale – Daily

Chart time length – 30 days

The rest of the information can remain as it is. Once the information is entered, click on Create New chart.

The screenshot shows the Jenkins Global Build Stats configuration interface. A modal window titled "Adding new chart" is open, prompting for chart details like title ("Demo"), dimensions (width 800, height 600), time scale (daily), and duration (30 days). It also includes filtering options for jobs, nodes, and launchers, and a list of build statuses to track (Success, Failure, Unstable, Aborted, Not Build). Below the modal are buttons for "Overview", "Create new chart", and "Cancel". At the bottom of the main page, there's a footer with localization help and version information.

You will now see the chart which displays the trends of the builds over time.



If you click on any section within the chart, it will give you a drill down of the details of the job and their builds.

The screenshot shows the Jenkins Global Build Search interface. At the top, there are navigation links: 'Back to Dashboard' and 'Back to Global Build Stats'. Below that is a search bar with placeholder text 'Search...'. Under the search bar, there are several filter options: 'Job filtering' (radio buttons for 'All jobs', 'Job name > regex', 'Name filtering', 'ALL Nodes', 'Master only', 'No Node Name Filter'), 'Launcher filtering' (radio buttons for 'ALL Users', 'System only', 'Username regex'), and 'Statuses when interacting' (checkboxes for 'Success', 'Failure', 'Unstable', 'Aborted', 'Not Started'). A 'Search' button is located below the filters. The main area is titled 'Search results' and contains a table with the following data:

Status	Job name	#	Date	Duration	Master name	Launcher & Job
Failure	HelloWorld	1	Oct 11, 2015 19:04:57 PM	5.6 sec	master	SHUTDOWN
Failure	HelloWorld	2	Oct 11, 2015 19:51:37 PM	4.6 sec	master	SHUTDOWN
Failure	HelloWorld	3	Oct 11, 2015 19:48:21 PM	4.2 sec	master	SHUTDOWN

At the bottom right of the table, it says 'Page generated: Oct 24, 2018 4:00:40 PM / 1007 API / Jenkins ver: 1.600.3'.

## Jenkins - Server Maintenance

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.

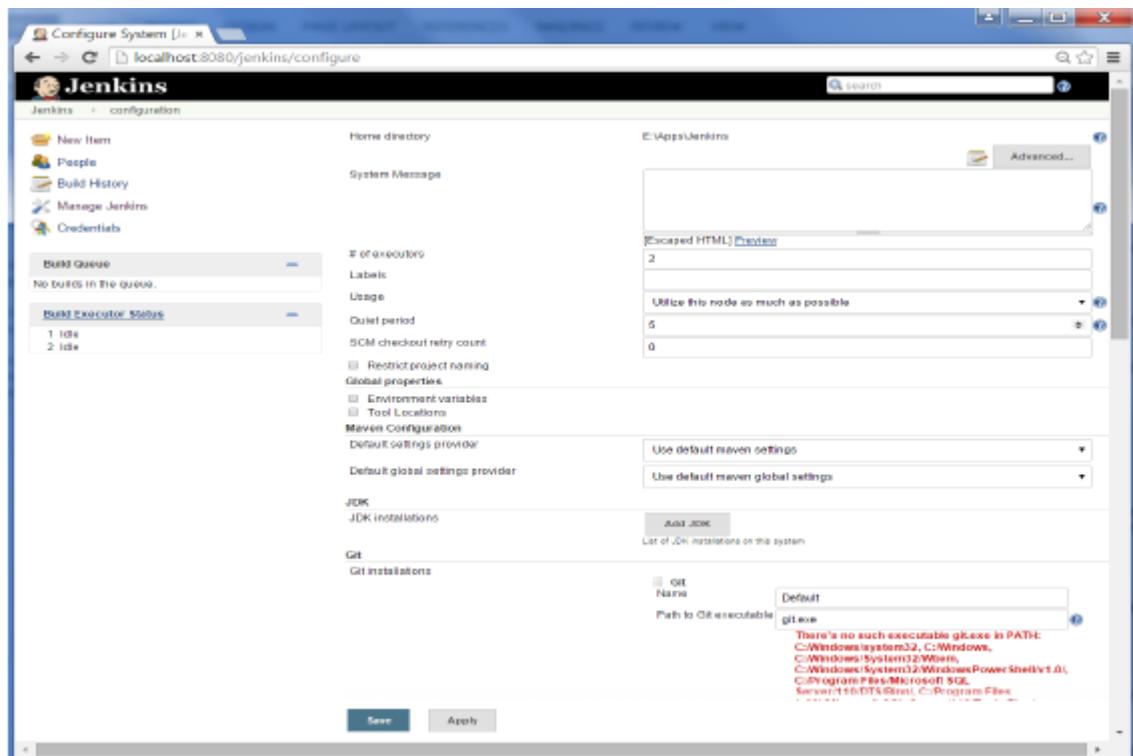
**http://localhost:8080/jenkins/exit** – shutdown jenkins

**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.

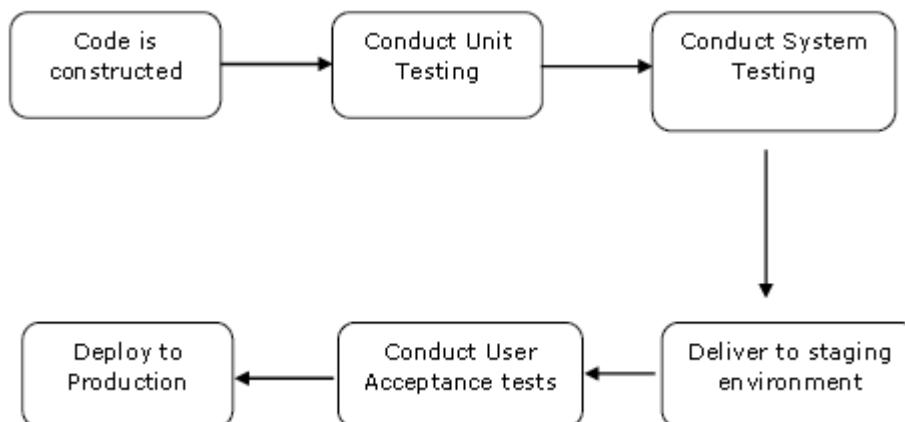


Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined and doing continuous builds, always ensure that Jenkins is setup on a drive that has enough hard disk space. If you hard disk runs out of space, then all builds on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

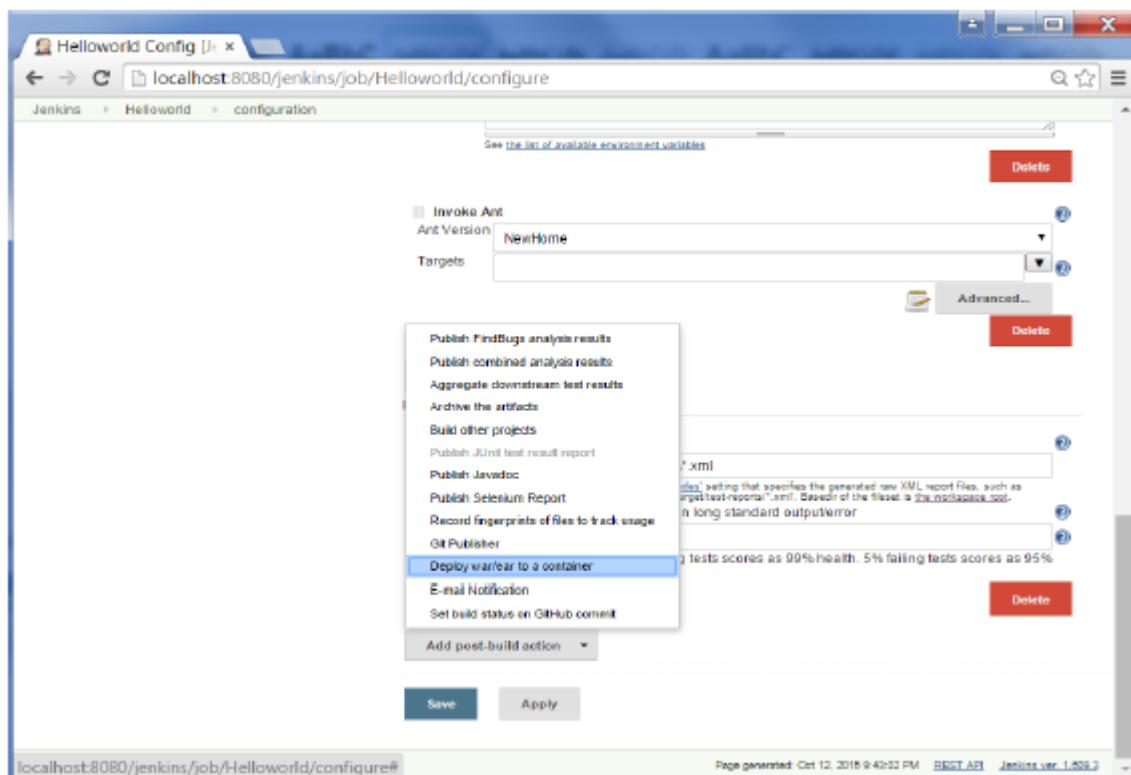
Another best practice is to write cron jobs or maintenance tasks that can carry out clean-up operations to avoid the disk where Jenkins is setup from becoming full.

## Jenkins - Continuous Deployment

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.



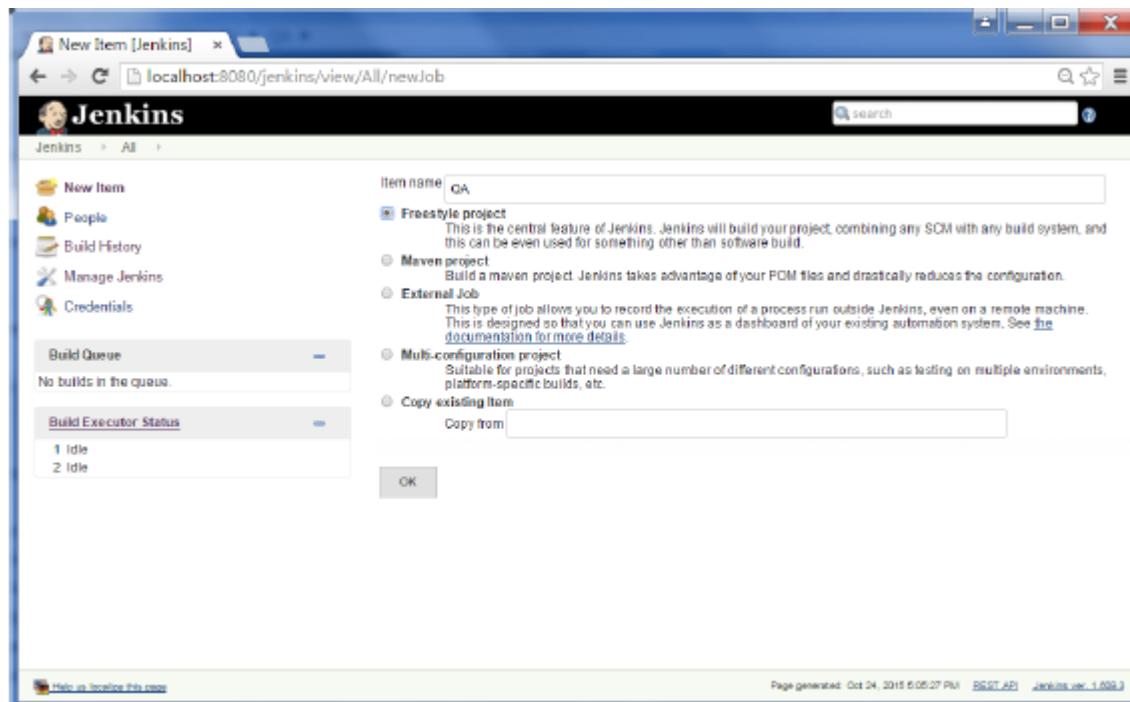
The main part of Continuous deployment is to ensure that the entire process which is shown above is automated. Jenkins achieves all of this via various plugins, one of them being the "Deploy to container Plugin" which was seen in the earlier lessons.



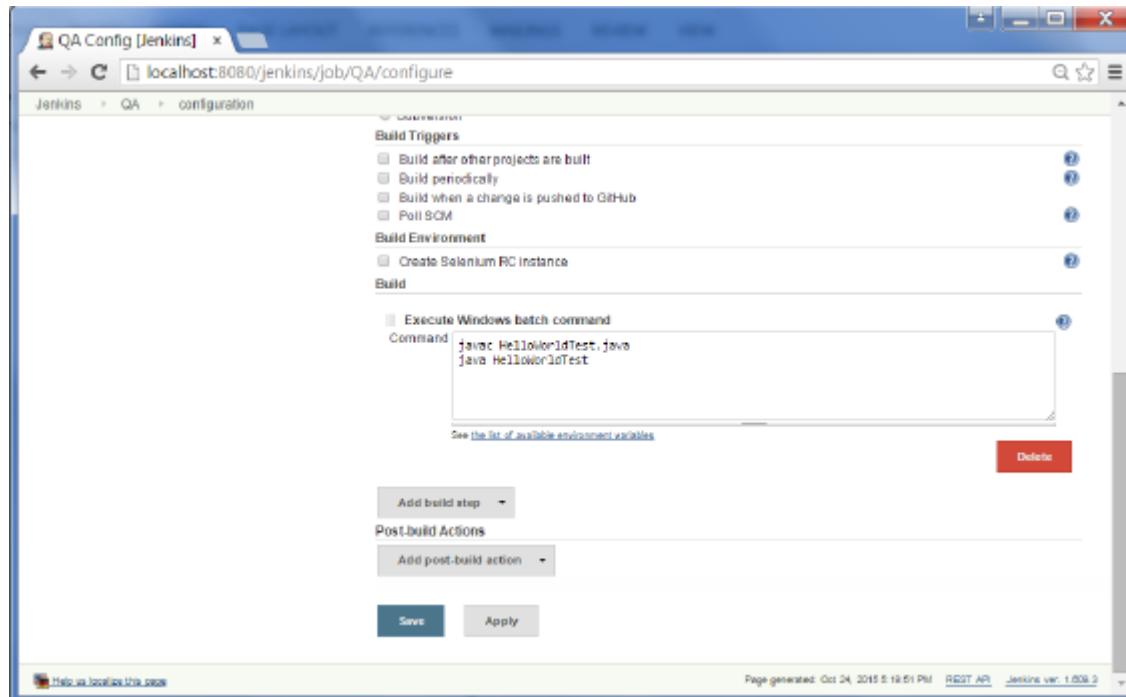
There are plugins available which can actually give you a graphical representation of the Continuous deployment process. But first lets create another project in Jenkins, so that we can see best how this works.

Let's create a simple project which emulates the QA stage, and does a test of the Helloworld application.

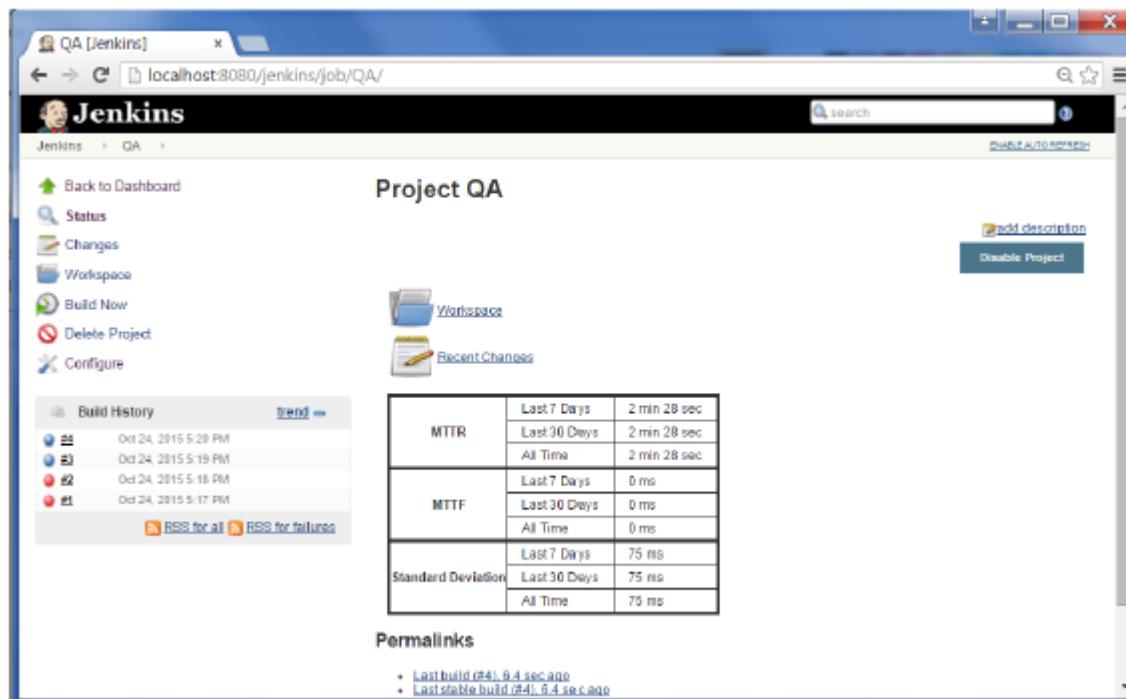
**Step 1** – Go to the Jenkins dashboard and click on New Item. Choose a 'Freestyle project' and enter the project name as 'QA'. Click on the Ok button to create the project.



**Step 2** – In this example, we are keeping it simple and just using this project to execute a test program for the Helloworld application.



So our project QA is now setup. You can do a build to see if it builds properly.



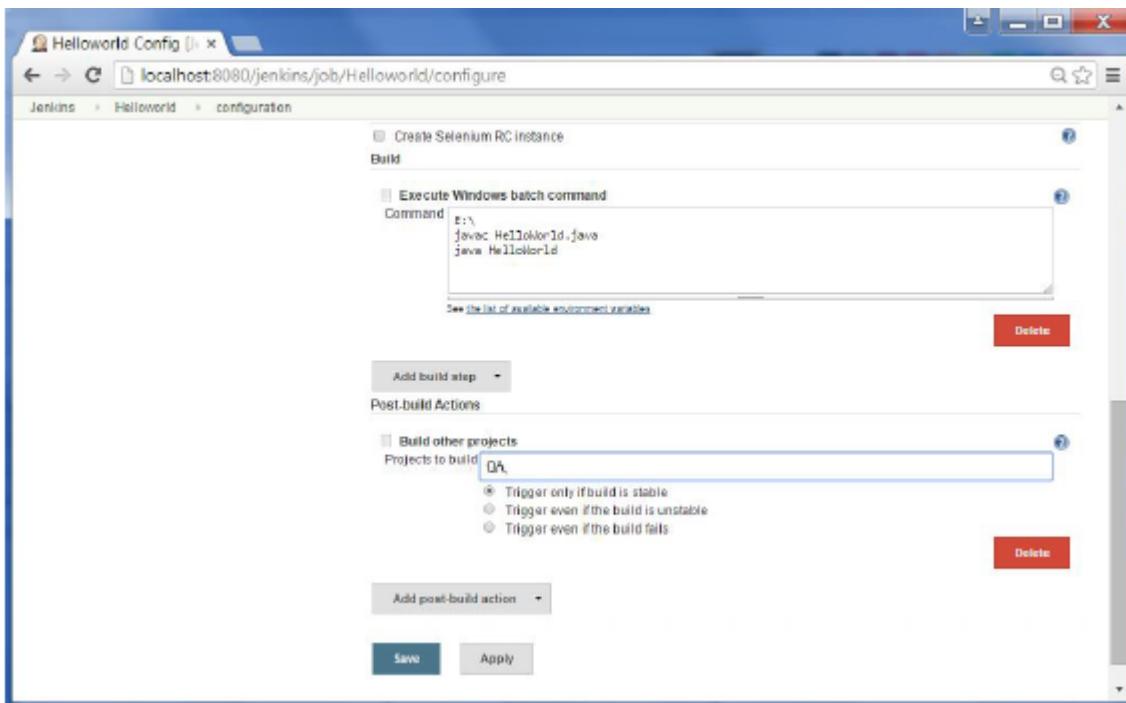
**Step 3** – Now go to your Helloworld project and click on the Configure option

The screenshot shows the Jenkins dashboard at [localhost:8080/jenkins/](http://localhost:8080/jenkins/). On the left sidebar, there are links for New Item, People, Build History, Manage Jenkins, and Credentials. Below these are sections for Build Queue (No builds in the queue) and Build Executor Status (1 idle). The main area displays a table of jobs, with 'Helloworld' selected. A context menu is open over the 'Helloworld' row, with 'Configure' highlighted.

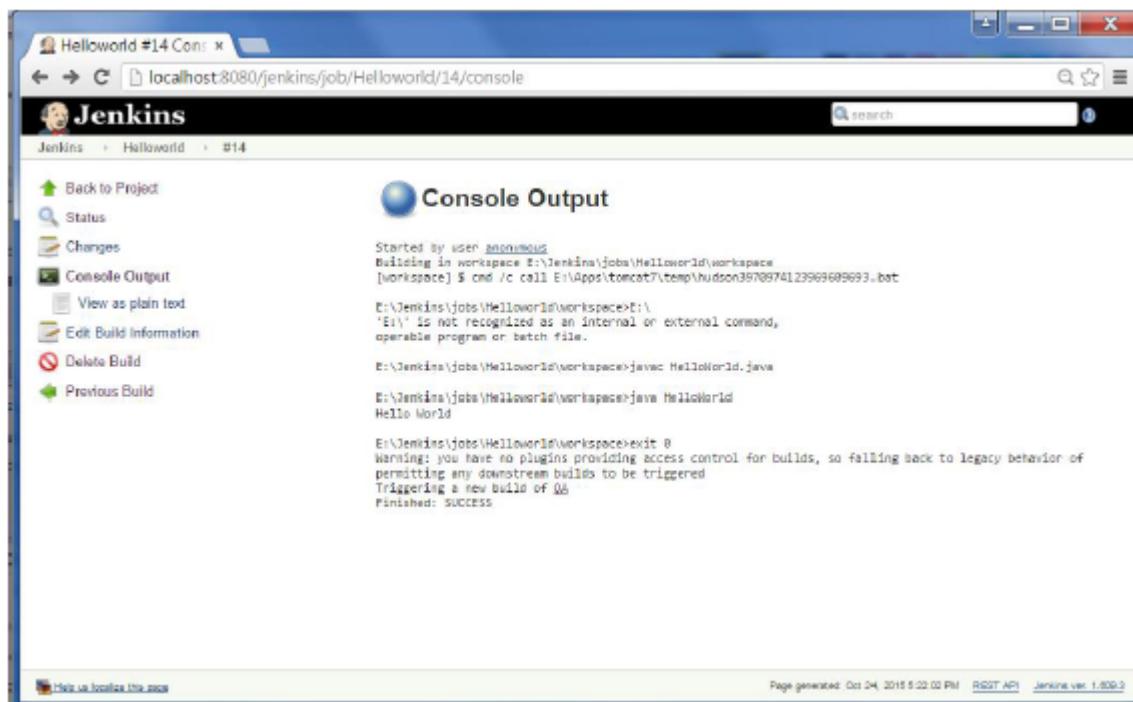
**Step 4** – In the project configuration, choose the ‘Add post-build action’ and choose ‘Build other projects’

The screenshot shows the configuration page for the 'Helloworld' job at [localhost:8080/jenkins/job/Helloworld/configure](http://localhost:8080/jenkins/job/Helloworld/configure). The 'Build other projects' option is selected from the 'Add post-build action' dropdown menu. The right side of the screen shows configuration details for the 'Build other projects' action, including a workspace path and threshold settings. At the bottom, there are 'Save' and 'Apply' buttons.

**Step 5** – In the ‘Project to build’ section, enter QA as the project name to build. You can leave the option as default of ‘Trigger only if build is stable’. Click on the Save button.



**Step 6** – Build the Helloworld project. Now if you see the Console output, you will also see that after the Helloworld project is successfully built, the build of the QA project will also happen.



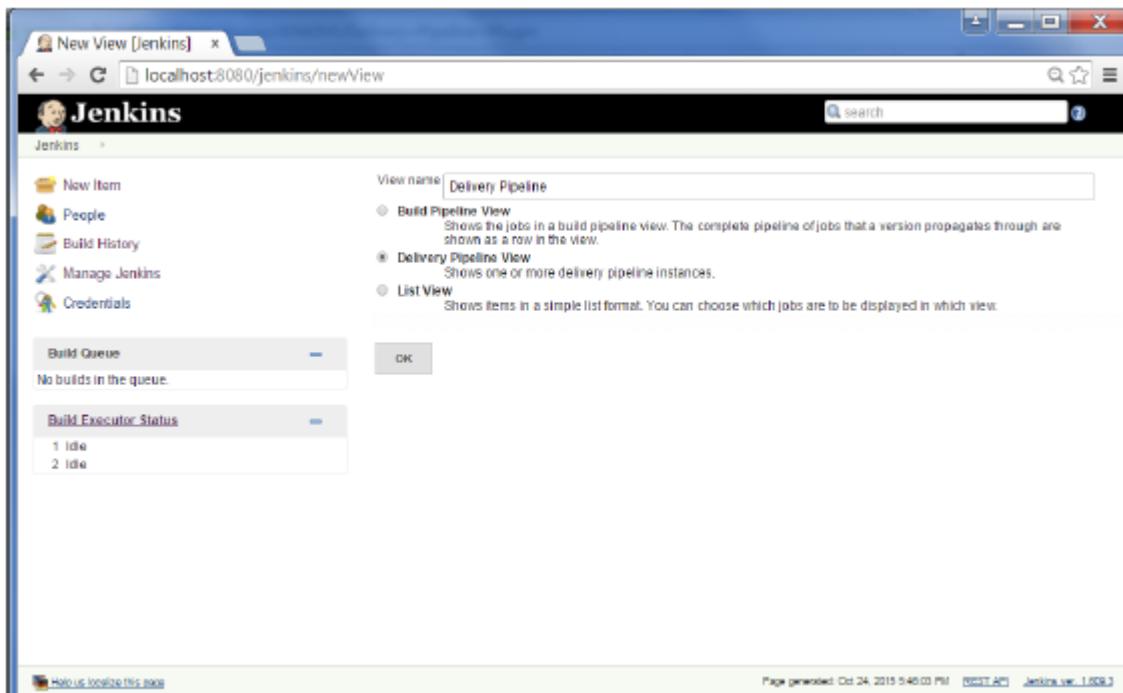
**Step 7** – Let now install the Delivery pipeline plugin. Go to Manage Jenkins → Manage Plugins. In the available tab, search for ‘Delivery Pipeline Plugin’. Click On Install without Restart. Once done, restart the Jenkins instance.

The screenshot shows the Jenkins Plugin Manager interface. The 'available' tab is selected. A checkbox for the 'Delivery Pipeline Plugin' is checked. Other visible plugins include 'ontrack Jenkins plug-in', 'Fail The Build Plugin', 'Runscope plugin', 'Build Graph View Plugin', 'Deployment Pipeline', 'CloudBees Docker Hub Notifications', 'Seed Jenkins plug-in', and 'Delivery Pipeline Plugin'. Buttons at the bottom include 'Install without restart' and 'Download now and install after restart'.

**Step 8** – To see the Delivery pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the 'All' Tab.

The screenshot shows the Jenkins Dashboard. The 'All' tab is selected. It displays two build items: 'HelloWorld' and 'DB'. The 'HelloWorld' item has a status of '25 min - #14', 'Last Success' at '25 min - #14', 'Last Failure' at '1 hr 40 min - #12', and 'Last Duration' at '1.4 sec'. The 'DB' item has a status of '25 min - #5', 'Last Success' at '28 min - #2', and 'Last Duration' at '1.4 sec'. A legend at the bottom right indicates 'R88 for all', 'R88 for failures', and 'R88 for just latest builds'.

**Step 9** – Enter any name for the View name and choose the option 'Delivery Pipeline View'.



**Step 10** – In the next screen, you can leave the default options. One can change the following settings –

Ensure the option 'Show static analysis results' is checked.

Ensure the option 'Show total build time' is checked.

For the Initial job – Enter the Helloworld project as the first job which should build.

Enter any name for the Pipeline

Click the OK button.

The screenshot shows the Jenkins 'Edit View [Jenkins]' configuration page for 'Delivery Pipeline'. The left sidebar includes links for New Item, People, Build History, Edit View, Delete View, View Fullscreen, Manage Jenkins, and Credentials. The main panel has sections for 'View settings' (Number of pipeline instances per pipeline: 3, Display aggregated pipeline for each pipeline: checked, Number of columns: 1, Sorting: None, Update interval: 2), 'Build Queue' (No builds in the queue), 'Build Executor Status' (1 Idle, 2 Idle), and 'Pipelines' (Components). Under Pipelines, there is a table with one row: Name: FirstJob, Initial Job: HelloWorld, Final Job (optional): (empty dropdown). Buttons for OK and Apply are at the bottom.

You will now see a great view of the entire delivery pipeline and you will be able to see the status of each project in the entire pipeline.

The screenshot shows the Jenkins interface for a 'Delivery Pipeline'. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', etc. The main area is titled 'Firstjob' and shows three build steps:

- #14 triggered by user anonymous started 29 minutes ago  
Total build time: 2 sec  
Helloworld → QA
- #13 triggered by user anonymous started an hour ago  
Total build time: 3 sec  
Helloworld → QA
- #12 triggered by user anonymous started 2 hours ago  
Total build time: 1 sec  
Helloworld → QA

Each step is represented by a box with a status bar at the bottom. The first two steps are green, while the third is red.

Another famous plugin is the **build pipeline plugin**. Let's take a look at this.

**Step 1** – Go to Manage Jenkins → Manage Plugin's. In the available tab, search for 'Build Pipeline Plugin'. Click On Install without Restart. Once done, restart the Jenkins instance.

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Jenki...]" and the address bar says "localhost:8080/jenkins/pluginManager/available". The main content area has a search bar with "search" and a filter bar with "Filter: Build pipeline". Below is a table with columns: Updates, Available, Installed, Advanced, Install, Name, and Version.

Updates	Available	Installed	Advanced	Install	Name	Version
	<input checked="" type="checkbox"/> <a href="#">Build Pipeline Plugin</a>				This plugin provides a _Build Pipeline View_ of upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins.	1.4.8
	<input type="checkbox"/> <a href="#">Fail The Build Plugin</a>				Set or change the build result to test job configurations - notifiers, publishers, promotions, build pipelines, etc.	1.0
	<input type="checkbox"/> <a href="#">Runscope plugin</a>				This plugin allows you to add a <a href="#">Runscope API</a> test as a build step into your Jenkins build pipeline. The plugin will trigger a specific API test (via <a href="#">Trigger URL</a> ) and wait for the test to complete. If the test passes, the build steps will continue. However, if it fails, the build will be marked as failed.	1.44
	<input type="checkbox"/> <a href="#">Build Graph View Plugin</a>				Shows a graph of builds that relates together (aka "build pipeline").	1.1.1
	<input type="checkbox"/> <a href="#">Delivery Pipeline Plugin</a>				Visualisation of Delivery/Build Pipelines, renders pipelines based on upstream/downstream jobs. Full screen view for information radiators.	0.9.7

At the bottom are three buttons: "Install without restart", "Download now and install after restart", and "Update info".

**Step 2** – To see the Build pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the 'All' Tab.

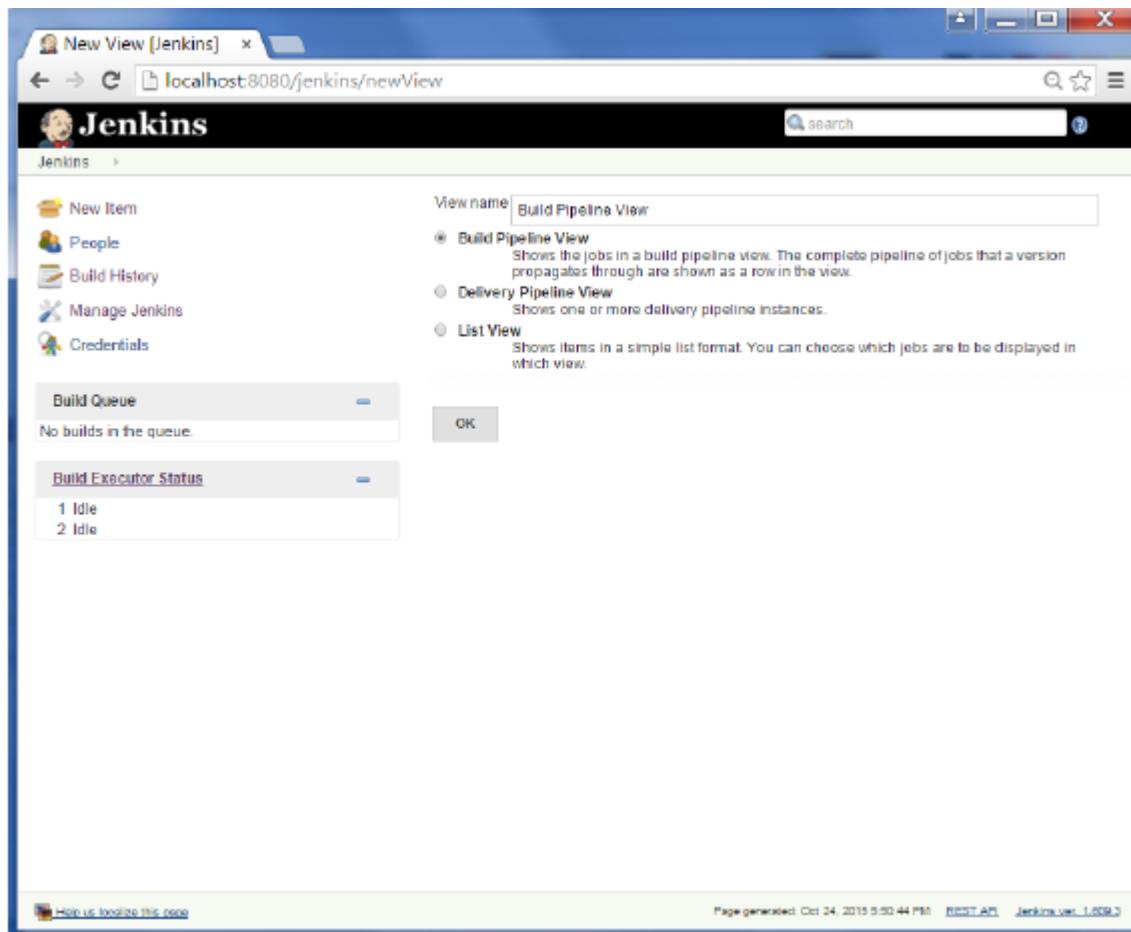
The screenshot shows the Jenkins Dashboard. The title bar says "Dashboard [Jenkins]" and the address bar says "localhost:8080/jenkins/". The main content area has a sidebar with links: New Item, People, Build History, Manage Jenkins, and Credentials. Below is a table with columns: All, W, Name, Last Success, Last Failure, and Last Duration. At the bottom right are RSS feed links: RSS for all, RSS for failures, and RSS for just latest builds.

All	W	Name	Last Success	Last Failure	Last Duration
5		HelloWorld	25 min - #14	1 hr 49 min - #12	1.4 sec
		QA	25 min - #5	28 min - #2	1.4 sec

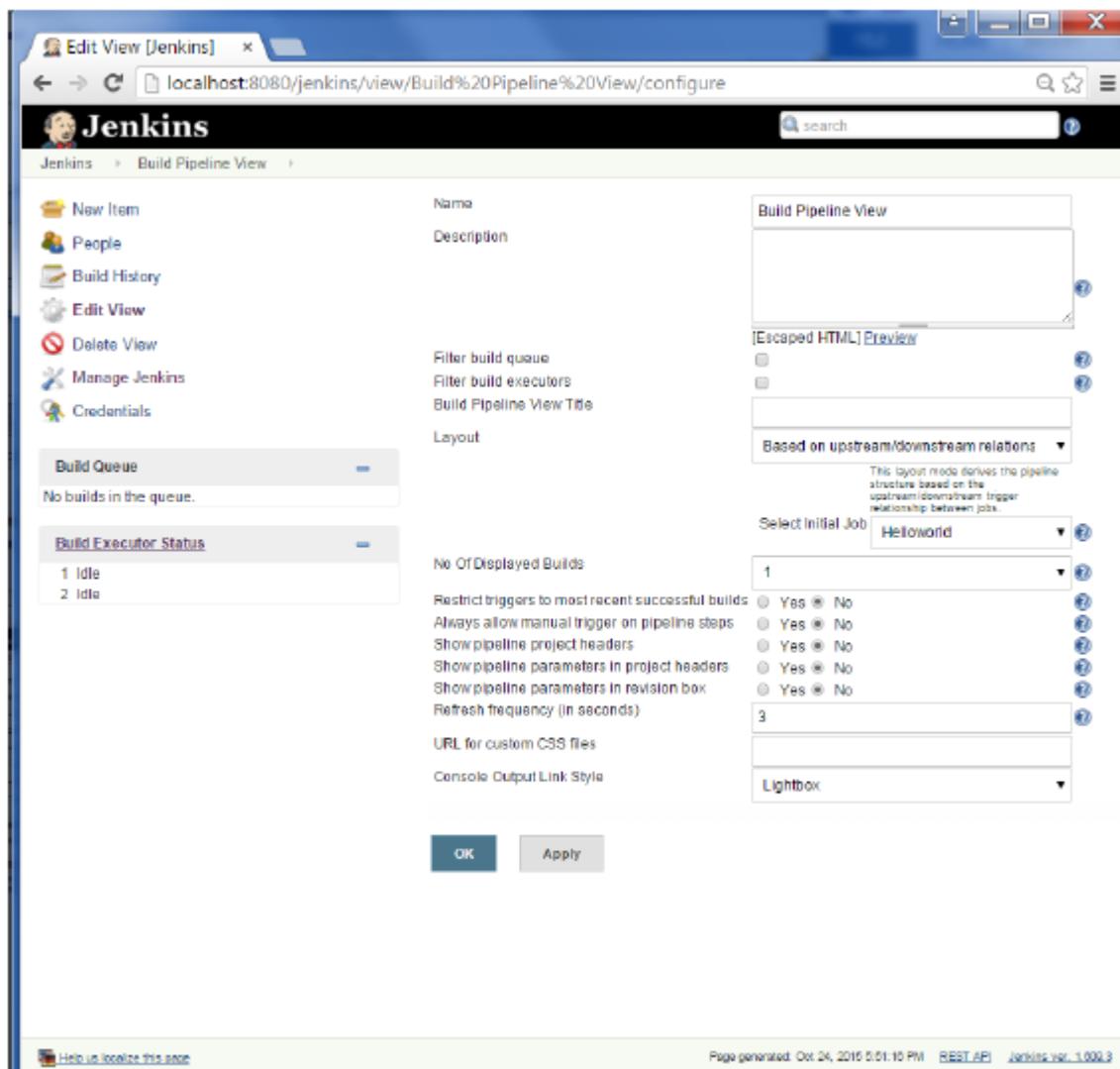
Legend: RSS for all, RSS for failures, RSS for just latest builds

At the bottom are links: Help us localize this page, Page generated: Oct 24, 2015 4:48:09 PM, REST API, Jenkins ver. 1.809.3

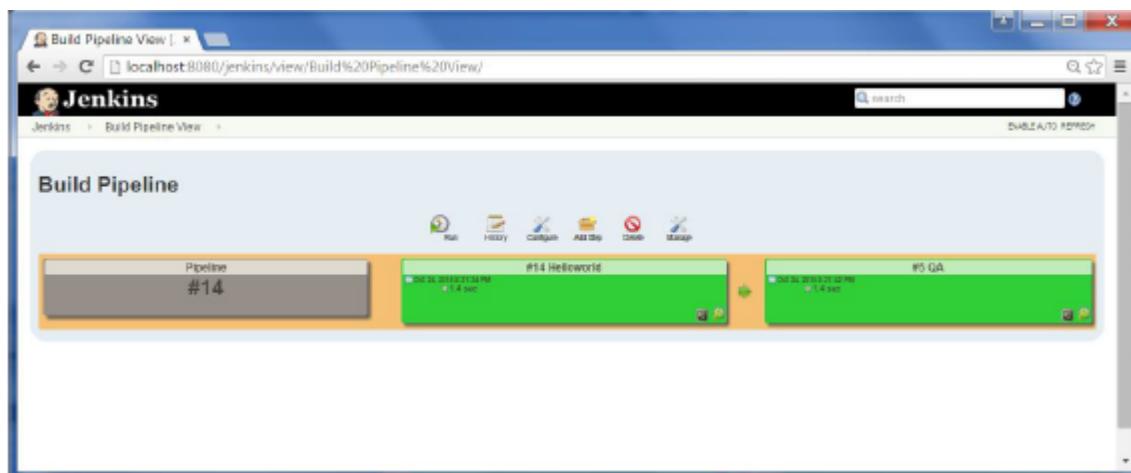
**Step 3** – Enter any name for the View name and choose the option 'Build Pipeline View'.



**Step 4** – Accept the default settings, just in the Selected Initial job, ensure to enter the name of the Helloworld project. Click on the Ok button.



You will now see a great view of the entire delivery pipeline and you will be able to see the status of each project in the entire pipeline.



## Jenkins - Managing Plugins

To get the list of all plugins available within Jenkins, one can visit the link – <https://wiki.jenkins-ci.org/display/JENKINS/Plugins>

The screenshot shows the Jenkins Plugins page. On the left, there's a sidebar with links to Jenkins Home, Mailing lists, Source code, Bugtracker, Security Advisories, Events, Donation, Commercial Support, and Wiki Site Map. Below that is a section for Documents with links to Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, and Servlet Container Notes. The main content area has a heading 'Plugins' with a sub-image of a person's head. It says 'Added by Kohsuke Kawaguchi, last edited by Sam Alsam on Jul 25, 2015 (view change)'. There are two main sections: 'How to install plugins' (with sub-links for interface usage, newest version, specific version, and by hand) and 'Getting notified of plugin releases'. Below these are sections for 'Developers', 'Plugins by topic', and a large list of individual plugins categorized under 'Source code management', 'Build triggers', 'Build tools', 'Build wrappers', 'Build notifiers', 'Slave launchers and controllers', 'Build reports', 'Artifact updaters', 'Other post-build actions', 'External site/tool integrations', 'UI plugins', 'List View column plugins', 'Page decorators', 'Authentication and user management', 'Cluster management and distributed build', 'CLI extensions', 'Maven', 'Parameters', 'iOS development', '.NET development', 'Android development', and 'Ruby development'.

We've already seen many instances for installing plugins, let's look at some other maintenance tasks with regards to plugins

## Uninstalling Plugins

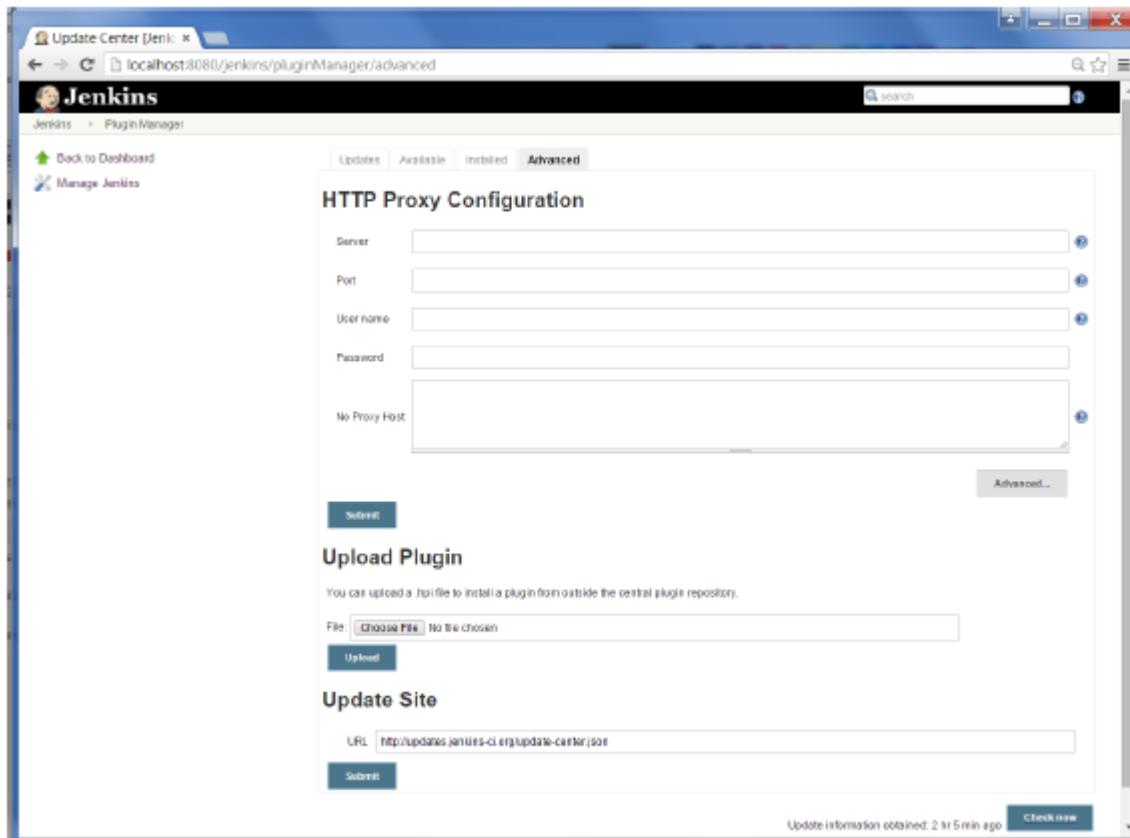
To uninstall a plugin, Go to Manage Jenkins → Manage plugins. Click on the Installed tab. Some of the plugins will have the Uninstall option. You can click these buttons to uninstall the plugins. Ensure to restart your Jenkins instance after the uninstallation.

The screenshot shows the Jenkins Plugin Manager. At the top, there are tabs for 'Updates', 'Available', 'Installed' (which is selected), and 'Advanced'. The 'Installed' tab displays a table of currently installed Jenkins plugins. The columns are 'Enabled', 'Name', 'Version', 'Previously installed version', 'Pinned', and 'Uninstall'. The plugins listed include:

Enabled	Name	Version	Previously installed version	Pinned	Uninstall
✓	<a href="#">Ant Plugin</a>	1.2			<a href="#">Uninstall</a>
✓	<a href="#">Build History Metrics Plugin</a>	1.2			<a href="#">Uninstall</a>
✓	<a href="#">Build Pipeline Plugin</a>	1.4.8			<a href="#">Uninstall</a>
✓	<a href="#">Credentials Plugin</a>	1.22	<a href="#">Downgrade to 1.18</a>	<a href="#">Unpin</a>	<a href="#">Uninstall</a>
✓	<a href="#">CVS Plugin</a>	2.13			<a href="#">Uninstall</a>
✓	<a href="#">Delivery Pipeline Plugin</a>	0.8.7			<a href="#">Uninstall</a>
✓	<a href="#">Deploy to container Plugin</a>	1.10			<a href="#">Uninstall</a>
✓	<a href="#">External Monitor Job Type Plugin</a>	1.8			<a href="#">Uninstall</a>
✓	<a href="#">External Notification Plugin</a>	1.1			<a href="#">Uninstall</a>
✓	<a href="#">FindBugs Plugin</a>	4.62			<a href="#">Uninstall</a>

# Installing another Version of a Plugin

Sometimes it may be required to install an older version of a plugin, in such a case, you can download the plugin from the relevant plugin page on the Jenkins web site. You can then use the **Upload** option to upload the plugin manually.

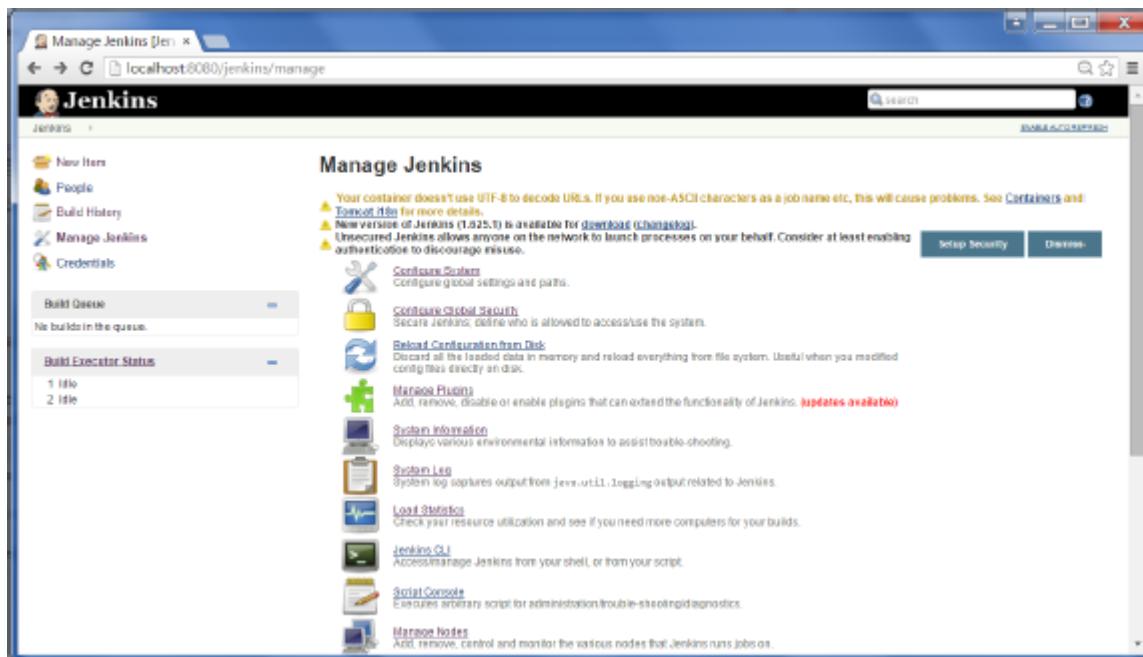


## Jenkins - Security

In Jenkins you have the ability to setup users and their relevant permissions on the Jenkins instance. By default you will not want everyone to be able to define jobs or other administrative tasks in Jenkins. So Jenkins has the ability to have a security configuration in place.

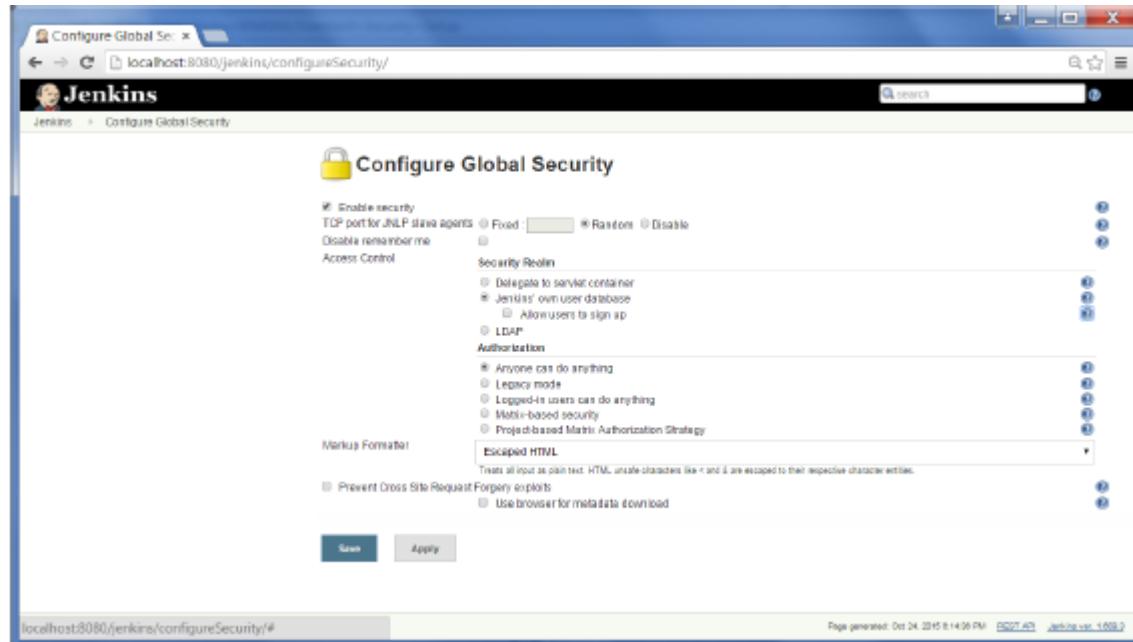
To configure Security in Jenkins, follow the steps given below.

**Step 1** – Click on Manage Jenkins and choose the 'Configure Global Security' option.



**Step 2** – Click on Enable Security option. As an example, let's assume that we want Jenkins to maintain it's own database of users, so in the Security Realm, choose the option of 'Jenkins' own user database'.

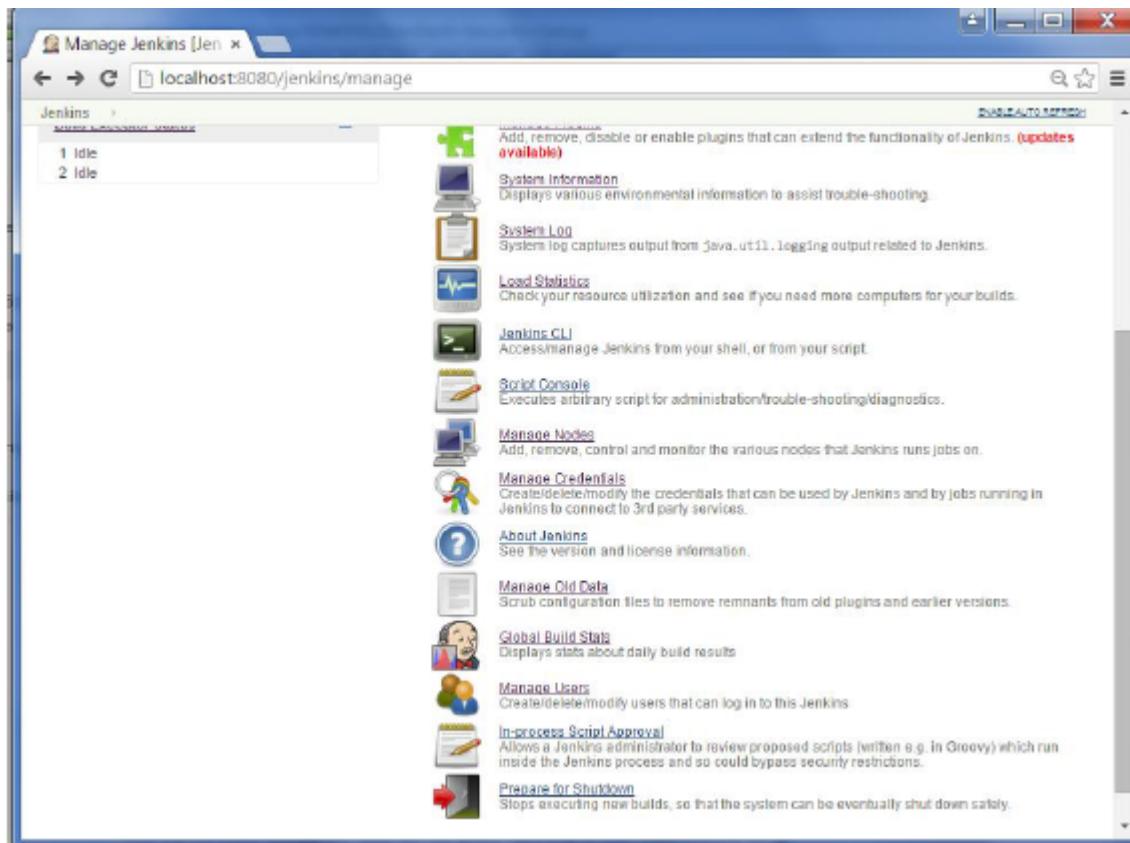
By default you would want a central administrator to define users in the system, hence ensure the 'Allow users to sign up' option is unselected. You can leave the rest as it is for now and click the Save button.



**Step 3** – You will be prompted to add your first user. As an example, we are setting up an admin users for the system.

The screenshot shows the Jenkins 'Sign up' page. At the top left, there's a 'Back to Dashboard' link, a 'Manage Jenkins' link, and a 'Create User' link. On the right, there's a search bar and a 'log in' link. The main title is 'Sign up'. Below it, there are four input fields: 'Username' (admin), 'Password' (\*\*\*\*), 'Confirm password' (\*\*\*\*), and 'E-mail address' (al@gmail.com). A blue 'Sign up' button is located at the bottom of these fields. At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 6:16:01 PM', 'REST API', and 'Jenkins ver. 1.608.3'.

**Step 4** – It's now time to setup your users in the system. Now when you go to Manage Jenkins, and scroll down, you will see a 'Manage Users' option. Click this option.



**Step 5** – Just like you defined your admin user, start creating other users for the system. As an example, we are just creating another user called ‘user’.

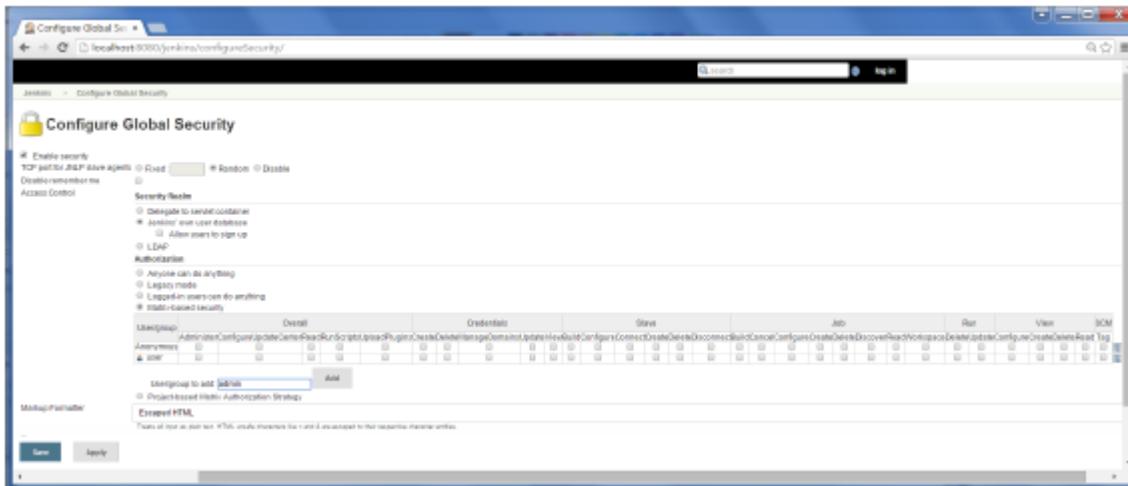
The screenshot shows the Jenkins Sign up page. The URL in the address bar is `localhost:8080/jenkins/securityRealm/addUser`. The page has a header 'Sign up' and a 'Back to Dashboard' link. On the left, there are links for 'Manage Jenkins' and 'Create User'. The form fields are:

Username:	<input type="text" value="user"/>
Password:	<input type="password" value="..."/>
Confirm password:	<input type="password" value="..."/>
Full name:	<input type="text" value="User"/>
E-mail address:	<input type="text" value="user@gmail.com"/>

A blue 'Sign up' button is at the bottom of the form. At the bottom of the page, there's a note 'Help us localize this page' and footer text 'Page generated: Oct 24, 2018 6:20:59 PM REST API Jenkins ver. 1.639.3'.

**Step 6** – Now it’s time to setup your authorizations, basically who has access to what. Go to Manage Jenkins → Configure Global Security.

Now in the Authorization section, click on 'Matrix based security'



**Step 7** – If you don't see the user in the user group list, enter the user name and add it to the list. Then give the appropriate permissions to the user.

Click on the Save button once you have defined the relevant authorizations.

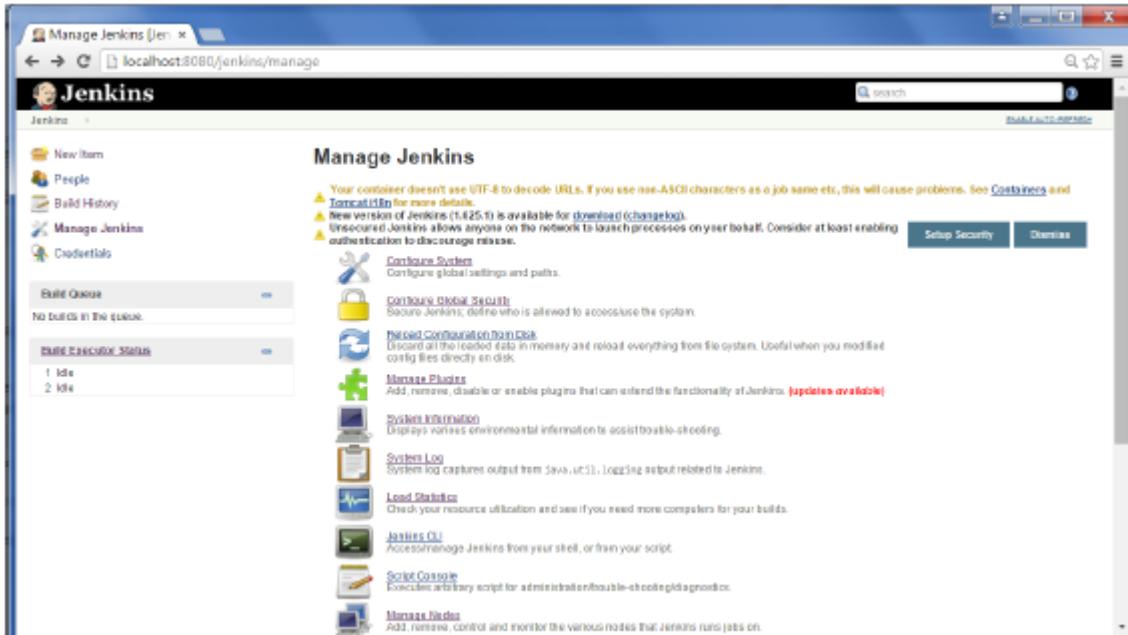
Your Jenkins security is now setup.

**Note** – For Windows AD authentication, one has to add the Active Directory plugin to Jenkins.

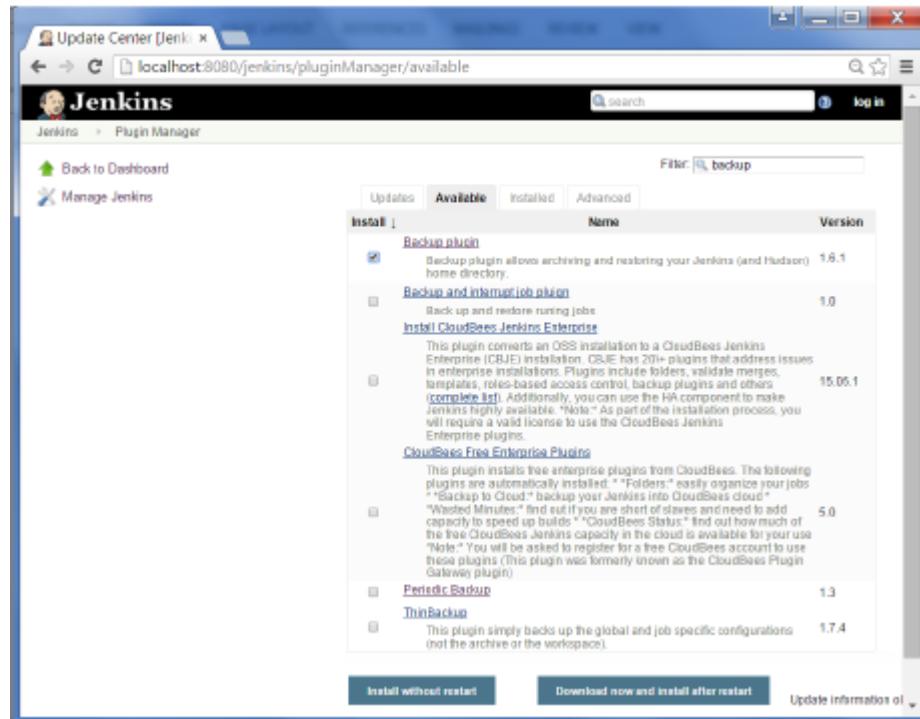
## Jenkins - Backup Plugin

Jenkins has a backup plugin which can be used to backup critical configuration settings related to Jenkins. Follow the steps given below to have a backup in place.

**Step 1** – Click on Manage Jenkins and choose the 'Manage Plugins' option.



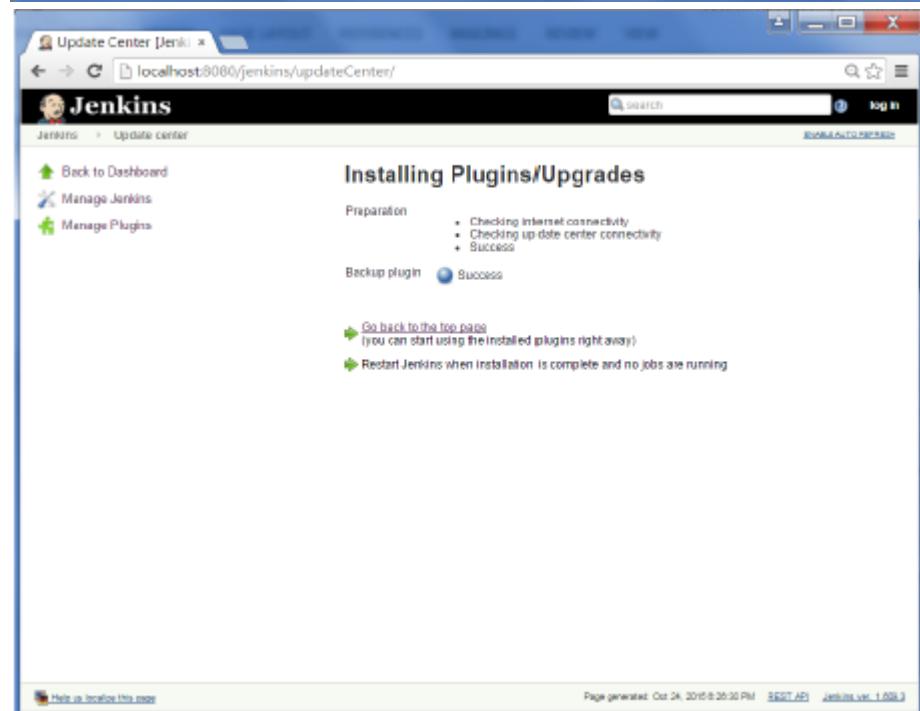
**Step 2** – In the available tab, search for 'Backup Plugin'. Click On Install without Restart. Once done, restart the Jenkins instance



The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected. A search bar at the top right contains the text 'backup'. Below it, a table lists available plugins:

Install	Name	Version
<input checked="" type="checkbox"/>	Backup plugin	1.6.1
<input type="checkbox"/>	Backup and interrupt job plugin	1.0
<b>Install CloudBees Jenkins Enterprise</b>		
<input type="checkbox"/>	This plugin converts an OSB installation to a CloudBees Jenkins Enterprise (CBJE) installation. CBJE has 20+ plugins that address issues in enterprise installations. Plugins include folders, validate merges, templates, roles-based access control, backup plugins and others.	15.05.1
<b>CloudBees Free Enterprise Plugins</b>		
<input type="checkbox"/>	This plugin installs free enterprise plugins from CloudBees. The following plugins are automatically installed: "Folders," easily organize your jobs "Backup to Cloud," backup your Jenkins into CloudBees cloud "Wasted Minutes," find out if you are short of slaves and need to add capacity or speed up builds "CloudBees Status," find out how much of the available Jenkins capacity in the cloud is available for your use "Note," You will be asked to register for a free CloudBees account to use these plugins (This plugin was formerly known as the CloudBees Plugin Gateway plugin)	5.0
<input type="checkbox"/>	Periodic Backup	1.3
<input type="checkbox"/>	ThinBackup	1.7.4

At the bottom are three buttons: 'Install without restart', 'Download now and install after restart', and 'Update information'.

The screenshot shows the Jenkins Update Center page. The title is 'Installing Plugins/Upgrades'. It displays the following information:

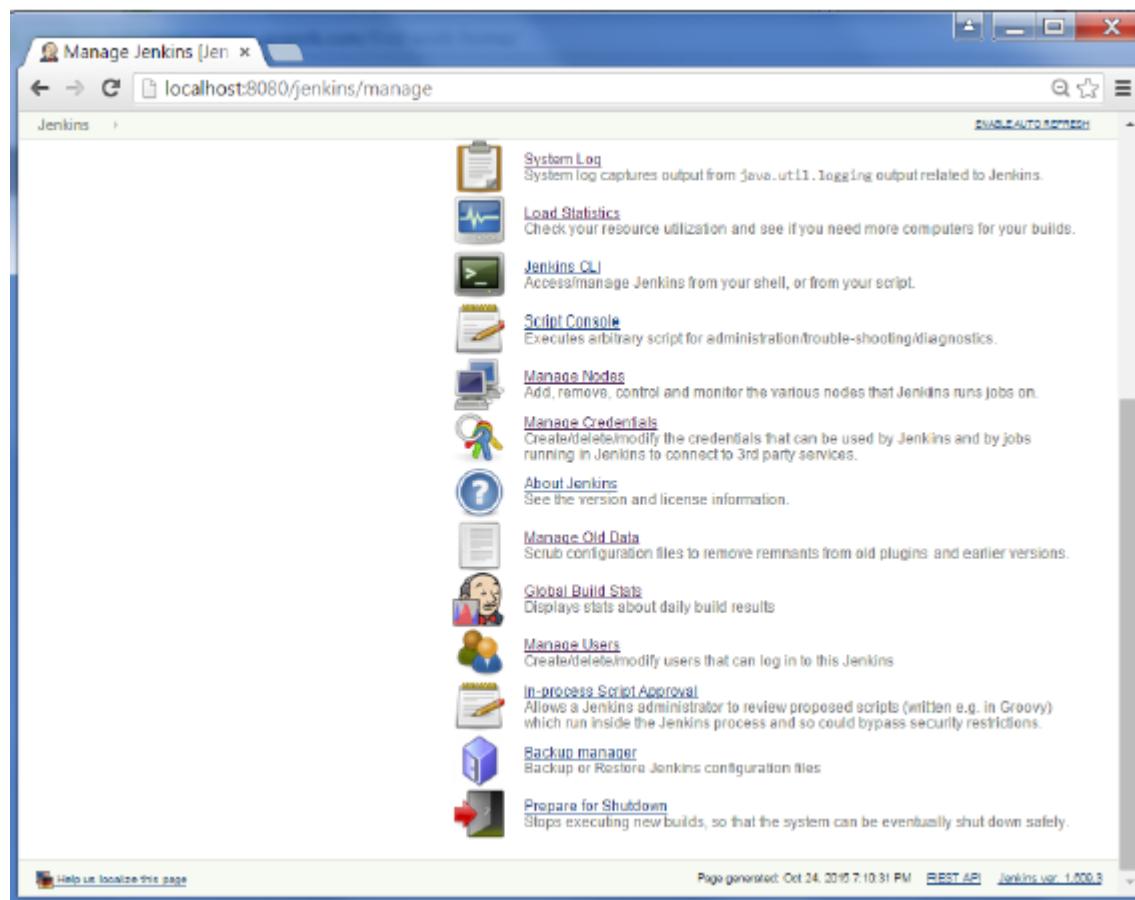
- Preparation**
  - Checking internet connectivity
  - Checking update center connectivity
  - Success
- Backup plugin**   Success

Below this, there are two green checkmark icons with instructions:

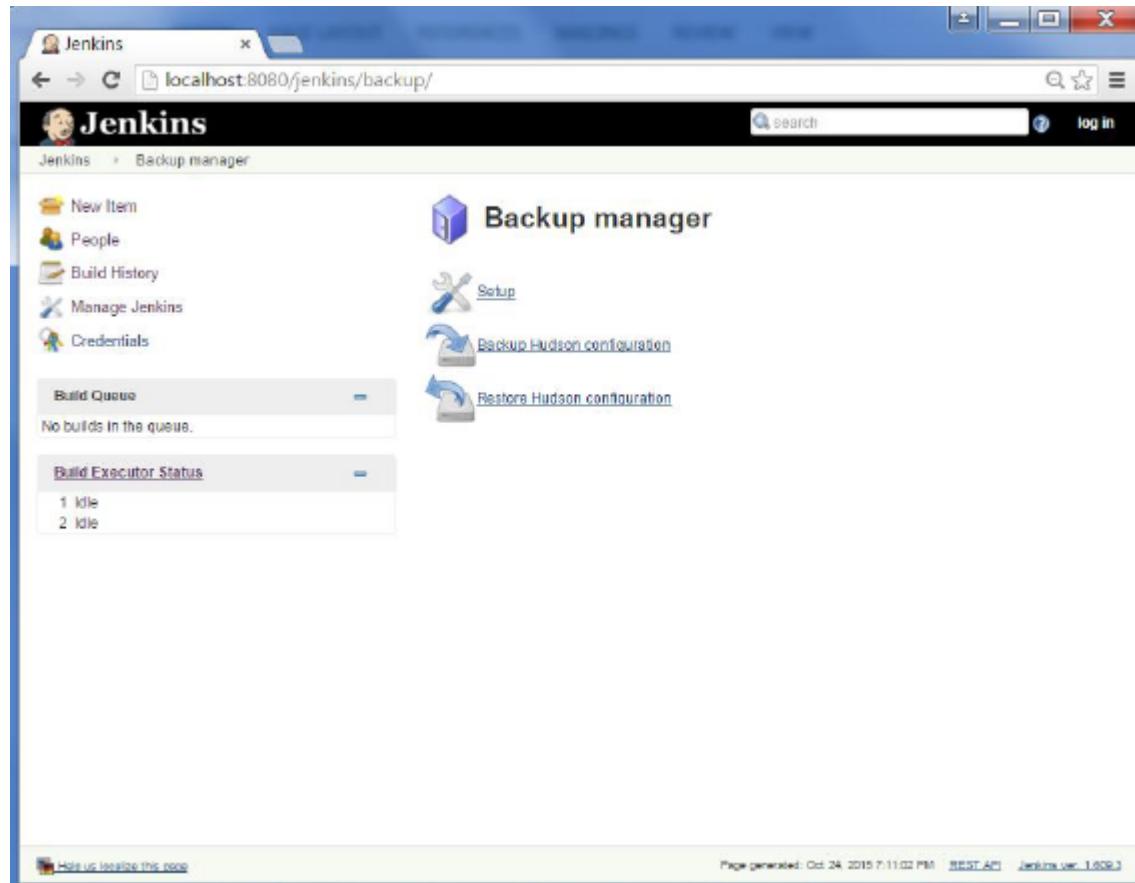
- Go back to the top page (you can start using the installed plugins right away)
- Restart Jenkins when installation is complete and no jobs are running

At the bottom left is a link 'Help us improve this page'. At the bottom right, it says 'Page generated: Oct 24, 2015 8:25:30 PM | REST API | Jenkins ver. 1.603'.

**Step 3** – Now when you go to Manage Jenkins, and scroll down you will see 'Backup Manager' as an option. Click on this option.

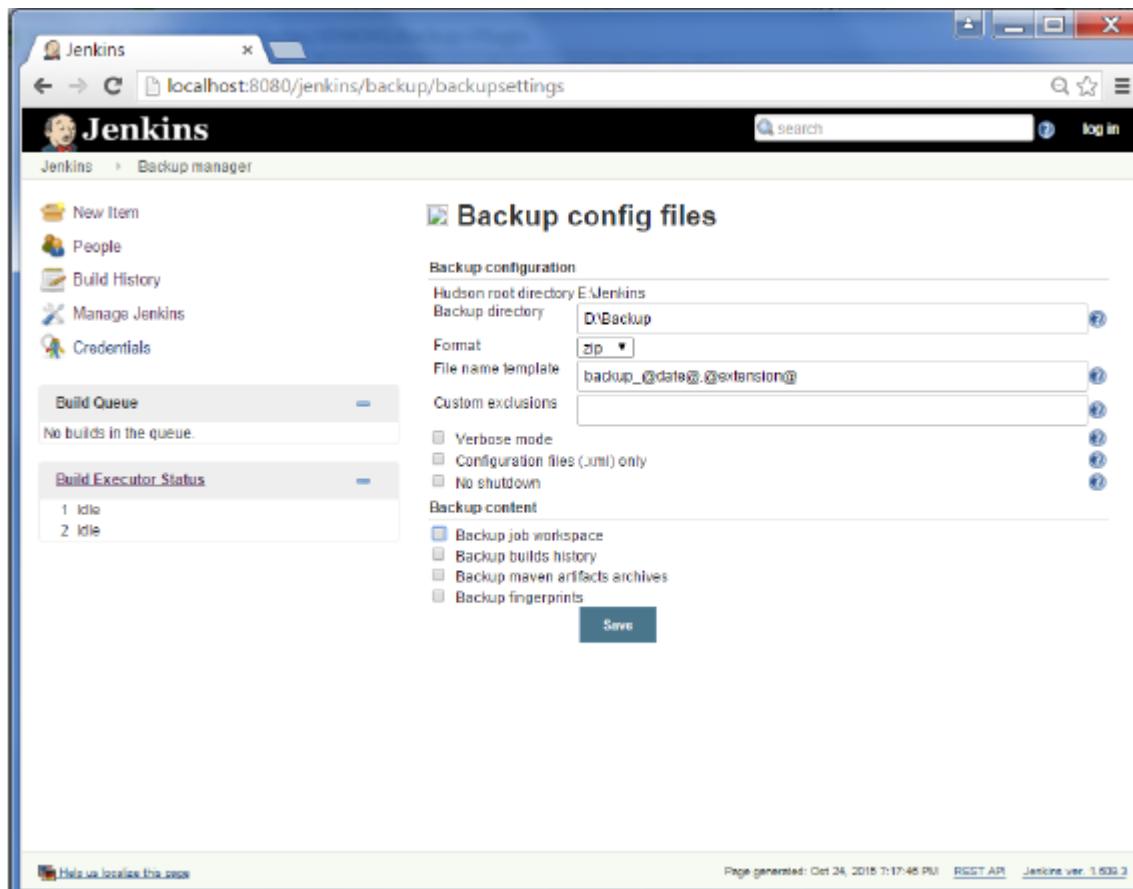


#### Step 4 – Click on Setup.

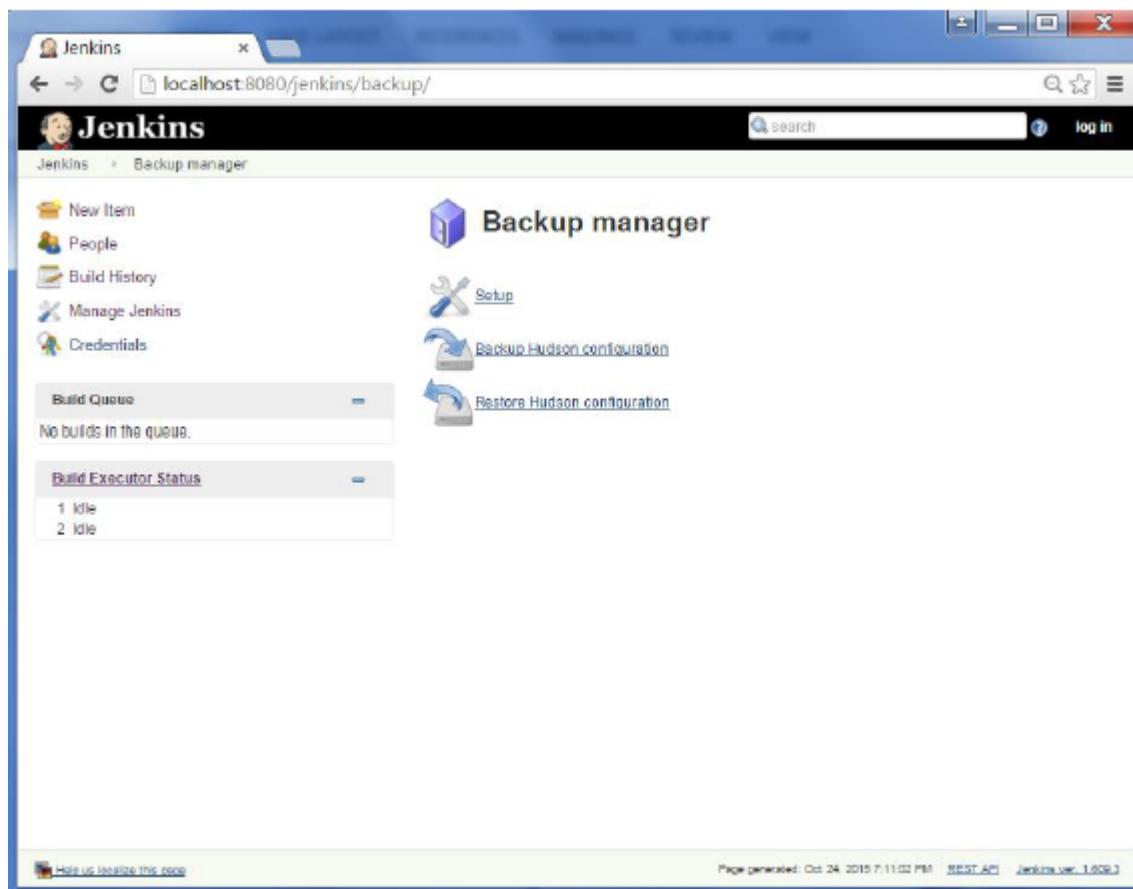


**Step 5** – Here, the main field to define is the directory for your backup. Ensure it's on another drive which is different from the drive where your Jenkins instance is setup. Click

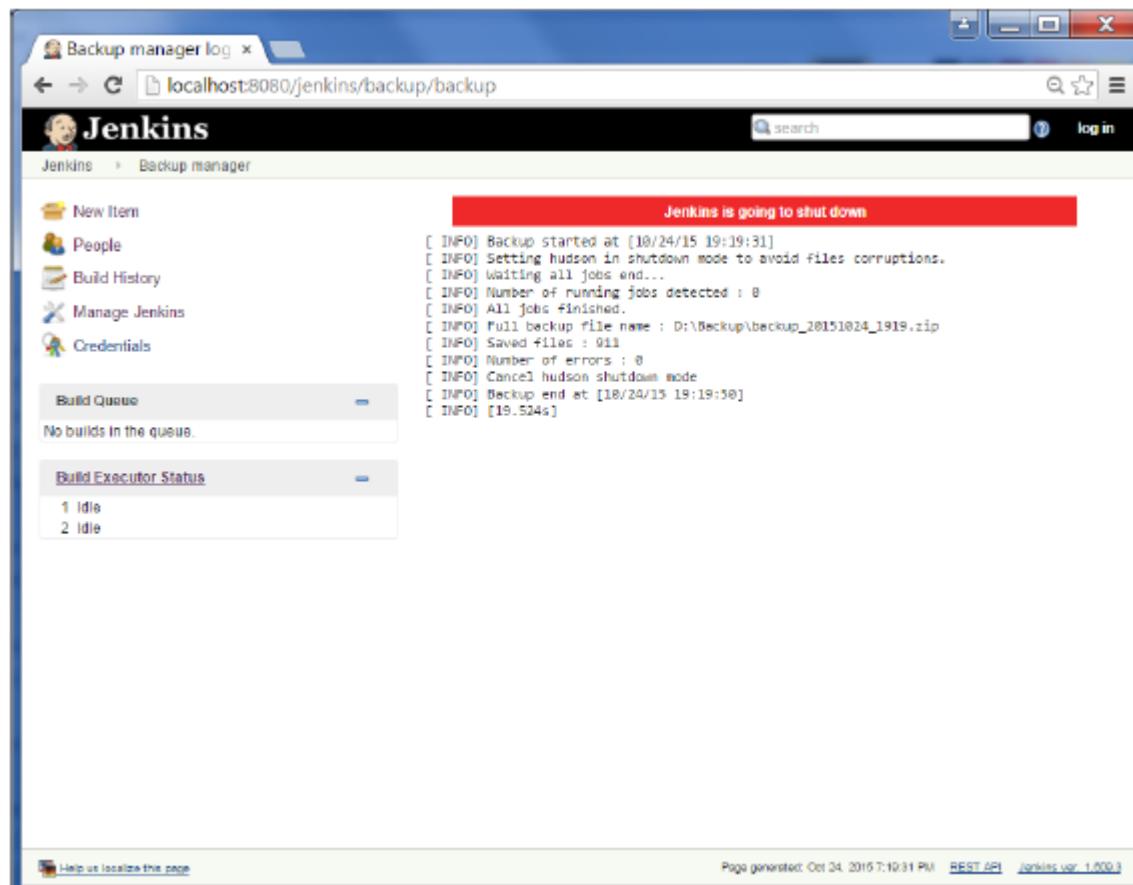
on the Save button.



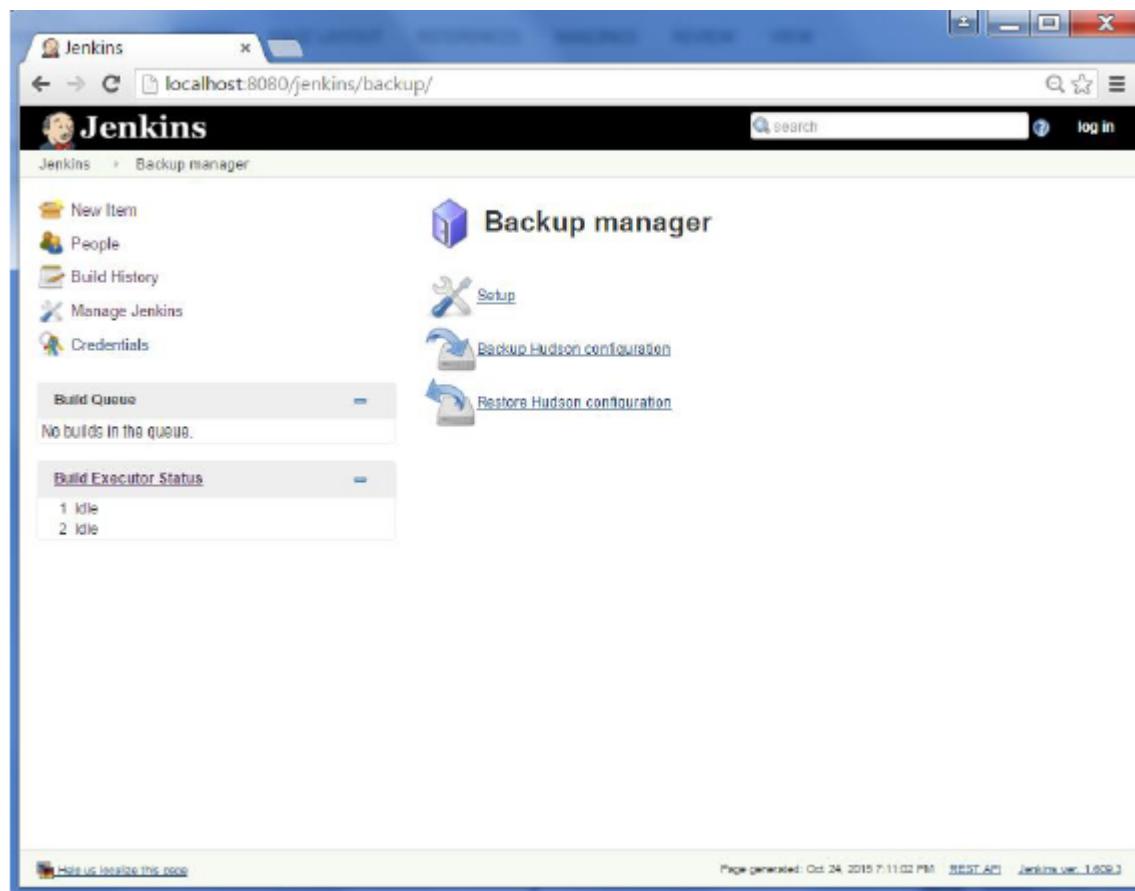
**Step 6** – Click on the ‘Backup Hudson configuration’ from the Backup manager screen to initiate the backup.



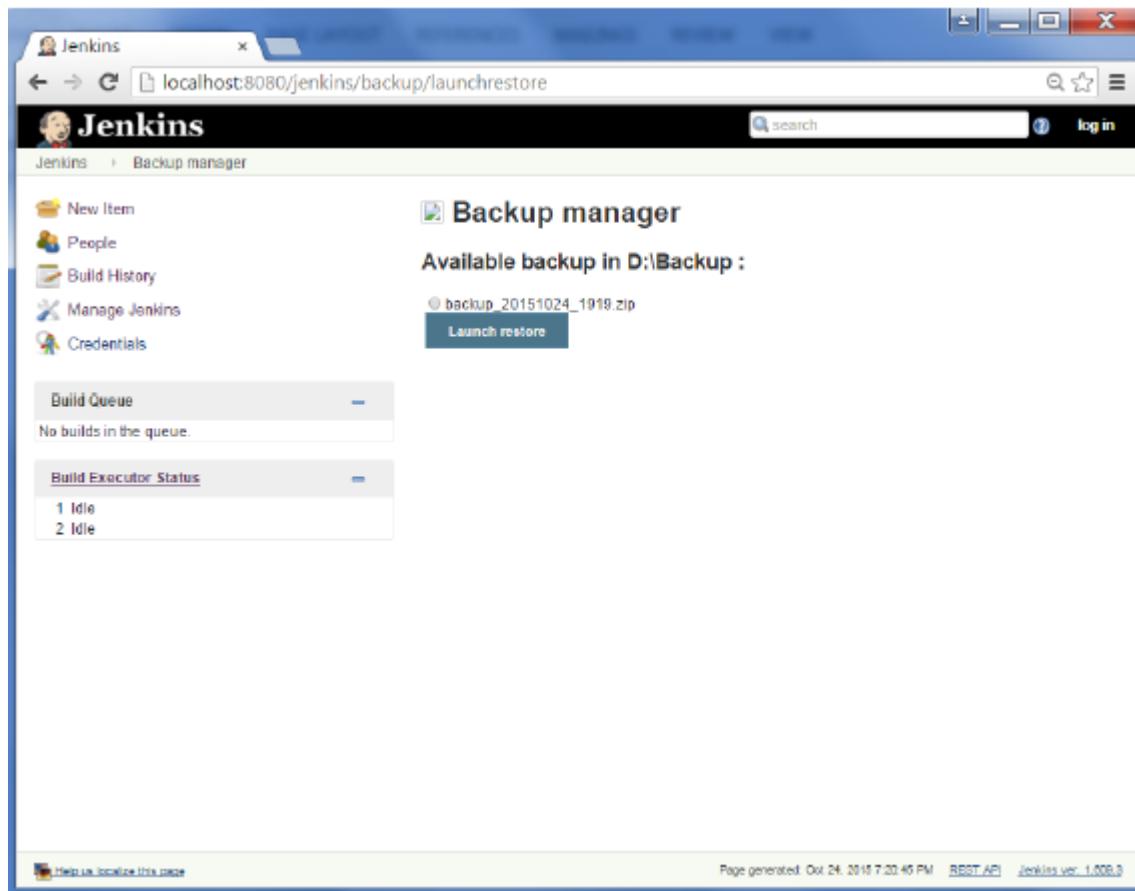
The next screen will show the status of the backup



To recover from a backup, go to the Backup Manager screen, click on Restore Hudson configuration.



The list of backup's will be shown, click on the appropriate one to click on Launch Restore to begin the restoration of the backup.



# Jenkins - Remote Testing

Web tests such as selenium tests can be run on remote slave machines via the master slave and selenium suite plugin installation. The following steps show how to run remote tests using this configuration.

**Step 1** – Ensure your master slave configuration is in place. Go to your master Jenkins server. Go to Manage Jenkins → Manage Nodes.

The screenshot shows the Jenkins Manage Jenkins page with the 'Manage Nodes' section selected. The section contains the following items:

- Manage Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Manage Credentials**: Create/delete/modify the credentials that can be used by Jenkins and by jobs running in Jenkins to connect to 3rd party services.
- About Jenkins**: See the version and license information.
- Manage Old Data**: Scrub configuration files to remove remnants from old plugins and earlier versions.
- In-process Script Approval**: Allows a Jenkins administrator to review proposed scripts (written e.g. in Groovy) which run inside the Jenkins process and so could bypass security restrictions.
- Prepare for Shutdown**: Stops executing new builds, so that the system can be eventually shut down safely.

In our node list, the DXBMEM30 label is the slave machine. In this example, both the master and slave machines are windows machines.

The screenshot shows the Jenkins Nodes page, displaying a list of available nodes:

Label	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Sp
DXBMEM30	DXBMEM30	Windows Server 2012 (x86)	In sync	112.79 GB	4.54 GB	13 min 112.79
master	master	Windows Server 2012 (x86)	In sync	94.20 GB	3.85 GB	94.20
	Data obtained	13 min	13 min	13 min	13 min	1

A blue button labeled "Refresh status" is visible at the bottom right of the table.

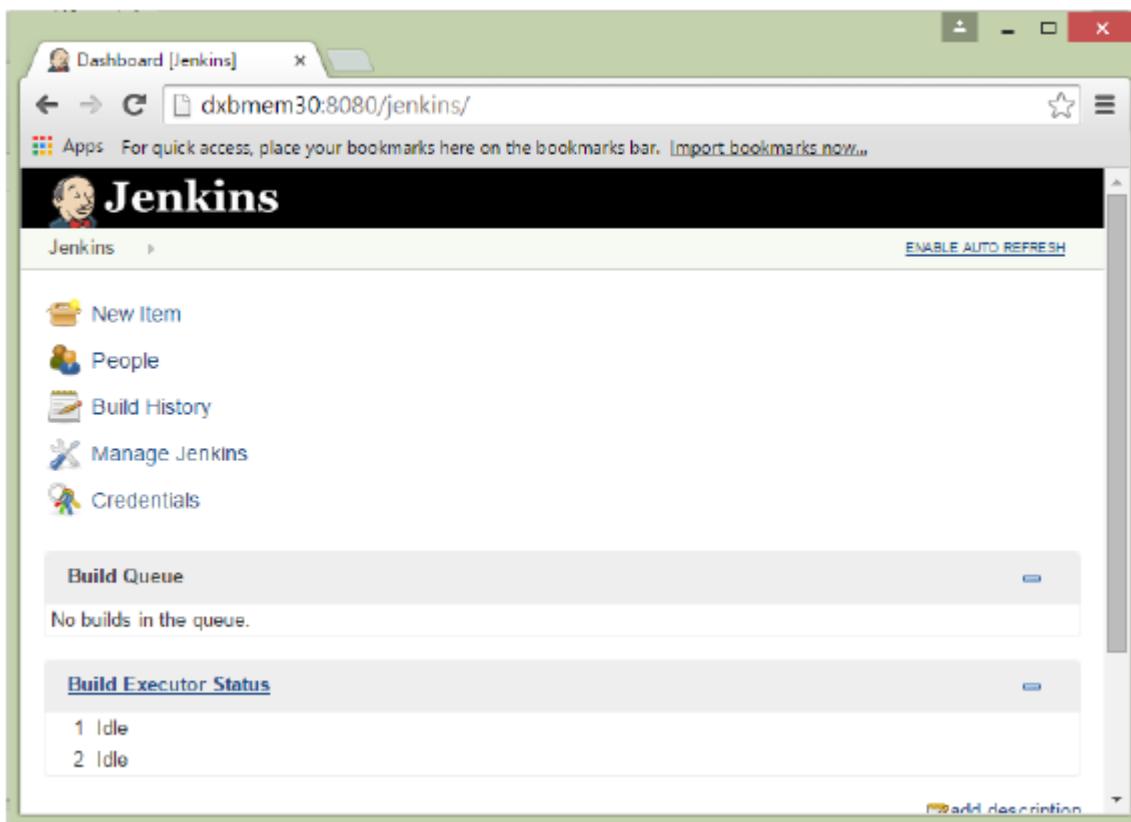
**Step 2 – Click on configure for the DXBMEM30 slave machine.**

The screenshot shows the Jenkins 'Nodes [Jenkins]' configuration page. The URL is `dxbmem20:8080/jenkins/computer/`. The left sidebar shows '2 Idle' nodes and one node named 'DXBMEM30'. The main table lists two nodes: 'DXBMEM30' (Windows Server 2012) and another unnamed node. The 'DXBMEM30' row has a context menu open with options: 'Delete Slave', 'Configure' (which is highlighted with a blue selection bar), and 'Build History'. A 'Refresh status' button is at the bottom right of the table.

**Step 3 – Ensure the launch method is put as 'Launch slave agents via Java Web Start'**

The screenshot shows the 'DXBMEM30 Configuration' page. The URL is `dxbmem20:8080/jenkins/computer/DXBMEM30/configure`. The configuration form includes fields for Name (DXBMEM30), Description, # of executors (1), Remote root directory (C:\users\administrator.EMIRATES\jenkins), Labels, Usage (Utilize this node as much as possible), and Launch method (set to 'Launch slave agents via Java Web Start'). A 'Save' button is at the bottom left, and an 'Advanced...' button is at the bottom right.

**Step 4 – Now go to your slave machine and from there, open a browser instance to your Jenkins master instance. Then go to Manage Jenkins → Manage Nodes. Go to DXBMEM30 and click on**



**Step 5** – Click on the DXBMEM30 instance.

A screenshot of the Jenkins 'Nodes' page. The URL is 'dxbmemp20:8080/jenkins/computer/'. The main content area shows 'Build Executor Status' for the 'master' node, which has '1 Idle' and '2 Idle' executors. Below this, the 'DXBMEM30 (offline)' node is listed. A table provides detailed information for both nodes:

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space
	<a href="#">DXBMEM30</a>	Windows Server 2012 (x86)	In sync	112.79 GB	4.54 GB
	<a href="#">master</a>	Windows Server 2012 (x86)	In sync	94.20 GB	3.85 GB
	Data obtained	45 min	45 min	45 min	45 min

Refresh status

**Step 6** – Scroll down and you will see the Launch option which is the option to Start 'Java Web Start'

Connect slave to Jenkins one of these ways:

- Launch agent from browser on slave
- Run from slave command line:  
`javaws http://localhost:8080/jenkins/computer/DXBMEM30/slave-agent.jnlp`
- Or if the slave is headless:  
`java -jar slave.jar -jnlpUrl http://localhost:8080/jenkins/computer/DXBMEM30/slave-agent.jnlp`

Created by anonymous user

### Projects tied to DXBMEM30

S	W	Name ↓	Last Success	Last Failure	Last Duration
		HelloWorld	43 min - #12	41 min - #13	7.3 sec

Icon: S M L      [Legend](#)    [RSS for all](#)    [RSS for failures](#)    [RSS for just latest builds](#)

**Step 7** – You will be presented with a Security Warning. Click on the Acceptance checkbox and click on run.

**Do you want to run this application?**

**Name:** Jenkins Remoting Agent

**Publisher:** Kohsuke Kawaguchi

**Locations:** <http://dxbmemp20:8080>  
Launched from downloaded JNLP file

**Running this application may be a security risk**

**Risk:** This application will run with unrestricted access which may put your computer and personal information at risk. The information provided is unreliable or unknown so it is recommended not to run this application unless you are familiar with its source

Unable to ensure the certificate used to identify this application has not been revoked.  
[More Information](#)

**Select the box below, then click Run to start the application**

I accept the risk and want to run this application.

**Run**    **Cancel**

You will now see a Jenkins Slave window opened and now connected.



**Step 8** – Configuring your tests to run on the slave. Here, you have to ensure that the job being created is meant specifically to only run the selenium tests.

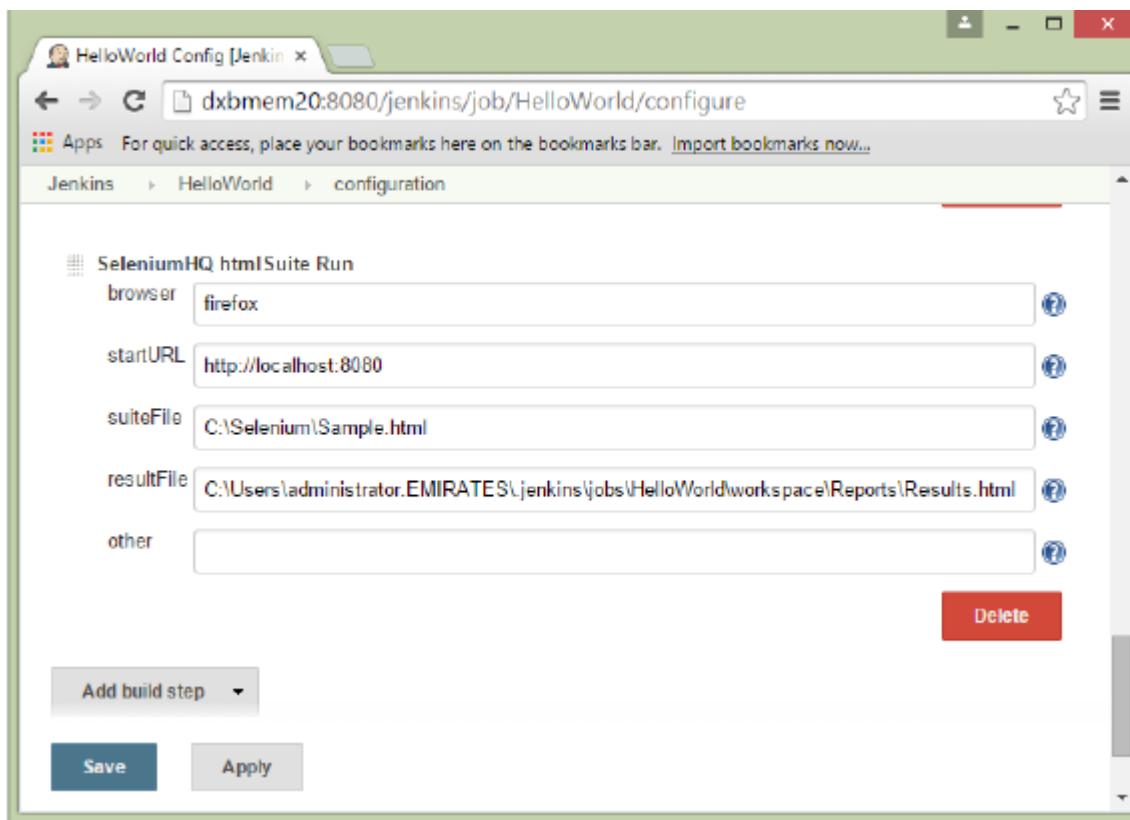
In the job configuration, ensure the option 'Restrict where this project can be run' is selected and in the Label expression put the name of the slave node.

A screenshot of a web browser displaying the Jenkins job configuration page for "HelloWorld". The URL in the address bar is "dxbmem20:8080/jenkins/job/HelloWorld/configure". The page shows the "configuration" section with several options:

- Discard Old Builds
- This build is parameterized
- Disable Build (No new builds will be executed until the project is re-enabled.)
- Execute concurrent builds if necessary
- Restrict where this project can be run

A "Label Expression" field contains the value "DXBMEM30". Below it, a note says "Slaves in label: 1". At the bottom of the configuration section are "Advanced Project Options" and an "Advanced..." button. At the very bottom of the page are "Save" and "Apply" buttons.

**Step 9** – Ensure the selenium part of your job is configured. You have to ensure that the Sample.html file and the selenium-server.jar file is also present on the slave machine.



Once you have followed all of the above steps, and click on Build, this project will run the Selenium test on the slave machine as expected.

[Previous Page](#)

[Next Page](#)

Advertisements

Ad ▾

## US Visa \$500,000 Required

A Path To U.S. Immigration Through  
Investment. Your Family Members Under 21  
May Qualify.



[FAQ's](#) [Cookies Policy](#) [Contact](#)

© Copyright 2018. All Rights Reserved.

Enter email for newsletter

go