



# Jenkins by Example

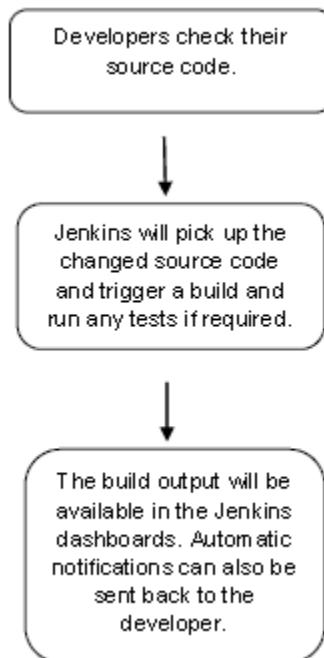
<u>JENKINS - OVERVIEW</u>	<b>4</b>
<b>WHY JENKINS?</b>	<b>4</b>
<b>WHAT IS CONTINUOUS INTEGRATION?</b>	<b>4</b>
<b>SYSTEM REQUIREMENTS</b>	<b>5</b>
<u>JENKINS - INSTALLATION</u>	<b>6</b>
<b>DOWNLOAD JENKINS</b>	<b>6</b>
<b>STARTING JENKINS</b>	<b>7</b>
<b>ACCESSING JENKINS</b>	<b>8</b>
<u>JENKINS – TOMCAT SETUP</u>	<b>9</b>
<b>STEP 1: VERIFYING JAVA INSTALLATION</b>	<b>9</b>
<b>STEP 2: VERIFYING JAVA INSTALLATION</b>	<b>10</b>
<b>STEP 3: DOWNLOAD TOMCAT</b>	<b>10</b>
<b>STEP 4: JENKINS AND TOMCAT SETUP</b>	<b>12</b>
<u>JENKINS – MAVEN SETUP</u>	<b>20</b>
<b>STEP 1: DOWNLOADING AND SETTING UP MAVEN</b>	<b>20</b>
<b>STEP 2: SETTING UP JENKINS AND MAVEN</b>	<b>21</b>
<u>JENKINS - CONFIGURATION</u>	<b>28</b>
<b>JENKINS HOME DIRECTORY</b>	<b>29</b>
<b># OF EXECUTORS</b>	<b>30</b>
<b>ENVIRONMENT VARIABLES</b>	<b>31</b>
<b>JENKINS URL</b>	<b>31</b>
<b>EMAIL NOTIFICATION</b>	<b>31</b>
<b>CONFIGURE SYSTEM</b>	<b>33</b>
<b>RELOAD CONFIGURATION FROM DISK</b>	<b>33</b>
<b>MANAGE PLUGIN</b>	<b>34</b>
<b>SYSTEM INFORMATION</b>	<b>34</b>
<b>SYSTEM LOG</b>	<b>35</b>
<b>LOAD STATISTICS</b>	<b>35</b>
<b>SCRIPT CONSOLE</b>	<b>35</b>
<b>MANAGE NODES</b>	<b>36</b>
<b>PREPARE FOR SHUTDOWN</b>	<b>36</b>
<u>JENKINS - SETUP BUILD JOBS</u>	<b>37</b>
<u>JENKINS - UNIT TESTING</u>	<b>48</b>
<b>EXAMPLE OF A JUNIT TEST IN JENKINS</b>	<b>50</b>
<u>JENKINS - AUTOMATED TESTING</u>	<b>59</b>
<u>JENKINS - NOTIFICATION</u>	<b>65</b>
<u>JENKINS - REPORTING</u>	<b>68</b>
<u>JENKINS - CODE ANALYSIS</u>	<b>69</b>

<u>JENKINS - DISTRIBUTED BUILDS</u>	<b>71</b>
<u>JENKINS - AUTOMATED DEPLOYMENT</u>	<b>76</b>
<u>JENKINS - METRICS &amp; TRENDS</u>	<b>79</b>
<u>JENKINS - SERVER MAINTENANCE</u>	<b>92</b>
<b>URL OPTIONS</b>	<b>92</b>
<b>BACKUP JENKINS HOME</b>	<b>92</b>
<u>JENKINS - CONTINUOUS DEPLOYMENT</u>	<b>94</b>
<u>JENKINS - MANAGING PLUGINS</u>	<b>106</b>
<b>UNINSTALLING PLUGINS</b>	<b>106</b>
<b>INSTALLING ANOTHER VERSION OF A PLUGIN</b>	<b>107</b>
<u>JENKINS - SECURITY</u>	<b>108</b>
<u>JENKINS - BACKUP PLUGIN</u>	<b>113</b>
<u>JENKINS - REMOTE TESTING</u>	<b>120</b>

# 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-ci.org). 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
```

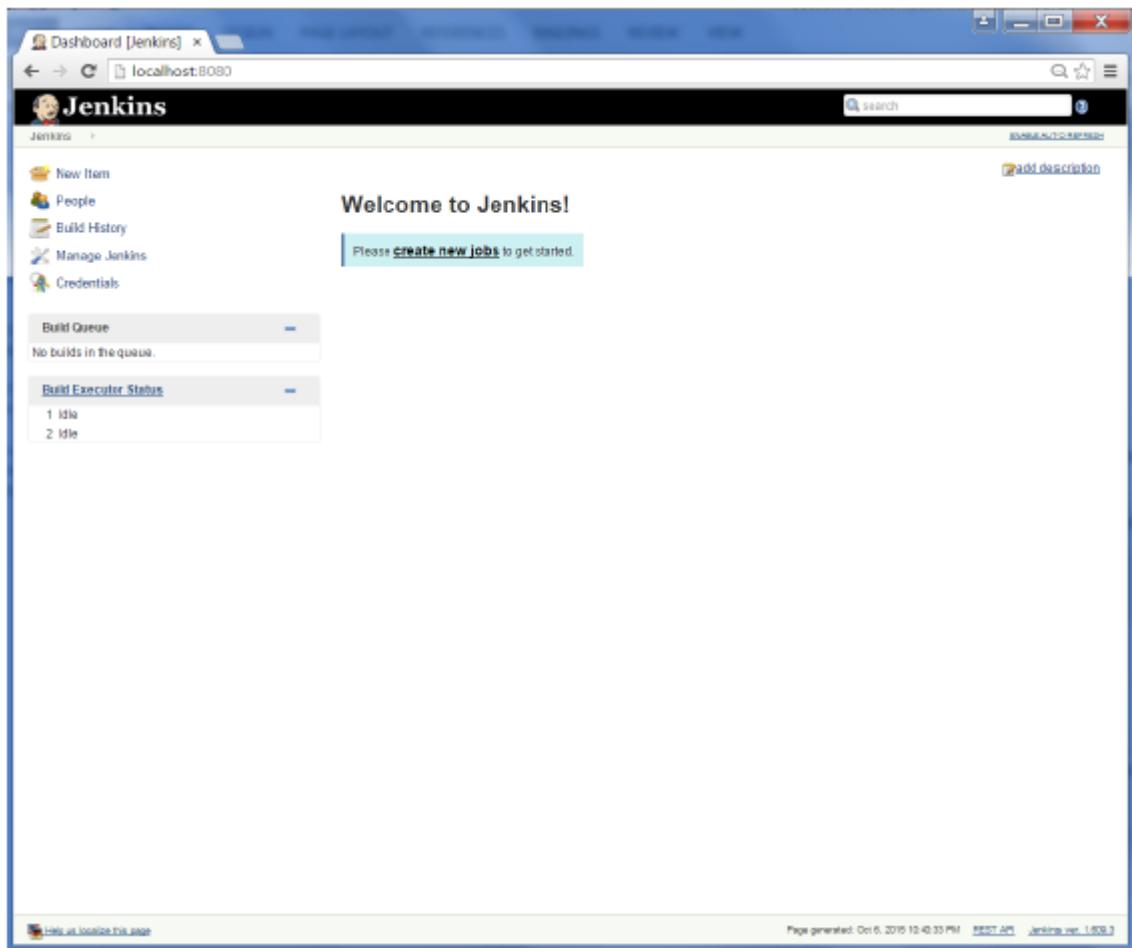
Once the processing is complete without major errors, the following line will come in the output of the command prompt.

```
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

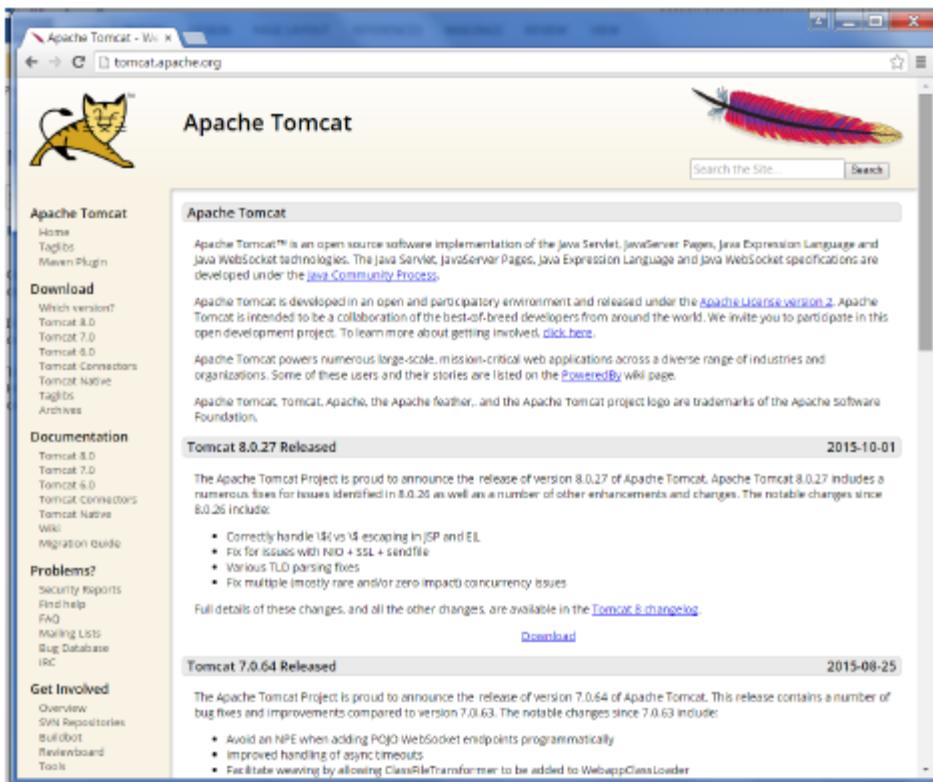
Append the full path of the Java compiler location to the System Path.

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.

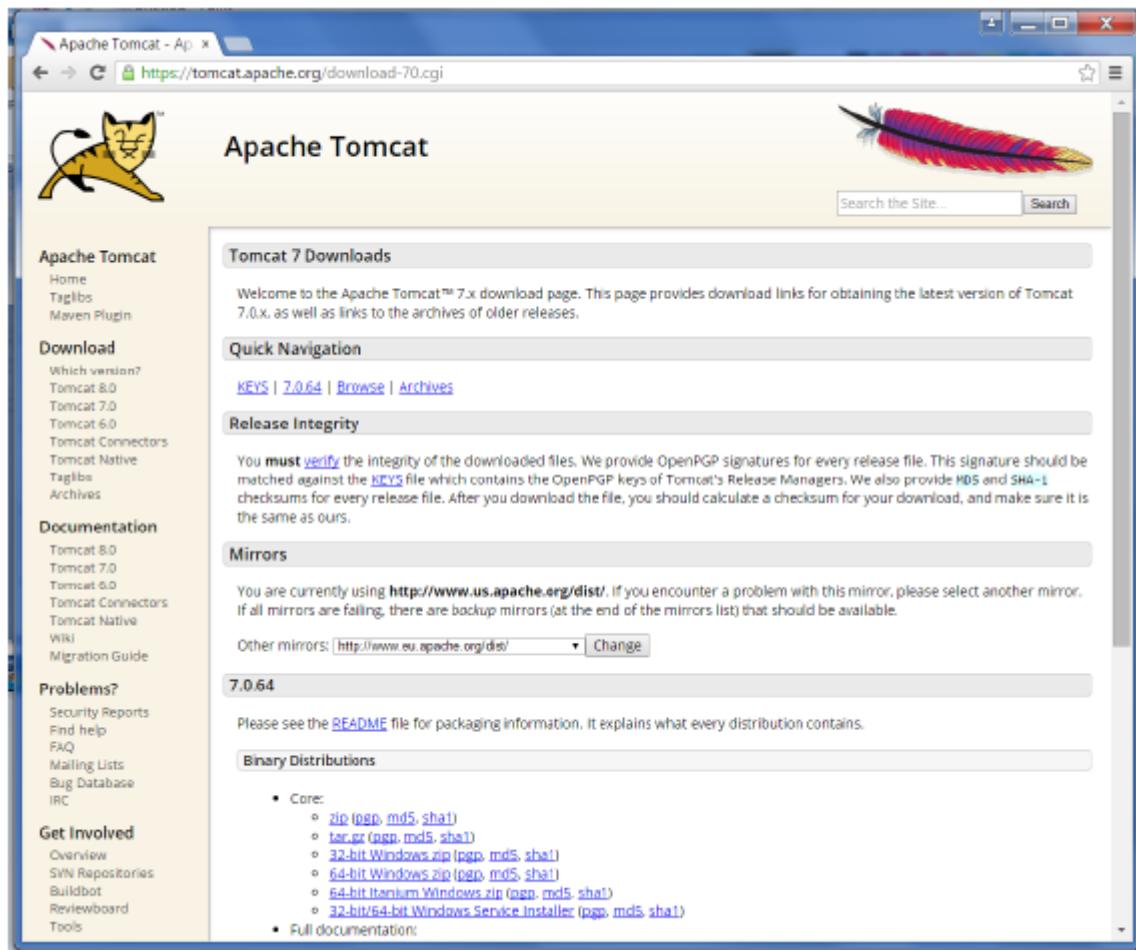
## Step 3: Download Tomcat

The official website for tomcat is [Tomcat](#). 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 homepage. The page features a yellow cat logo on the left and a red feather logo on the right. The main content area is titled "Apache Tomcat". It includes a search bar and a "Search" button. The left sidebar contains links for "Apache Tomcat", "Download", "Documentation", "Problems?", and "Get Involved". The "Download" section lists versions 8.0, 7.0, and 6.0, along with "Tomcat Native", "Wiki", "Migration Guide", "FAQ", "Mailing Lists", "Bug Database", and "IRC". The "Documentation" section links to "Tomcat 8.0.27 Released" (2015-10-01) and "Tomcat 7.0.64 Released" (2015-08-25). The "Problems?" section links to "Security Reports", "Find help", "FAQ", "Mailing Lists", "Bug Database", and "IRC". The "Get Involved" section links to "Overview", "SVN Repositories", "Buildbot", "Reviewboard", and "Tools". The central content area discusses the Apache Tomcat project, its history, and its impact on the Java ecosystem. It also highlights the release of version 8.0.27, listing several bug fixes and improvements.

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



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 Jenkins.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 located. 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.

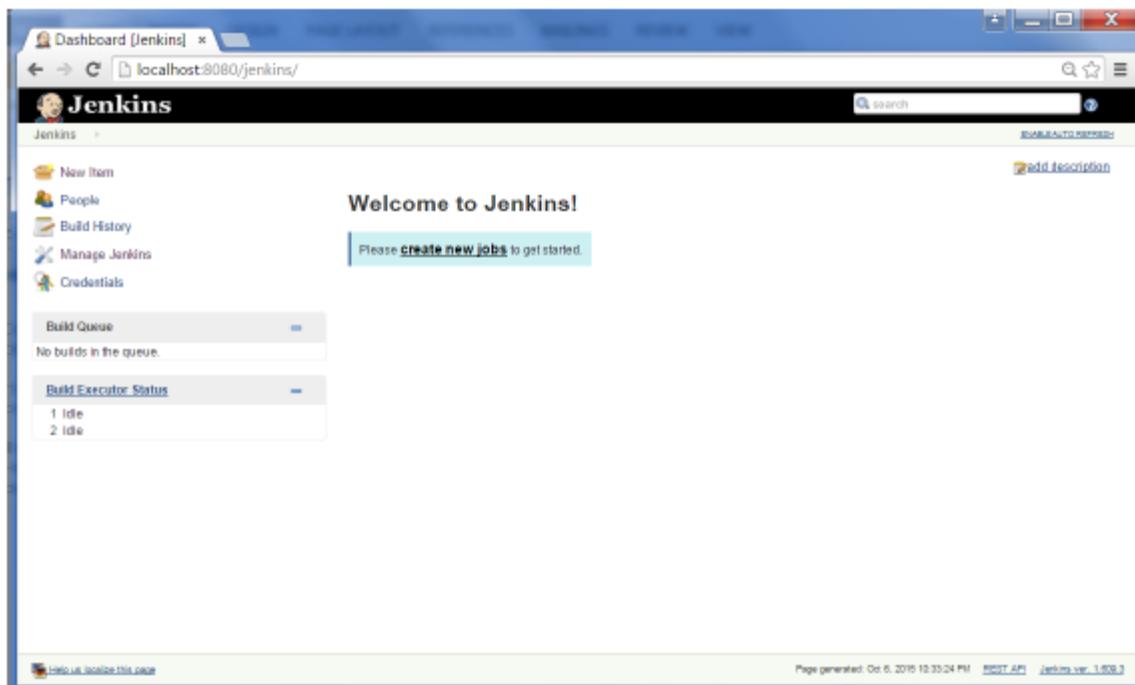
```
INFO: Server startup in 1302 ms
```

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

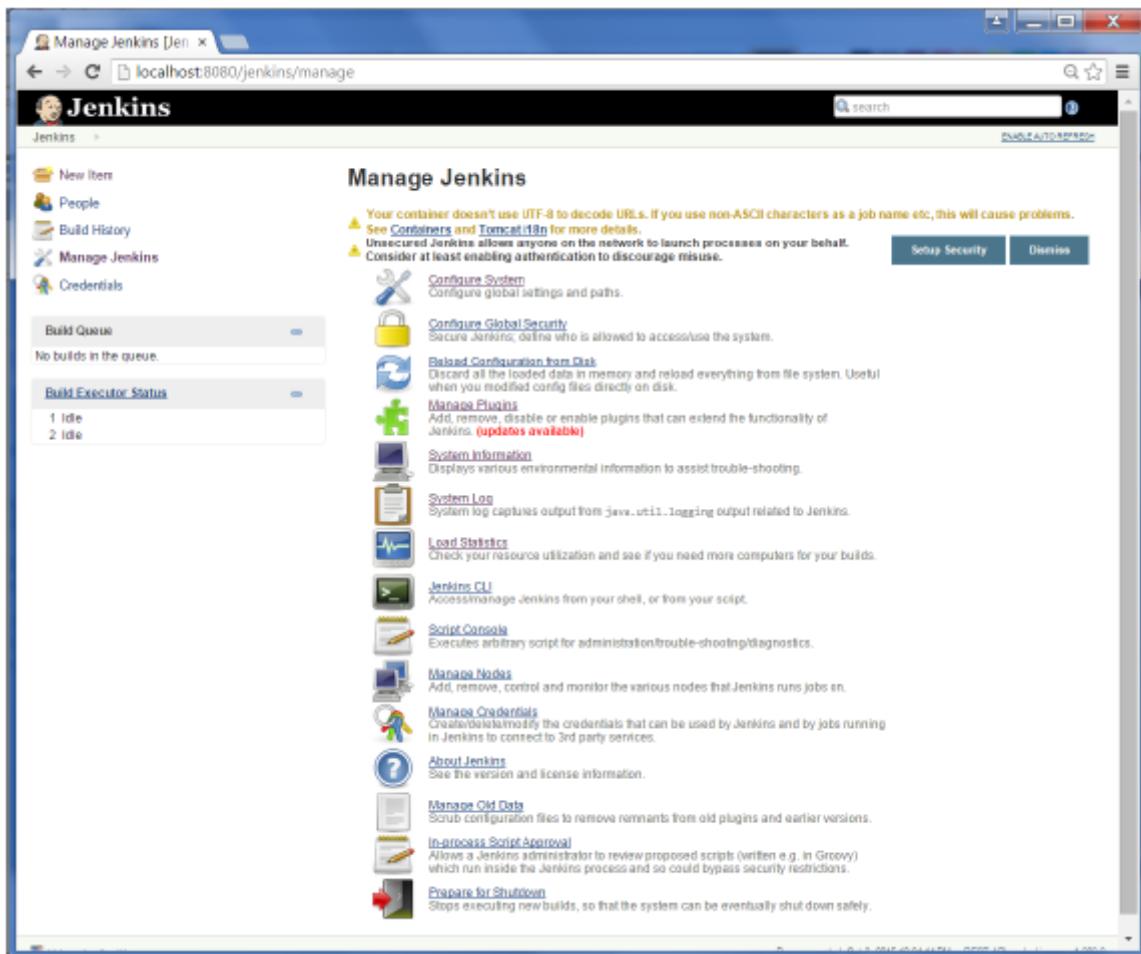


# Jenkins - Git Setup

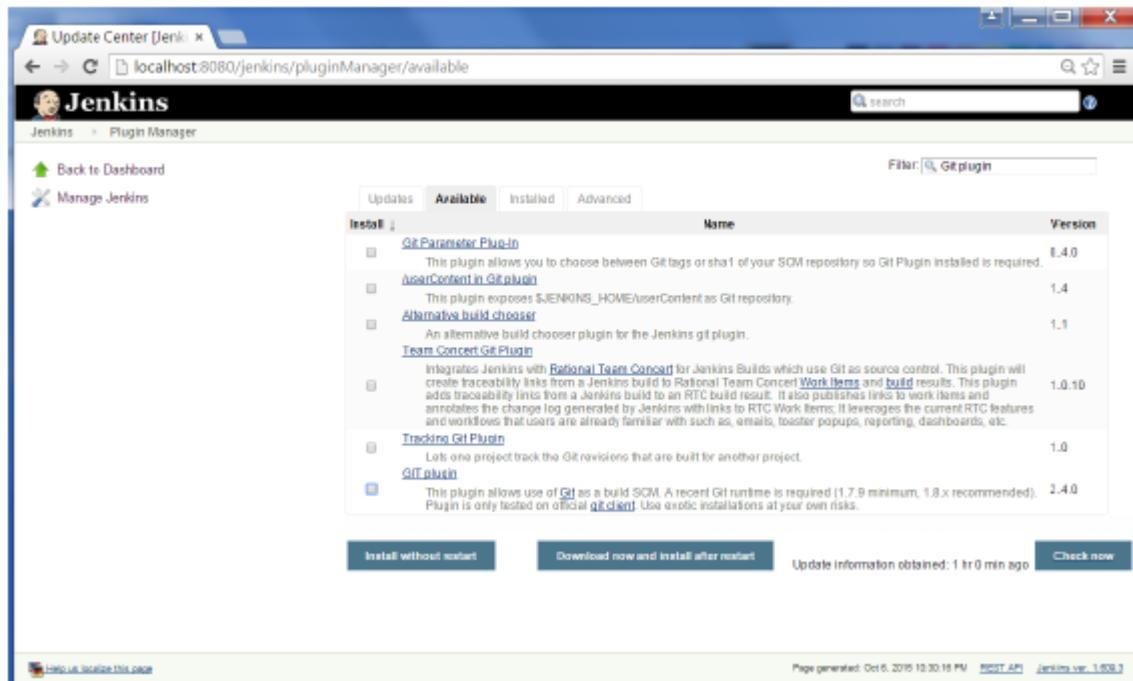
For this exercise, you have to ensure that Internet connectivity is present from the machine on which Jenkins is installed. In your Jenkins Dashboard (Home screen), click the Manage Jenkins option on the left hand side.



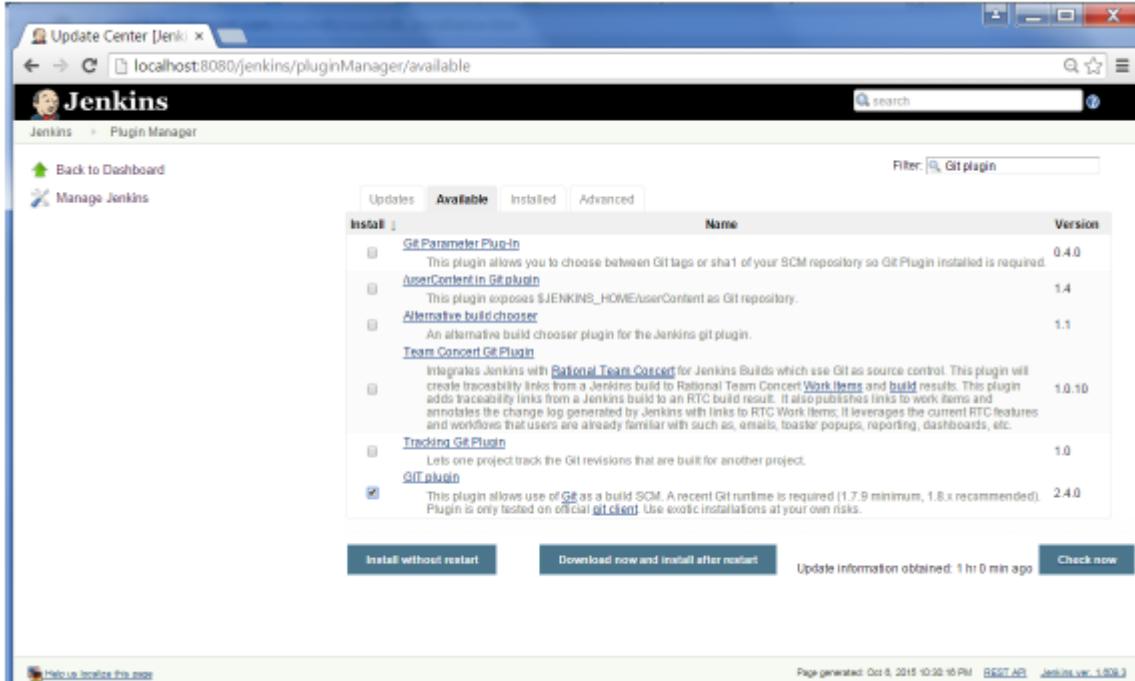
In the next screen, click the 'Manage Plugins' option.



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

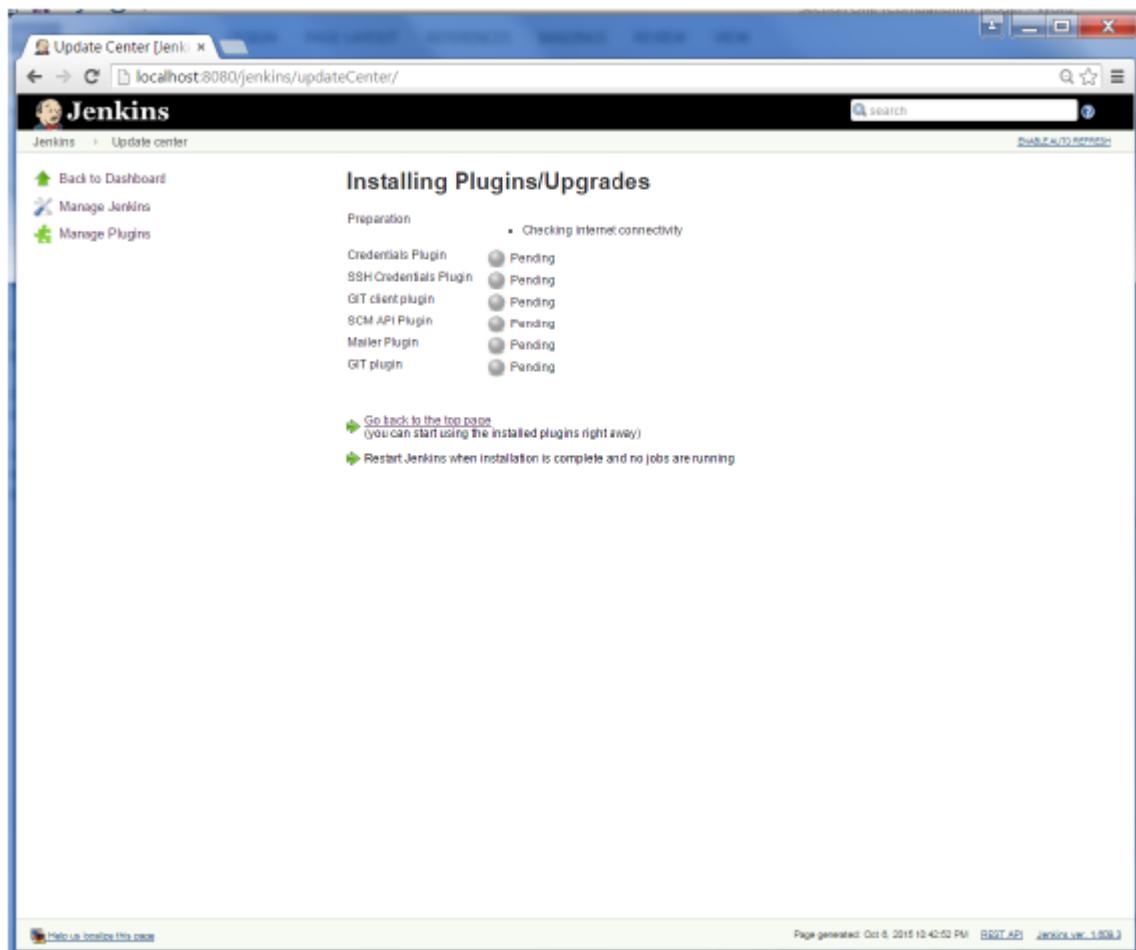


The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected, and a search bar at the top right contains the text 'Git plugin'. Below the search bar, a table lists several Git-related plugins. The 'GIT plugin' entry is highlighted with a checked checkbox in its 'Install' column. At the bottom of the table, there are three buttons: 'Install without restart' (highlighted in blue), 'Download now and install after restart', and 'Check now'.

Install	Name	Version
<input type="checkbox"/>	Git Parameter Plugin	0.4.0
<input type="checkbox"/>	UserContent in Git plugin	1.4
<input type="checkbox"/>	Alternative build chooser	1.1
<input type="checkbox"/>	Team Concert Git Plugin	1.0.10
<input type="checkbox"/>	Tracking Git Plugin	1.0
<input checked="" type="checkbox"/>	GIT plugin	2.4.0

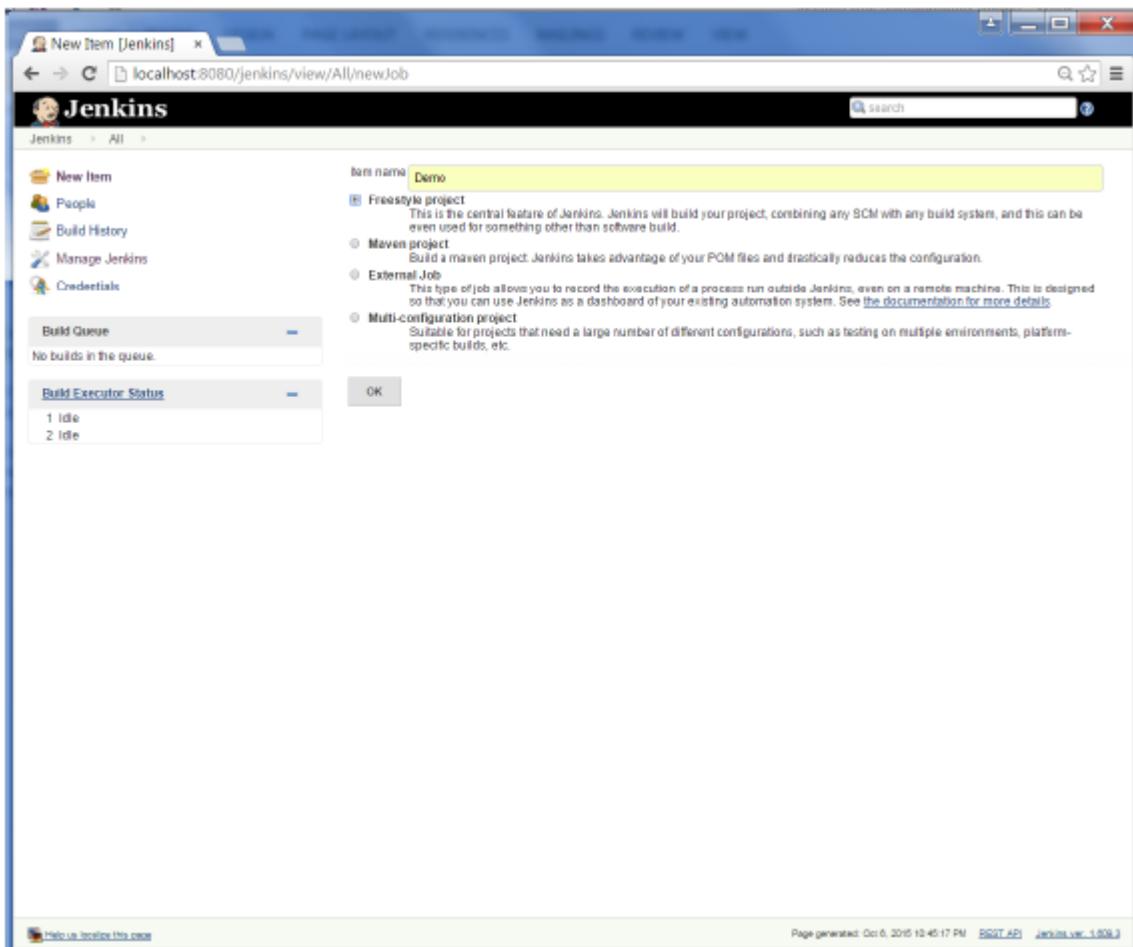
At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 8, 2015 10:32:16 PM', 'Build API', and 'Jenkins ver. 1.808'.

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.

Demo Config Jenkins x

localhost:8080/jenkins/job/Demo/configure

# Jenkins

Back to Dashboard Status Changes Workspace Build Now Delete Project Configure

Build History trend RSS for all RSS for failures

Project name: Demo

Description:   
 [Escaped HTML] Preview

Advanced Project 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

Source Code Management

None

CVS

CVS Projectset

Git

Subversion

Build Triggers

Build after other projects are built

Build periodically

Poll SCM

Build

Add build step

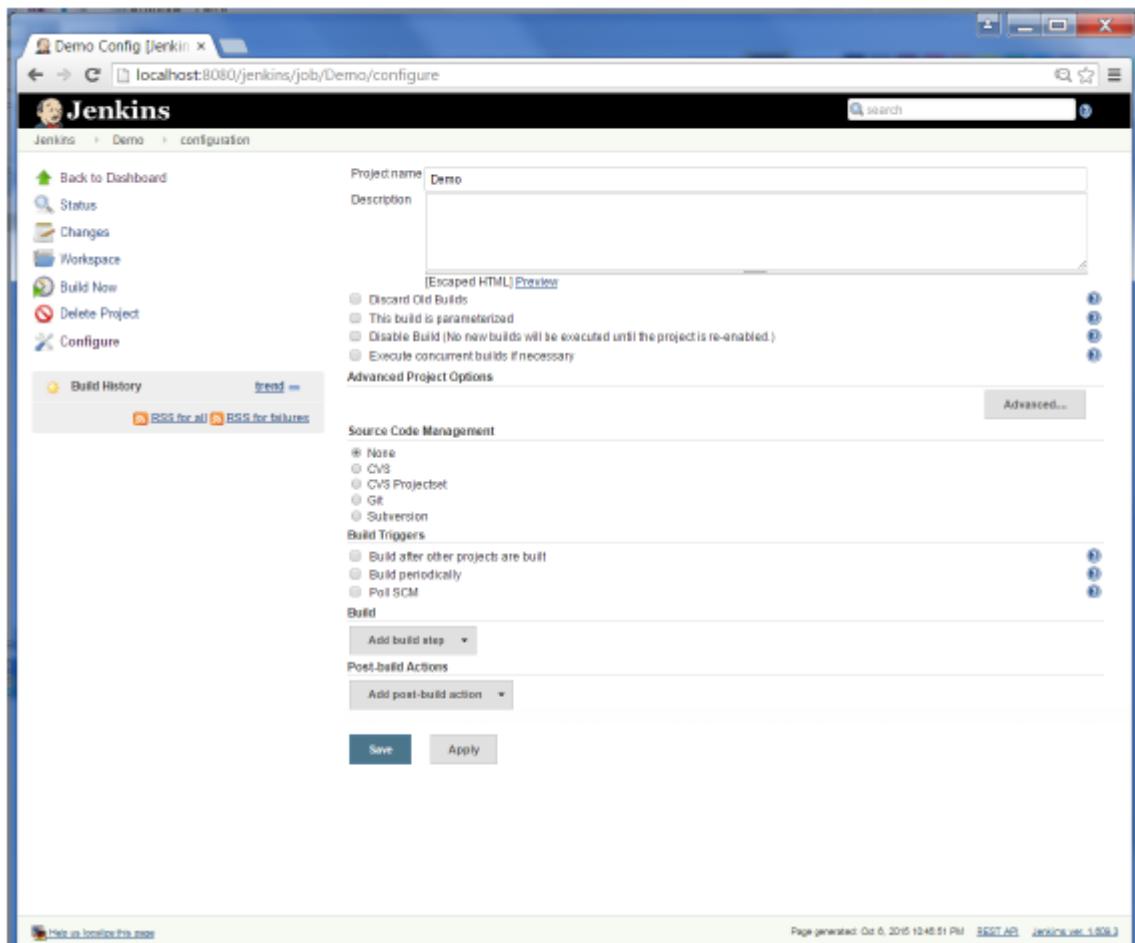
Post-build Actions

Add post-build action

Save Apply

Help is located [here](#).

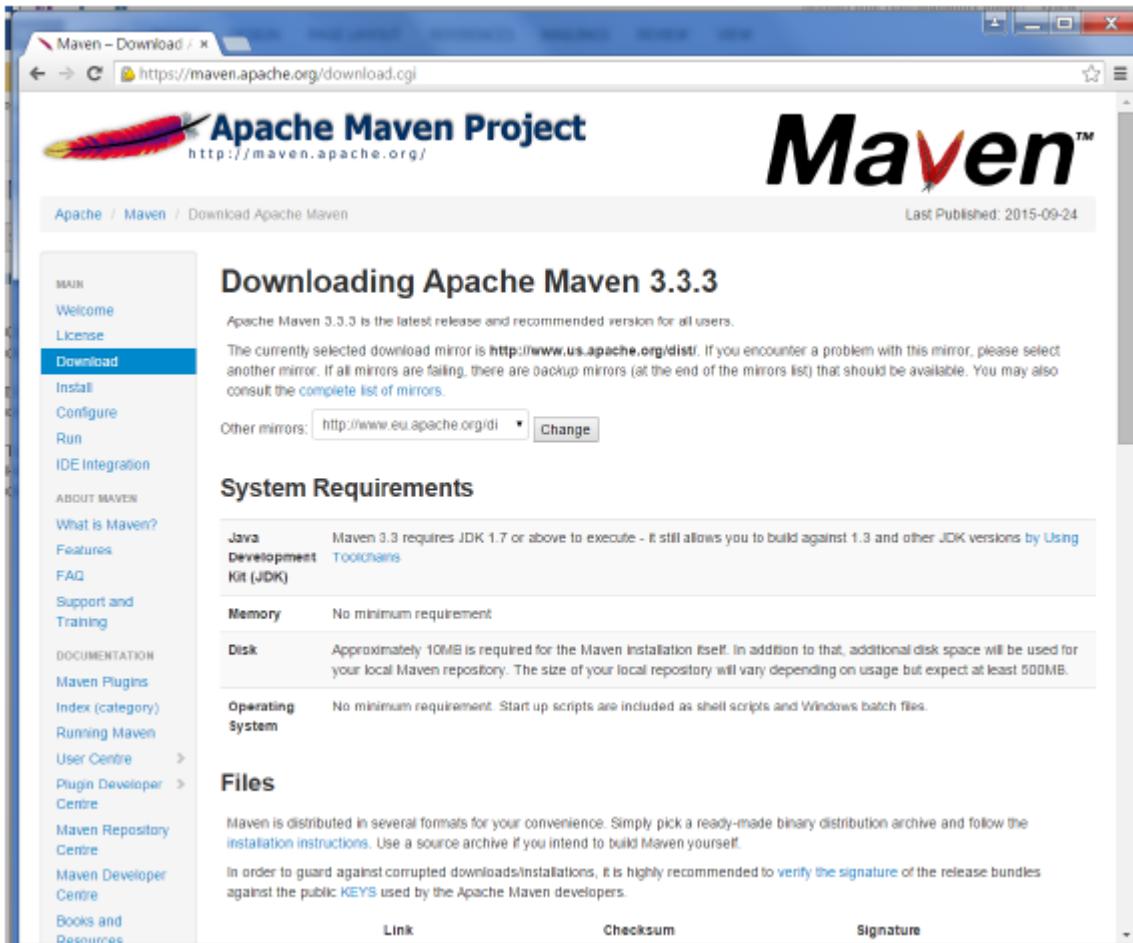
Page generated: Oct 6, 2015 10:45:51 PM REST API Jenkins ver. 1.608.3



# Jenkins – Maven Setup

## Step 1: Downloading and Setting Up Maven

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



The screenshot shows a web browser window with the URL <https://maven.apache.org/download.cgi>. The page is titled "Apache Maven Project" and features the "Maven" logo. The main content is titled "Downloading Apache Maven 3.3.3". It includes a note that Maven 3.3.3 is the latest release. A "System Requirements" table lists the following: Java Development Kit (JDK) requires JDK 1.7 or above, Memory has no minimum requirement, Disk requires approximately 10MB, and Operating System has no minimum requirement. The "Files" section provides links for download, checksum, and signature. The left sidebar has a "Download" link highlighted in blue.

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

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  
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  
IVY

Memory No minimum requirement  
Disk Approximately 10MB is required for the Maven installation itself. In addition to that, additional disk space will be used for your local Maven repository. The size of your local repository will vary depending on usage but expect at least 500MB.  
Operating System No minimum requirement. Start up scripts are included as shell scripts and Windows batch files.

## Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [Installation Instructions](#). Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

Link	Checksum	Signature
Binary tar.gz archive <a href="#">apache-maven-3.3.3-bin.tar.gz</a>	apache-maven-3.3.3-bin.tar.gz.md5	apache-maven-3.3.3-bin.tar.gz.asc
Binary zip archive <a href="#">apache-maven-3.3.3-bin.zip</a>	apache-maven-3.3.3-bin.zip.md5	apache-maven-3.3.3-bin.zip.asc
Source tar.gz archive <a href="#">apache-maven-3.3.3-src.tar.gz</a>	apache-maven-3.3.3-src.tar.gz.md5	apache-maven-3.3.3-src.tar.gz.asc
Source zip archive <a href="#">apache-maven-3.3.3-src.zip</a>	apache-maven-3.3.3-src.zip.md5	apache-maven-3.3.3-src.zip.asc

- Release Notes
- Reference Documentation
- Apache Maven Website As Documentation Archive
- All sources (plugins, shared libraries, ...) available at <http://www.apache.org/dist/maven/>
- Distributed under the Apache License, version 2.0

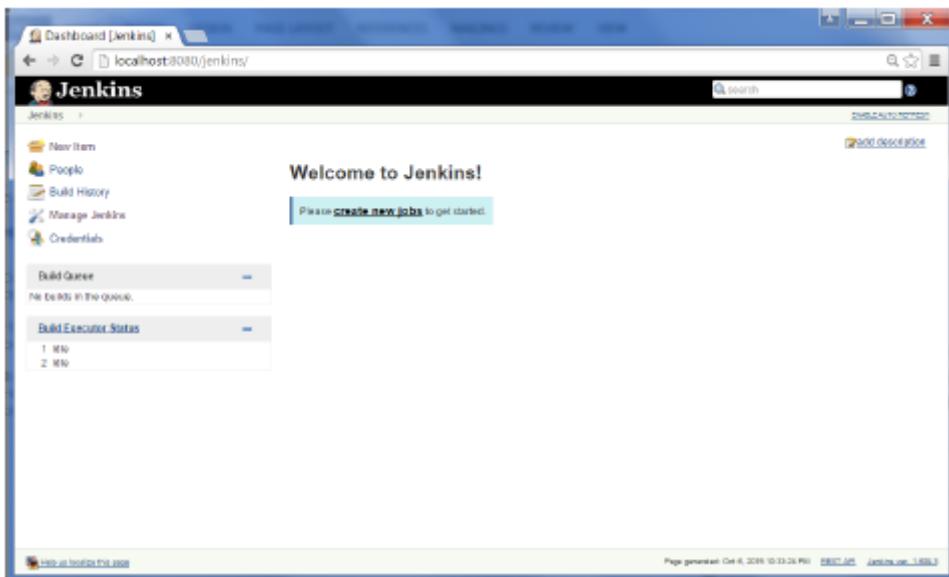
## Previous Releases

It is strongly recommended to use the latest release version of Apache Maven to take advantage of newest features and bug fixes. If you still want to use an old version you can find more information in the [Maven Releases History](#) and can download files from the [archives](#) for versions 3.0.4+ and [legacy archives](#) for earlier releases.

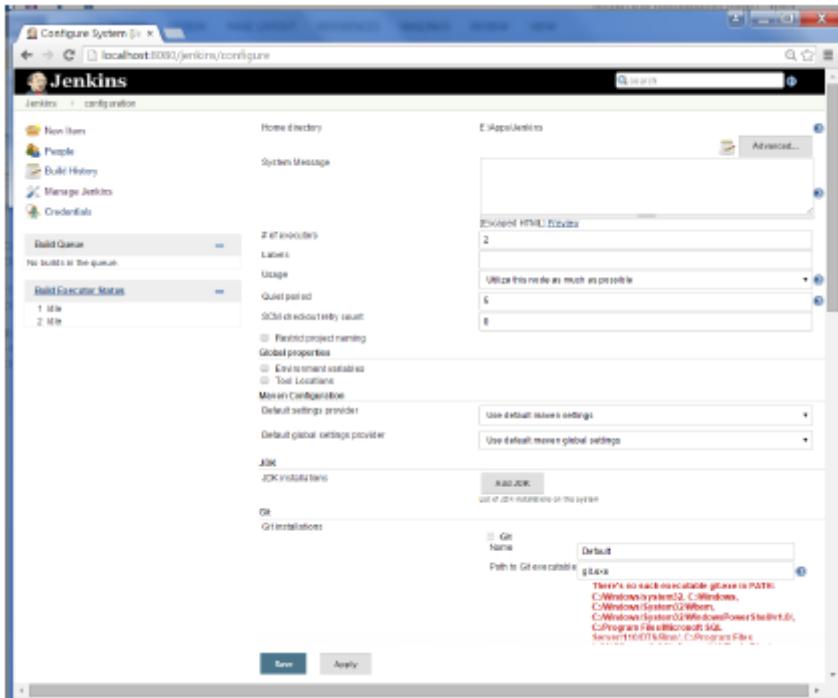
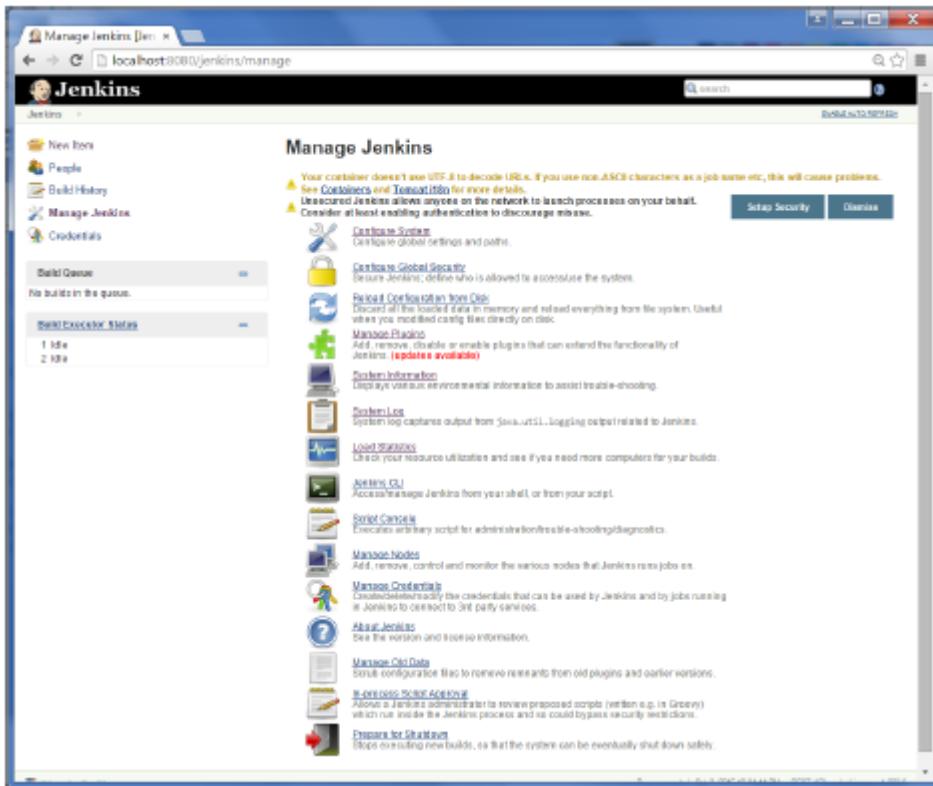
Once the file is downloaded, extract the files to the relevant application folder. For this purpose, the maven files will be placed in E:\Apps\apache-maven-3.3.3.

## Step 2: Setting up Jenkins and Maven

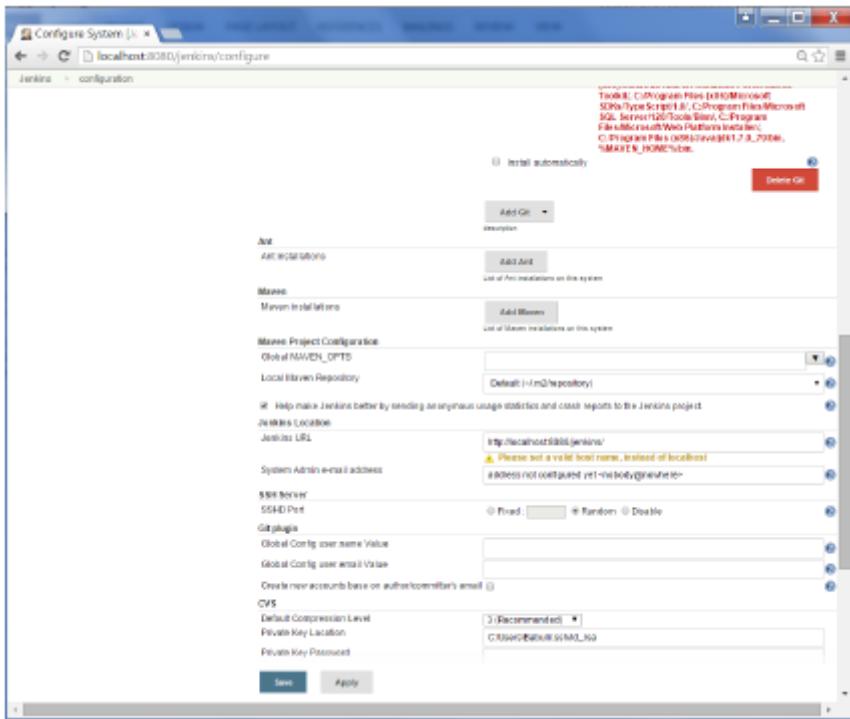
In the Jenkins dashboard (Home screen), click Manage Jenkins from the left-hand side menu.



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



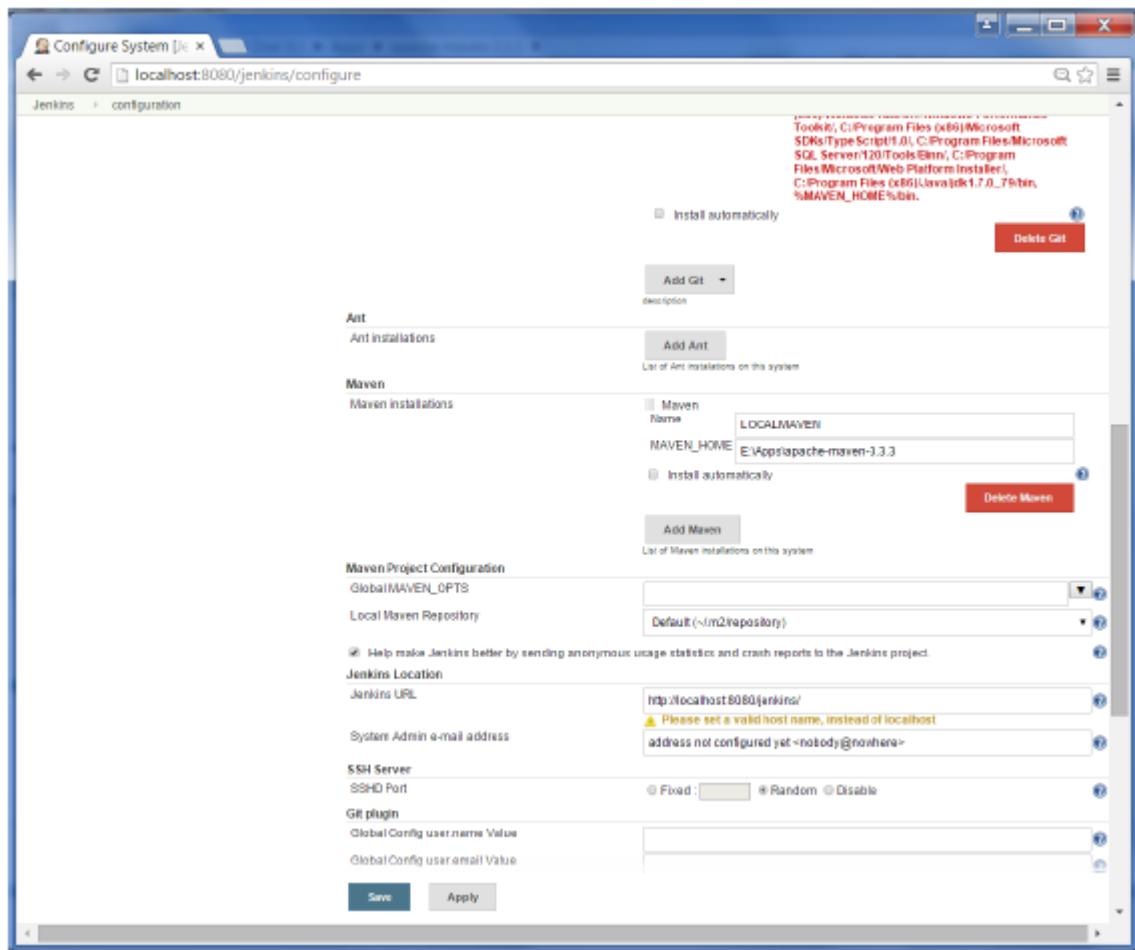
In the Configure system screen, scroll down till you see the Maven section and then click on the 'Add Maven' button.



Uncheck the 'Install automatically' option.

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

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



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

Dashboard [Jenkins] x

localhost:8080/jenkins/

# Jenkins

New Item People Build History Manage Jenkins Credentials

All

S	W	Name	Last Success	Last Failure	Last Duration
Grey	Yellow	Demo	N/A	N/A	N/A

ken: SML

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

Build Queue

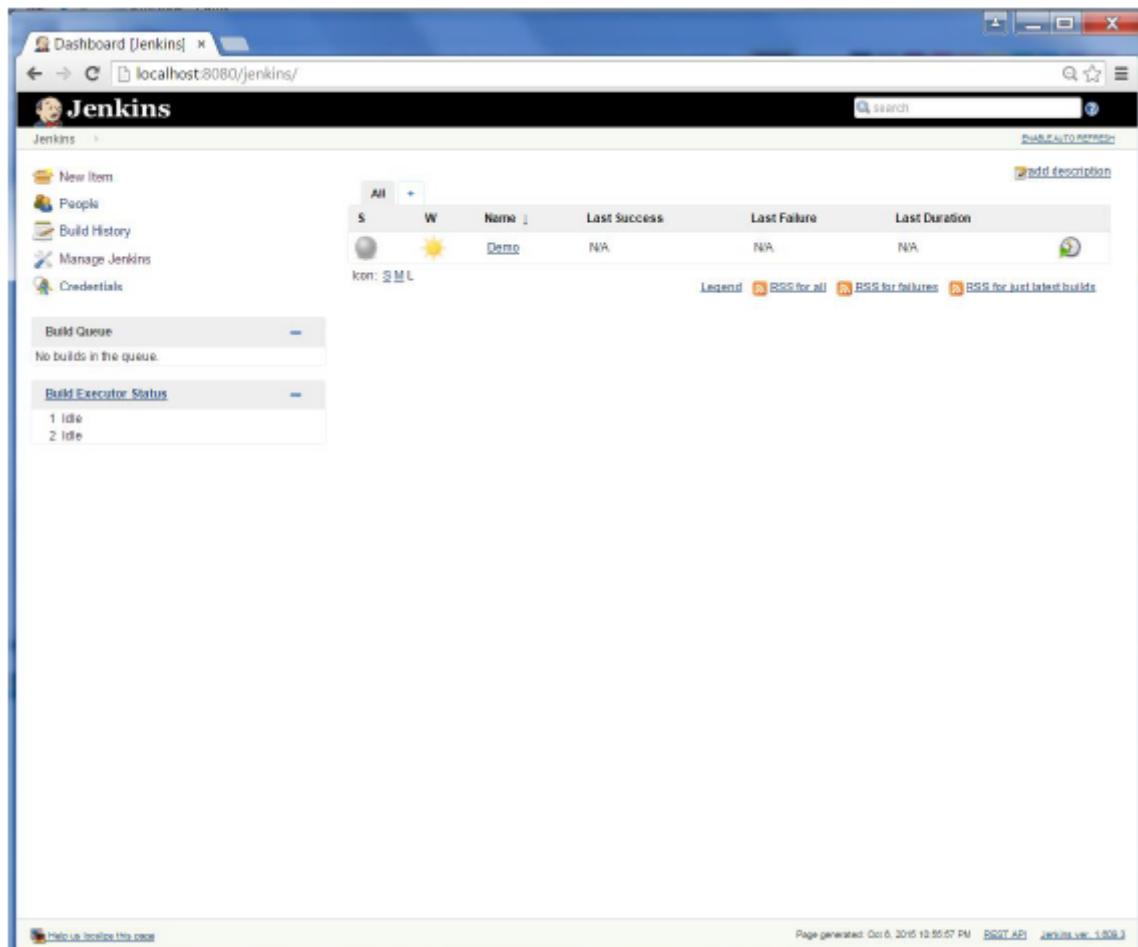
No builds in the queue.

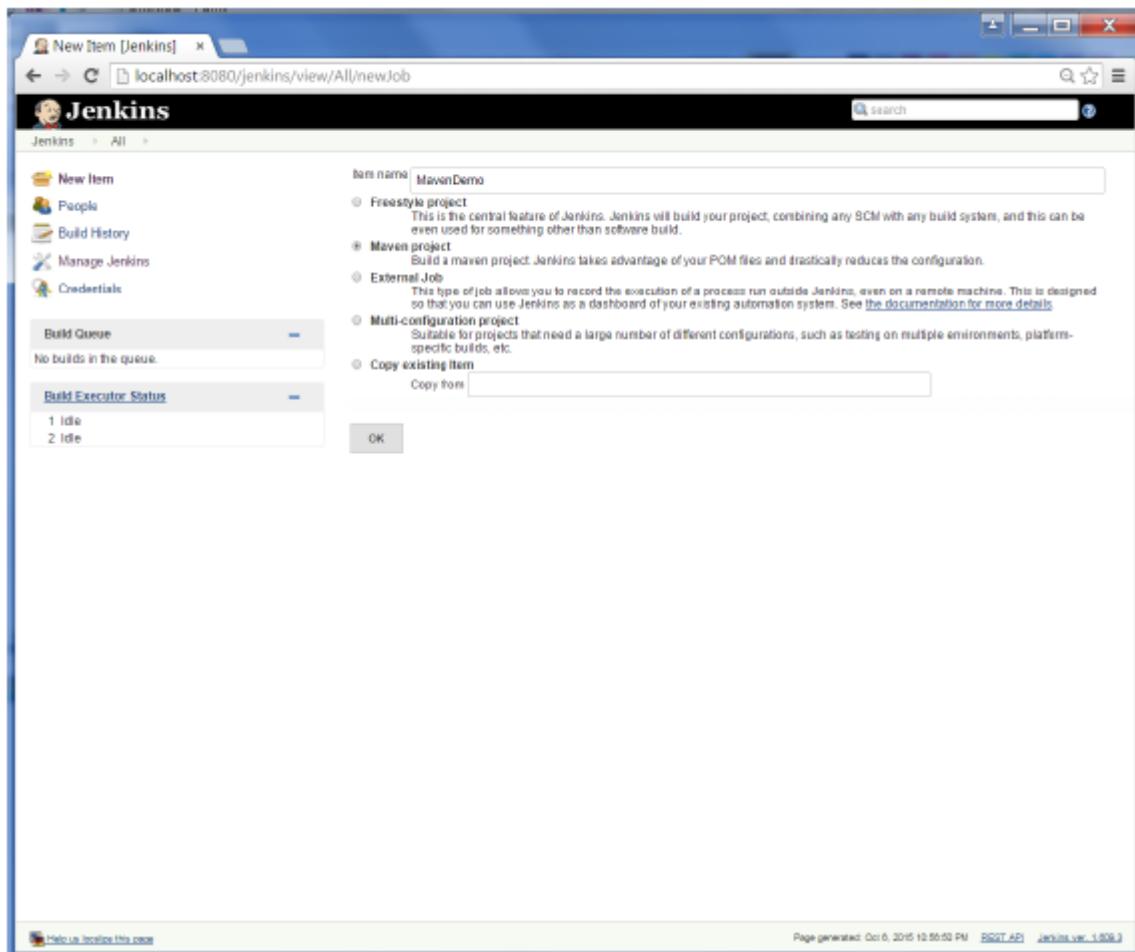
Build Executor Status

1 Idle  
2 Idle

Help us decide the case

Page generated: Oct 6, 2015 12:55:57 PM REST API Jenkins ver. 1.608.3

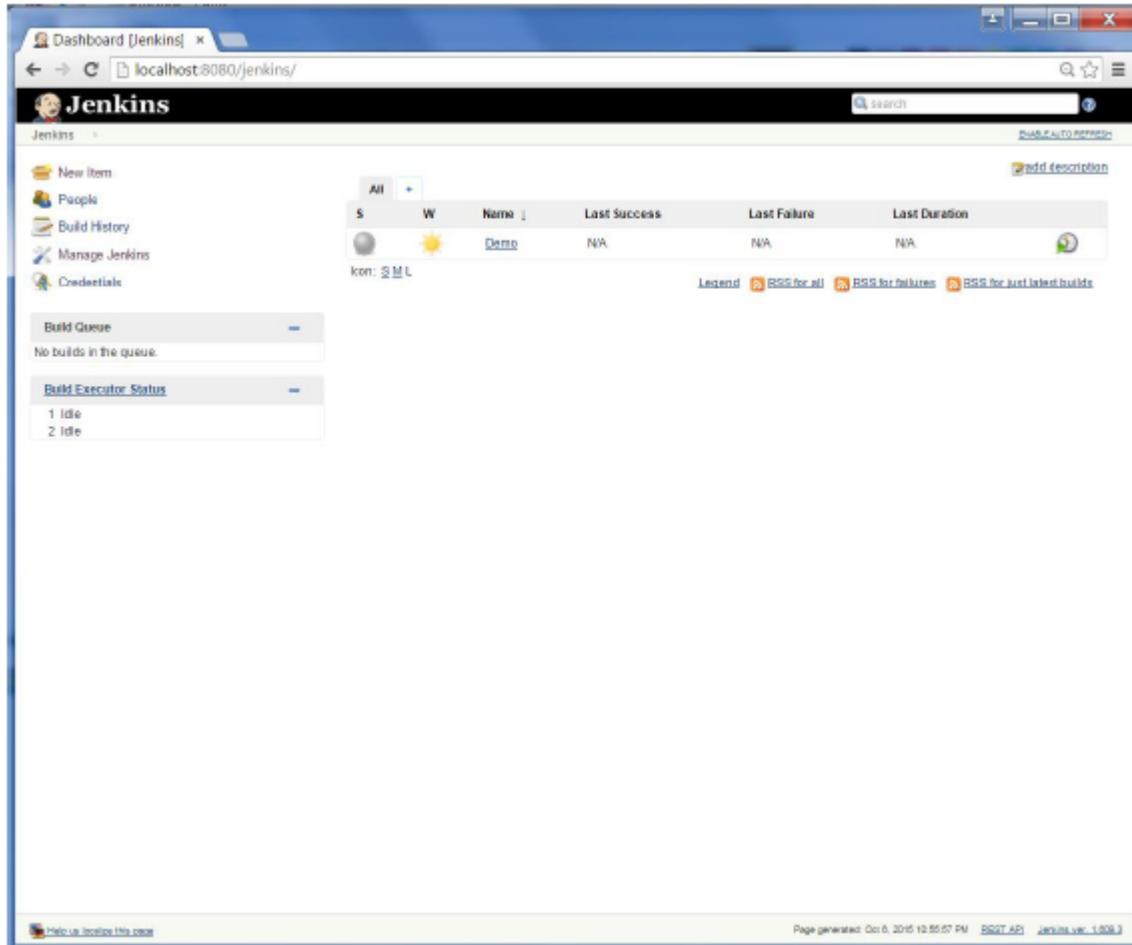




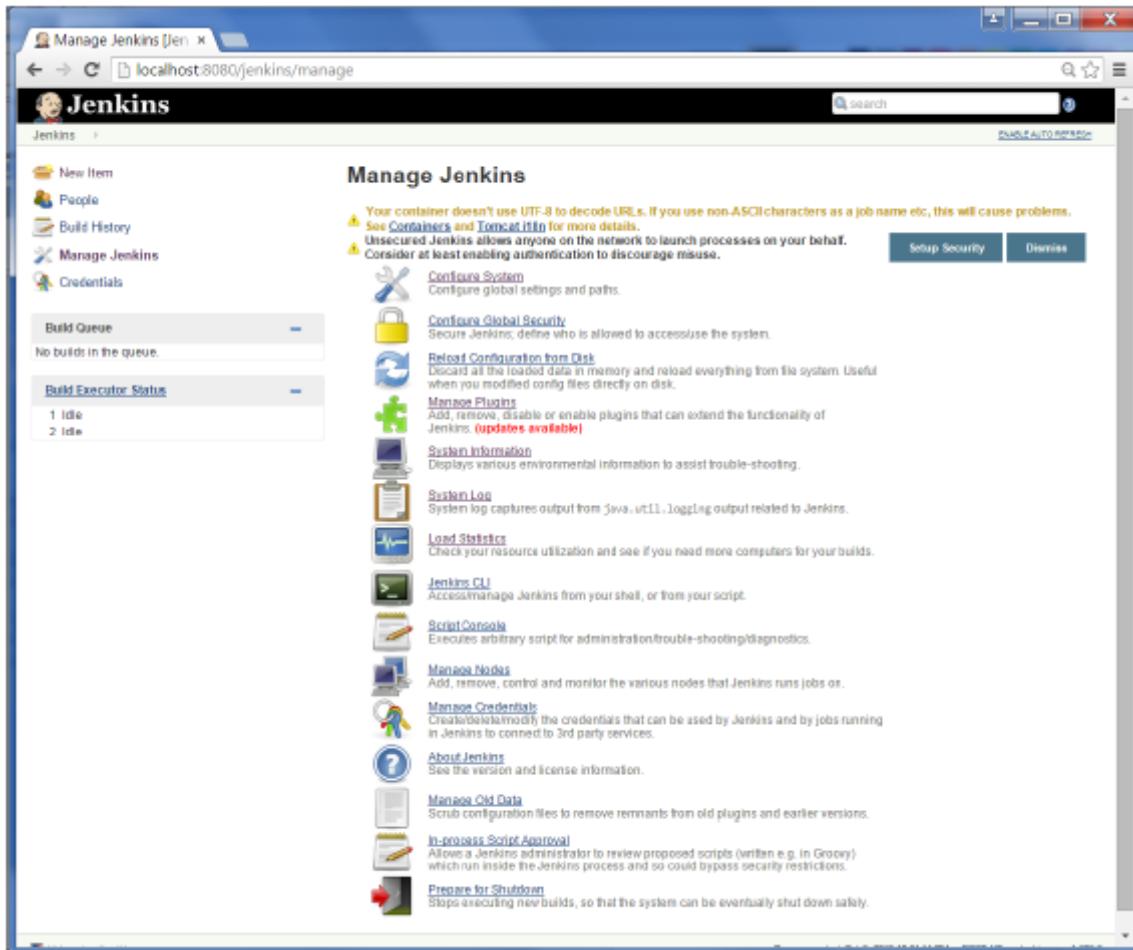
# Jenkins - Configuration

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.



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.

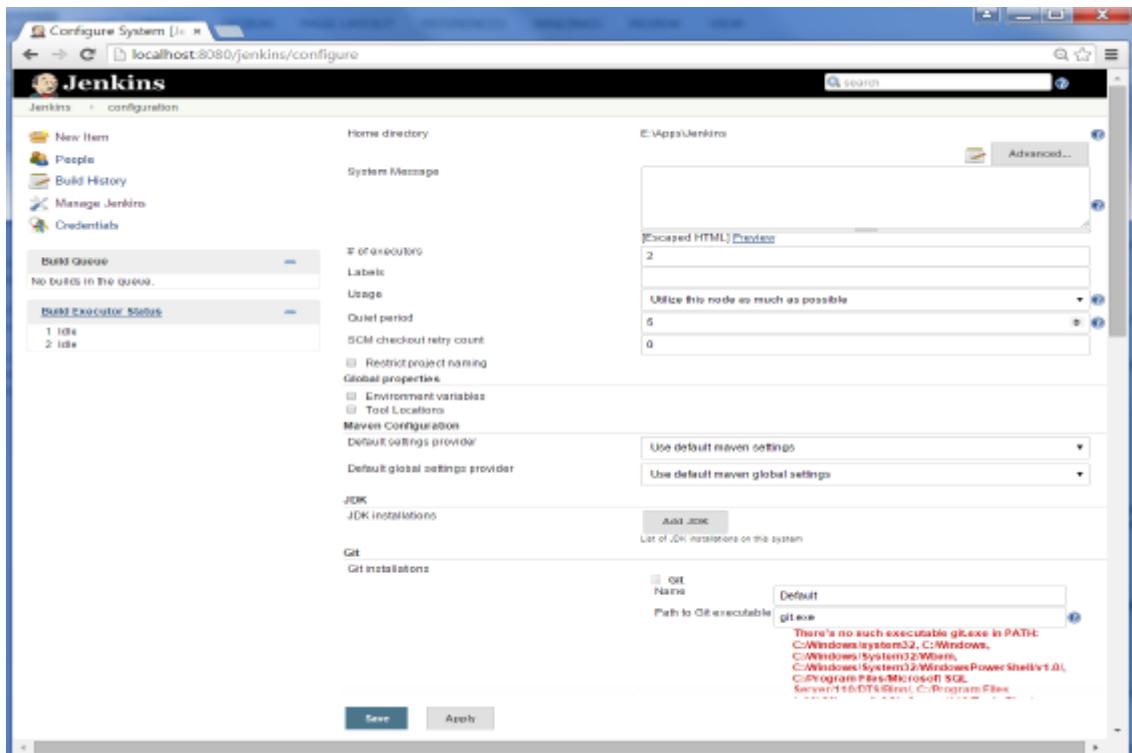
First create a new folder E:\Apps\Jenkins. Copy all the contents from the existing ~/.jenkins to this new directory.

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**

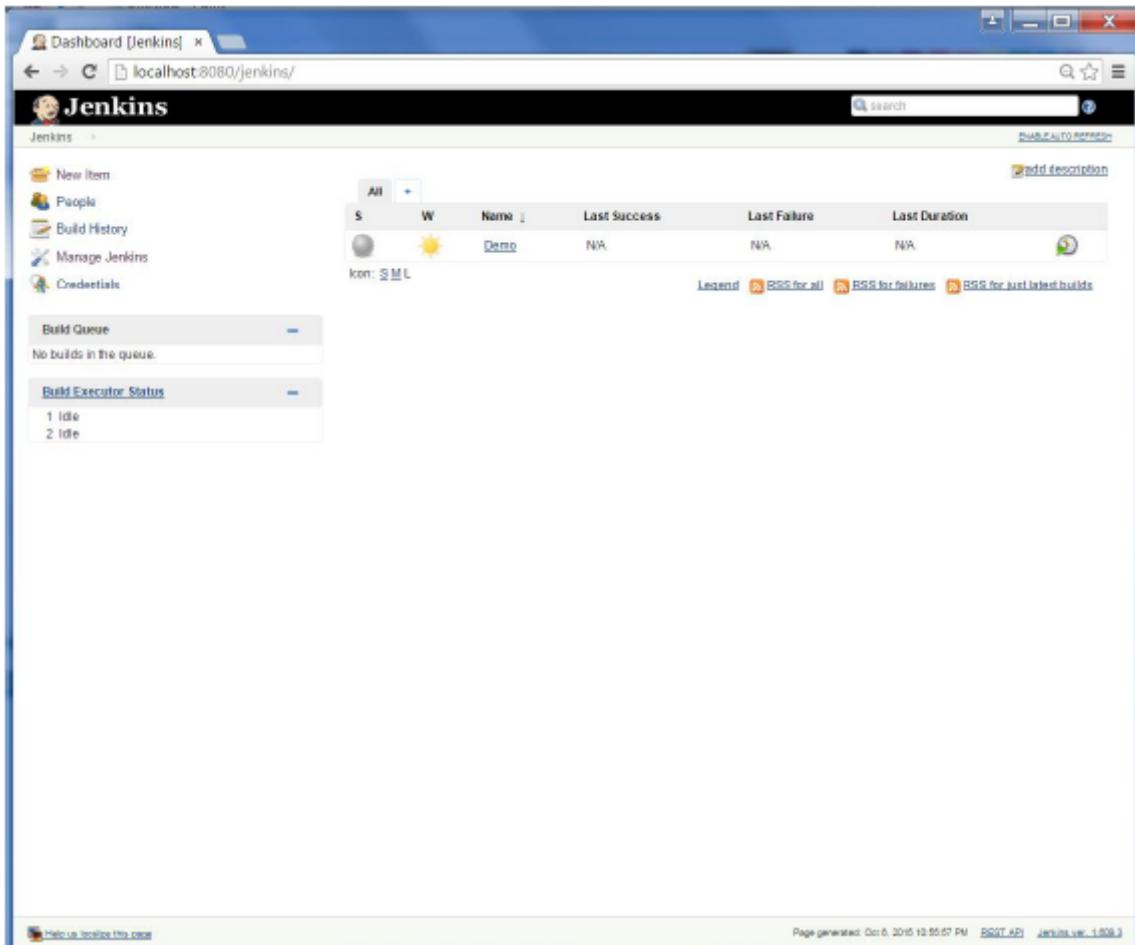
By default, the Jenkins URL points to localhost. If you have a domain name setup for your machine, set this to the domain name else overwrite localhost with IP of machine. This will help in setting up slaves and while sending out links using the email as you can directly access the Jenkins URL using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

## **Email Notification**

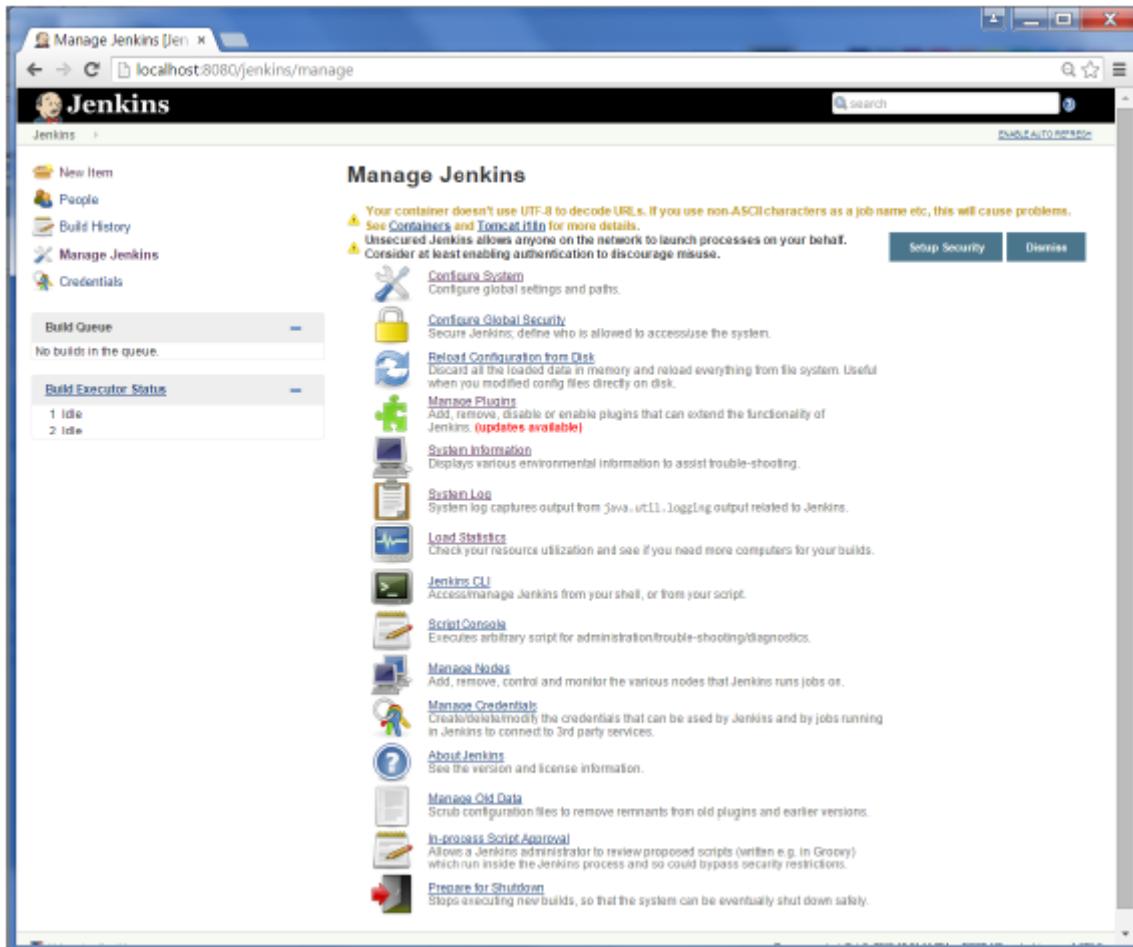
In the email Notification area, you can configure the SMTP settings for sending out emails. This is required for Jenkins to connect to the SMTP mail server and send out emails to the recipient list.

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

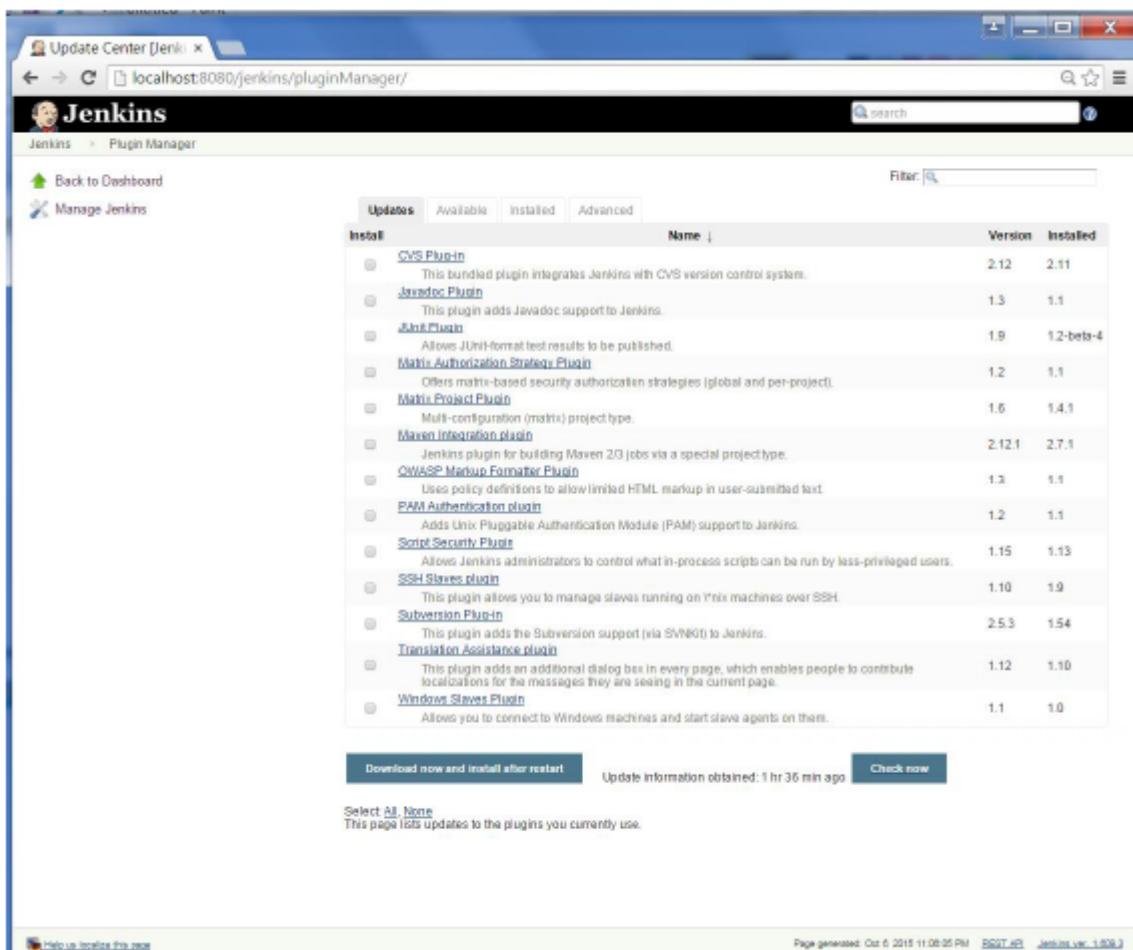
This is where one can manage paths to the various tools to use in builds, such as the JDKs, the versions of Ant and Maven, as well as security options, email servers, and other system-wide configuration details. When plugins are installed, Jenkins will add the required configuration fields dynamically 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 Update Center interface. The title bar says "Update Center Jenkins". The address bar shows "localhost:8080/jenkins/pluginManager/". The main header is "Jenkins > Plugin Manager". On the left, there are links for "Back to Dashboard" and "Manage Jenkins". The central area has a "Updates" tab selected, with other tabs for "Available", "Installed", and "Advanced". A "Filter" input field is at the top right. Below is a table of available plugins:

Name	Version	Installed
<a href="#">CVS Plugin</a> This bundled plugin integrates Jenkins with CVS version control system.	2.12	2.11
<a href="#">Javadoc Plugin</a> This plugin adds Javadoc support to Jenkins.	1.3	1.1
<a href="#">JUnit Plugin</a> Allows JUnit-format test results to be published.	1.9	1.2-beta-4
<a href="#">Matrix Authorization Strategy Plugin</a> Offers matrix-based security authorization strategies (global and per-project).	1.2	1.1
<a href="#">Matrix Project Plugin</a> Multi-configuration (matrix) project type.	1.6	1.4.1
<a href="#">Maven Integration plugin</a> Jenkins plugin for building Maven 2.0 jobs via a special project type.	2.12.1	2.7.1
<a href="#">OWASP Markup Formatter Plugin</a> Uses policy definitions to allow limited HTML markup in user-submitted text.	1.3	1.1
<a href="#">PAM Authentication plugin</a> Adds Unix Pluggable Authentication Module (PAM) support to Jenkins.	1.2	1.1
<a href="#">Script Security Plugin</a> Allows Jenkins administrators to control what in-process scripts can be run by less-privileged users.	1.15	1.13
<a href="#">SSH Slave plugin</a> This plugin allows you to manage slaves running on Unix machines over SSH.	1.10	1.9
<a href="#">Subversion Plugin</a> This plugin adds the Subversion support (via SVNKit) to Jenkins.	2.5.3	1.54
<a href="#">Translation Assistance plugin</a> This plugin adds an additional dialog box in every page, which enables people to contribute localizations for the messages they are seeing in the current page.	1.12	1.10
<a href="#">Windows Slave Plugin</a> Allows you to connect to Windows machines and start slave agents on them.	1.1	1.0

At the bottom, there are buttons for "Download now and install after restart", "Check now", and a status message "Update information obtained: 1 hr 36 min ago". A note says "Select All None" and "This page lists updates to the plugins you currently use." The footer includes links for "Help Us Improve This Page", "Page generated: Oct 6, 2015 11:08:25 PM", "RS2748", and "Jenkins ver. 1.605".

# System Information

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

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

System Information   x																																																																
localhost:8080/jenkins/systemInfo																																																																
 Jenkins <span style="float: right;">search</span>																																																																
Jenkins >																																																																
 New Item																																																																
 People																																																																
 Build History																																																																
 Manage Jenkins																																																																
 Credentials																																																																
<b>Build Queue</b>	<b>No builds in the queue.</b>																																																															
<b>Build Executor Status</b>	<b>1 Idle</b> 2 Idle																																																															
<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>awt.toolkit</td> <td>sun.awt.windows.WToolkit</td></tr> <tr> <td>catalina.base</td> <td>E:\App\tomcat7</td></tr> <tr> <td>catalina.home</td> <td>E:\App\tomcat7</td></tr> <tr> <td>catalina.useNaming</td> <td>true</td></tr> <tr> <td>common.loader</td> <td>\$catalina.base/lib/\$catalina.base/lib/*jar;\$catalina.home/lib/\$catalina.home/lib/*jar</td></tr> <tr> <td>file.encoding</td> <td>CP1252</td></tr> <tr> <td>file.encoding.pkg</td> <td>sun.io</td></tr> <tr> <td>file.separator</td> <td>\</td></tr> <tr> <td>java.awt.graphicsenv</td> <td>sun.awt.Win32GraphicsEnvironment</td></tr> <tr> <td>java.awt.printerjob</td> <td>sun.awt.windows.WPrinterJob</td></tr> <tr> <td>java.class.path</td> <td>E:\App\tomcat7\bin\bootstrap.jar;E:\App\tomcat7\bin\tomcat-juli.jar</td></tr> <tr> <td>java.class.version</td> <td>51.0</td></tr> <tr> <td>java.endorsed.dirs</td> <td>E:\App\tomcat7\endorsed</td></tr> <tr> <td>java.ext.dirs</td> <td>C:\Program Files (x86)\Java\jdk1.7.0_79\jre\lib\ext;C:\Windows\Sun\Java\lib\ext</td></tr> <tr> <td>java.home</td> <td>C:\Program Files (x86)\Java\jdk1.7.0_79\jre</td></tr> <tr> <td>java.io.tmpdir</td> <td>E:\App\tomcat7\temp</td></tr> <tr> <td>java.library.path</td> <td>C:\Program Files (x86)\Java\jdk1.7.0_79\bin;C:\Windows\Sun\Java\bin;C:\Windows\System32;C:\Windows;C:\Windows\system32;C:\Windows;C:\Windows\System32;C:\Windows\system32\Windows\PowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Program Files (x86)\Microsoft Visual Studio\10.0\Common7\IDE\PrivateAssemblies\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Windows\Performance Toolkit\;C:\Program Files\100\Microsoft TypeScript\1.0\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files\Microsoft\Windows Installer\;C:\Program Files\86\Java\jdk1.7.0_79\bin;%MAVEN_HOME%\bin..</td></tr> <tr> <td>java.naming.factory.initial</td> <td>org.apache.naming.java.javaURLContextFactory</td></tr> <tr> <td>java.naming.factory.url.pkgs</td> <td>org.apache.naming</td></tr> <tr> <td>java.runtime.name</td> <td>Java(TM) SE Runtime Environment</td></tr> <tr> <td>java.runtime.version</td> <td>1.7.0_79-b15</td></tr> <tr> <td>java.specification.name</td> <td>Java Platform API Specification</td></tr> <tr> <td>java.specification.vendor</td> <td>Oracle Corporation</td></tr> <tr> <td>java.specification.version</td> <td>1.7</td></tr> <tr> <td>java.util.logging.config.file</td> <td>E:\App\tomcat7\conf\logging.properties</td></tr> <tr> <td>java.util.logging.manager</td> <td>org.apache.juli.ClassLoaderLogManager</td></tr> <tr> <td>java.vendor</td> <td>Oracle Corporation</td></tr> <tr> <td>java.vendor.url</td> <td>http://java.oracle.com/</td></tr> <tr> <td>java.vendor.url_bugurl</td> <td>http://bugreport.sun.com/bugreport/</td></tr> <tr> <td>java.version</td> <td>1.7.0_79</td></tr> <tr> <td>java.vm.info</td> <td>mixed mode; sharing</td></tr> </tbody> </table>	Name	Value	awt.toolkit	sun.awt.windows.WToolkit	catalina.base	E:\App\tomcat7	catalina.home	E:\App\tomcat7	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.io	file.separator	\	java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment	java.awt.printerjob	sun.awt.windows.WPrinterJob	java.class.path	E:\App\tomcat7\bin\bootstrap.jar;E:\App\tomcat7\bin\tomcat-juli.jar	java.class.version	51.0	java.endorsed.dirs	E:\App\tomcat7\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:\App\tomcat7\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;C:\Windows;C:\Windows\System32;C:\Windows\system32\Windows\PowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Program Files (x86)\Microsoft Visual Studio\10.0\Common7\IDE\PrivateAssemblies\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Windows\Performance Toolkit\;C:\Program Files\100\Microsoft TypeScript\1.0\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files\Microsoft\Windows Installer\;C:\Program Files\86\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:\App\tomcat7\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_bugurl	http://bugreport.sun.com/bugreport/	java.version	1.7.0_79	java.vm.info	mixed mode; sharing
Name	Value																																																															
awt.toolkit	sun.awt.windows.WToolkit																																																															
catalina.base	E:\App\tomcat7																																																															
catalina.home	E:\App\tomcat7																																																															
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.io																																																															
file.separator	\																																																															
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment																																																															
java.awt.printerjob	sun.awt.windows.WPrinterJob																																																															
java.class.path	E:\App\tomcat7\bin\bootstrap.jar;E:\App\tomcat7\bin\tomcat-juli.jar																																																															
java.class.version	51.0																																																															
java.endorsed.dirs	E:\App\tomcat7\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:\App\tomcat7\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;C:\Windows;C:\Windows\System32;C:\Windows\system32\Windows\PowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Program Files (x86)\Microsoft Visual Studio\10.0\Common7\IDE\PrivateAssemblies\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\ManagementStudio\;C:\Windows\Performance Toolkit\;C:\Program Files\100\Microsoft TypeScript\1.0\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files\Microsoft\Windows Installer\;C:\Program Files\86\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:\App\tomcat7\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_bugurl	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.

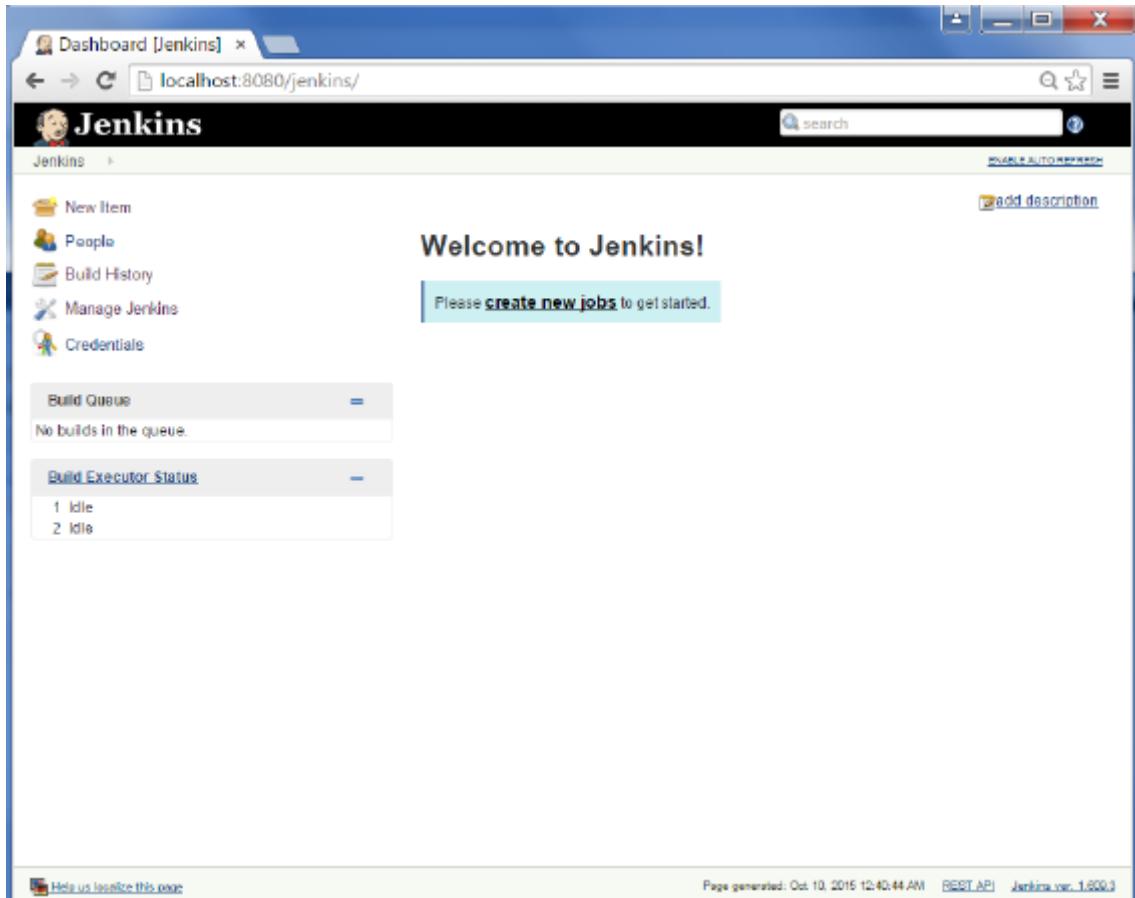
## Prepare for Shutdown

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.

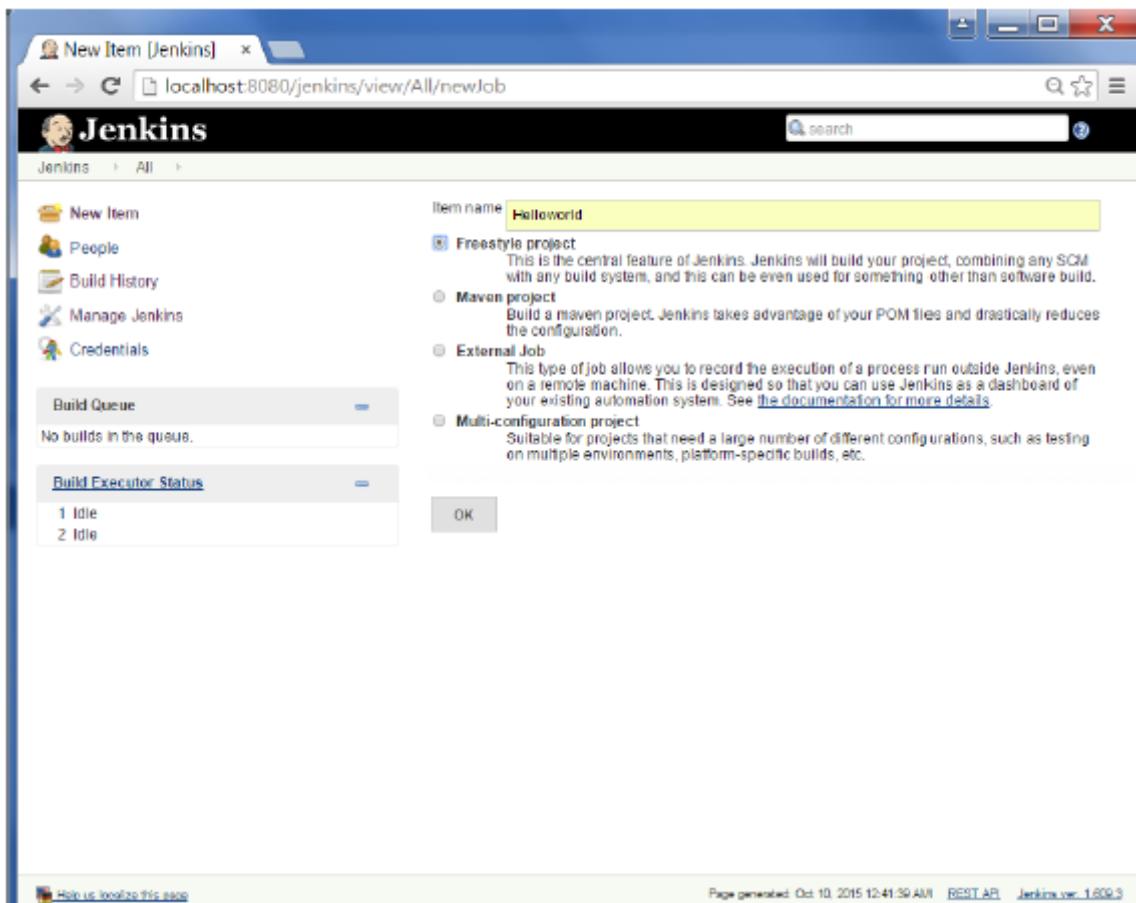
# Jenkins - Setup Build Jobs

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

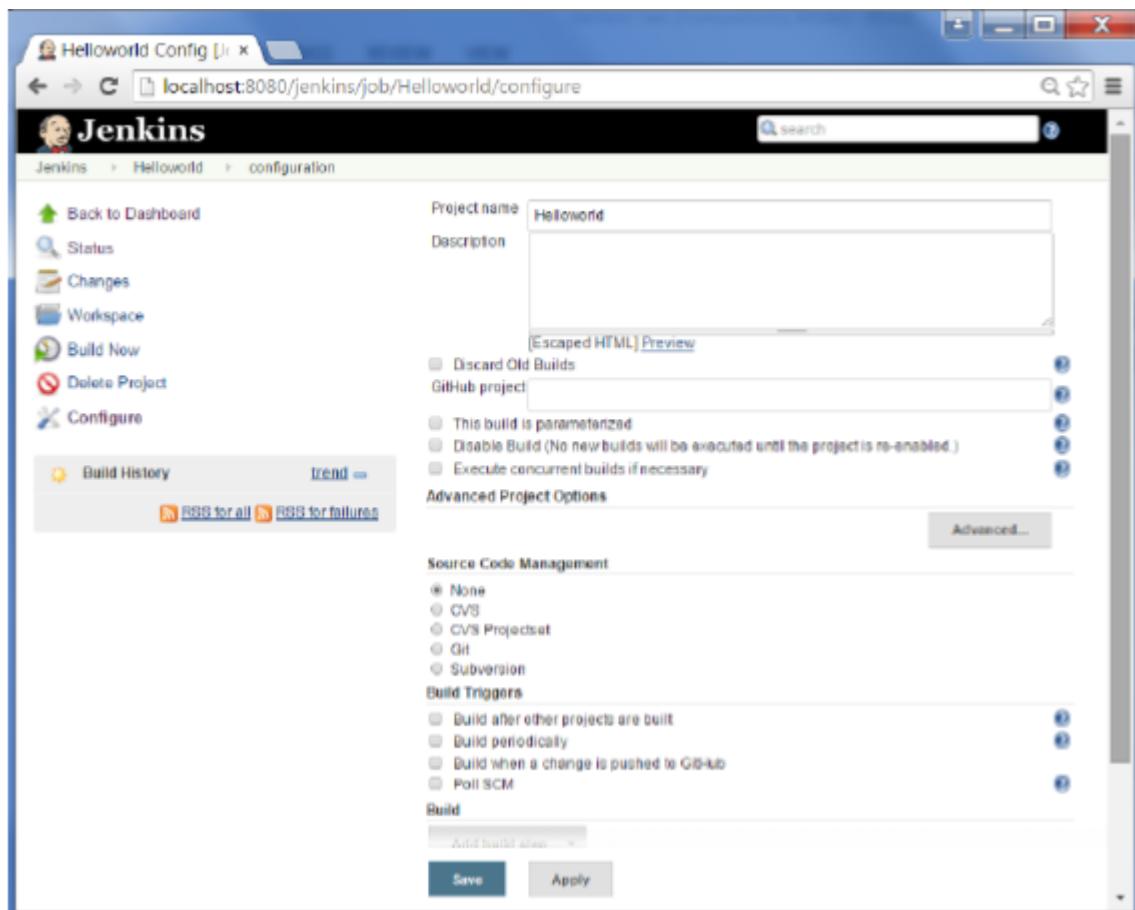
**Step 1** – Go to the Jenkins dashboard and Click on New Item



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

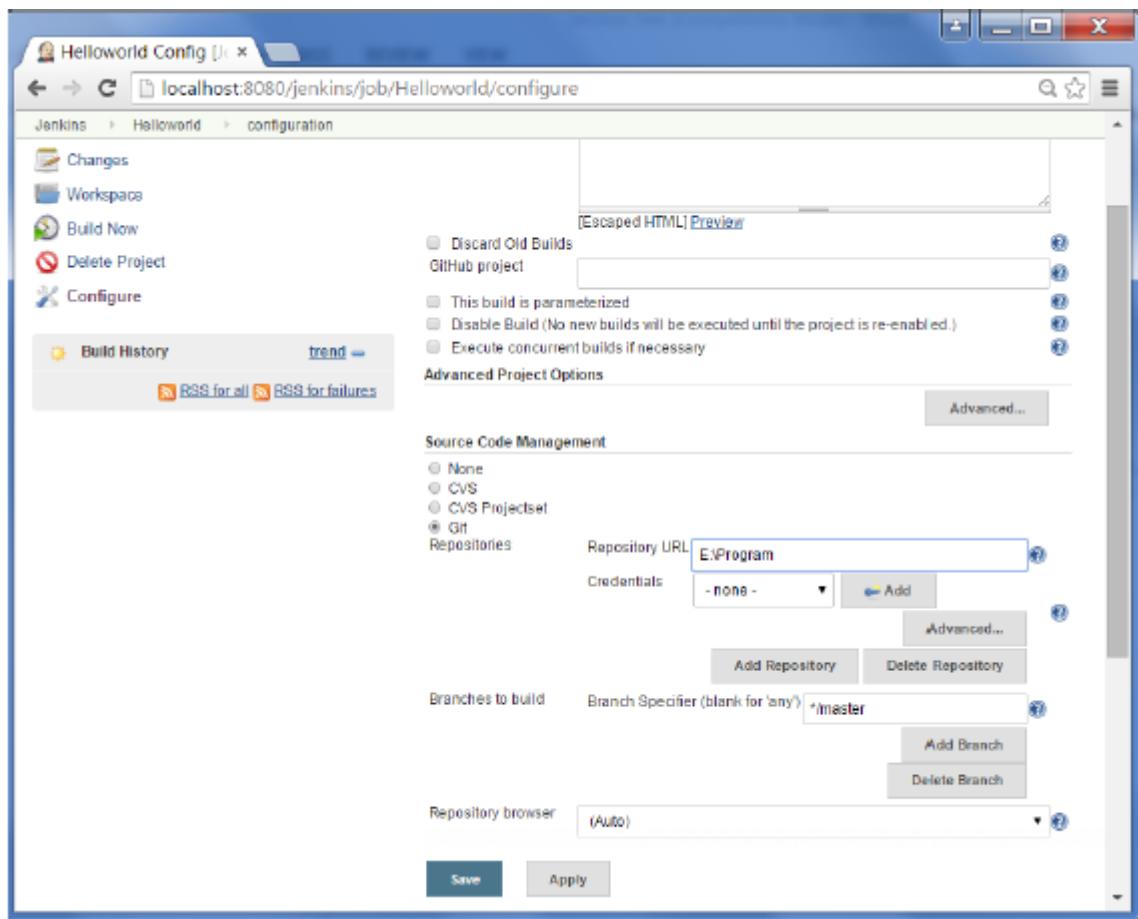


**Step 3** – The following screen will come up in which you can specify the details of the job.

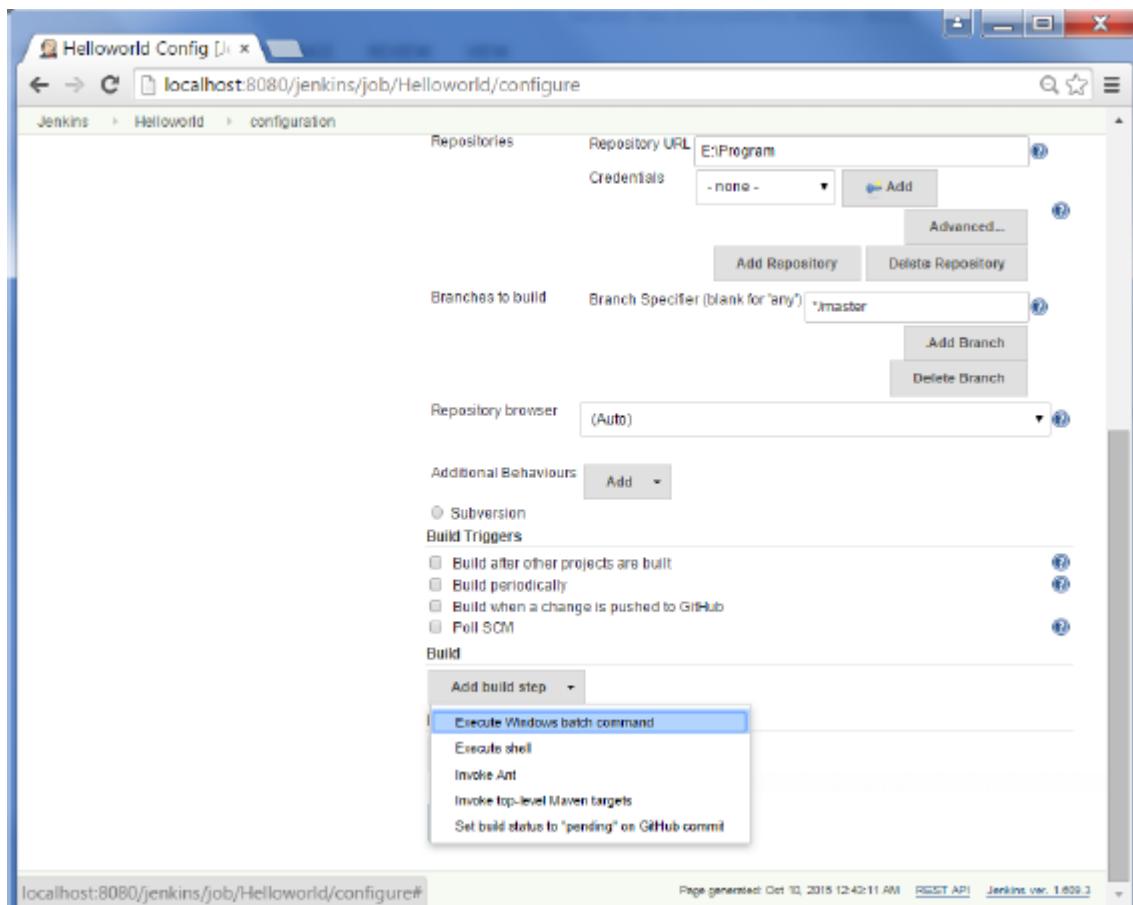


**Step 4** – We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a 'HelloWorld.java' file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

**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.

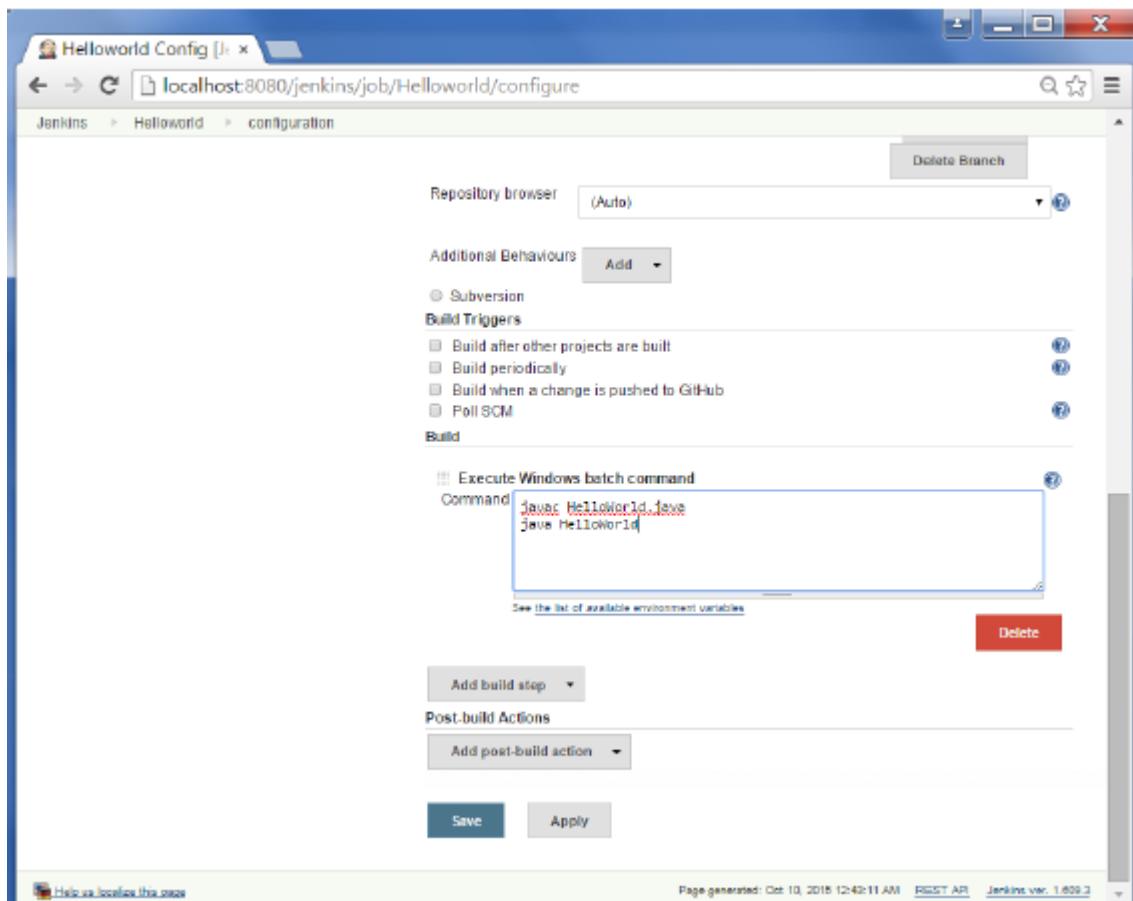


**Step 5** – Now go to the Build section and click on Add build step → Execute Windows batch command

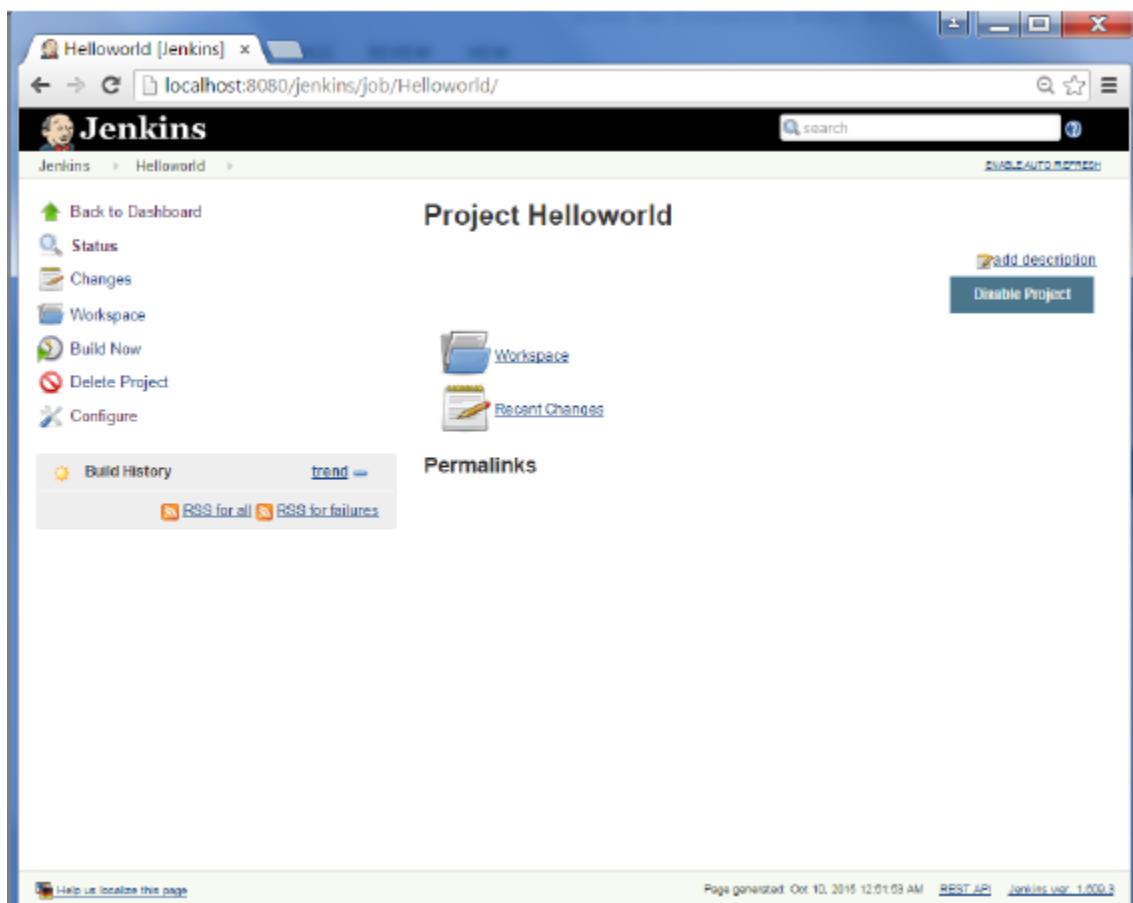


**Step 6** – In the command window, enter the following commands and then click on the Save button.

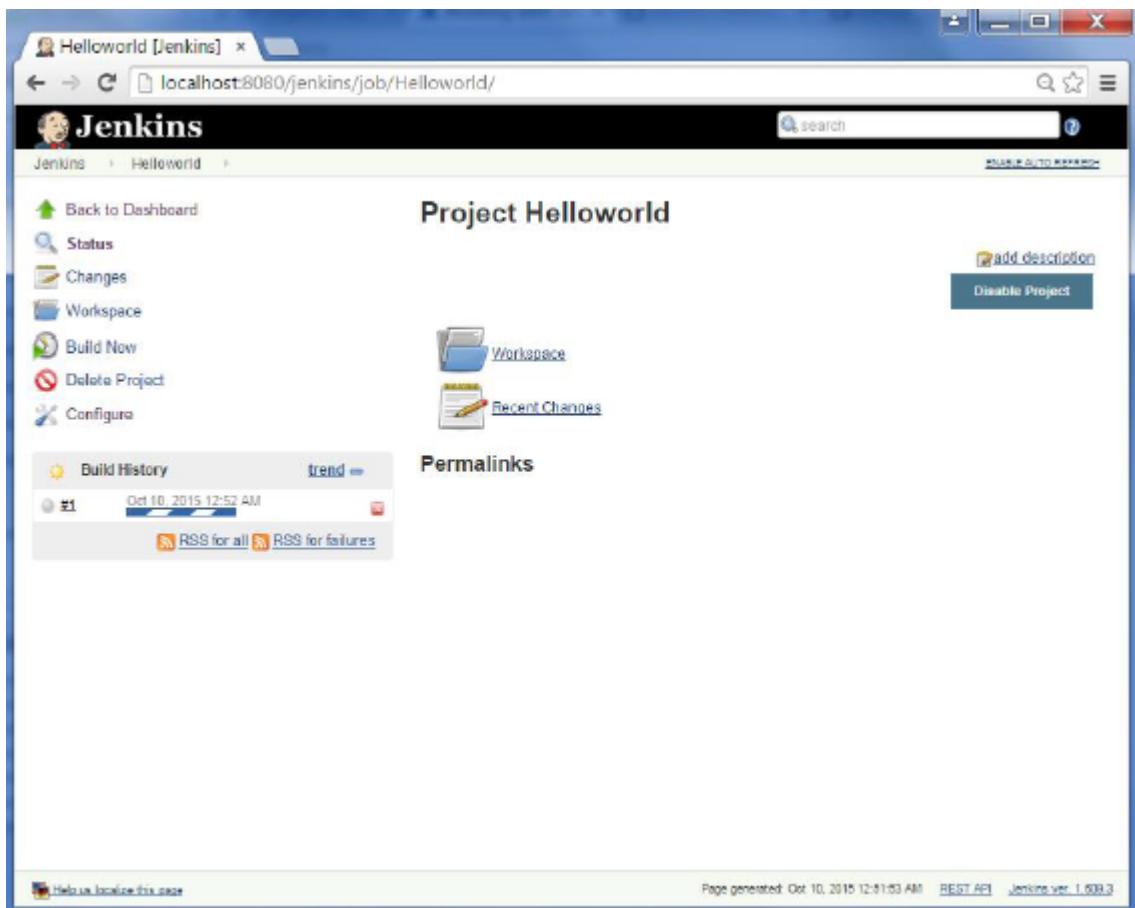
```
Javac HelloWorld.java
Java HelloWorld
```



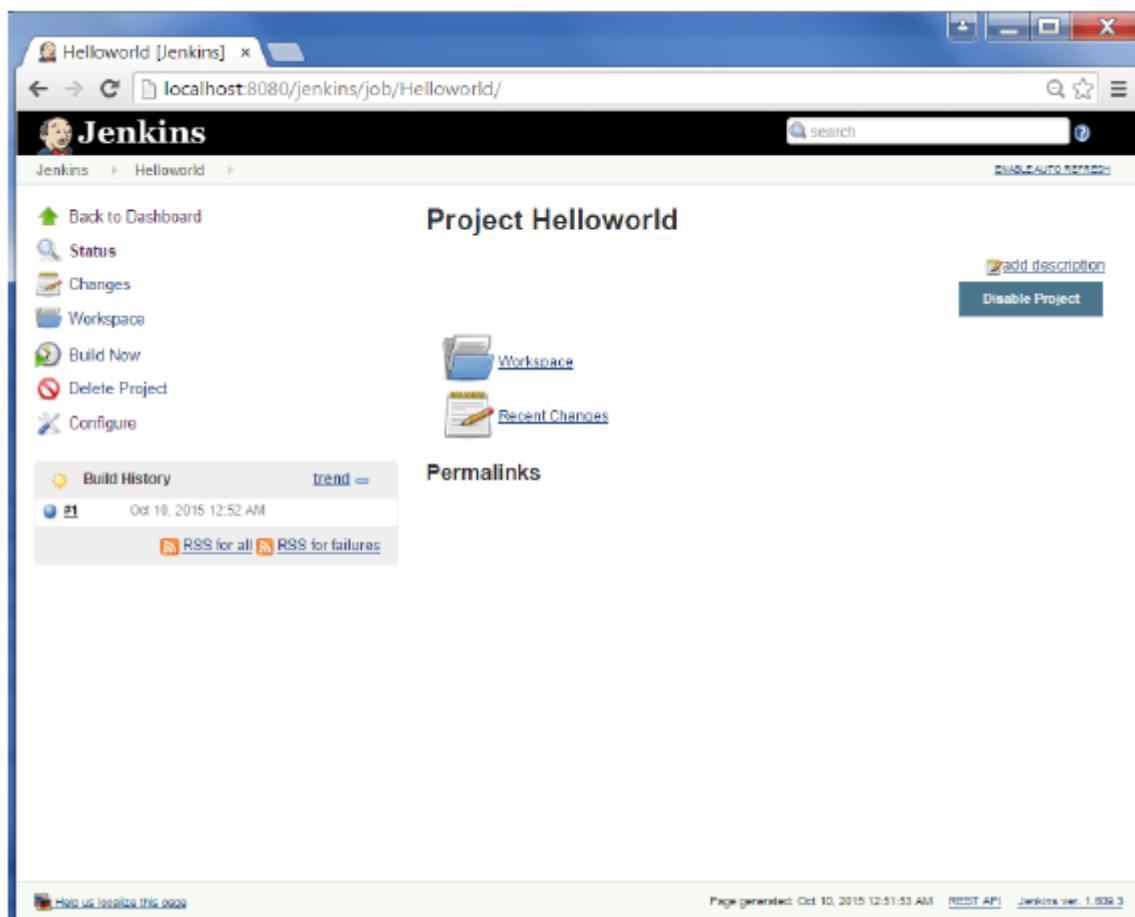
**Step 7** – Once saved, you can click on the Build Now option to see if you have successfully defined the job.



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



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



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

localhost:8080/jenkins/job/Helloworld/1/

## Jenkins

Build #1 (Oct 10, 2015 12:52:50 AM)

Started 4 min 40 sec ago  
Took 4.7 sec

No changes.

Started by anonymous user

Revision: 42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f  
• ref/remotes/origin/master

Page generated: Oct 10, 2015 12:57:31 AM RESTART Jenkins ver. 1.609.3

localhost:8080/jenkins/job/Helloworld/12/console

## Jenkins

### Console Output

```
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/master"(commit) # timeout=10
Checking out Revision 42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
(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
42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
> C:\Program Files\Git\bin\git.exe rev-list
42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f # timeout=10
[workspace] $ cmd /c call E:\Appsvtomcat7\temp\hudson1928478766077504681.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
```

Page generated: Oct 10, 2015 10:14:21 PM RESTART Jenkins ver. 1.609.3

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

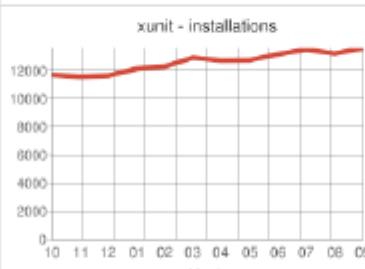
<https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin>

Dashboard > Jenkins > Plugins > xUnit Plugin

**xUnit Plugin**

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

**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	2014-Oct 11592 2014-Nov 11557 2014-Dec 11631 2015-Jan 12106 2015-Feb 12262 2015-Mar 12891 2015-Apr 12694 2015-May 12716 2015-Jun 13143 2015-Jul 13470 2015-Aug 13192 2015-Sep 13563

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

**CppUnit output**

The screenshot shows a web browser window with the title 'xUnit Plugin - Jenkins'. The URL in the address bar is <https://wiki.jenkins-ci.org/display/JENKINS/xUnit+Plugin>. The page content is as follows:

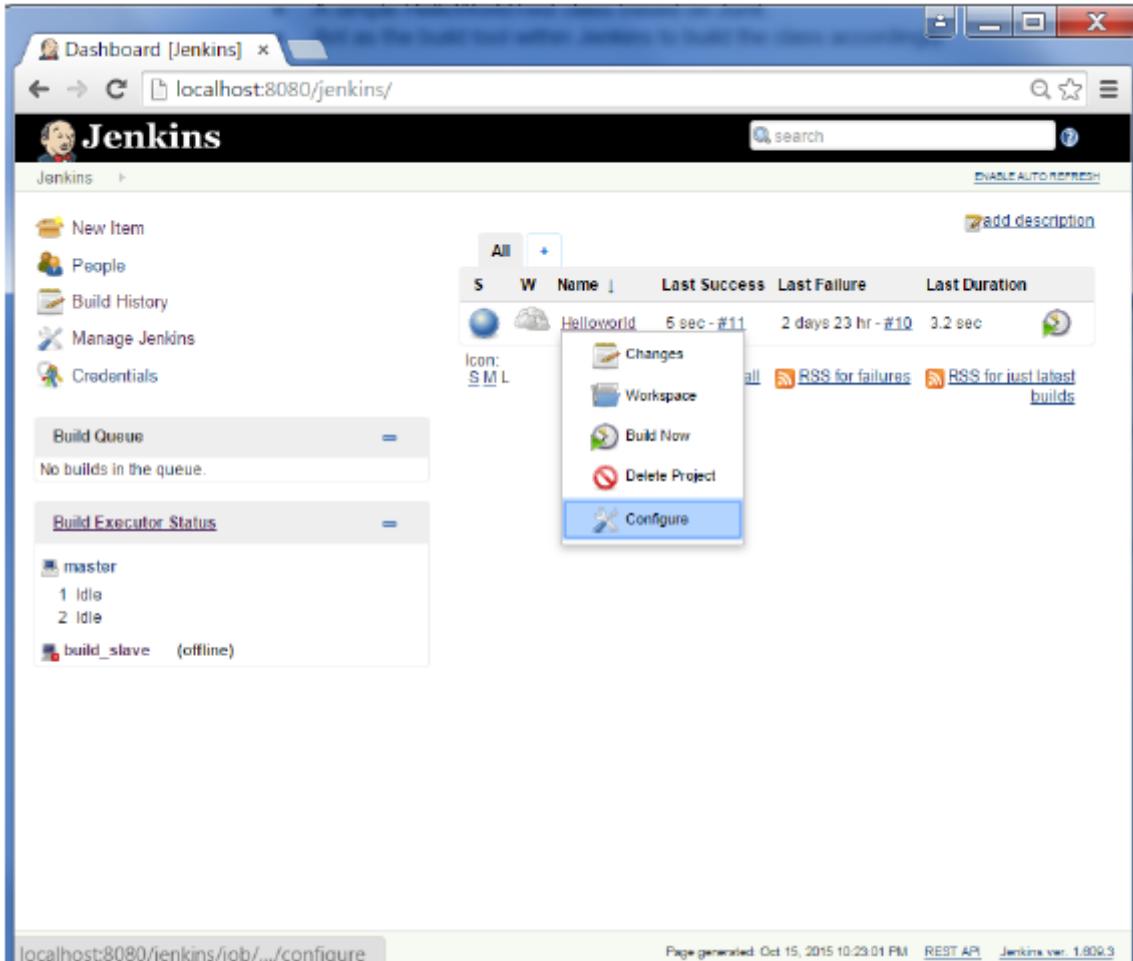
- Features**
  - Records xUnit tests
  - Mark the build unstable or fail according to threshold values
- Supported tools**
  - JUnit itself
  - [JUnit](#)
  - [MSTest \(imported from MSTest Plugin\)](#)
  - [NUnit \(imported from NUnit Plugin\)](#)
  - [UnitTests++](#)
  - [Boost Test Library](#)
  - [PHPUnit](#)
  - [Free Pascal Unit](#)
  - [CppUnit](#)
  - [MUnit](#)
  - [GoogleTest](#)
  - [EmUnit](#)
  - [gtester/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

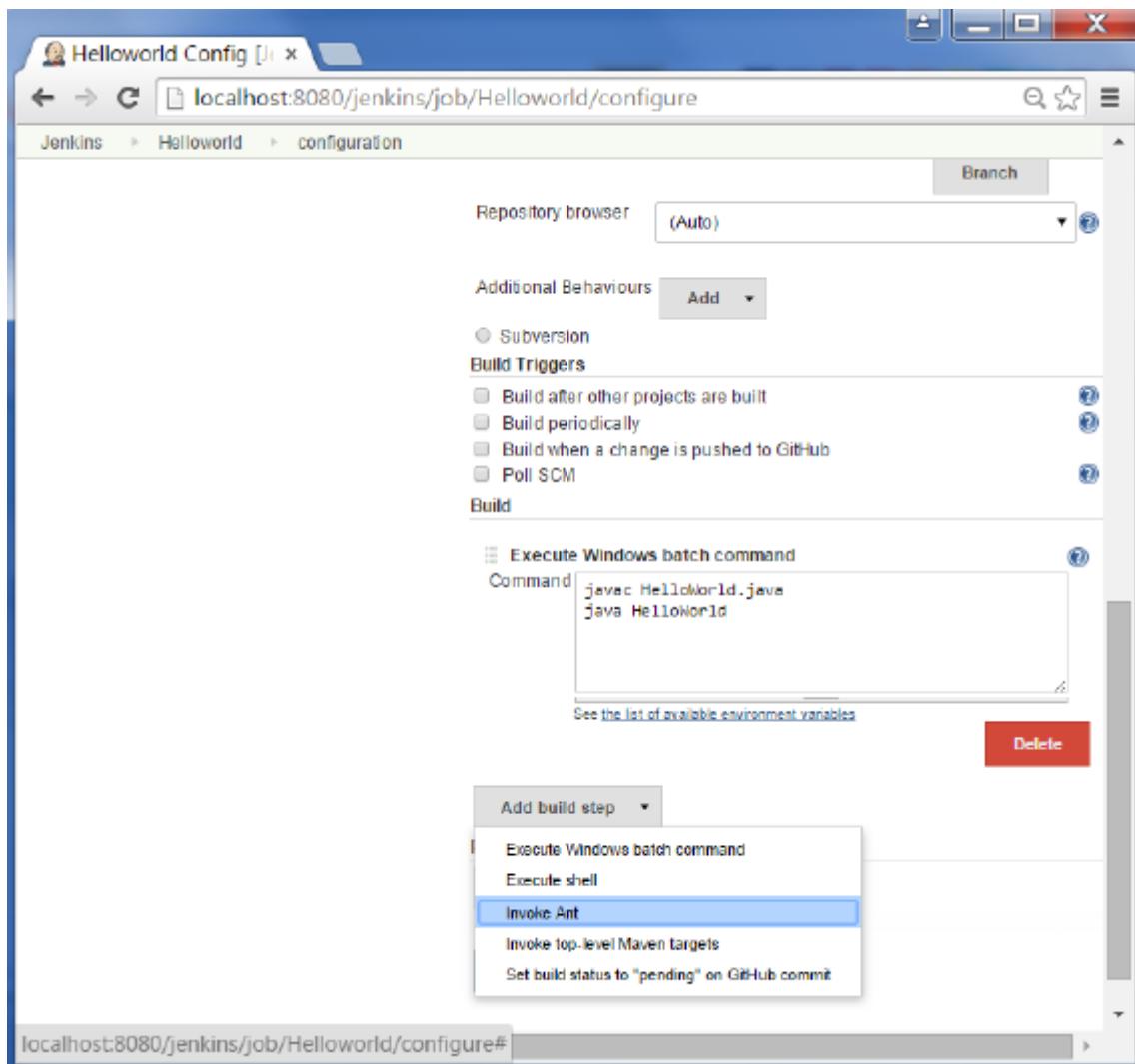
- A simple HelloWorldTest class based on Junit.
- Ant as the build tool within Jenkins to build the class accordingly.

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

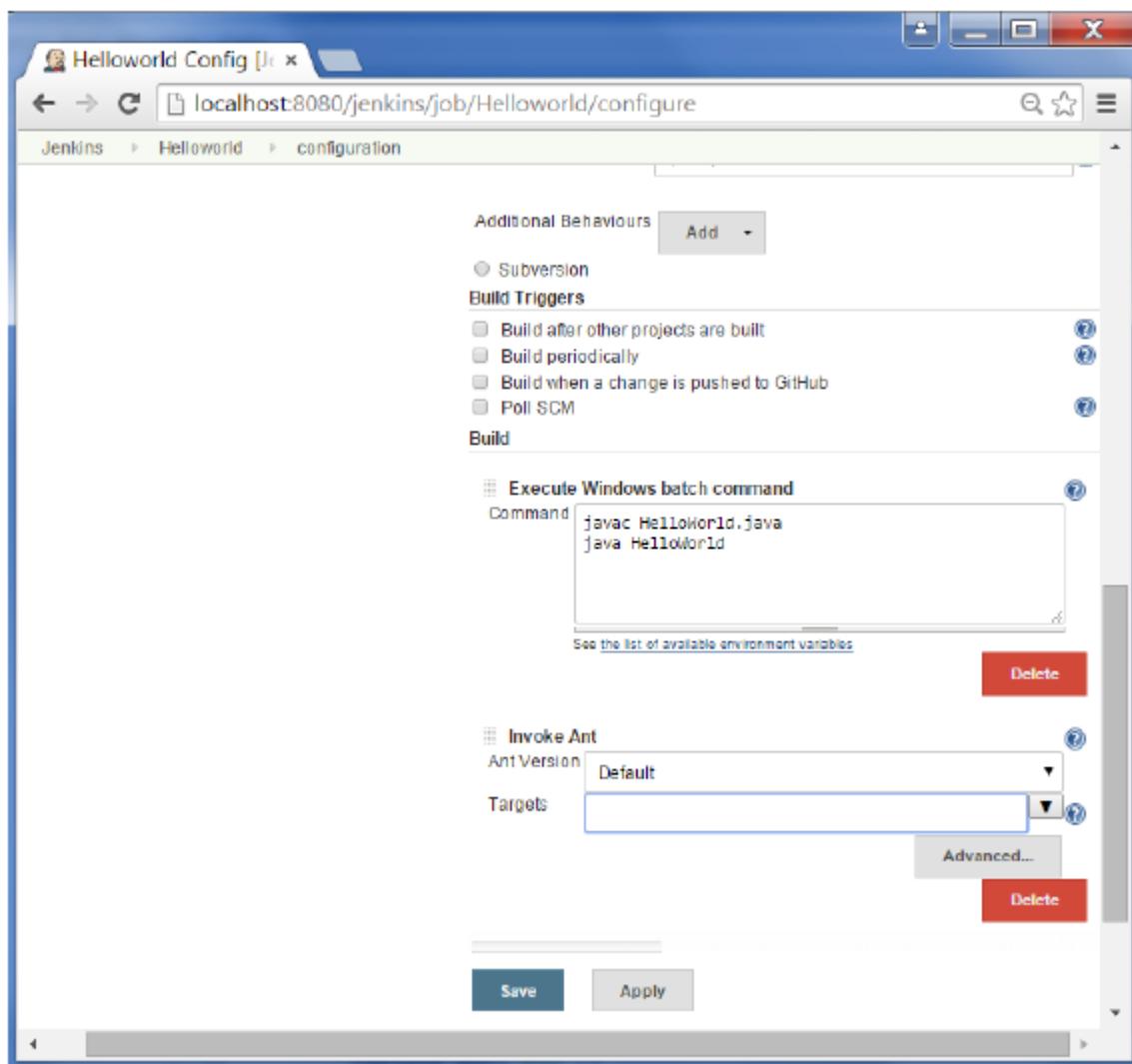


The screenshot shows the Jenkins Dashboard at [localhost:8080/jenkins/](http://localhost:8080/jenkins/). The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. The main area displays a table of projects with columns for S, W, Name, Last Success, Last Failure, and Last Duration. The 'HelloWorld' project is listed with a success status (6 sec - #11), a failure status (2 days 23 hr - #10), and a duration of 3.2 sec. Below the table are links for Changes, Workspace, Build Now, Delete Project, and Configure. The 'Configure' link is highlighted with a blue box. The bottom status bar shows the URL [localhost:8080/jenkins/job/.../configure](http://localhost:8080/jenkins/job/.../configure), the page generation time (Oct 15, 2015 10:23:01 PM), and Jenkins version (1.809.3).

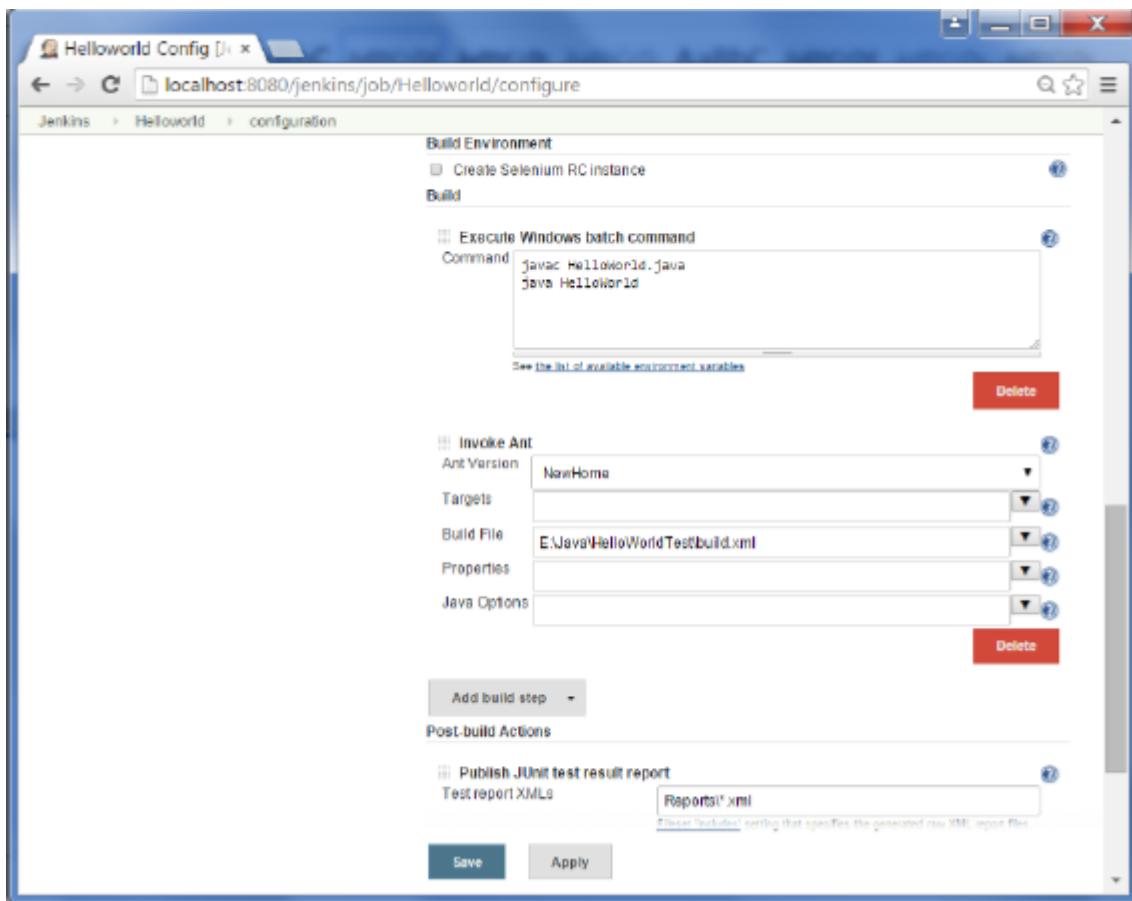
**Step 2** – Browse to the section to Add a Build step and choose the option to Invoke Ant.



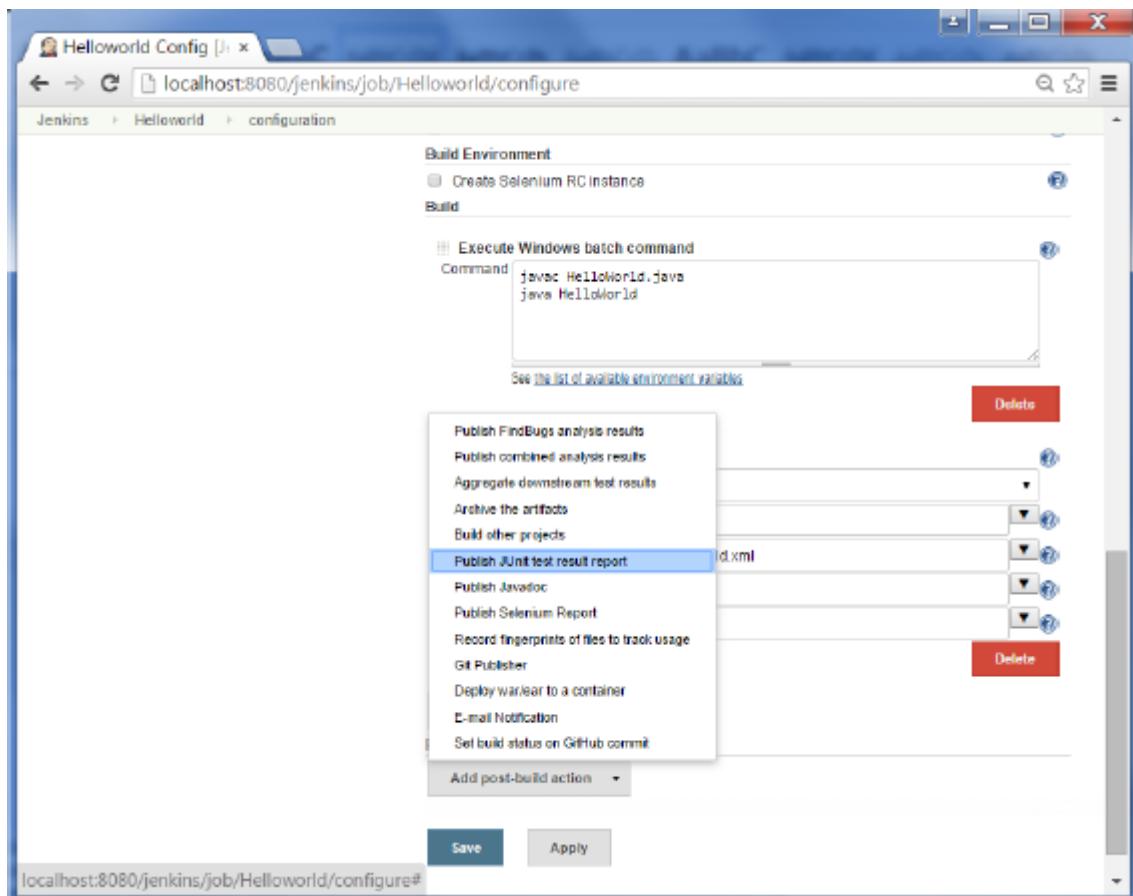
**Step 3** – Click on the Advanced button.



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

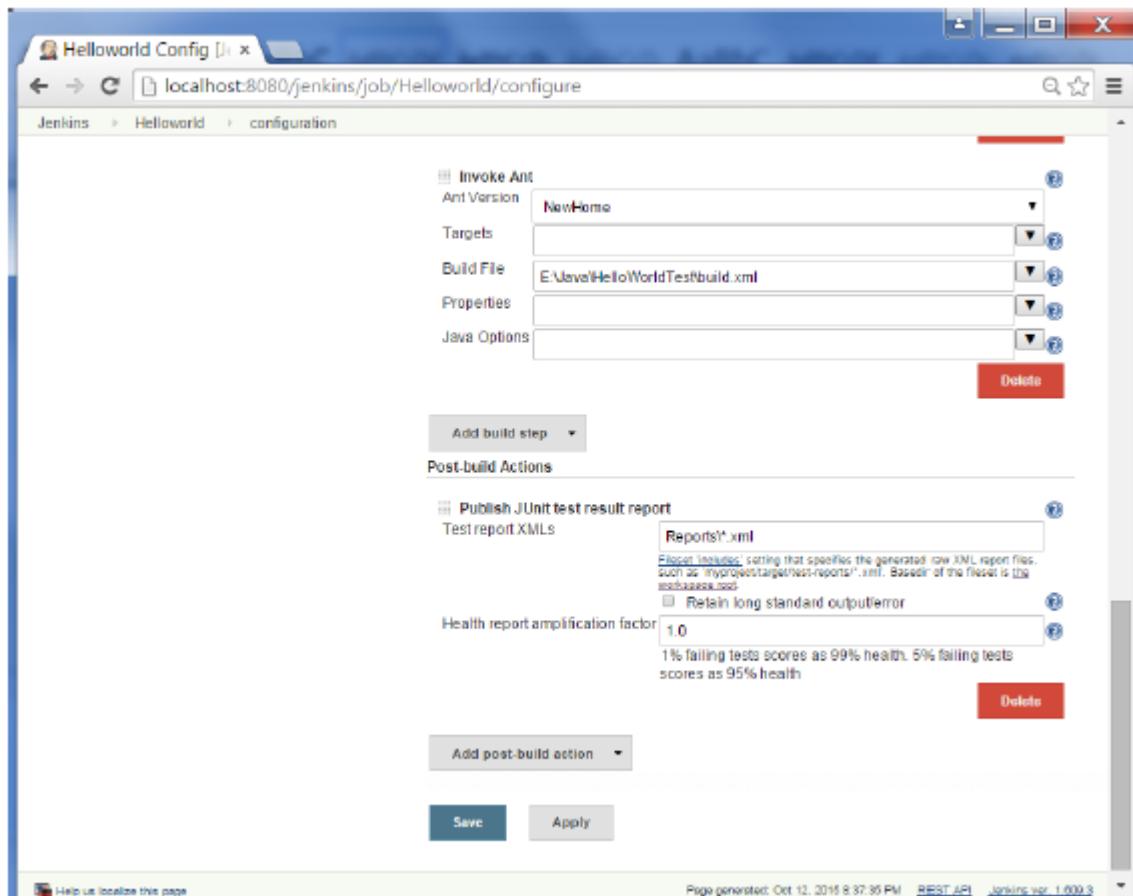


**Step 5** – Next click the option to Add post-build option and choose the option of “Publish Junit test result report”



**Step 6** – In the Test reports 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 a Jenkins build page for the 'Helloworld' project, specifically build #4. The page is titled 'Build #4 (Oct 12, 2015) 8:33:16 PM'. It displays the following information:

- Build Status:** Started 3 days 1 hr ago Took 3.9 sec on master
- Actions:** Back to Project, Status, Changes, Console Output, Edit Build Information, Delete Build, Git Build Data, No Tags, Test Result, Previous Build, Next Build.
- Log:** No changes.
- Git:** Started by anonymous user, Revision: 42f9a82ffadd86fb5c3a0dfae40e731a807fc8f, refs/remotes/origin/master.
- Test Result:** Test Result (1 failure) [HelloWorldTestCase.InitializationError](#)

At the bottom, there are links for 'Help us localize this page', 'Page generated: Oct 15, 2015 10:24:38 PM', 'REST API', and 'Jenkins ver. 1.604.3'.

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.

Helloworld #4 Test R x

localhost:8080/jenkins/job/Helloworld/4/testReport/

# Jenkins

Jenkins > Helloworld > #4 > Test Results

Test Result

1 failures

1 tests  
Took 10 ms  
Add description

All Failed Tests

Test Name	Duration	Age
<a href="#">HelloWorldTestCase</a> InitializationError	10 ms	1

All Tests

Package	Duration	Fail	(#)	Skip	(#)	Pass	(#)	Total	(#)
[root]	10 ms	1	+1	0	(#)	0	(#)	1	+1

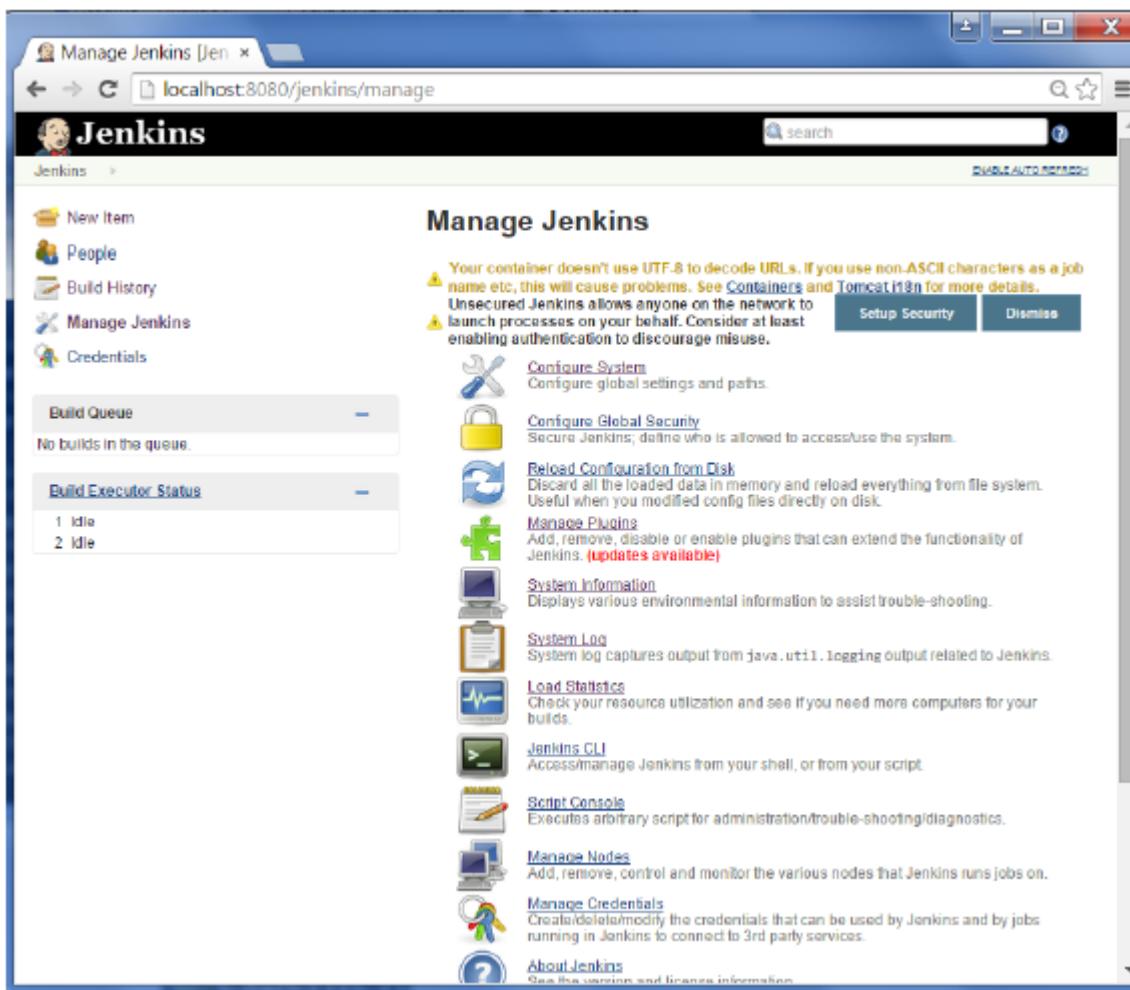
Help us localize this page

Page generated: Oct 12, 2015 8:45:49 PM REST API Jenkins ver. 1.609.3

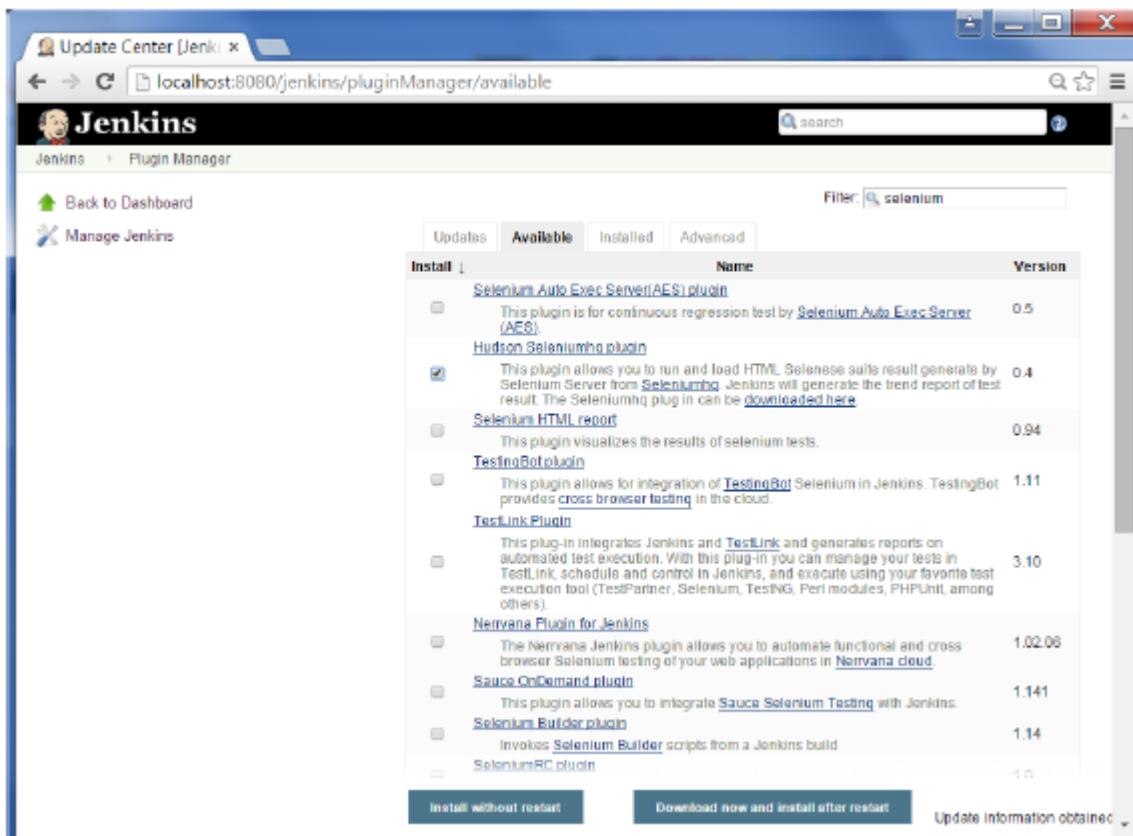
# Jenkins - Automated Testing

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.



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

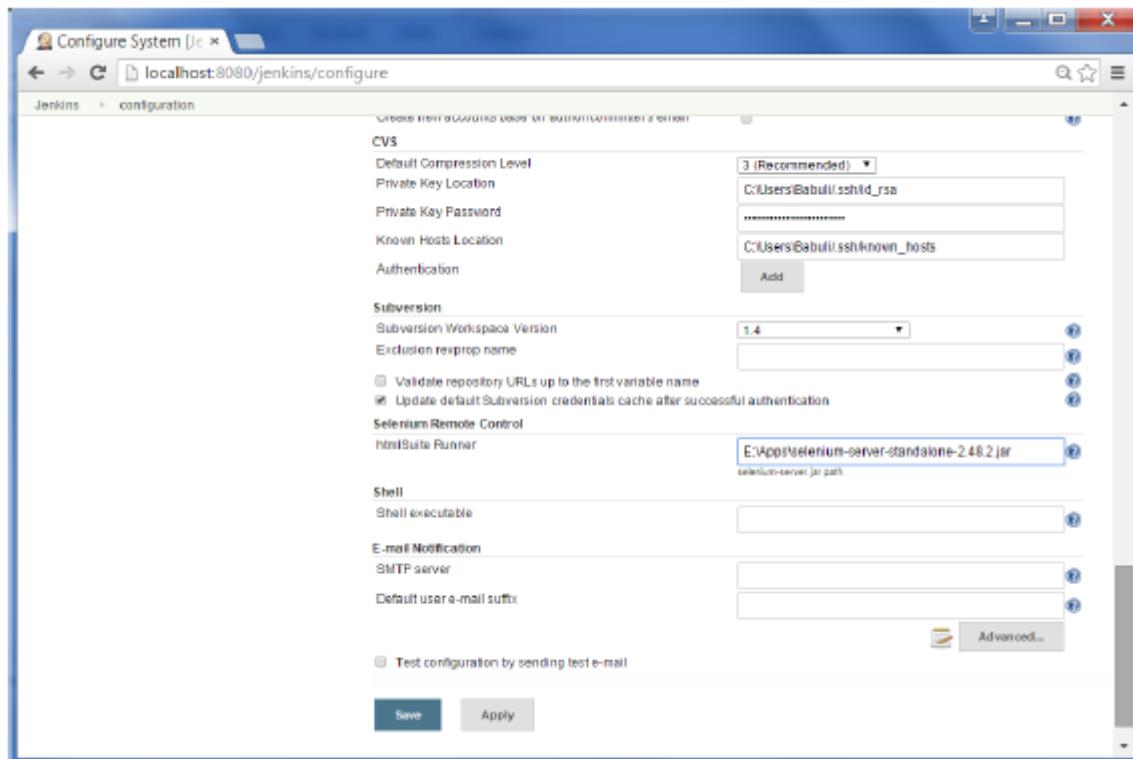


Install	Name	Version
<input type="checkbox"/>	Selenium Auto Exec Server(AES) plugin This plugin is for continuous regression test by <a href="#">Selenium Auto Exec Server (AES)</a> .	0.5
<input checked="" type="checkbox"/>	Hudson SeleniumHQ plugin This plugin allows you to run and load HTML Selenium suite result generates by Selenium Server from <a href="#">SeleniumHQ</a> . Jenkins will generate the trend report of test result. The SeleniumHQ plug in can be <a href="#">downloaded here</a> .	0.4
<input type="checkbox"/>	Selenium HTML report This plugin visualizes the results of selenium tests.	0.94
<input type="checkbox"/>	TestingBot plugin This plugin allows for integration of <a href="#">TestingBot</a> Selenium in Jenkins. TestingBot provides cross browser testing in the cloud.	1.11
<input type="checkbox"/>	TestLink Plugin This plug-in integrates Jenkins and <a href="#">TestLink</a> and generates reports on automated test execution. With this plug-in you can manage your tests in TestLink, schedule and control in Jenkins, and execute using your favorite test execution tool (TestPartner, Selenium, TestNG, Perl modules, PHPUnit, among others).	3.10
<input type="checkbox"/>	Nervana Plugin for Jenkins The Nervana Jenkins plugin allows you to automate functional and cross browser Selenium testing of your web applications in <a href="#">Nervana cloud</a> .	1.02.06
<input type="checkbox"/>	Sauce OnDemand plugin This plugin allows you to integrate <a href="#">Sauce Selenium Testing</a> with Jenkins.	1.141
<input type="checkbox"/>	Selenium Builder plugin Invokes <a href="#">Selenium Builder</a> scripts from a Jenkins build	1.14
<input type="checkbox"/>	SeleniumRC plugin	1.0

### Step 3 – Go to Configure system.

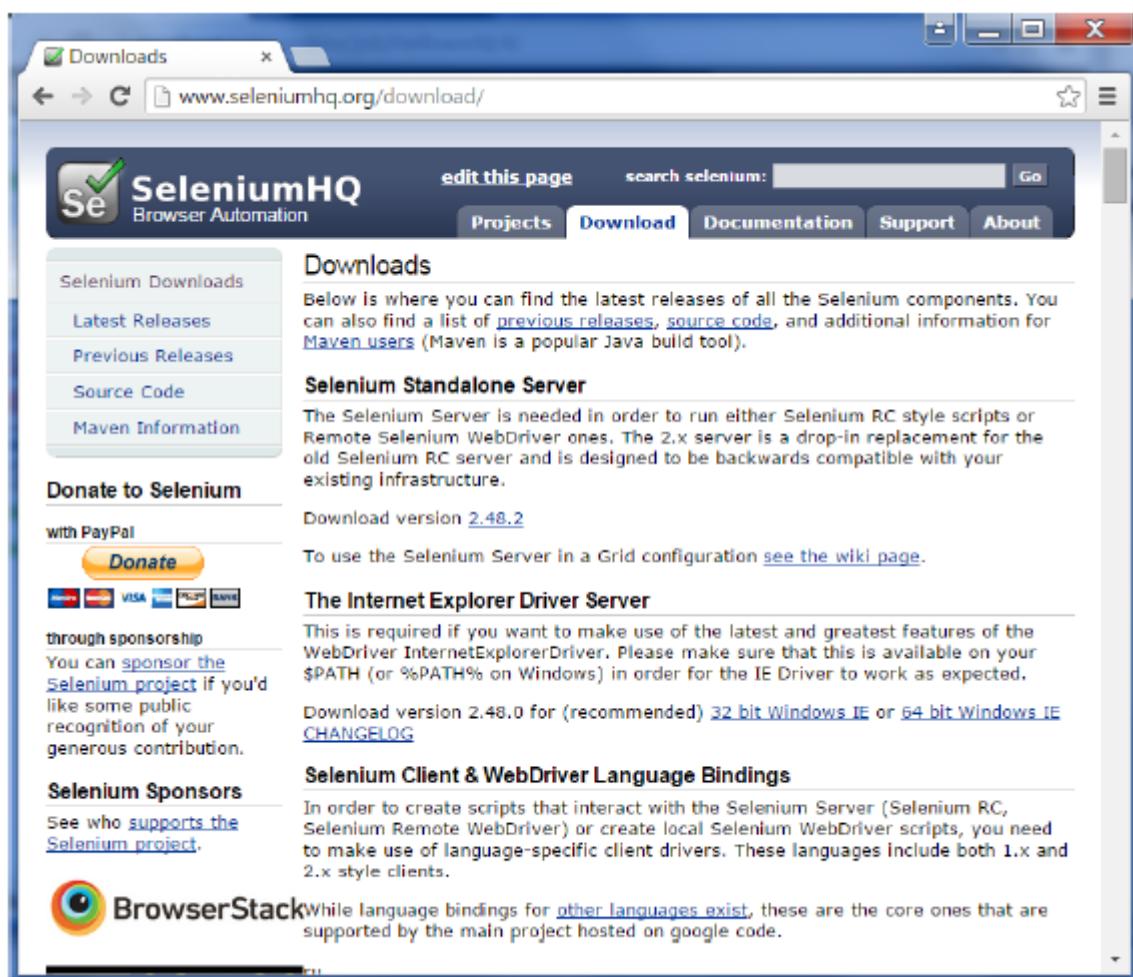
(updates available)), 'System Information' (Displays various environmental information to assist trouble-shooting), 'System Log' (System log captures output from java.util.logging output related to Jenkins), 'Load Statistics' (Check your resource utilization and see if you need more computers for your builds), 'Jenkins CLI' (Access/manage Jenkins from your shell, or from your script), 'Script Console' (Executes arbitrary script for administration/trouble-shooting/diagnostics), and 'Manage Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on). A 'Setup Security' button is visible in the top right of the configuration section." data-bbox="140 490 854 861"/&gt;

### Step 4 – Configure the selenium server jar and click on the Save button.



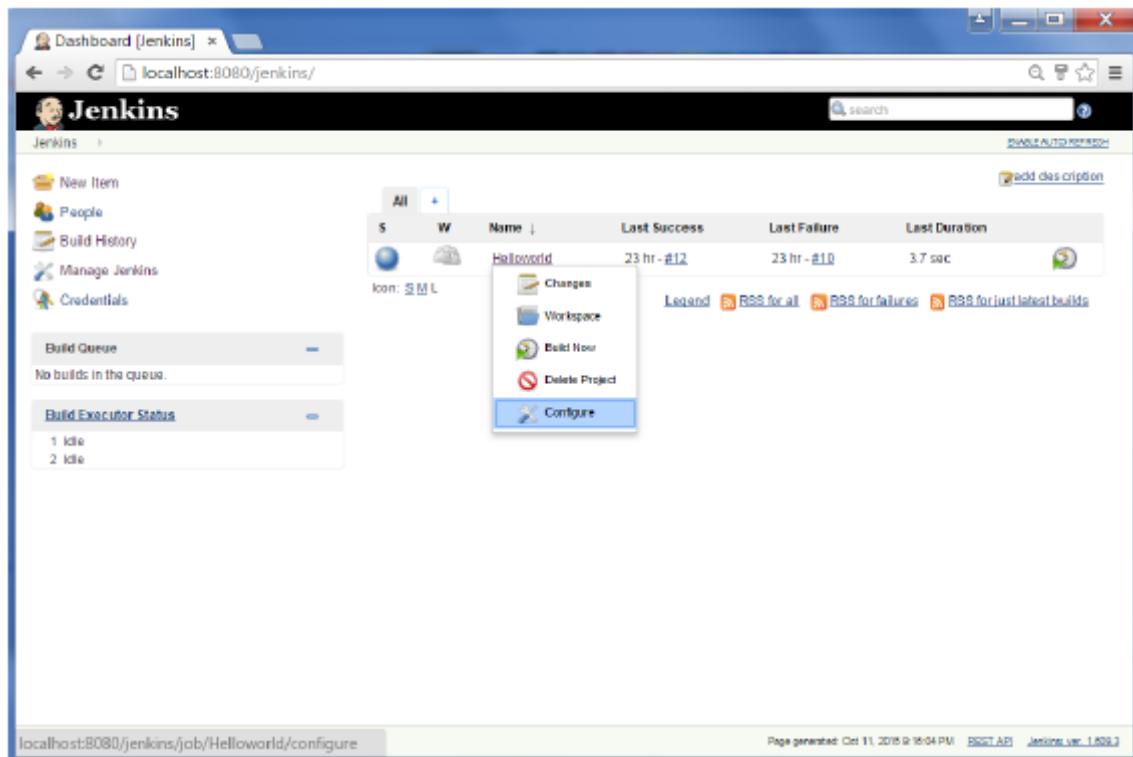
**Note** – The selenium jar file can be downloaded from the location [SeleniumHQ](#)

Click on the download for the Selenium standalone server.



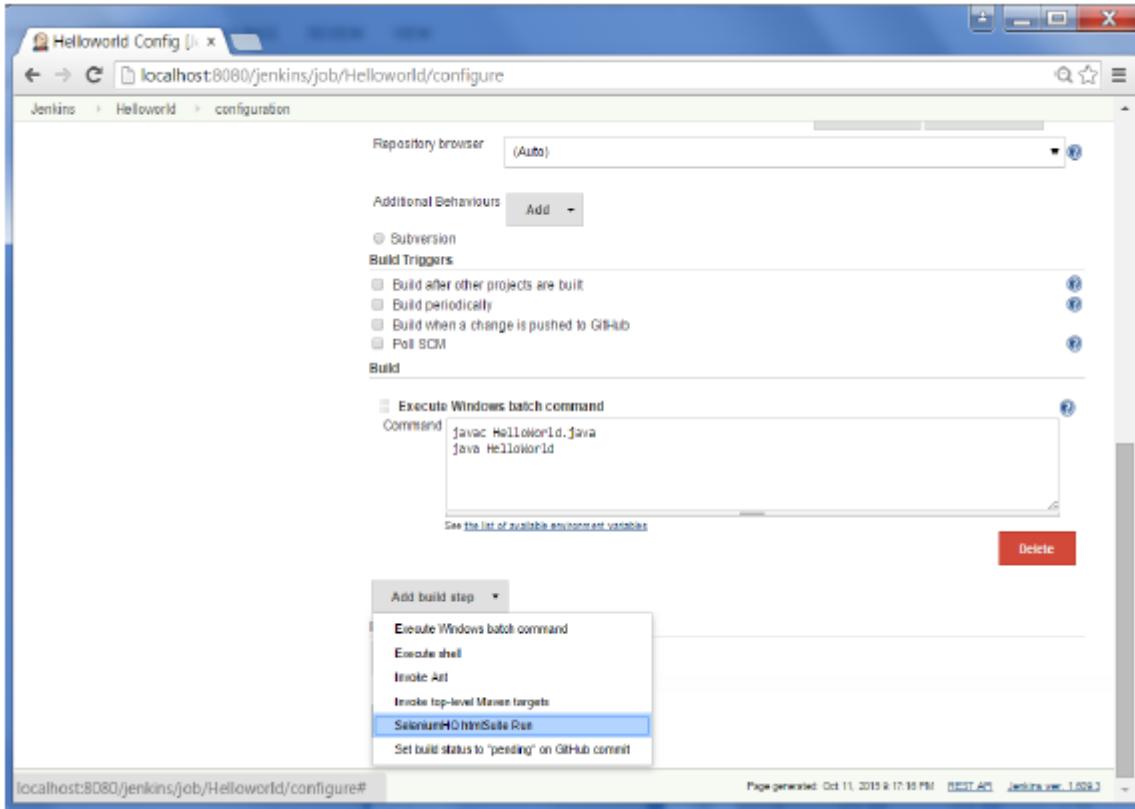
The screenshot shows the SeleniumHQ website with the URL [www.seleniumhq.org/download/](http://www.seleniumhq.org/download/) in the address bar. The page is titled "SeleniumHQ Browser Automation". The "Download" tab is selected in the top navigation bar. The "Downloads" section is highlighted, showing information about the latest releases of Selenium components. It includes links for "Latest Releases", "Previous Releases", "Source Code", and "Maven Information". There is a "Donate to Selenium" section with a "Donate" button and payment method icons (PayPal, Credit Card, VISA, MasterCard, American Express, Discover). A "Selenium Sponsors" section lists "BrowserStack" as a sponsor. The "Selenium Standalone Server" section provides instructions for running the server and links to download versions 2.48.2 and 2.48.0 for Internet Explorer. The "Selenium Client & WebDriver Language Bindings" section explains how to create scripts that interact with the Selenium Server or WebDriver, and lists supported languages.

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

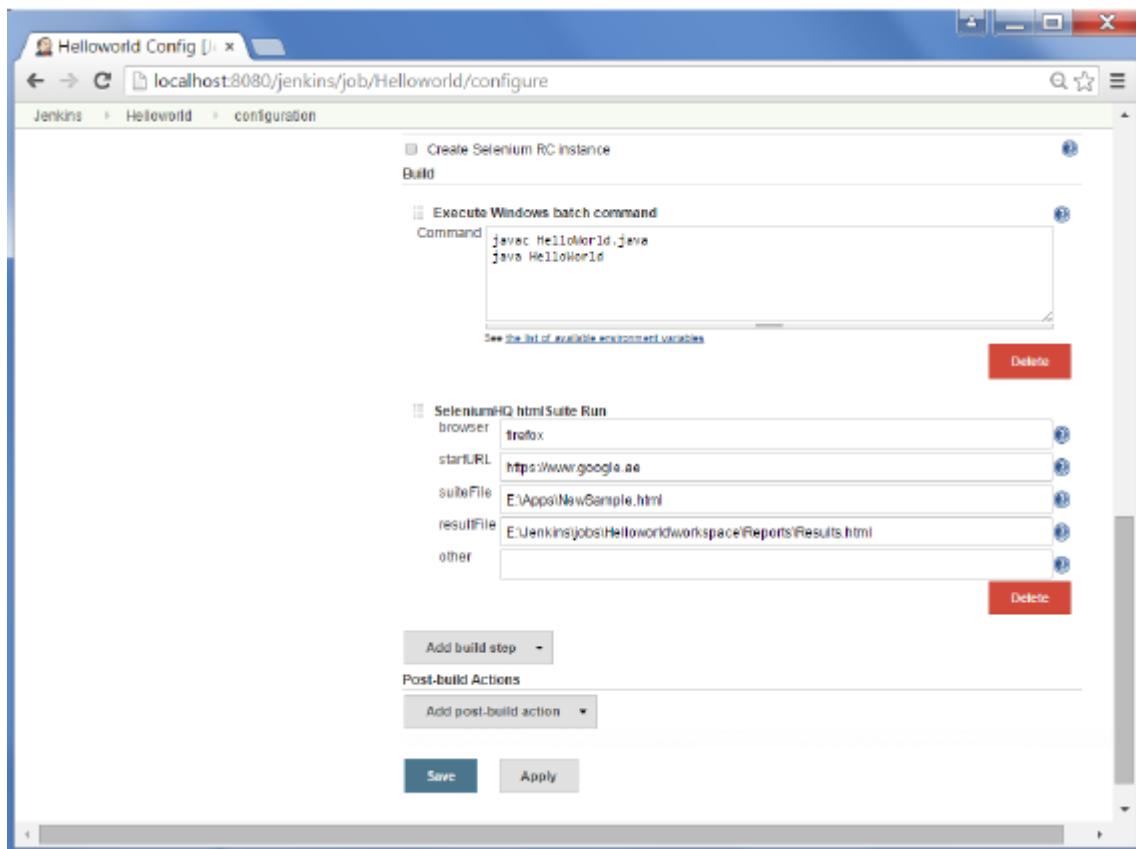


The screenshot shows the Jenkins dashboard at [localhost:8080/jenkins/](http://localhost:8080/jenkins/). The "Dashboard" link is in the top left. The main area shows a list of projects: "HelloWorld" (last success 23 hr - #12, last failure 23 hr - #10, last duration 3.7 sec). A context menu is open over the "HelloWorld" project, with the "Configure" option highlighted. The left sidebar includes links for "New Item", "People", "Build History", "Manage Jenkins", and "Credentials". The "Build Queue" and "Build Executor Status" sections are also visible.

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



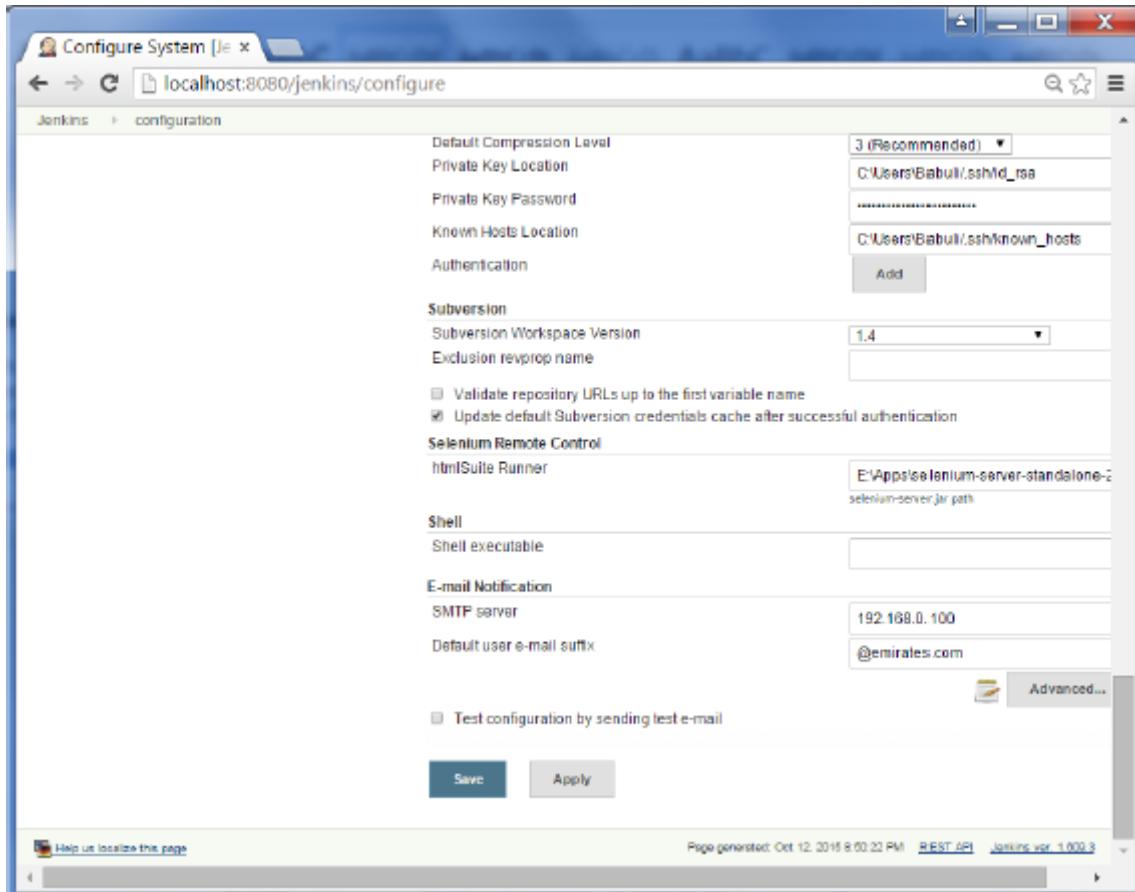
**Step 7** – Add the necessary details for the selenium test. Here the suiteFile is the TestSuite generated by using the Selenium IDE. Click on Save and execute a build. Now the post build will launch the selenium driver, and execute the html test.



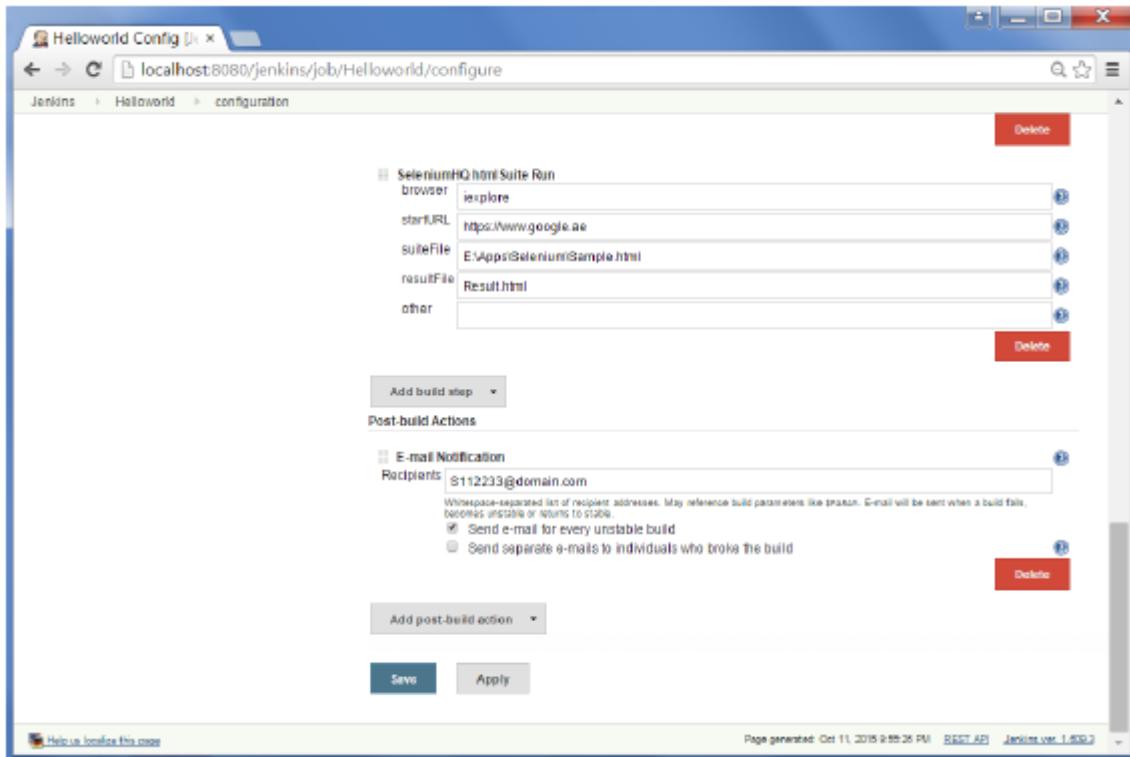
# Jenkins - Notification

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.

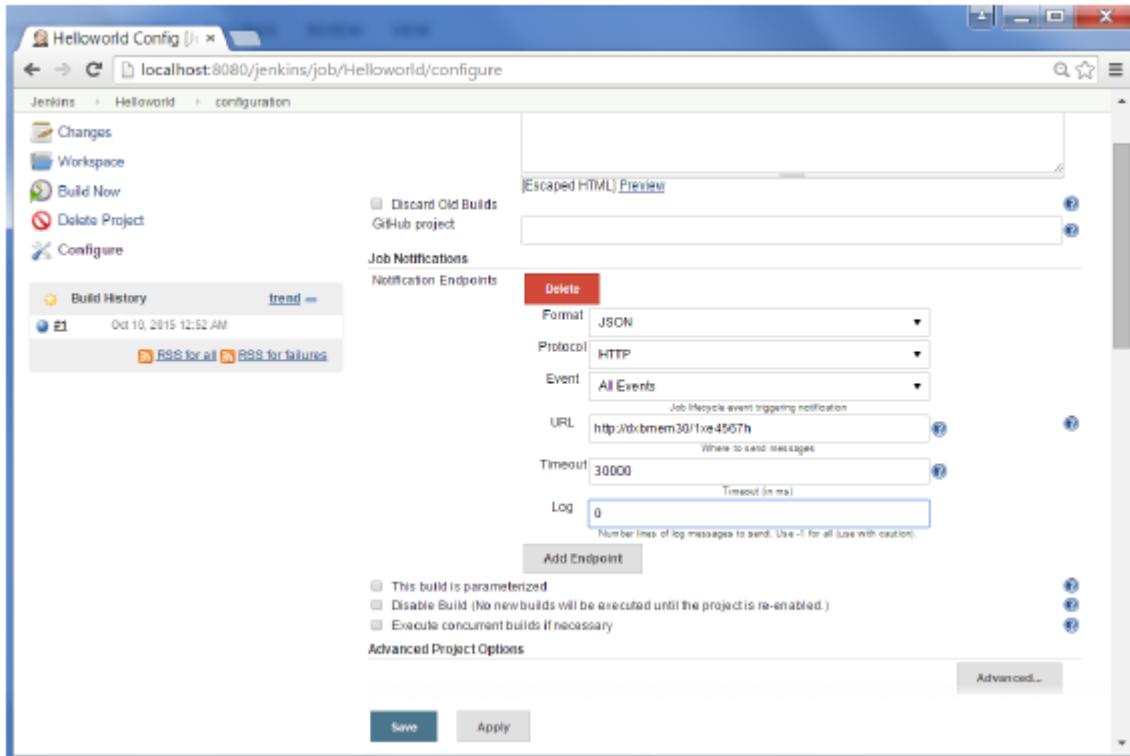


**Step 2** – Configure the recipients in the Jenkins project - When you configure any Jenkins build project, right at the end is the ability to add recipients who would get email notifications for unstable or broken builds. Then click on the Save button.



The screenshot shows the Jenkins job configuration page for 'Helloworld'. The 'Build Steps' section contains a 'SeleniumHQ html Suite Run' step with the following parameters: browser (set to 'explore'), startURL ('https://www.google.ae'), suiteFile ('E:\Apps\Selenium\Sample.html'), resultFile ('Result.html'), and other. The 'Post-build Actions' section contains an 'E-mail Notification' step with the recipient 'S112233@domain.com'. The 'Send e-mail for every unstable build' and 'Send separate e-mails to individuals who broke the build' checkboxes are checked. Buttons for 'Save' and 'Apply' are at the bottom.

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.



The screenshot shows the Jenkins job configuration page for 'Helloworld'. The 'Job Notifications' section is expanded, showing a 'Notification Endpoints' table with one entry. The entry is titled 'Delete' and has the following details: Format (JSON), Protocol (HTTP), Event (All Events), URL ('http://10.10.10.10:4567'), and Timeout (30000). A note below says 'Job lifecycle event triggering notification'. The 'Log' field is set to 0. Buttons for 'Save' and 'Apply' are at the bottom. A sidebar on the left shows 'Build History' with a single entry from Oct 10, 2015, 12:52 AM, and links for 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'.

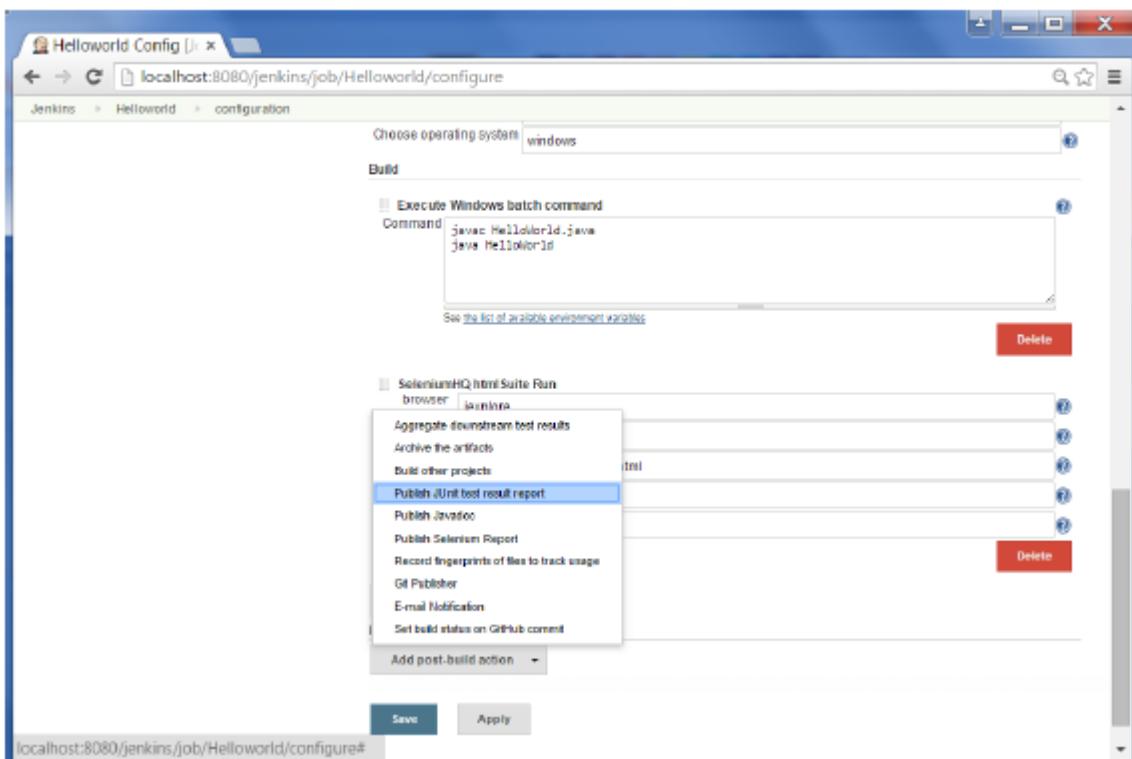
Here are the details of each option –

- **"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

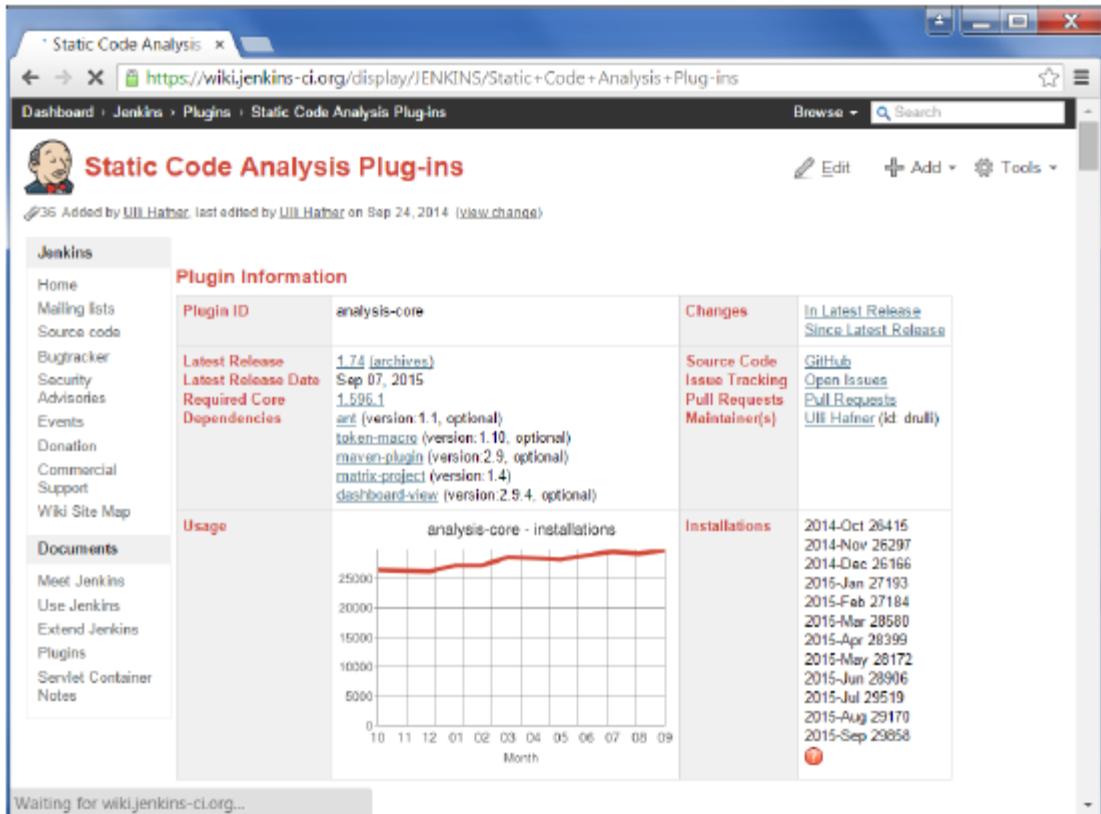
As demonstrated in the earlier section, there are many reporting plugins available with the simplest one being the reports available for jUnit tests.

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 Plug-ins page. The main content is the 'analysis-core' plugin page. The 'Plugin Information' section shows the following details:

Plugin ID	analysis-core	Changes	In Latest Release Since Latest Release
Latest Release	1.78 [archives]	Source Code	<a href="#">GitHub</a>
Latest Release Date	Sep 07, 2015	Issue Tracking	<a href="#">Open Issues</a>
Required Core Dependencies	ant (version: 1.1, optional) token-macro (version: 1.10, optional) maven-plugin (version: 2.9, optional) matrix-project (version: 1.4) dashboard-view (version: 2.9.4, optional)	Pull Requests	<a href="#">Pull Requests</a>
Maintainer(s)	Ulli Hafner (id: drull)		

The 'Usage' section contains a line graph titled 'analysis-core - installations' showing the number of installations over time. The x-axis represents the months from October 2014 to September 2015, and the y-axis represents the number of installations from 0 to 25,000. The graph shows a steady increase from approximately 25,000 installations in October 2014 to about 28,000 installations in September 2015.

Month	Installations
10	25000
11	25000
12	25000
01	25000
02	25000
03	25000
04	25000
05	25000
06	25000
07	25000
08	25000
09	28000

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.

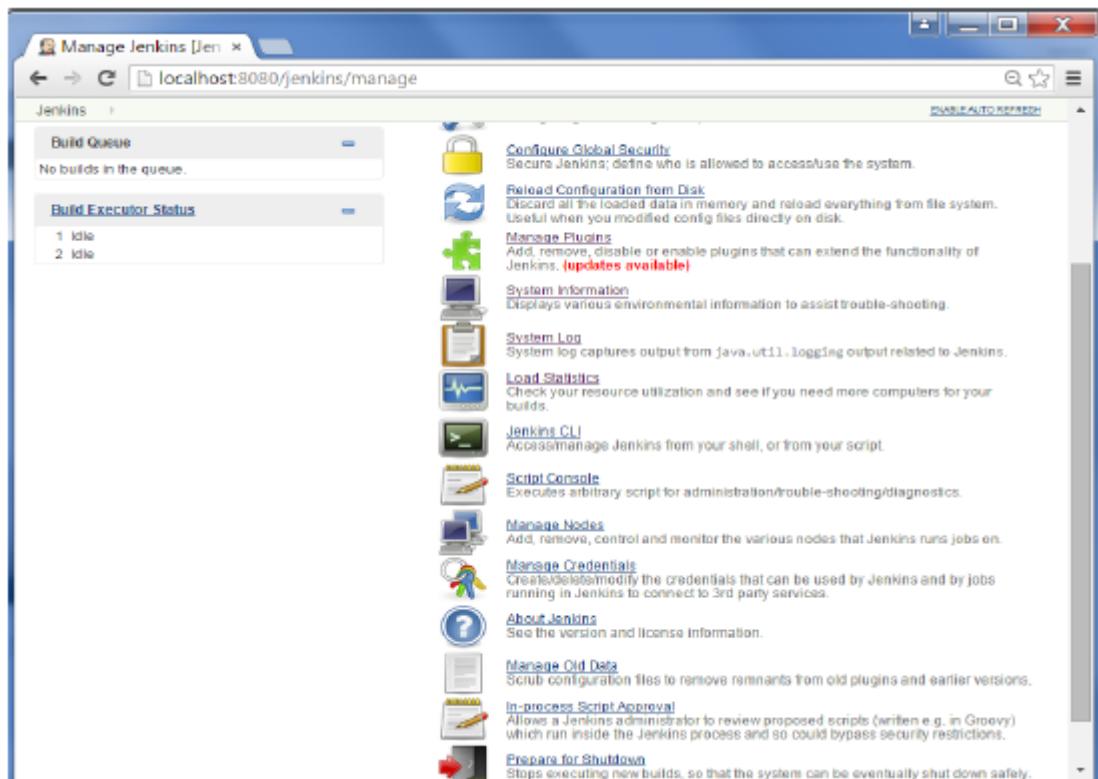
Sometimes you might also need several different environments to test your builds. In this case using a slave to represent each of your required environments is almost a must.

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.

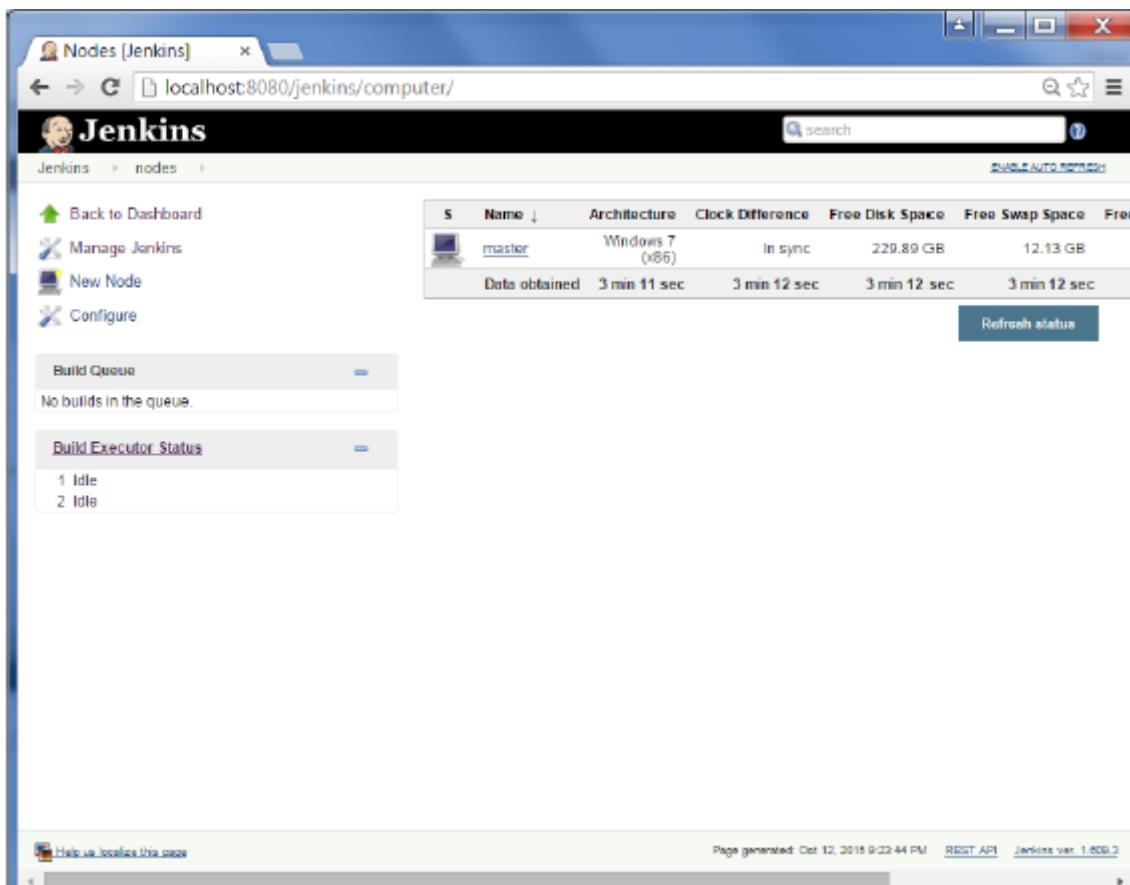
Since each slave runs a separate program called a "slave agent" there is no need to install the full Jenkins (package or compiled binaries) on a slave. There are various ways to start slave agents, but in the end the slave agent and Jenkins master needs to establish a bi-directional communication link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

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

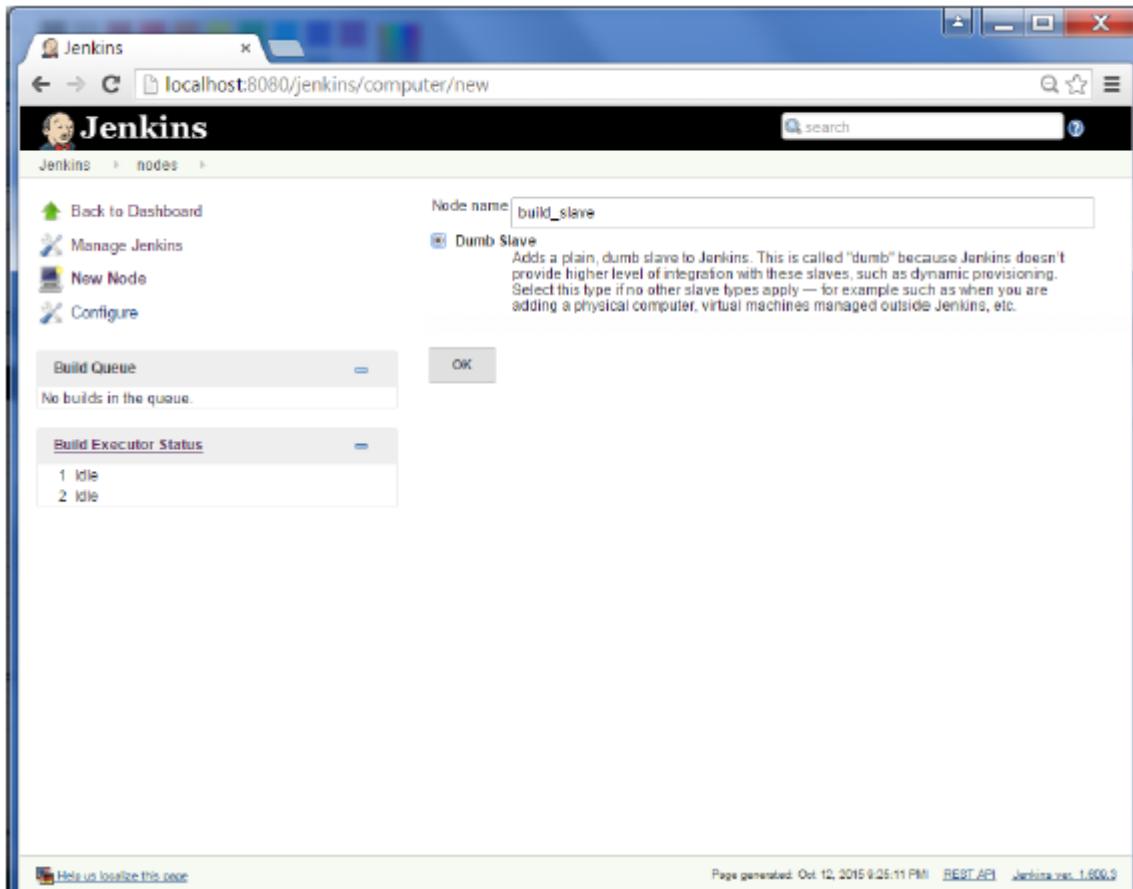
**Step 1** – Go to the Manage Jenkins section and scroll down to the section of Manage Nodes.



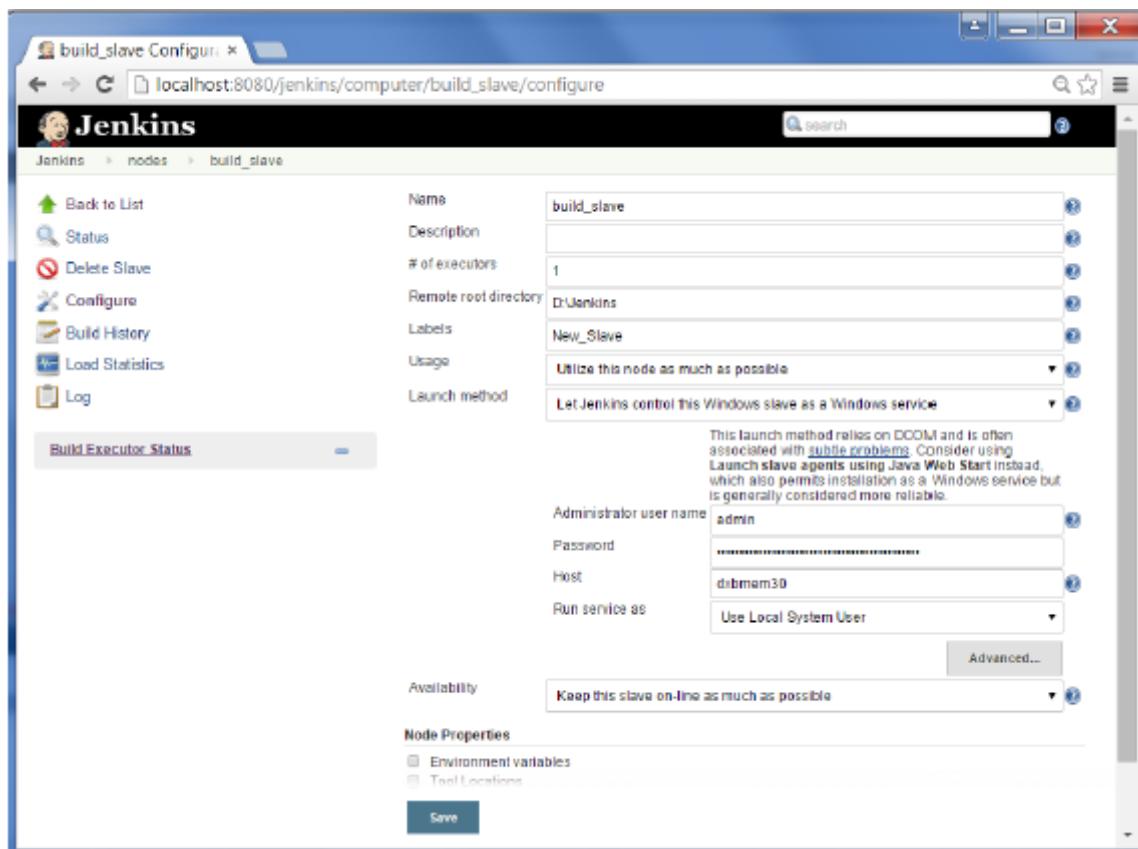
## Step 2 – Click on New Node



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



**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.

Nodes [Jenkins] X

localhost:8080/jenkins/computer/

# Jenkins

Jenkins > nodes > ENABLE AUTOMATIC UPDATES

[Back to Dashboard](#)

[Manage Jenkins](#)

[New Node](#)

[Configure](#)

**Build Queue** -

No builds in the queue.

**Build Executor Status** -

**master**  
1 Idle  
2 Idle

**build\_slave** (offline)

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp S
1	<a href="#">build_slave</a>		N/A	N/A	N/A	N/A
2	<a href="#">master</a>	Windows 7 (x86)	In sync	229.89 GB	12.13 GB	229.89 GB
		Data obtained	3 ms	2 ms	1 ms	11 min

Refresh status

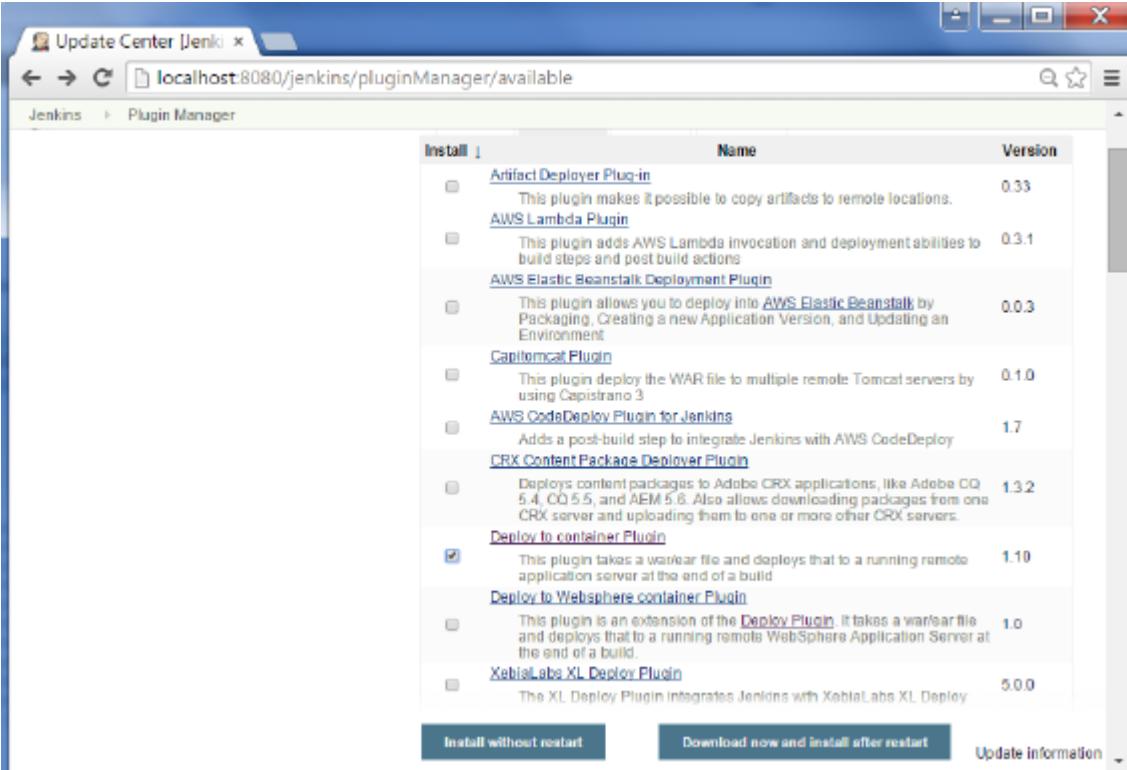
[Help us localize this page](#)

Page generated: Oct 12, 2015 9:31:43 PM [REST API](#) 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 URL in the address bar is `localhost:8080/jenkins/pluginManager/available`. The 'Available' tab is selected. A list of plugins is displayed, with the 'Deploy to container Plugin' checked for installation. The table includes columns for Name and Version. The 'Deploy to container Plugin' is version 1.10 and is described as taking a war/ear file and deploying it to a running remote application server at the end of a build.

Install	Name	Version
<input type="checkbox"/>	<a href="#">Artifact Deployer Plugin</a> This plugin makes it possible to copy artifacts to remote locations.	0.33
<input type="checkbox"/>	<a href="#">AWS Lambda Plugin</a> This plugin adds AWS Lambda invocation and deployment abilities to build steps and post build actions	0.3.1
<input type="checkbox"/>	<a href="#">AWS Elastic Beanstalk Deployment Plugin</a> This plugin allows you to deploy into <a href="#">AWS Elastic Beanstalk</a> by Packaging, Creating a new Application Version, and Updating an Environment	0.0.3
<input type="checkbox"/>	<a href="#">Capitomatic Plugin</a> This plugin deploys the WAR file to multiple remote Tomcat servers by using Capistrano 3	0.1.0
<input type="checkbox"/>	<a href="#">AWS CodeDeploy Plugin for Jenkins</a> Adds a post-build step to integrate Jenkins with AWS CodeDeploy	1.7
<input type="checkbox"/>	<a href="#">CRX Content Package Deployer Plugin</a> Deploys content packages to Adobe CRX applications, like Adobe CQ 5.4, CQ 5.5, and AEM 6.0. Also allows downloading packages from one CRX server and uploading them to one or more other CRX servers.	1.3.2
<input checked="" type="checkbox"/>	<a href="#">Deploy to container Plugin</a> This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build	1.10
<input type="checkbox"/>	<a href="#">Deploy to WebSphere container Plugin</a> This plugin is an extension of the <a href="#">Deploy Plugin</a> . It takes a war/ear file and deploys that to a running remote WebSphere Application Server at the end of a build.	1.0
<input type="checkbox"/>	<a href="#">Xebialabs XL Deploy Plugin</a> The XL Deploy Plugin integrates Jenkins with Xebialabs XL Deploy	5.0.0

**Buttons at the bottom:**  
Install without restart   Download now and install after restart   Update information

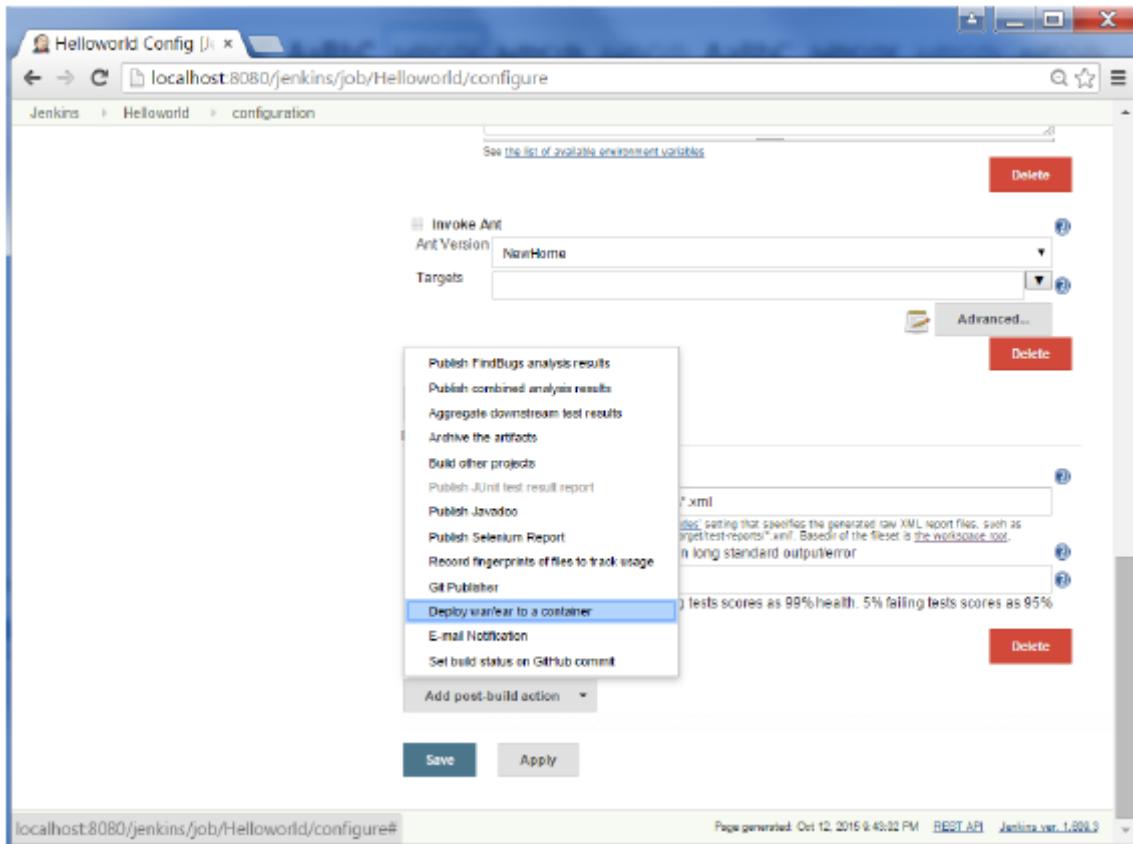
This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build.

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

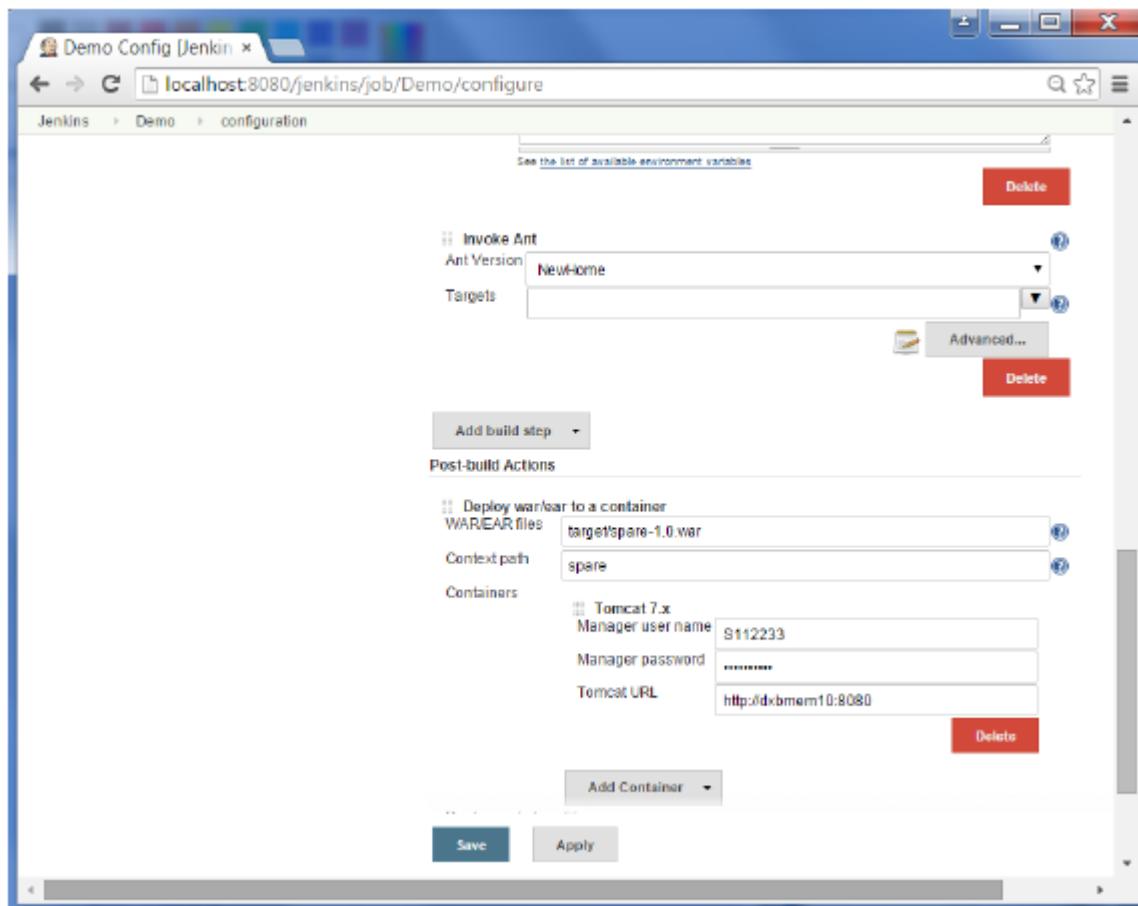
JBoss 3.x/4.x

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.



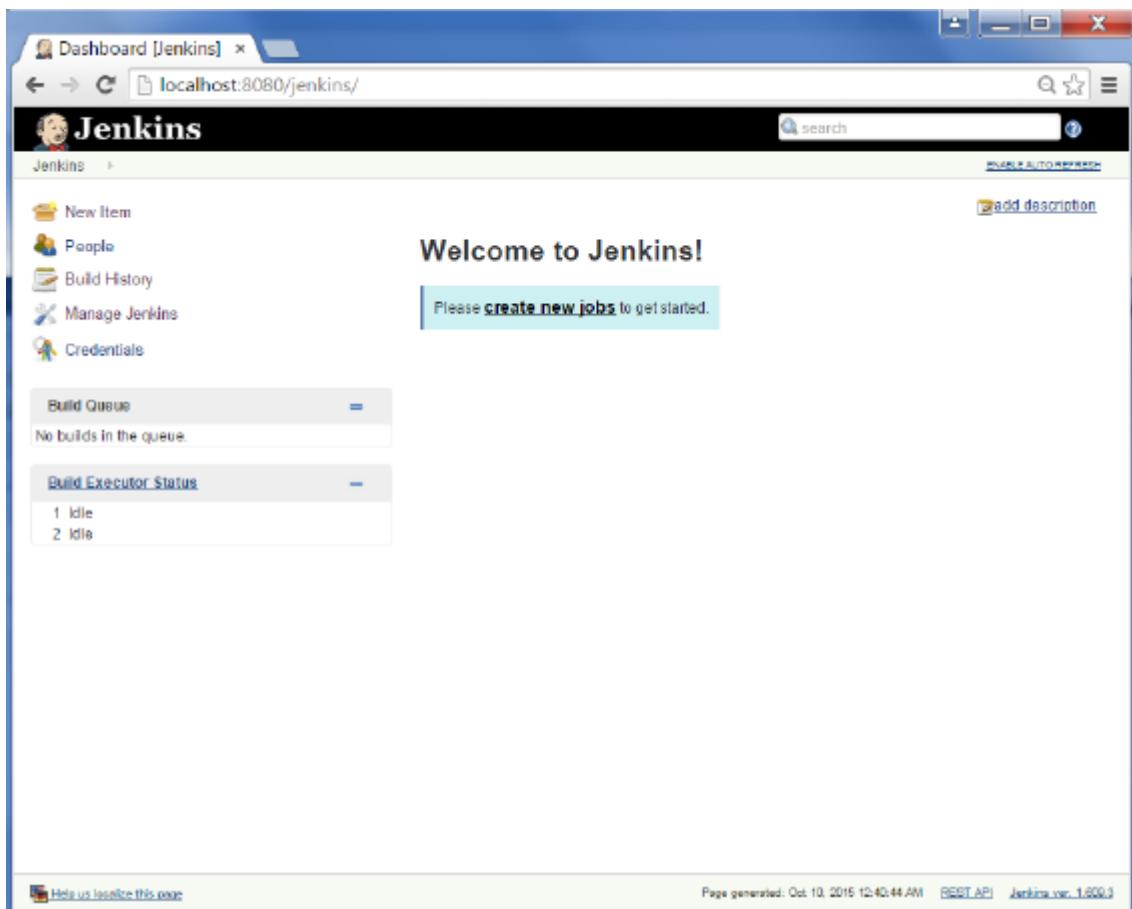
# Jenkins - Metrics & Trends

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'.

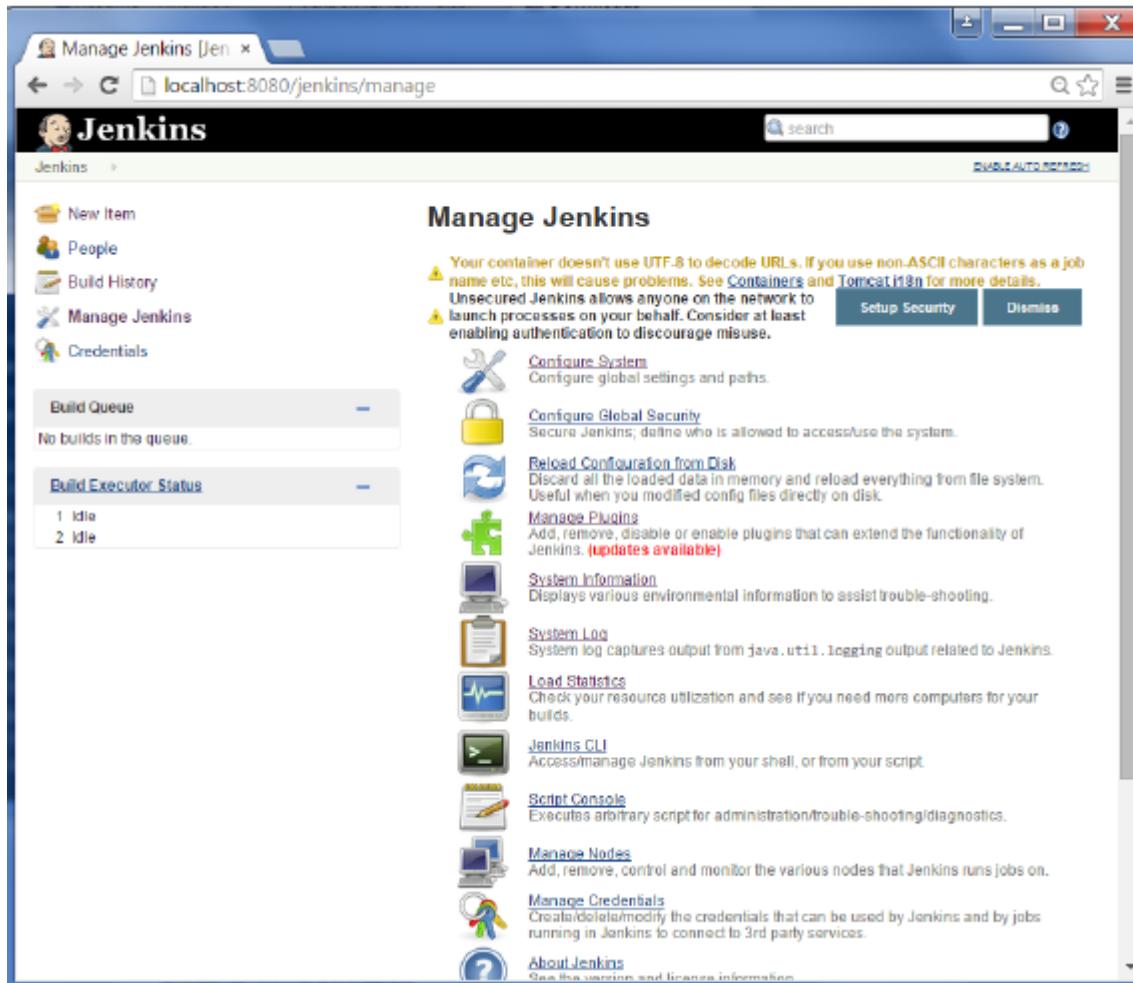
This plugin calculates the following metrics for all of the builds once installed

- 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

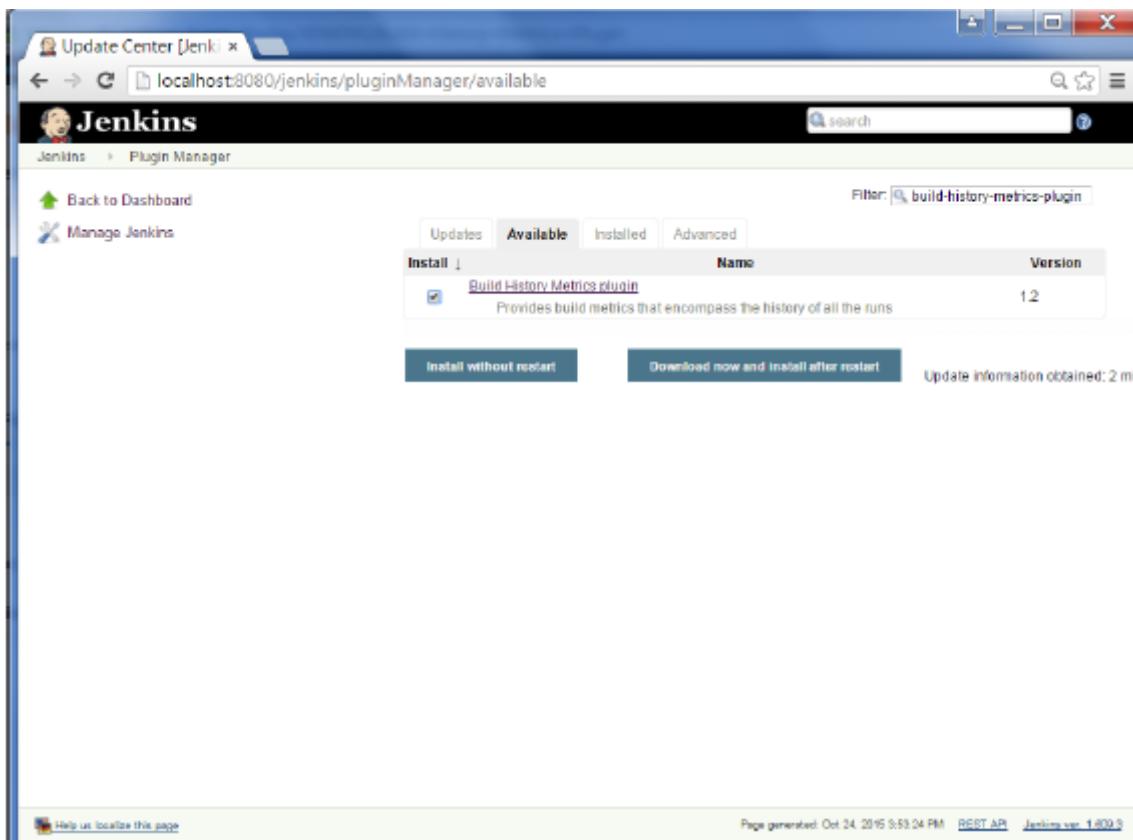


**Step 2** – Go to the Manage Plugins option.



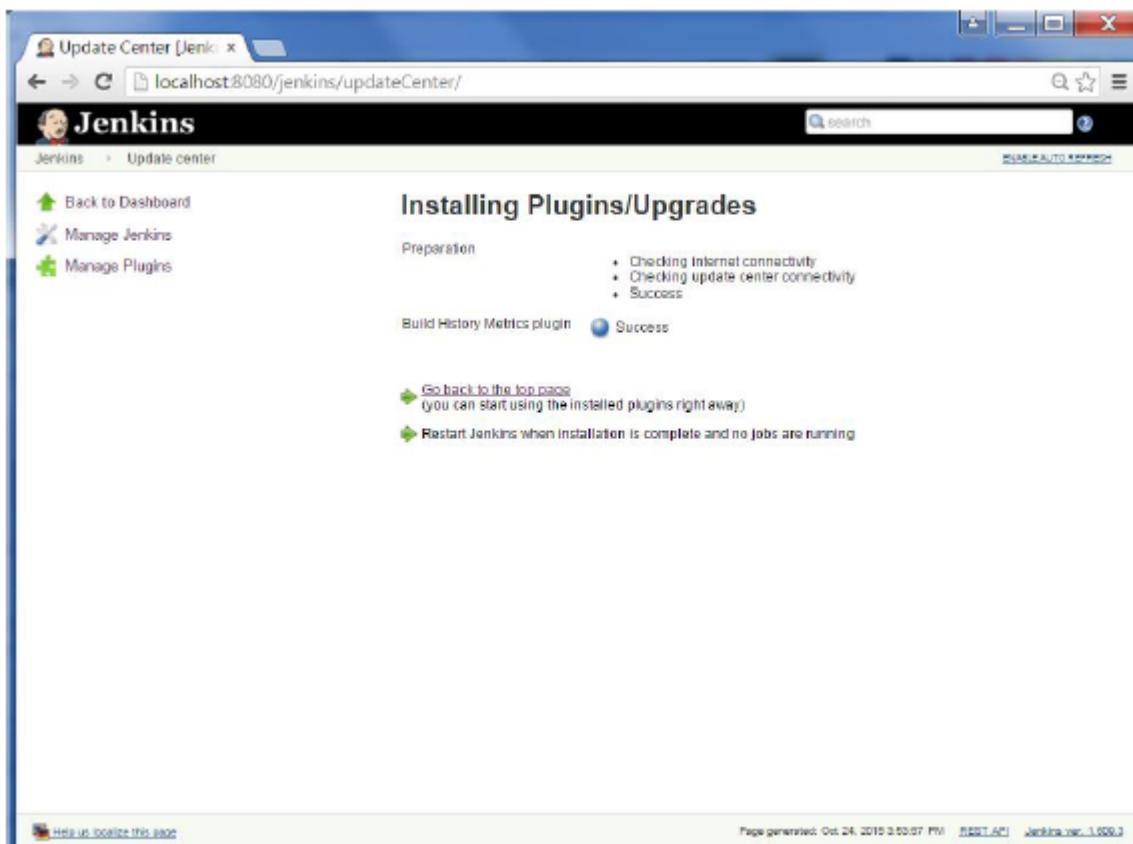
The screenshot shows the Jenkins 'Manage Jenkins' interface. The left sidebar includes links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Under 'Build Queue' and 'Build Executor Status', there are no builds listed. The main content area is titled 'Manage Jenkins' and contains several configuration links with icons: 'Configure System' (wrench), 'Configure Global Security' (padlock), 'Reload Configuration from Disk' (refresh), 'Manage Plugins' (puzzle), 'System Information' (monitor), 'System Log' (text), 'Load Statistics' (graph), 'Jenkins CLI' (terminal), 'Script Console' (script), 'Manage Nodes' (computer), 'Manage Credentials' (key), and 'About Jenkins' (info). A warning message at the top states: 'Your container doesn't use UTF-8 to decode URLs. If you use non-ASCII characters as a job name etc, this will cause problems. See [Containers](#) and [Tomcat 8](#) for more details.' It also mentions 'Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse.' Buttons for 'Setup Security' and 'Dismiss' are shown.

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



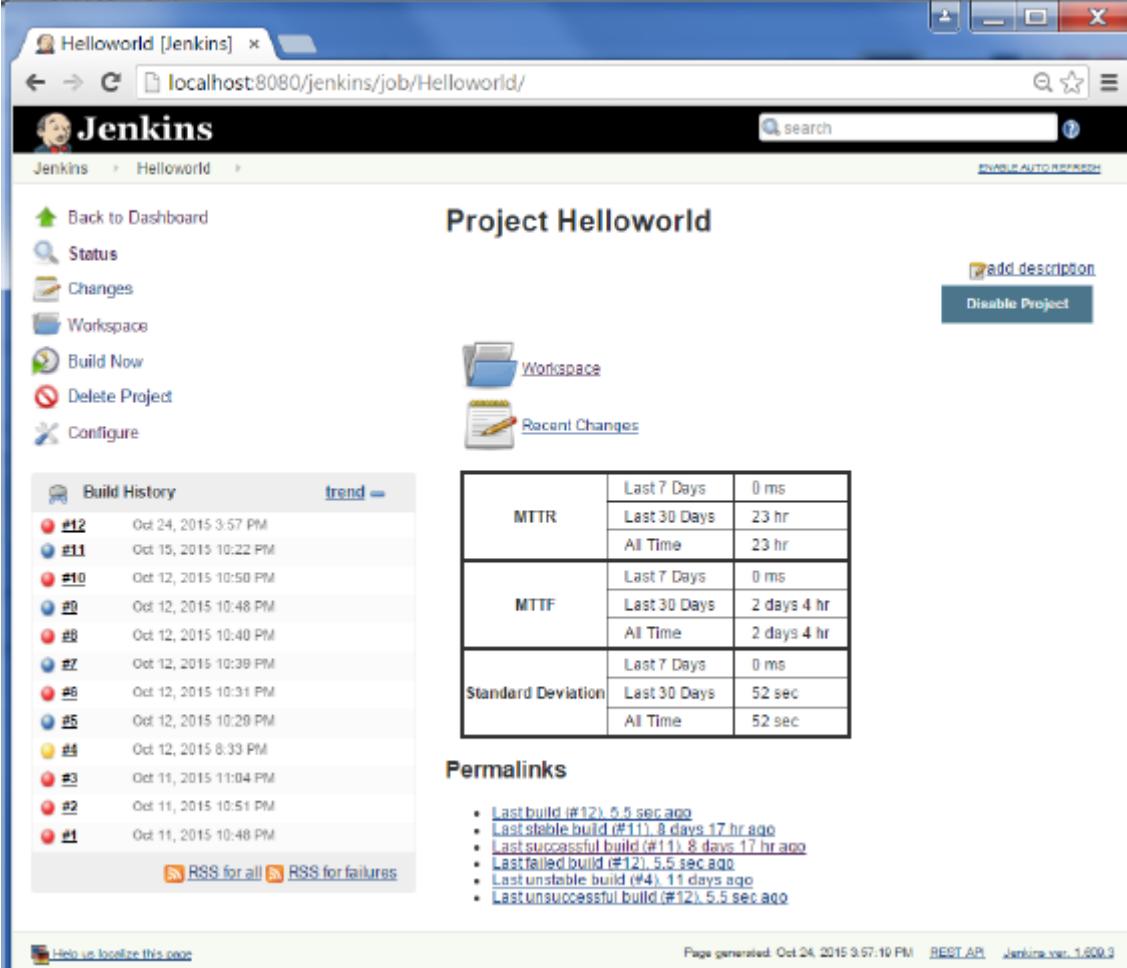
The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Jenk]". The address bar shows "localhost:8080/jenkins/pluginManager/available". The main content area has a "Available" tab selected. A table lists the "Build History Metrics plugin" with version 1.2. The table has columns for "Name" and "Version". Below the table are two buttons: "Install without restart" and "Download now and install after restart". A status message "Update information obtained: 2 mi" is shown. The bottom of the page includes a "Help us localize this page" link, a timestamp "Page generated: Oct 24, 2015 3:59:24 PM", and "REST API Jenkins ver. 1.600.3".

**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 "Installing Plugins/Upgrades" page. The title bar says "Update Center [Jenk]". The address bar shows "localhost:8080/jenkins/updateCenter/". The main content area has a heading "Installing Plugins/Upgrades". It shows a "Preparation" section with a bulleted list: "• Checking Internet connectivity", "• Checking update center connectivity", and "• Success". Below that is a "Build History Metrics plugin" section with a "Success" status icon. At the bottom, there are two green checkmark icons with text: "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 bottom of the page includes a "Help us localize this page" link, a timestamp "Page generated: Oct 24, 2015 3:59:57 PM", and "REST API Jenkins ver. 1.600.3".

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 interface for the 'Helloworld' project. On the left, a sidebar lists navigation options: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main content area is titled 'Project Helloworld'. It features a 'Build History' table with 12 entries from Oct 24, 2015, to Oct 11, 2015. Below the history is a 'Permalinks' section with links to various build logs. To the right, there are three tables for 'MTTR', 'MTTF', and 'Standard Deviation', each showing data for Last 7 Days, Last 30 Days, and All Time. A 'Recent Changes' section is also present. At the bottom, there are links for 'RSS for all' and 'RSS for failures'.

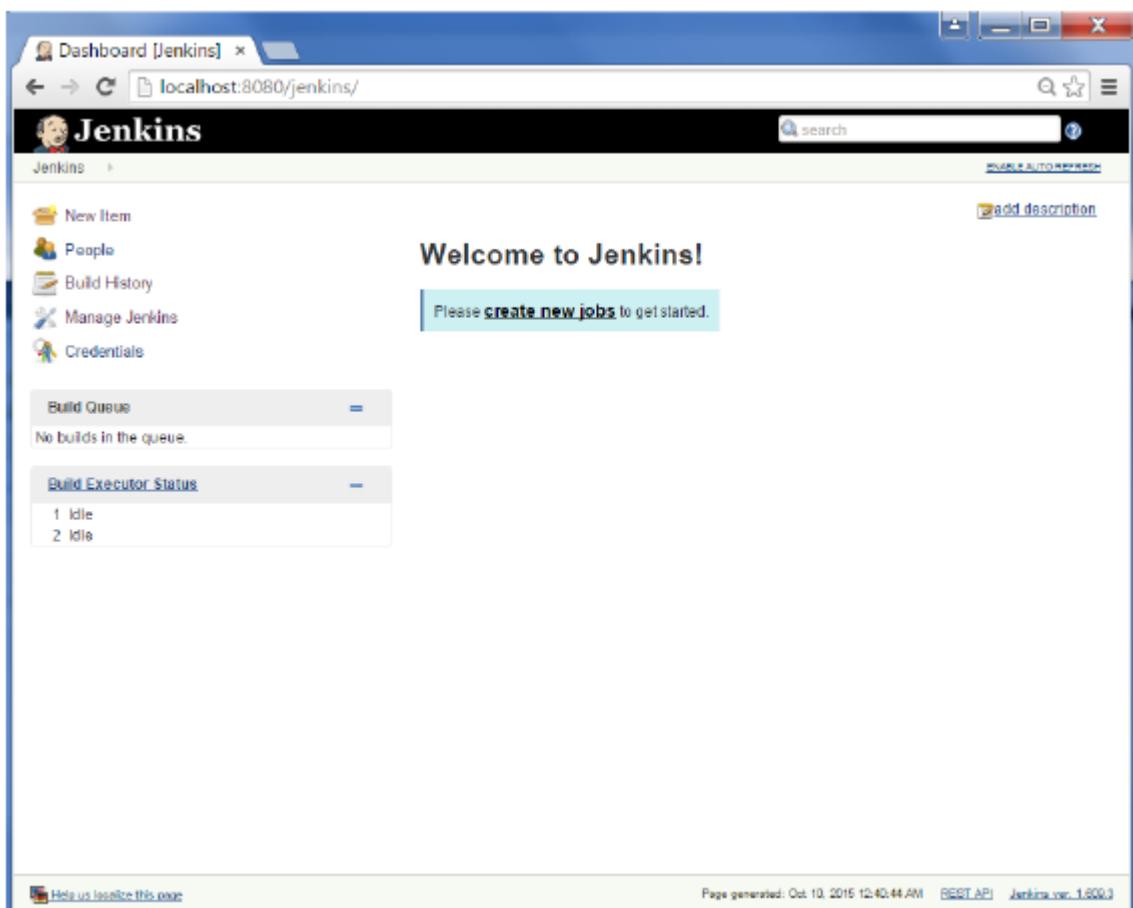
	Last 7 Days	0 ms
MTTR	Last 30 Days	23 hr
	All Time	23 hr
MTTF	Last 7 Days	0 ms
	Last 30 Days	2 days 4 hr
	All Time	2 days 4 hr
Standard Deviation	Last 7 Days	0 ms
	Last 30 Days	52 sec
	All Time	52 sec

**Permalinks**

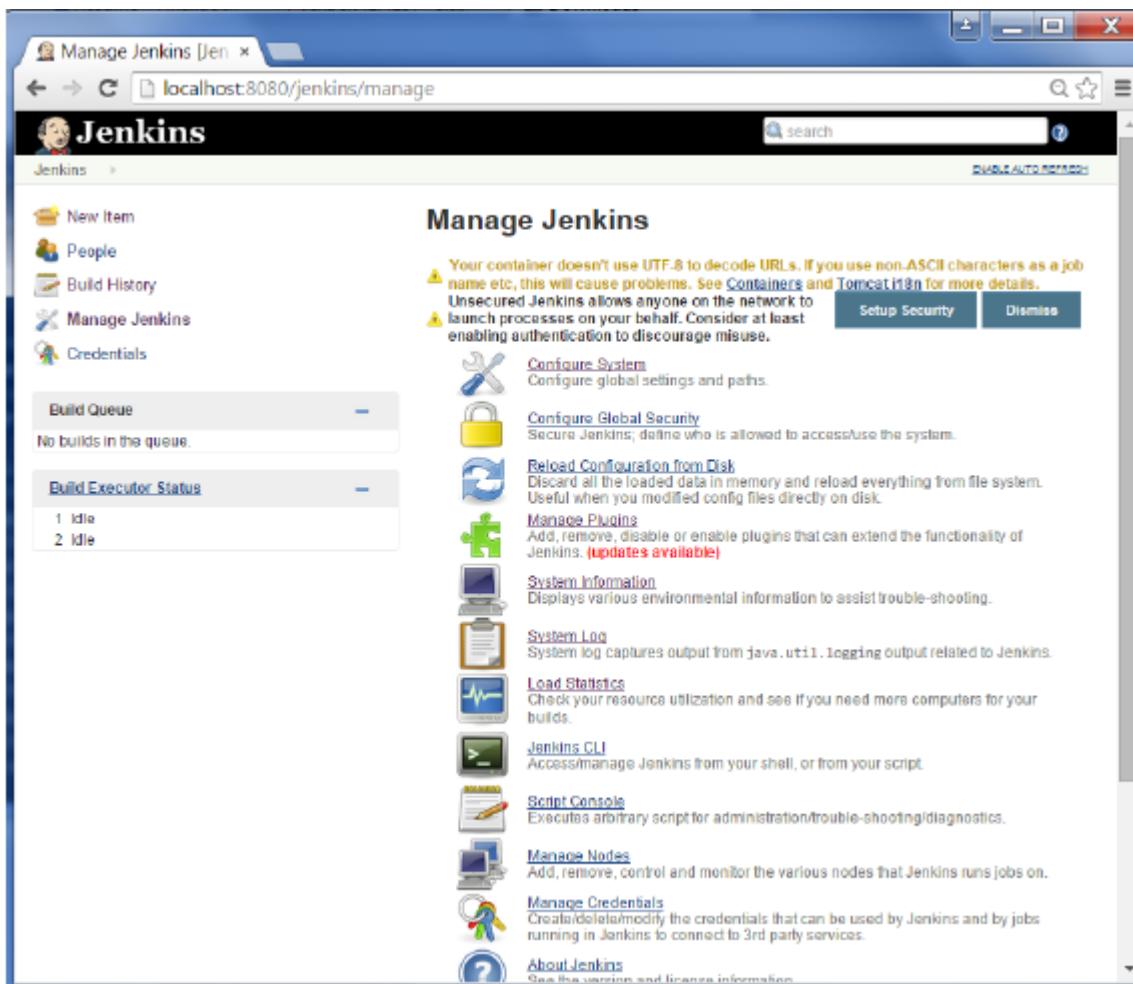
- Last build (#12) 5.5 sec ago
- Last stable build (#11) 8 days 17 hr ago
- Last successful build (#11) 8 days 17 hr ago
- Last failed build (#12) 5.5 sec ago
- Last unstable build (#4) 11 days ago
- Last unsuccessful build (#12) 5.5 sec ago

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.

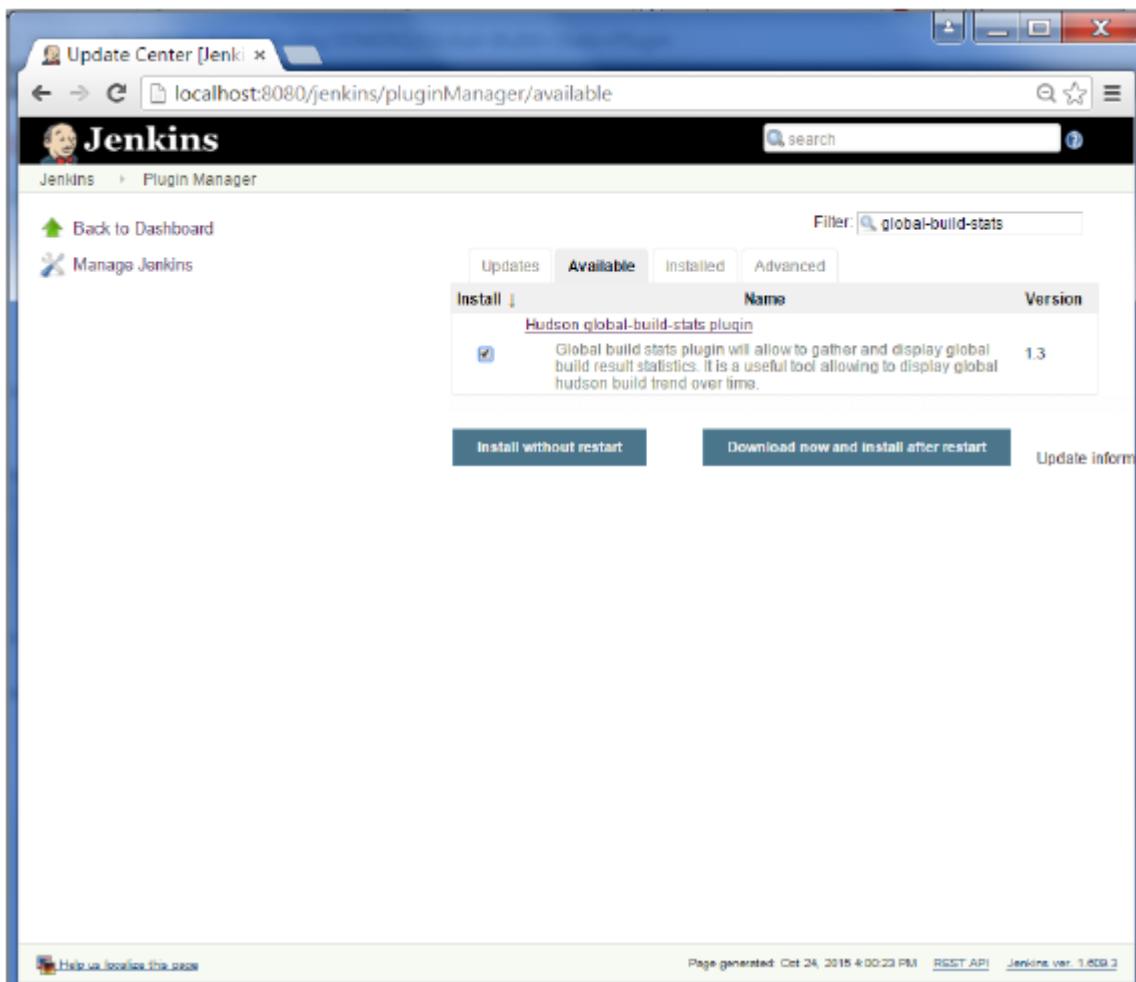
**Step 1** – Go to the Jenkins dashboard and click on Manage Jenkins



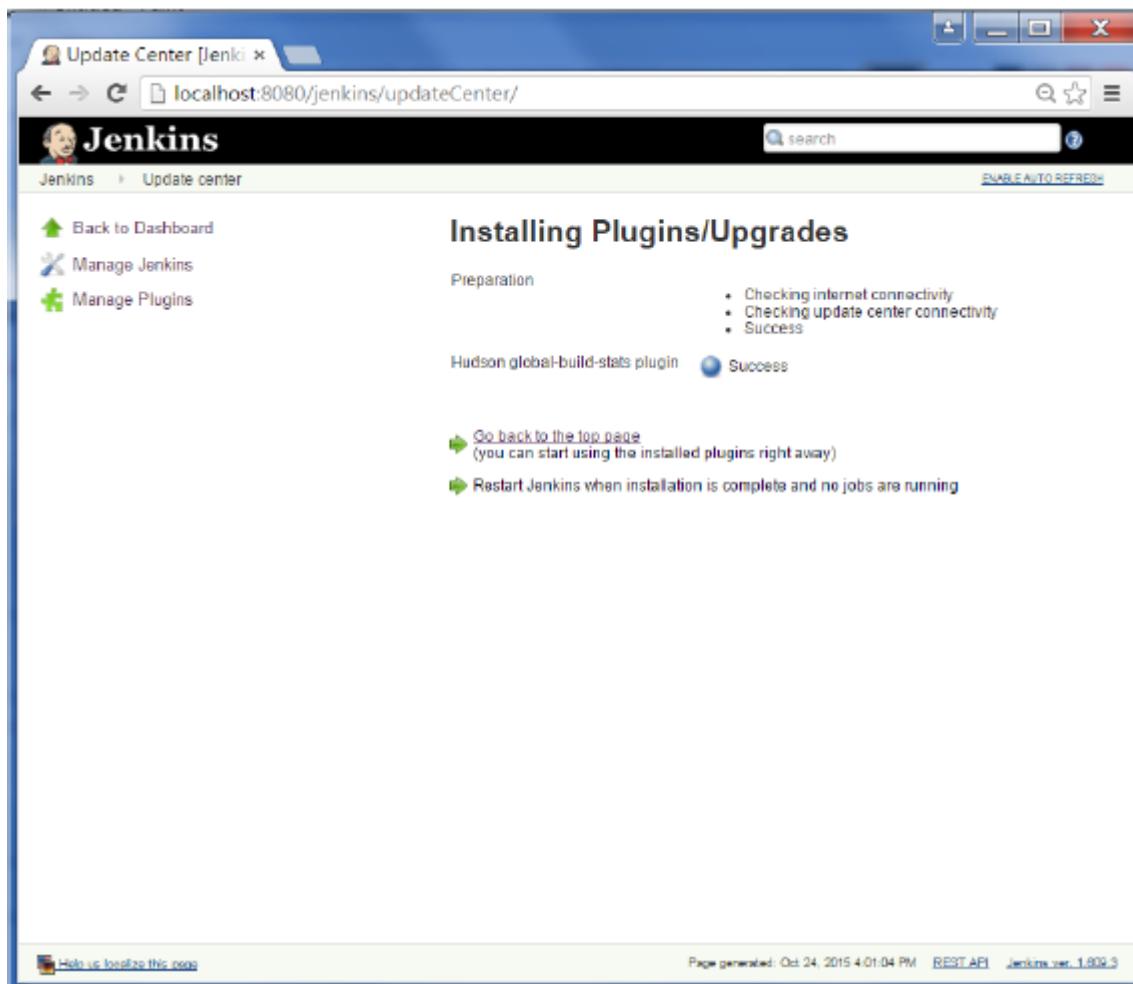
**Step 2** – Go to the Manage Plugins option



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

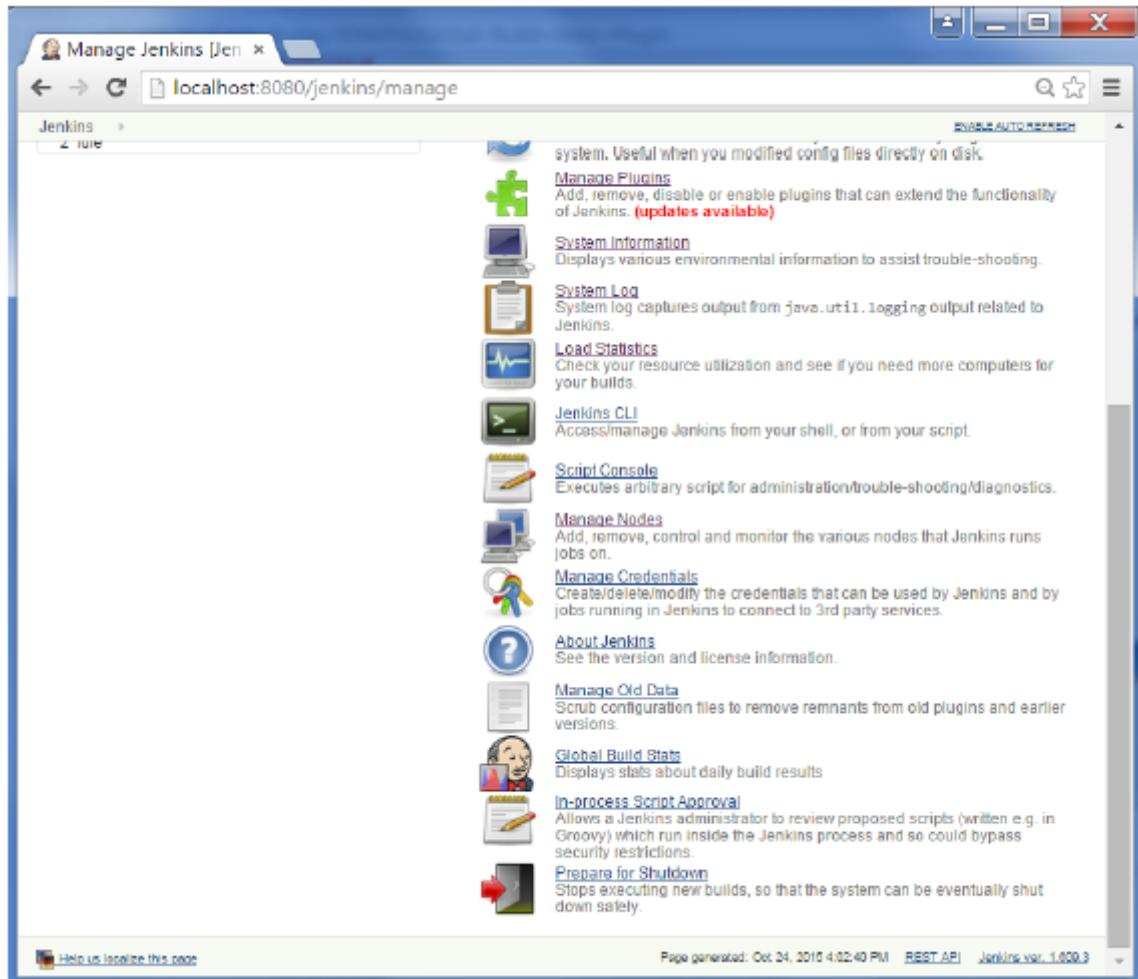


**Step 4** – The following screen shows up to confirm successful installation of the plugin. Restart the Jenkins instance.



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

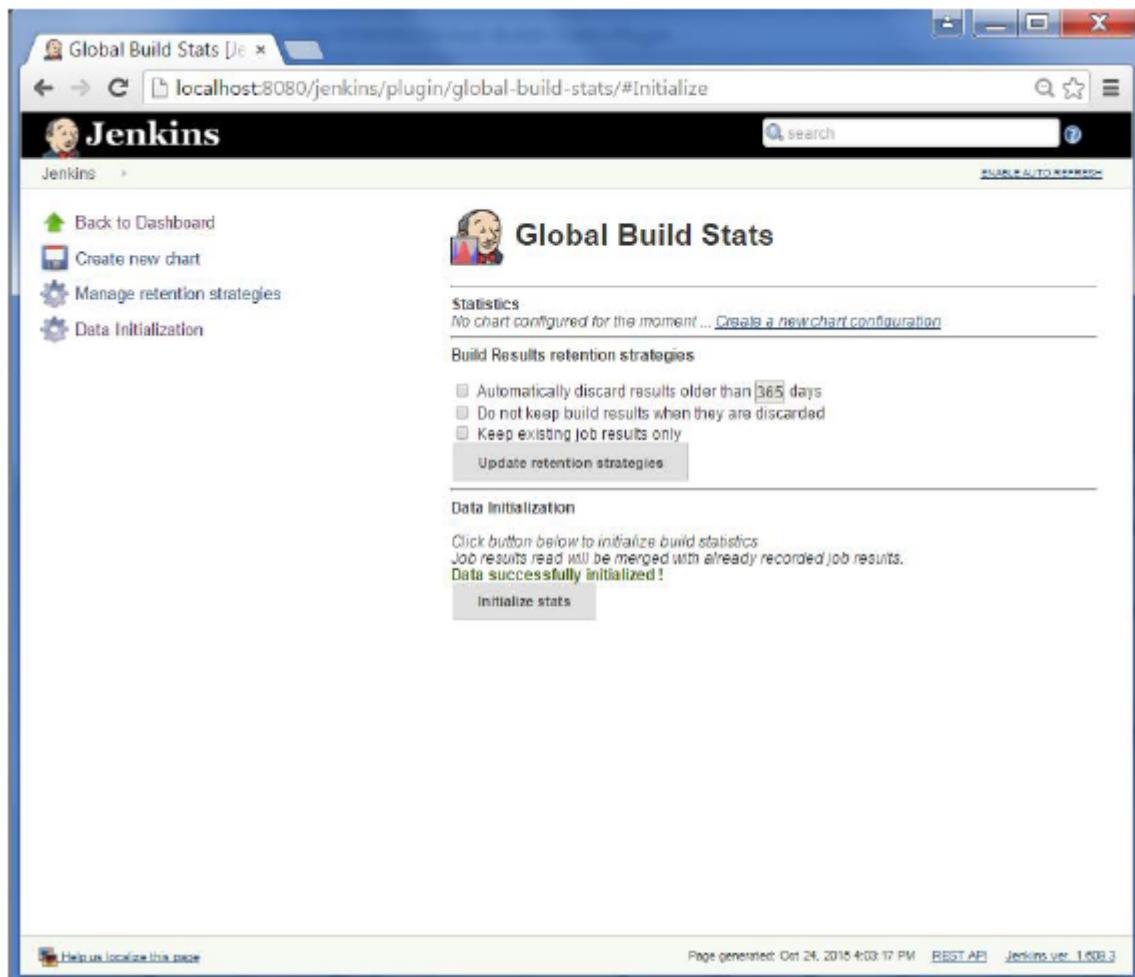
**Step 5** – Go to the Jenkins dashboard and click on Manage Jenkins. In the Manage Jenkins screen, scroll down and now you will now see an option called 'Global Build Stats'. Click on this link.



**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 is a navigation bar with links for 'Back to Dashboard', 'Create new chart', 'Manage retention strategies', and 'Data Initialization'. The main content area is titled 'Global Build Stats' and features a 'Statistics' section with a message: 'No chart configured for the moment ... [Create a new chart configuration](#)'. Below this is a 'Build Results retention strategies' section with three checkboxes: 'Automatically discard results older than 285 days', 'Do not keep build results when they are discarded', and 'Keep existing job results only'. A 'Update retention strategies' button is located below these checkboxes. The next section is 'Data Initialization', which contains a message: '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 is a footer with 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.

Global Build Stats | x

localhost:8080/jenkins/plugin/global-build-stats/#

**Jenkins**

Back to Dashboard | Create new chart | Manage retention strategies | Data Initialization

**Global Build Stats**

Statistics  
No chart configured for the moment ... [Create a new chart configuration](#)

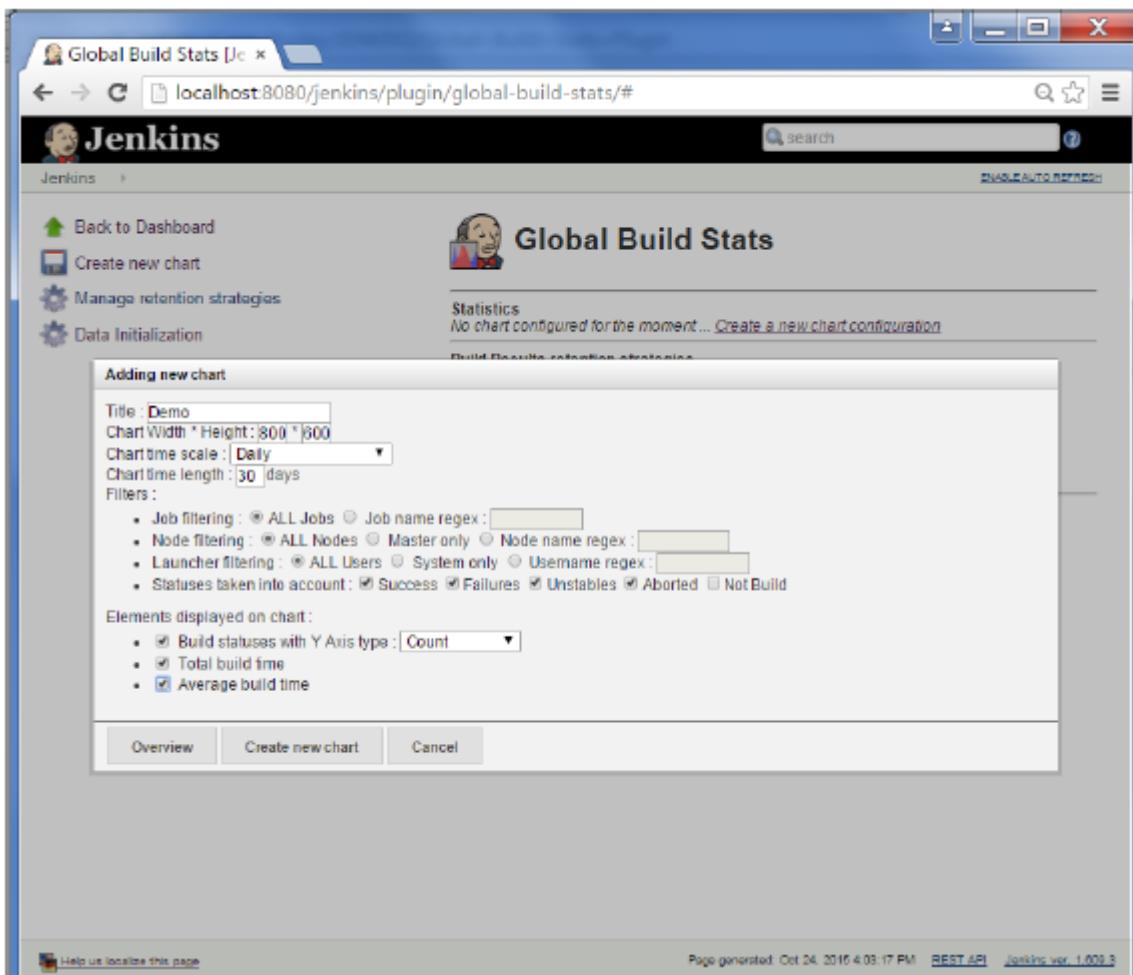
**Adding new chart**

Title : Demo  
Chart Width \* Height : 800 \* 600  
Chart time scale : Daily  
Chart time length : 30 days  
Filters :  
• Job filtering :  ALL Jobs  Job name regex :   
• Node filtering :  ALL Nodes  Master only  Node name regex :   
• Launcher filtering :  ALL Users  System only  Username regex :   
• Statuses taken into account :  Success  Failures  Unstables  Aborted  Not Build

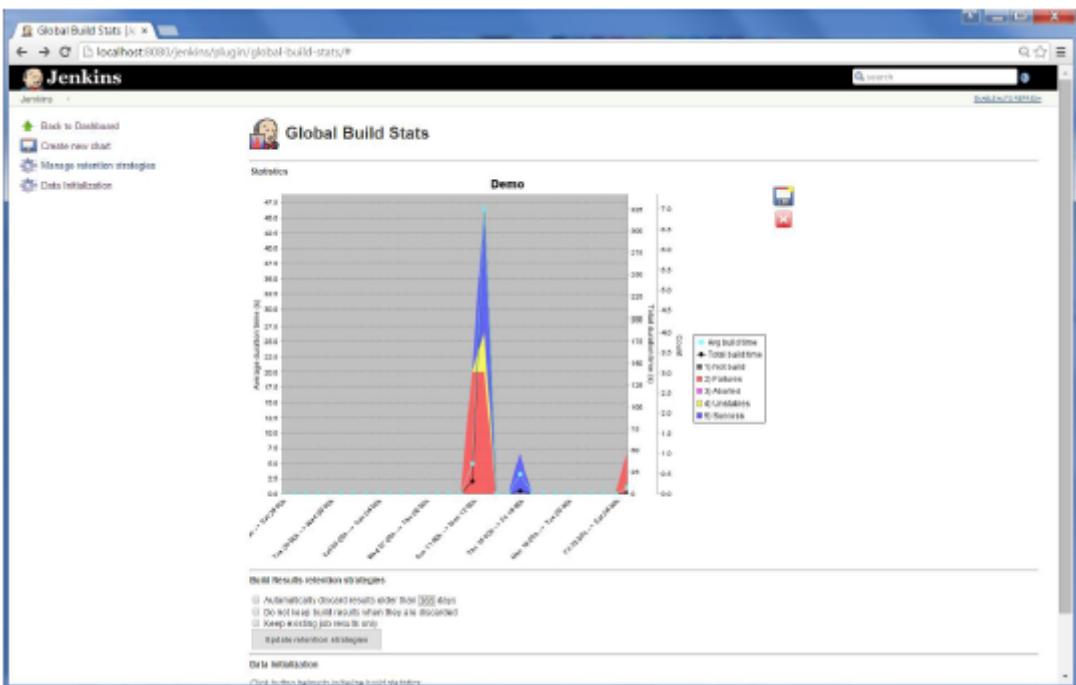
Elements displayed on chart :  
•  Build statuses with Y Axis type : Count  
•  Total build time  
•  Average build time

Overview | Create new chart | Cancel

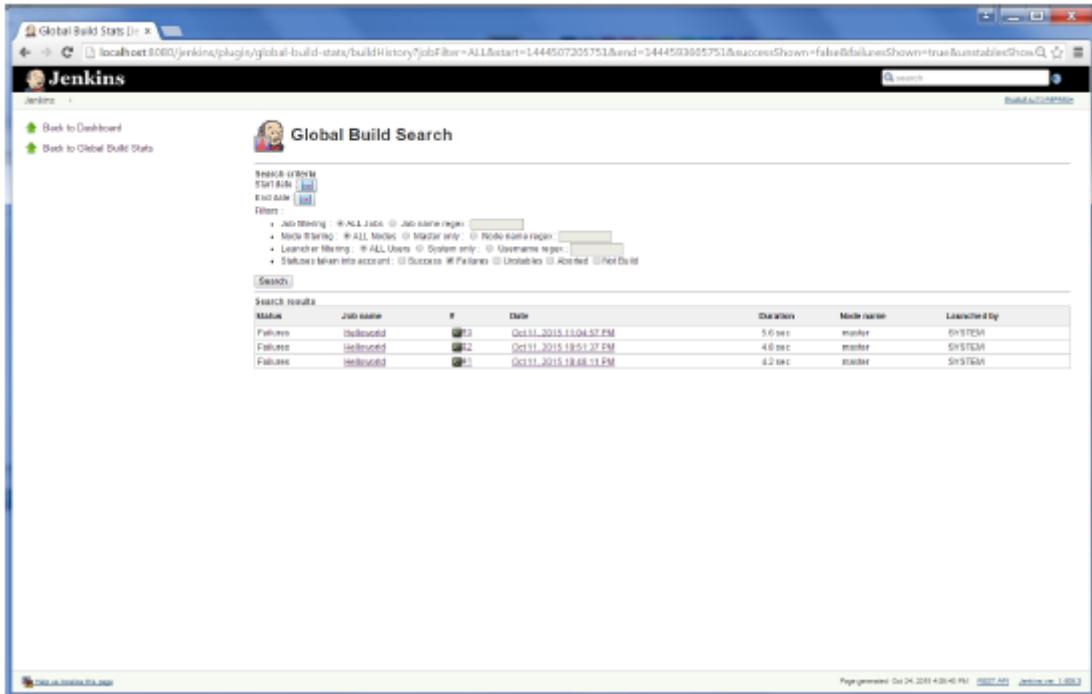
Help us localize this page | Page generated: Oct 24, 2015 4:03:17 PM | REST API | Jenkins ver. 1.609.3



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. The left sidebar has links for 'Back to Dashboard' and 'Back to Global Build Stats'. The main area is titled 'Global Build Search' with a sub-header 'Global Build Stats'. It includes sections for 'Search criteria', 'Start date' (with a date picker), 'End date' (with a date picker), and 'Filters'. The 'Filters' section contains several dropdowns and checkboxes. Below these is a 'Search' button. The main content area is titled 'Search Results' and shows a table with the following data:

Status	Job Name	#	Date	Duration	Master name	Last modified by
Failure	HelloWorld	13	Oct 11, 2015 11:04:57 PM	5.6 sec	master	SYSTEM
Failure	HelloWorld	12	Oct 11, 2015 11:51:37 PM	4.6 sec	master	SYSTEM
Failure	HelloWorld	11	Oct 11, 2015 11:48:11 PM	0.2 sec	master	SYSTEM

At the bottom of the page, there is a link 'Data in Jenkins XML page' and a footer with 'Page generated: Oct 24, 2015 4:00:40 PM' and 'Jenkins v1.603.3'.

# 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.

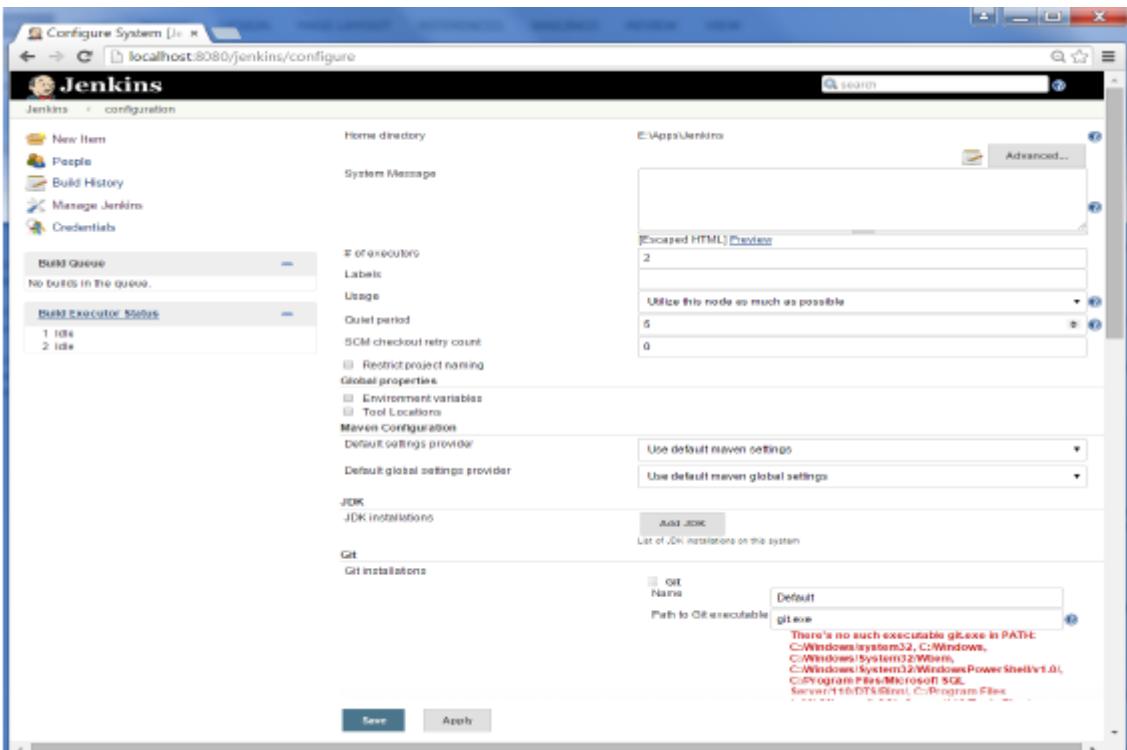
**http://localhost:8080/jenkins/exit** – shutdown jenkins

**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.



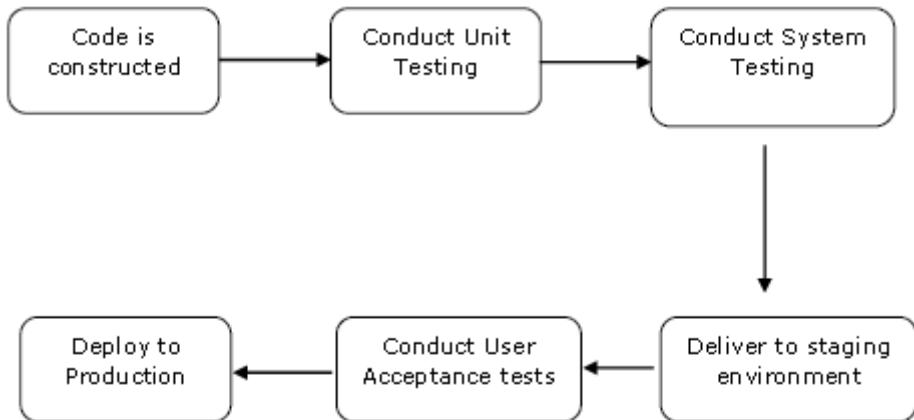
Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined

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.

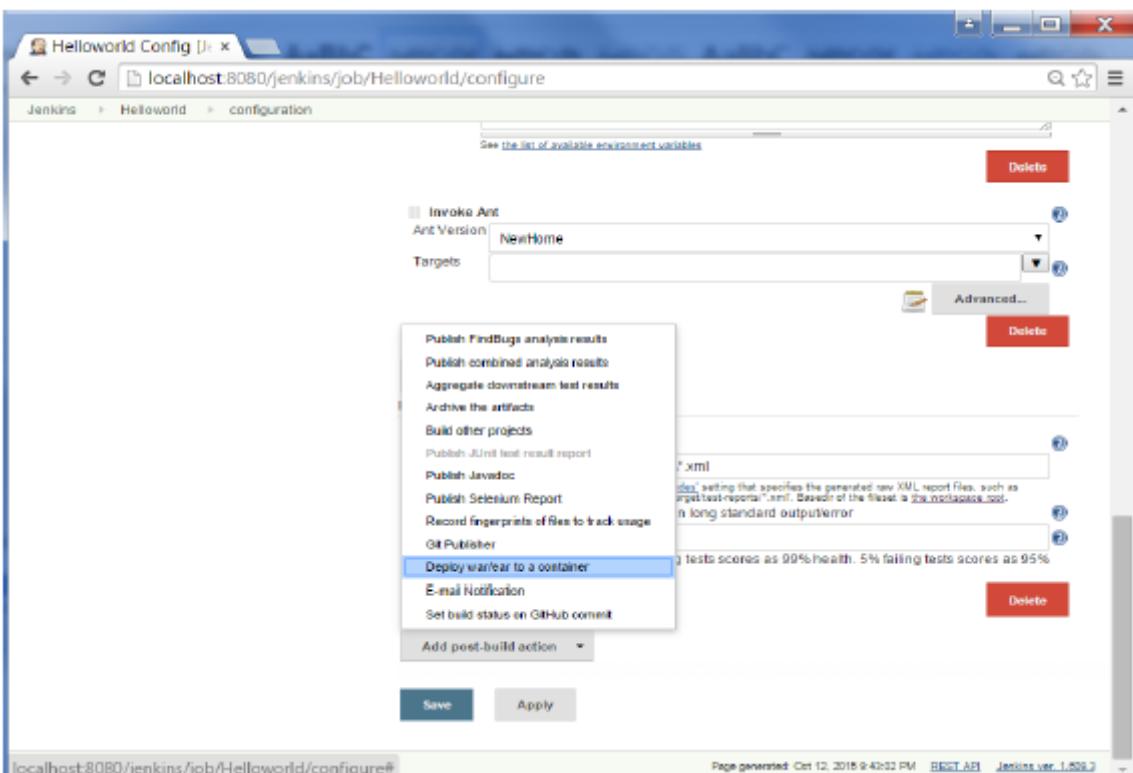
Another best practice is to write cron jobs or maintenance tasks that can carry out clean-up operations to avoid the disk where Jenkins is setup from becoming full.

# 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.

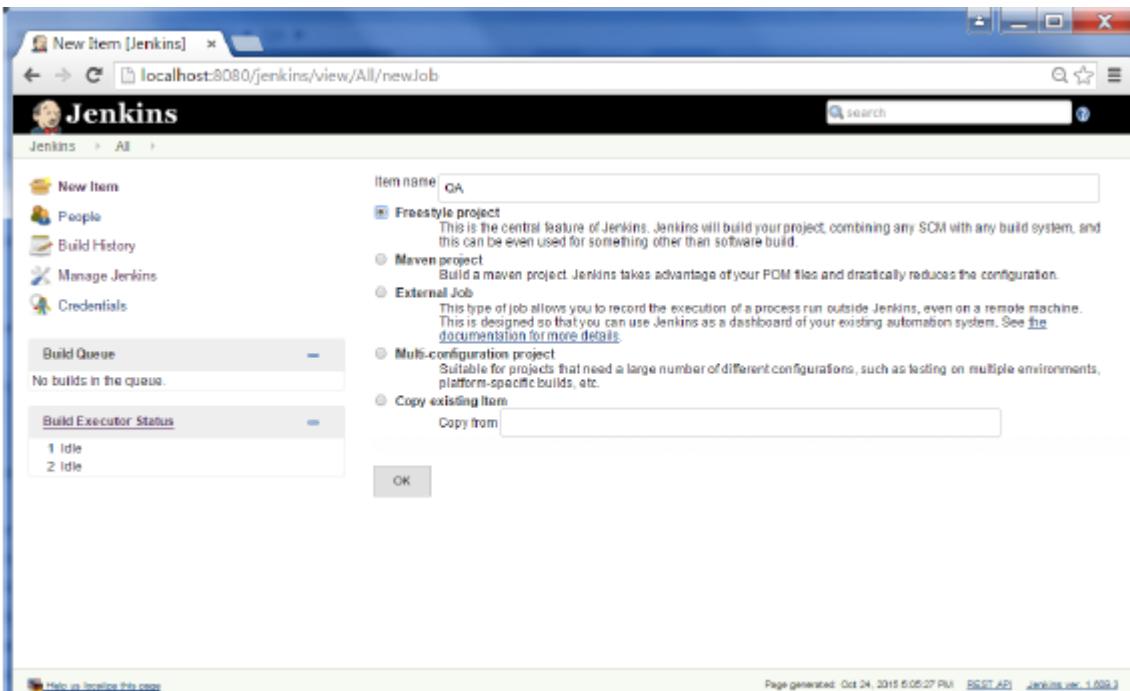


There are plugins available which can actually give you a graphical representation of the Continuous deployment process. But first lets

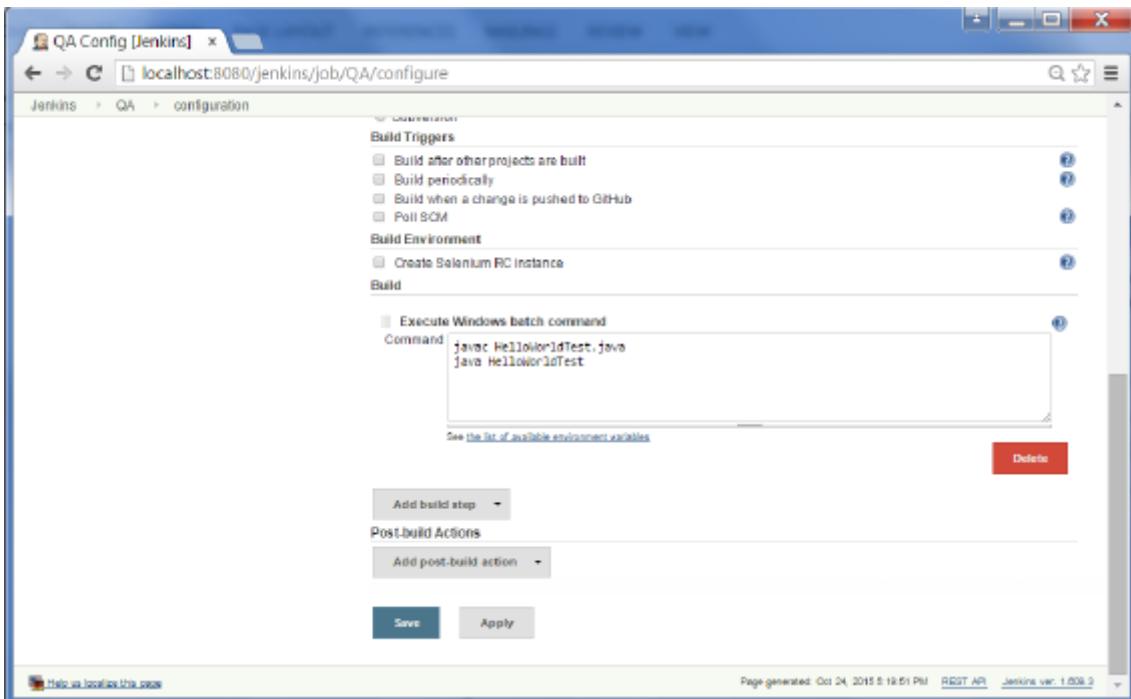
create another project in Jenkins, so that we can see best how this works.

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.



QA Config [Jenkins] ×

localhost:8080/jenkins/job/QA/configure

Jenkins > QA > configuration

Build Triggers

- Build after other projects are built
- Build periodically
- Build when a change is pushed to GitHub
- Poll SCM

Build Environment

- Create Selenium RC instance

Build

Execute Windows batch command

Command:

```
javac HelloWorldTest.java
java HelloWorldTest
```

See the list of available environment variables

Add build step

Post-build Actions

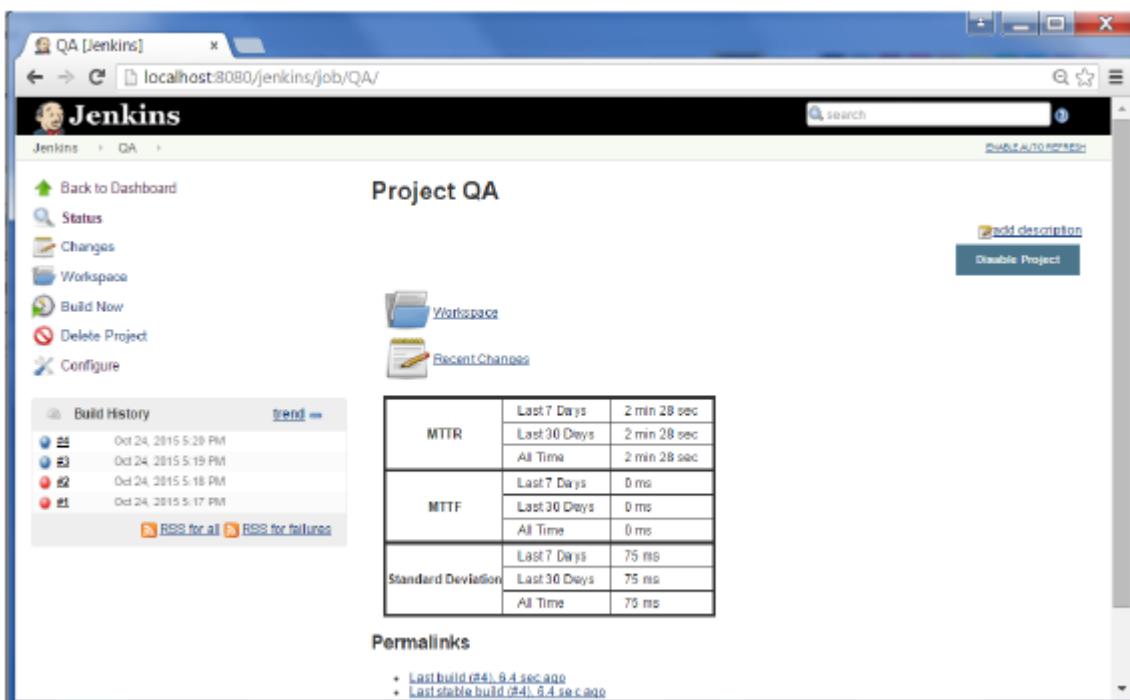
Add post-build action

Save Apply

Help us locate this issue

Page generated: Oct 24, 2015 5:18:51 PM REST API Jenkins ver. 1.609.2

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



QA [Jenkins] ×

localhost:8080/jenkins/job/QA/

Jenkins > QA >

Project QA

Back to Dashboard

Status

Changes

Workspace

Build Now

Delete Project

Configure

Build History

trend →

Oct 24, 2015 5:29 PM

Oct 24, 2015 5:19 PM

Oct 24, 2015 5:18 PM

Oct 24, 2015 5:17 PM

RSS for all RSS for failures

Workspace

Recent Changes

MTTR

	Last 7 Days	2 min 28 sec
Last 30 Days	2 min 28 sec	
All Time	2 min 28 sec	

MTTF

	Last 7 Days	0 ms
Last 30 Days	0 ms	
All Time	0 ms	

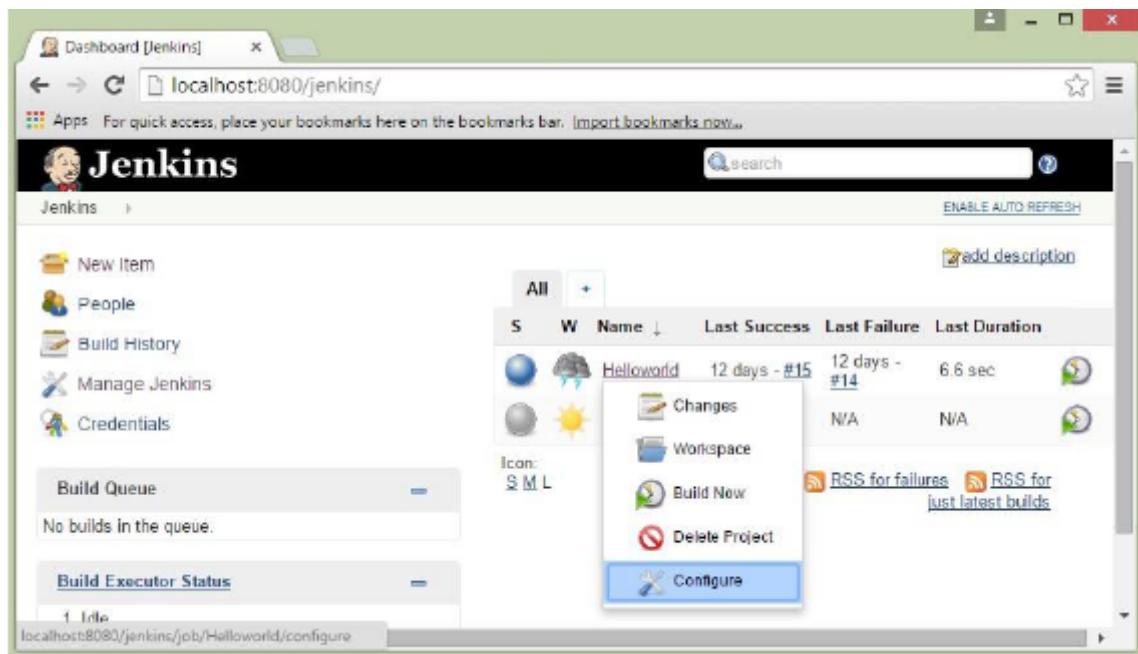
Standard Deviation

	Last 7 Days	75 ms
Last 30 Days	75 ms	
All Time	75 ms	

Permalinks

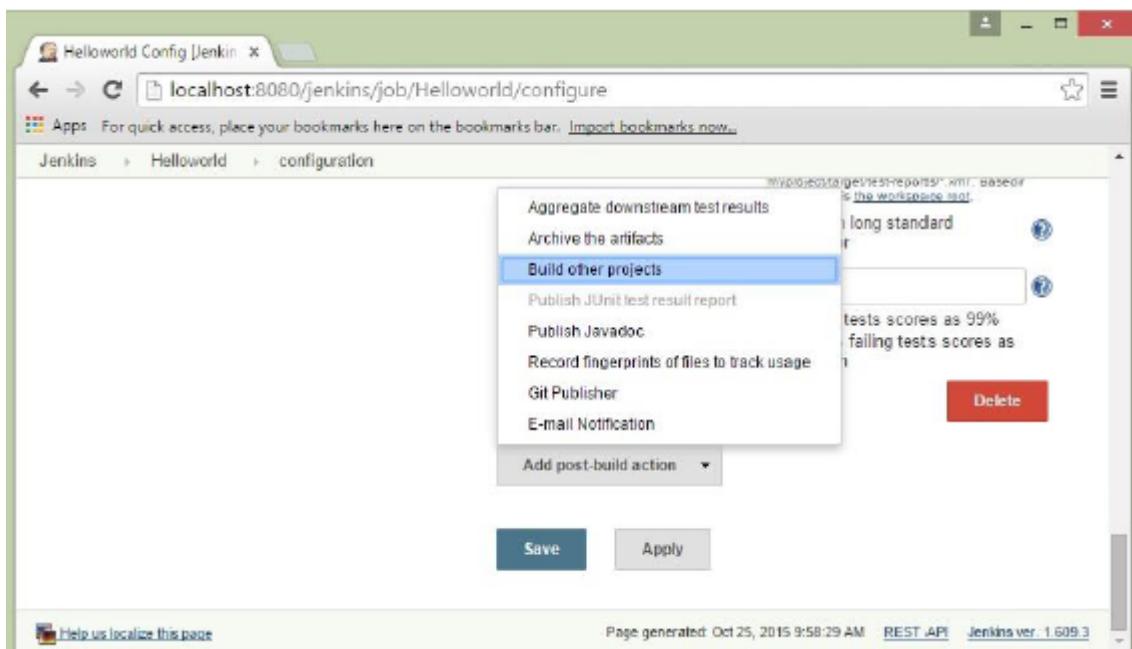
- Last build (24), 6.4 sec ago
- Last stable build (24), 6.4 sec ago

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



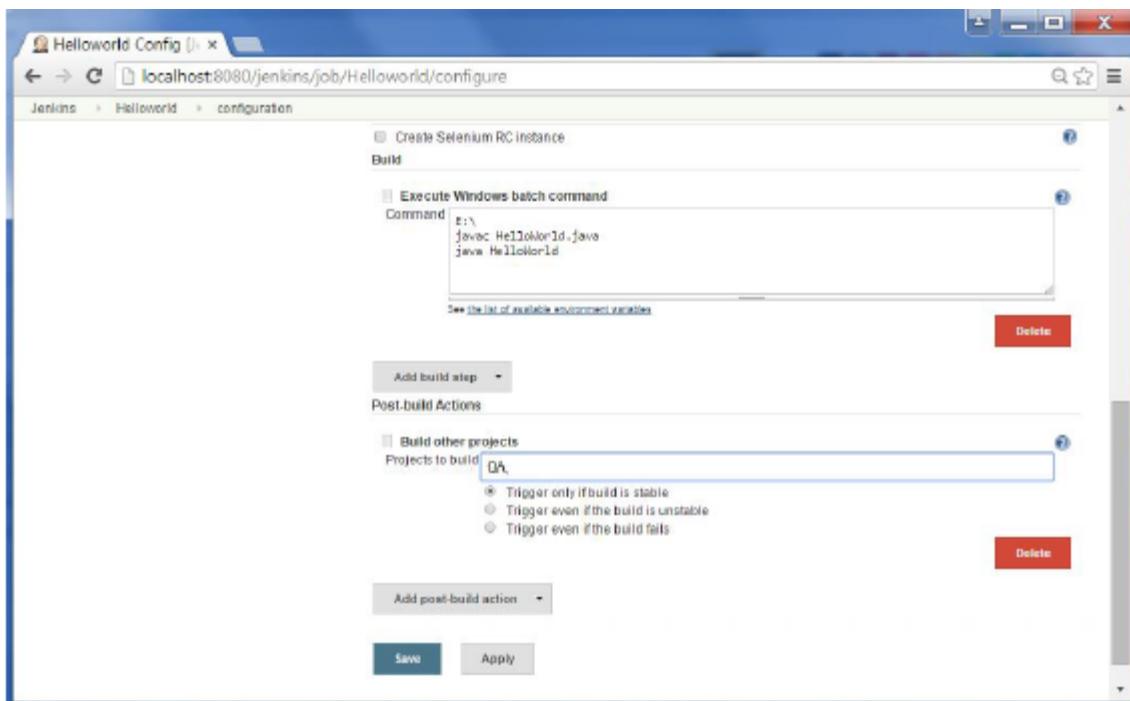
The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status'. The main area displays a table of jobs. The 'Helloworld' job is listed with the following details: Name: Helloworld, Last Success: 12 days - #15, Last Failure: 12 days - #14, Last Duration: 6.6 sec. Below the table are links for 'RSS for failures' and 'RSS for just latest builds'. The 'Configure' button for the 'Helloworld' job is highlighted with a blue box.

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

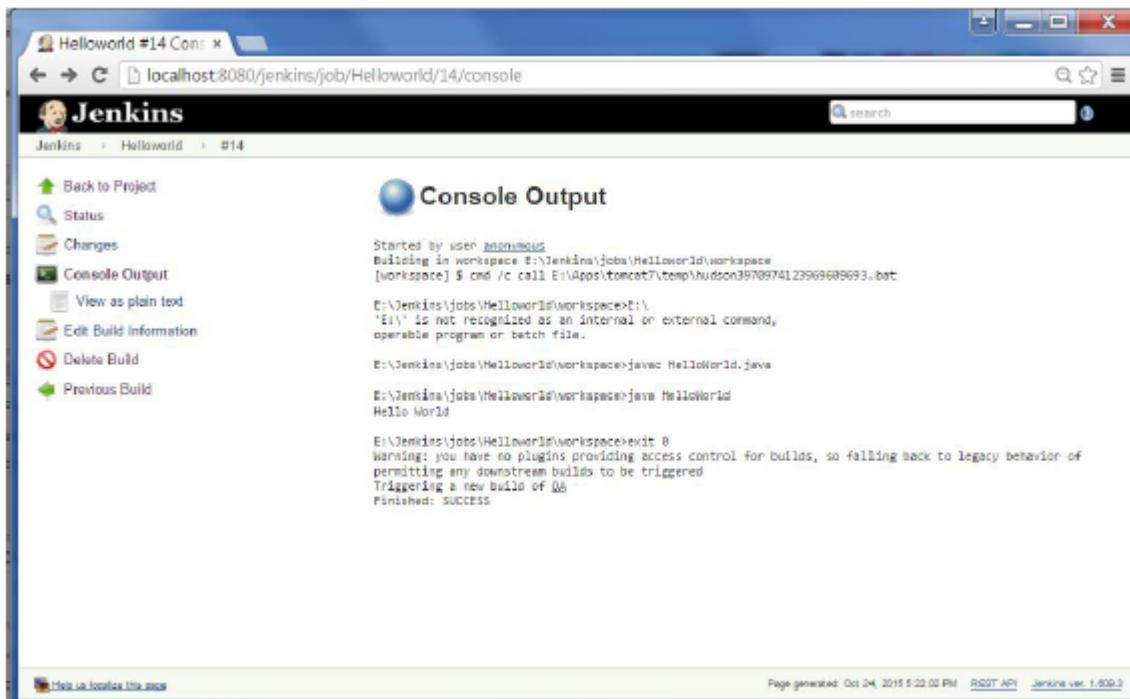


The screenshot shows the configuration page for the 'Helloworld' job. The URL is 'localhost:8080/jenkins/job/Helloworld/configure'. The 'configuration' section is open. A dropdown menu under 'Add post-build action' shows several options: 'Aggregate downstream test results', 'Archive the artifacts', 'Build other projects' (which is highlighted with a blue box), 'Publish JUnit test result report', 'Publish Javadoc', 'Record fingerprints of files to track usage', 'Git Publisher', and 'E-mail Notification'. Below the dropdown are 'Save' and 'Apply' buttons. To the right of the dropdown, there is a section for 'Post-build triggers' with a 'Delete' button.

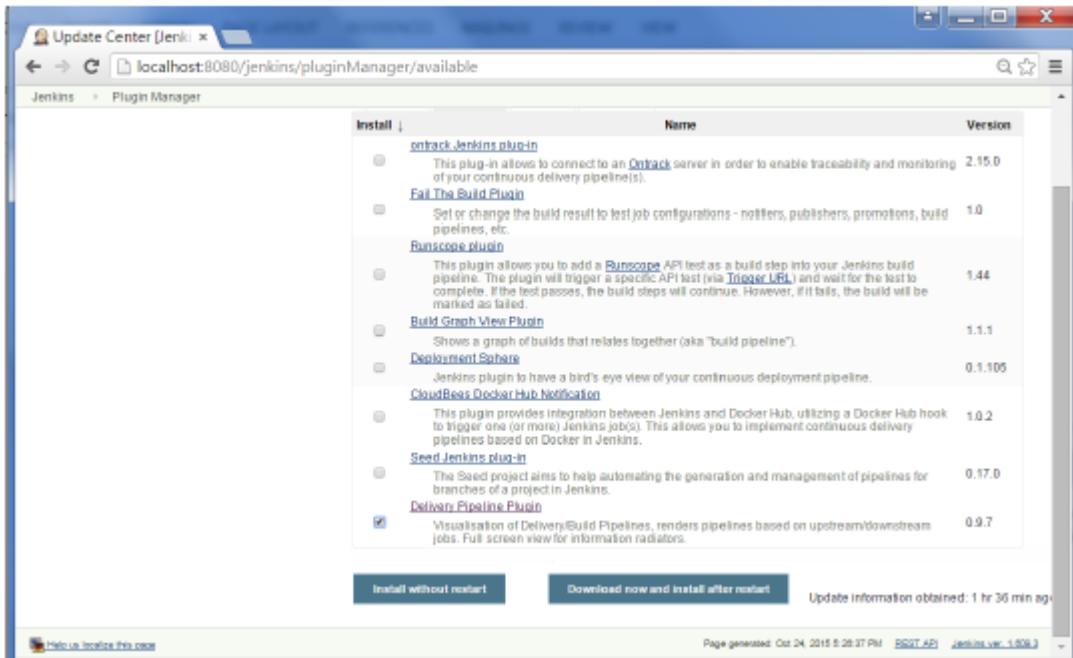
**Step 5** – In the ‘Project to build’ section, enter QA as the project name to build. You can leave the option as default of ‘Trigger only if build is stable’. Click on the Save button.



**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 Plugin's. In the available tab, search for 'Delivery Pipeline Plugin'. Click On Install without Restart. Once done, restart the Jenkins instance.

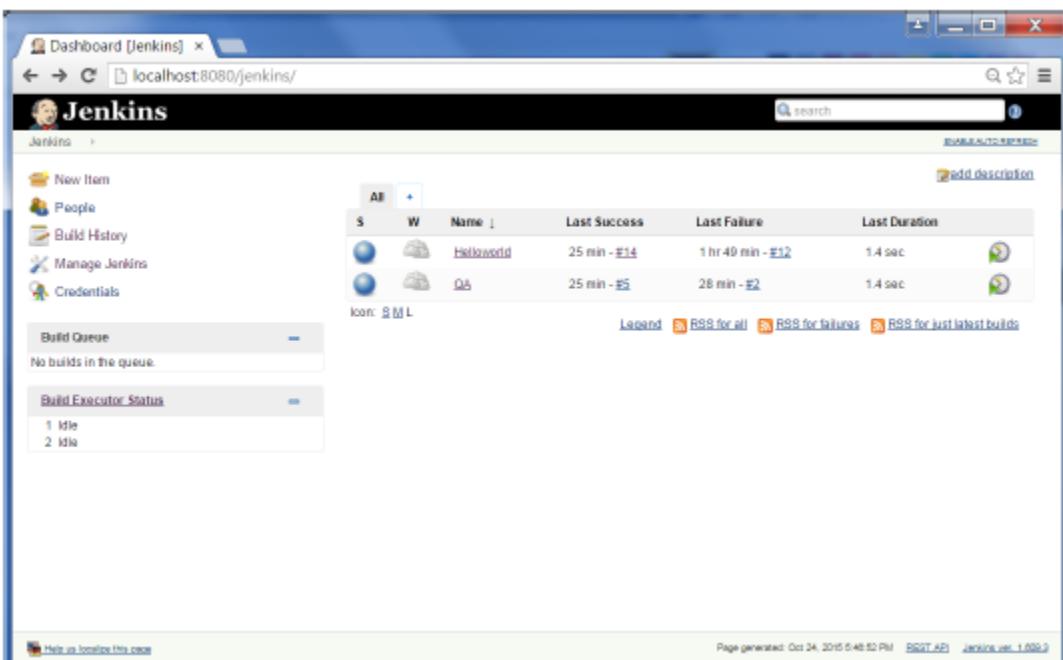


The screenshot shows the Jenkins Update Center interface. The 'available' tab is selected. A list of available plugins is displayed, with the 'Delivery Pipeline Plugin' checked for installation. The plugin details are as follows:

Name	Version
ontrack Jenkins plugin	2.15.0
Fail The Build Plugin	1.0
Runscope plugin	1.44
Build Graph View Plugin	1.1.1
Deployment Square	0.1.105
CloudBees Docker Hub Notification	1.0.2
Seed Jenkins plugin	0.17.0
<b>Delivery Pipeline Plugin</b>	0.9.7

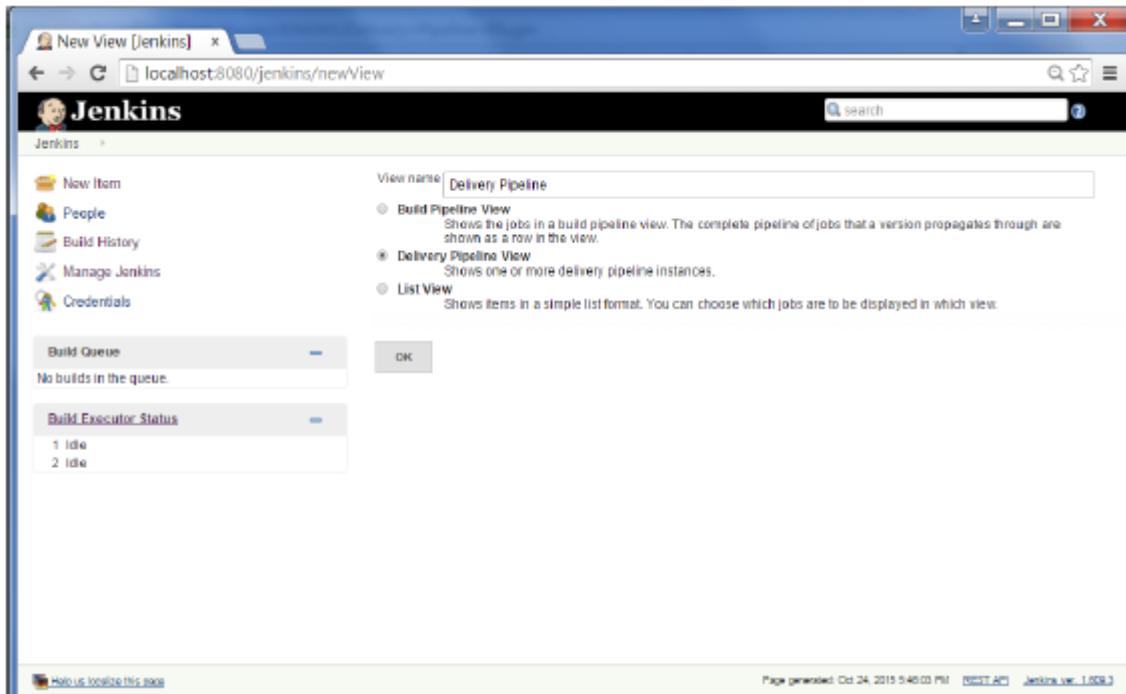
At the bottom, there are two buttons: 'Install without restart' and 'Download now and install after restart'. A status message indicates 'Update information obtained: 1 hr 36 min ago'.

**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.



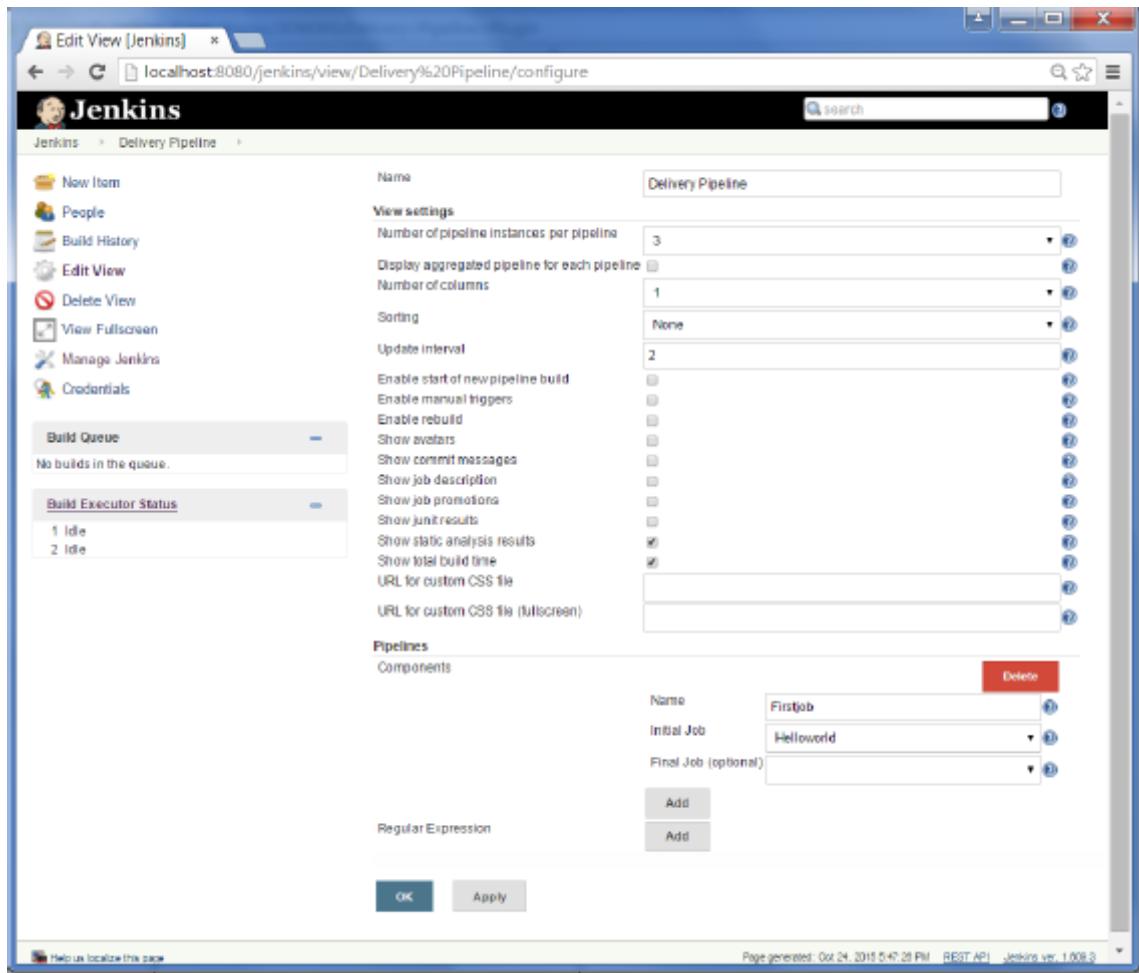
The screenshot shows the Jenkins Dashboard. The 'All' tab is selected. On the right, a table displays Jenkins jobs: 'HelloWorld' and 'QA'. Below the table, there are tabs for 'Build Queue' (which is empty) and 'Build Executor Status' (which shows 1 Idle and 2 Idle). At the bottom right, there is a 'Delivery Pipeline View' tab.

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

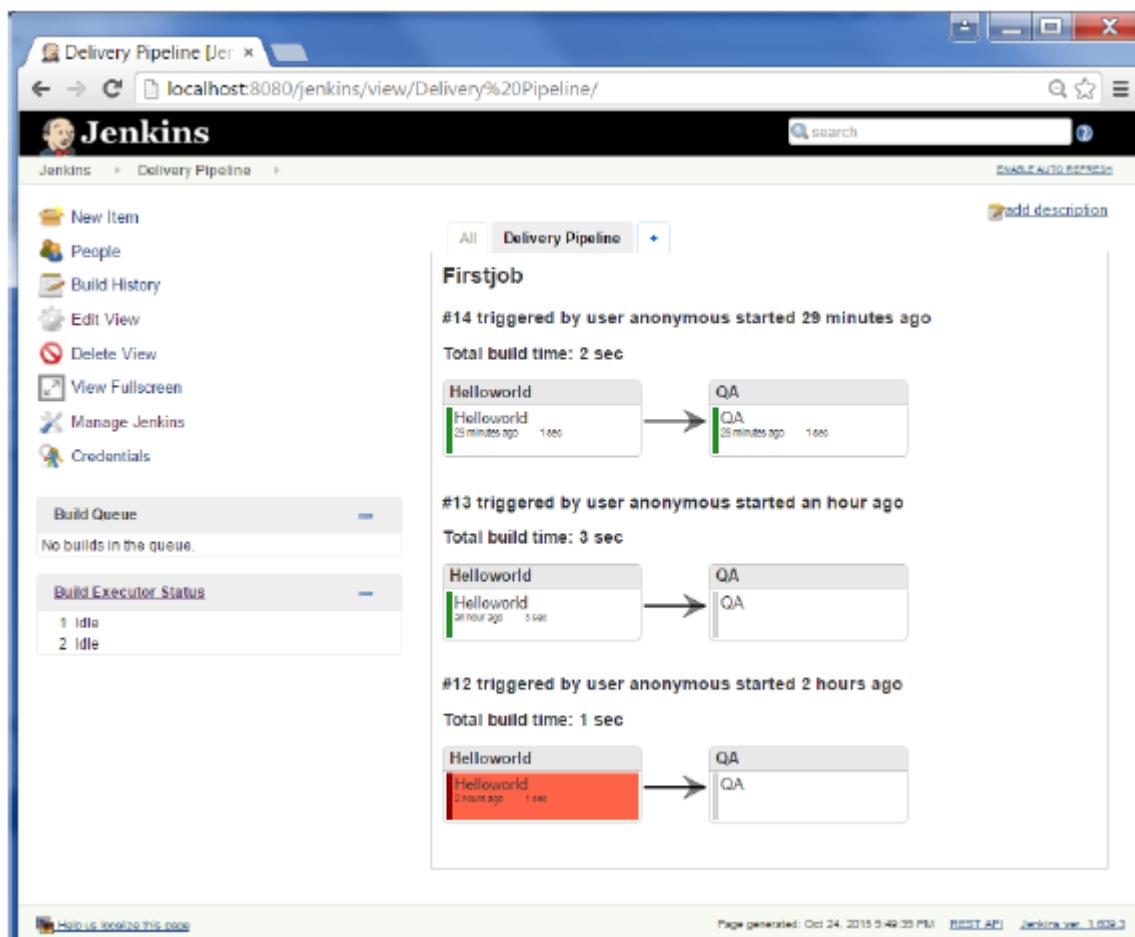


**Step 10** – In the next screen, you can leave the default options. One can change the following settings –

- 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.

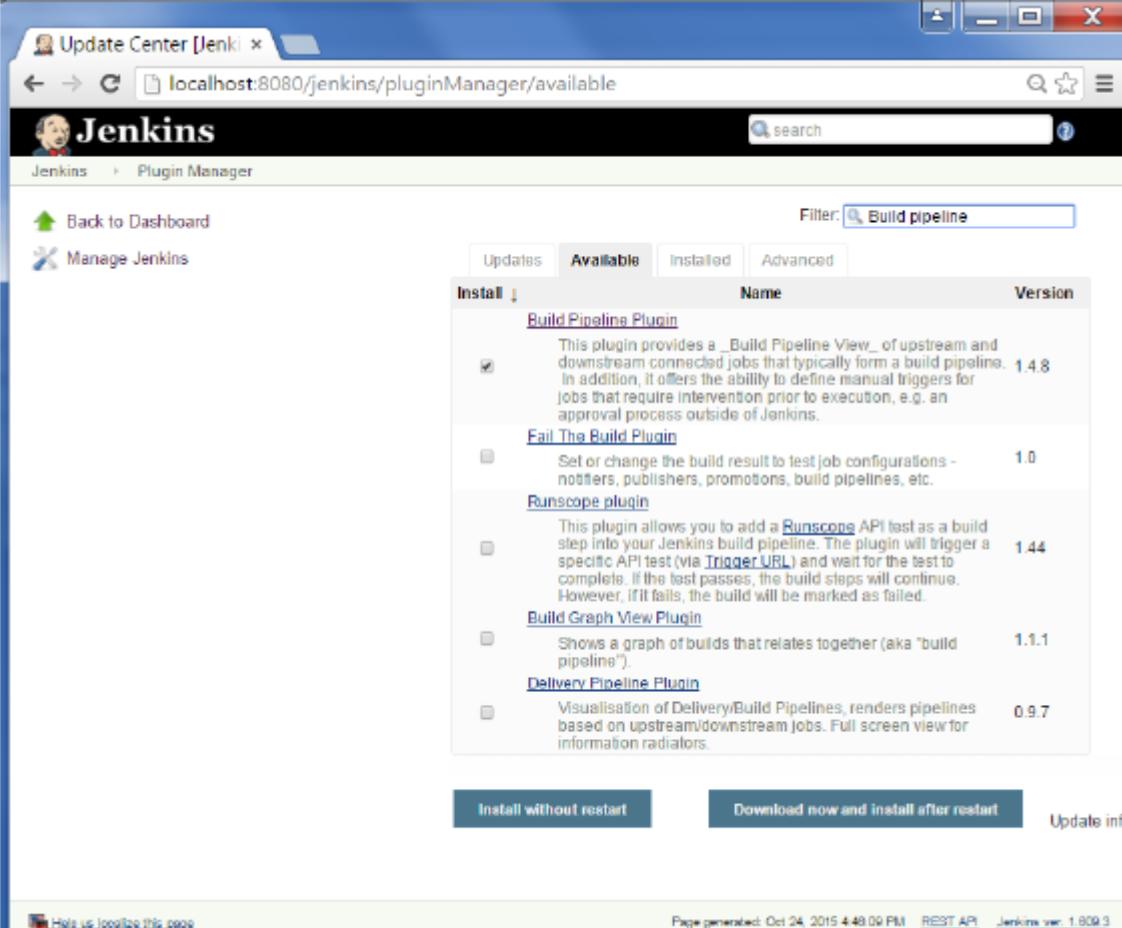


You will now see a great view of the entire delivery pipeline and you will be able to see the status of each project in the entire pipeline.



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.



Update Center [Jenkins] localhost:8080/jenkins/pluginManager/available

**Jenkins** Plugin Manager

Back to Dashboard Manage Jenkins

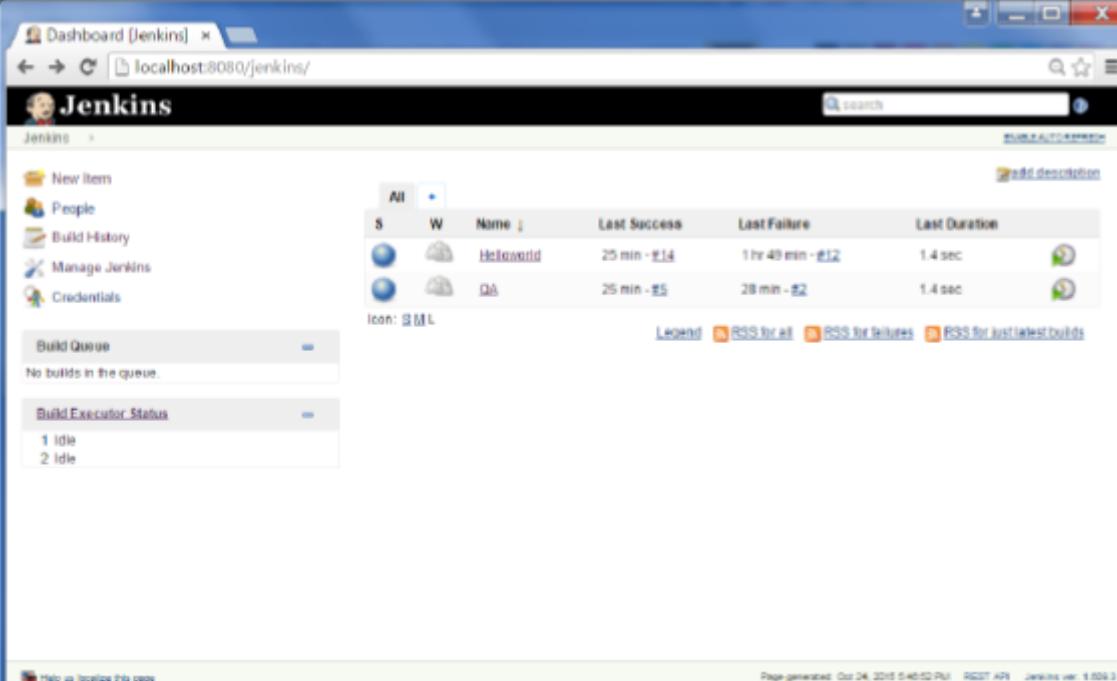
Updates Available Installed Advanced Filter:

Install	Name	Version
<input checked="" type="checkbox"/>	<a href="#">Build Pipeline Plugin</a> This plugin provides a <code>_Build Pipeline View_</code> 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 <code>Runscope</code> API test as a build step into your Jenkins build pipeline. The plugin will trigger a specific API test (via <code>Trigger URL</code> ) 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 pipelines").	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

[Install without restart](#) [Download now and install after restart](#) [Update info](#)

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

**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.



Dashboard [Jenkins] localhost:8080/jenkins/

**Jenkins**

New Item add description

People

Build History

Manage Jenkins

Credentials

Build Queue No builds in the queue.

Build Executor Status 1 Idle 2 Idle

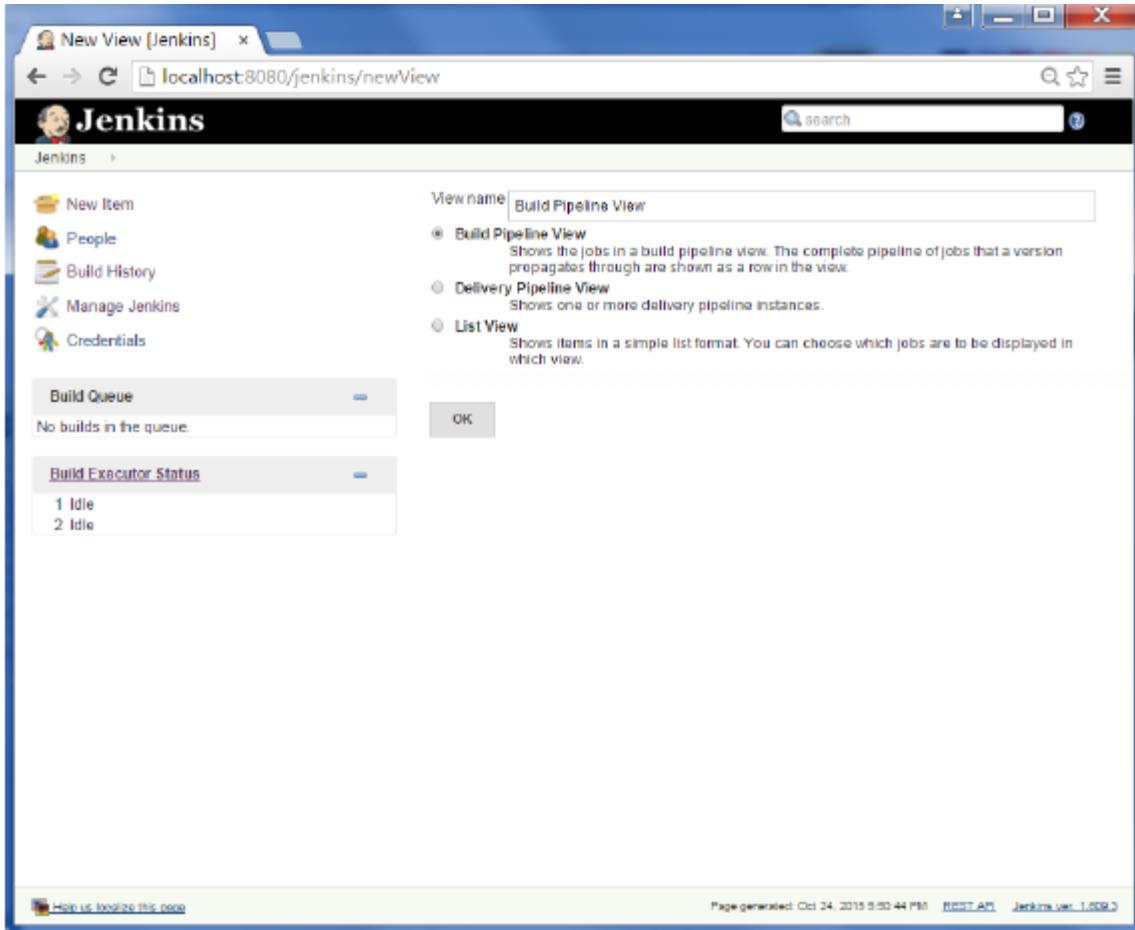
All	W	Name	Last Success	Last Failure	Last Duration
<input checked="" type="checkbox"/>	<input type="checkbox"/>	Helleworld	25 min - 14	1 hr 48 min - 52	1.4 sec
<input type="checkbox"/>	<input type="checkbox"/>	QA	25 min - 5	28 min - 52	1.4 sec

Icon:  

Legend:  RSS for all  RSS for failures  RSS for last failed builds

Help us localize this page Page generated: Oct 24, 2015 5:48:52 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.

localhost:8080/jenkins/view/Build%20Pipeline%20View/configure

**Jenkins**

New Item People Build History Edit View Delete View Manage Jenkins Credentials

**Build Queue**  
No builds in the queue.

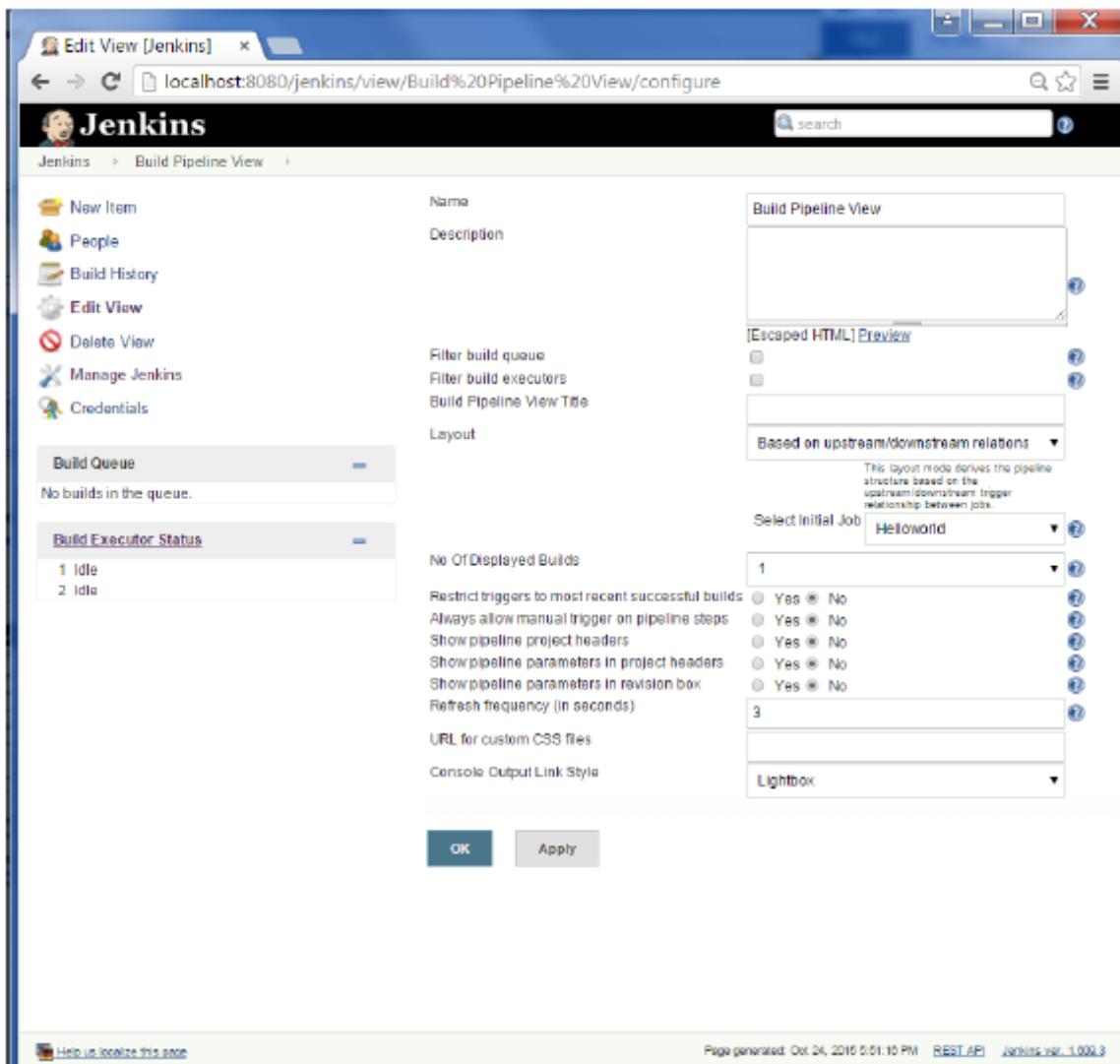
**Build Executor Status**  
1 Idle  
2 Idle

**Build Pipeline View**

Name: Build Pipeline View  
Description:   
[Escaped HTML] Preview  
Filter build queue  
Filter build executors  
Build Pipeline View Title  
Layout: Based on upstream/downstream relations  
This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs.  
Select Initial Job: HelloWorld  
1  
2  
3  
URL for custom CSS files  
Console Output Link Style: Lightbox

OK Apply

Help us localize this page Page generated: Oct 24, 2016 5:01:10 PM REST API Jenkins ver. 1.000.3



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.

localhost:8080/jenkins/view/Build%20Pipeline%20View/

**Jenkins**

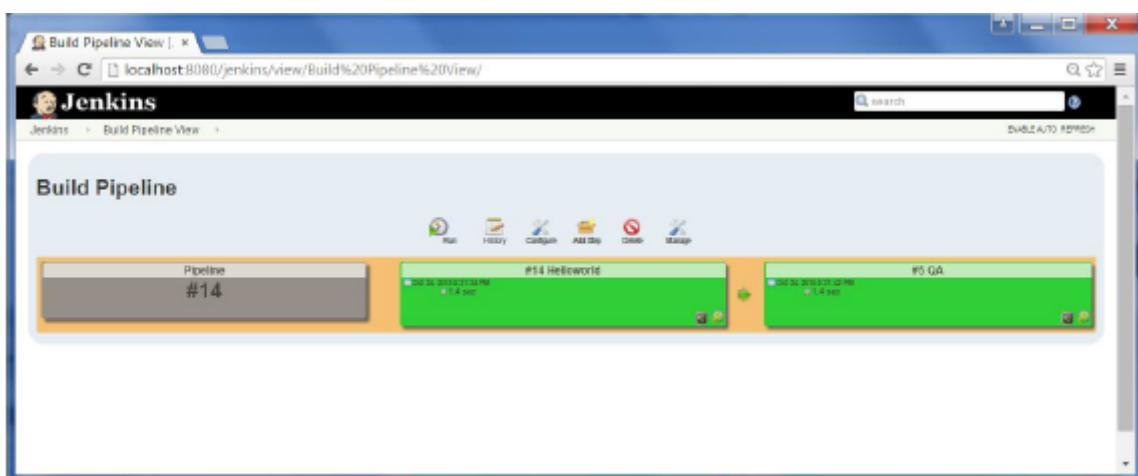
Build Pipeline View

Build Pipeline

Pipeline #14

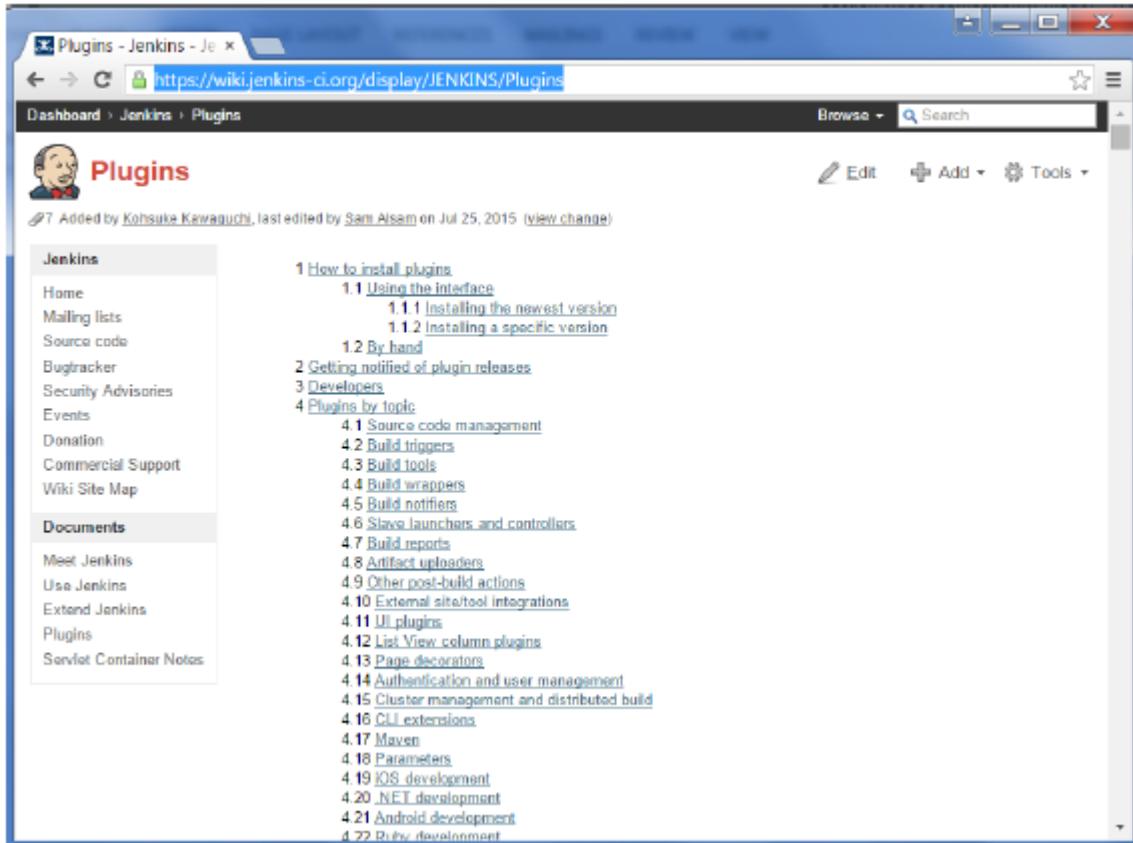
#14 HelloWorld

#15 QA



# 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>

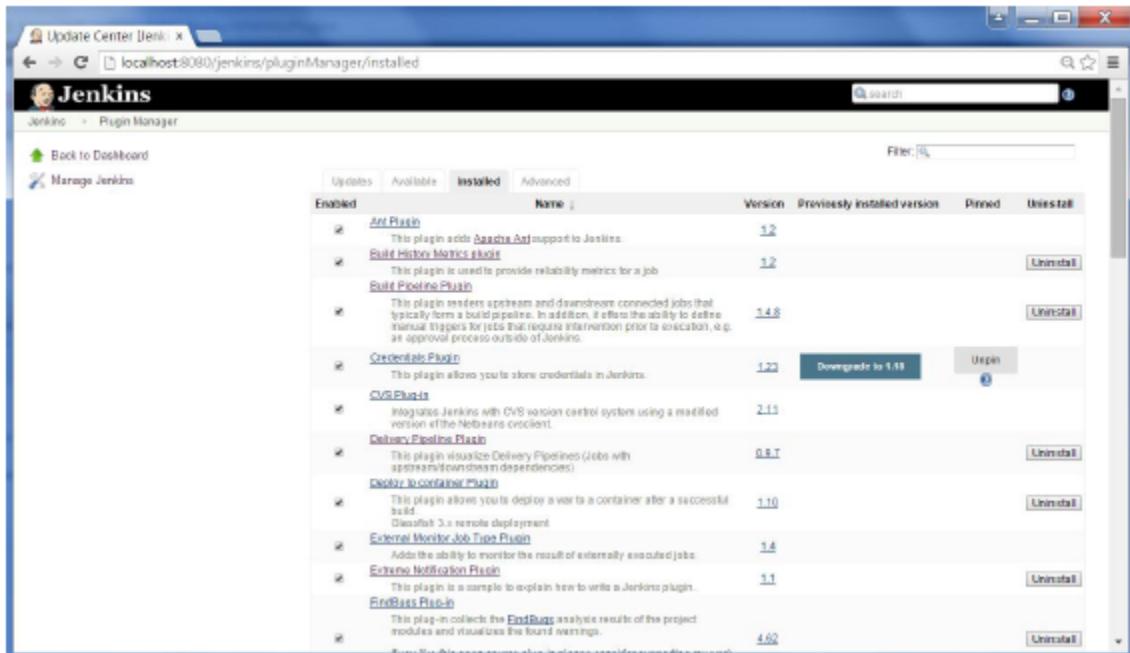


The screenshot shows a web browser window with the title 'Plugins - Jenkins - Je'. The URL in the address bar is <https://wiki.jenkins-ci.org/display/JENKINS/Plugins>. The page content is organized into sections: 'Jenkins' (Home, Mailing lists, Source code, Bugtracker, Security Advisories, Events, Donation, Commercial Support, Wiki Site Map) and 'Documents' (Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, Servlet Container Notes). The main content area is titled 'Plugins' and contains a table of contents for plugin management, including 'How to install plugins' (Using the interface, By hand), 'Getting notified of plugin releases', 'Developers', and 'Plugins by topic' (Source code management, Build triggers, Build tools, Build wrappers, Build notifiers, Slave launchers and controllers, Build reports, Artifact uploaders, 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, 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

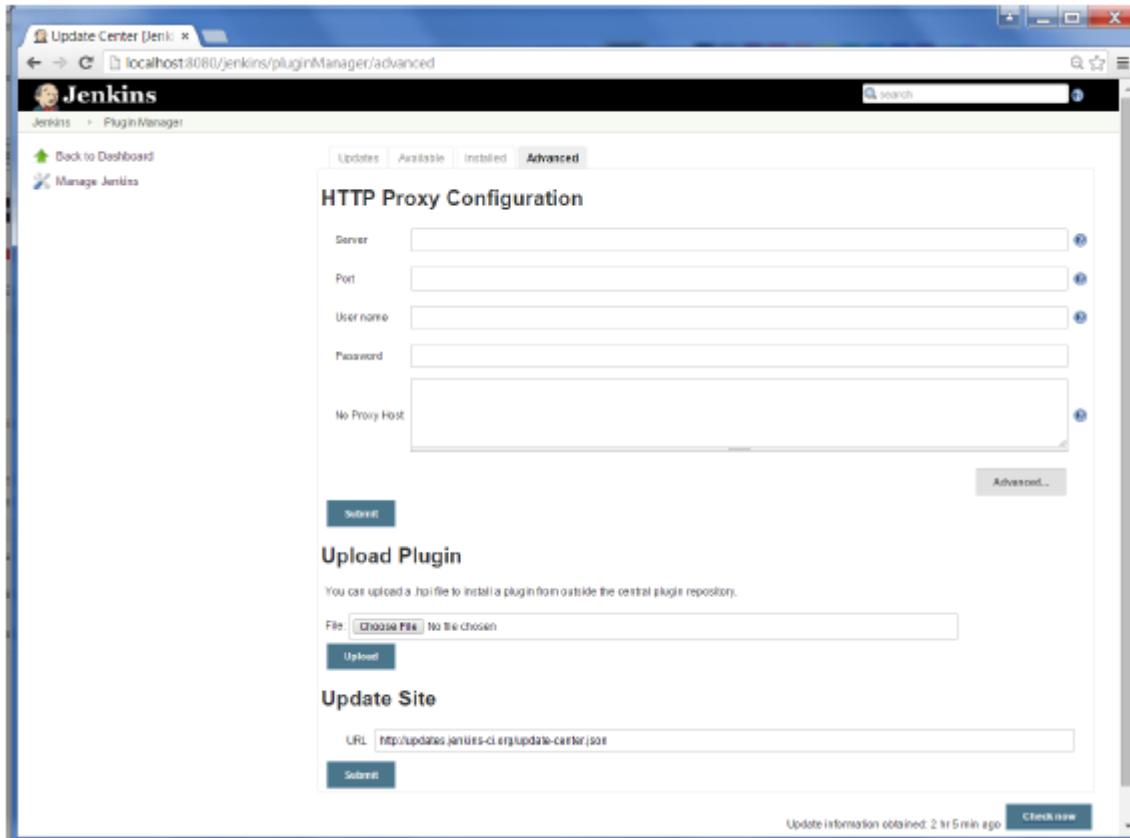
To uninstall a plugin, Go to Manage Jenkins → Manage plugins. Click on the Installed tab. Some of the plugins will have the Uninstall option. You can click these buttons to uninstall the plugins. Ensure to restart your Jenkins instance after the uninstallation.



Enabled	Name	Version	Previously installed version	Pinned	Uninstall
<input checked="" type="checkbox"/>	<a href="#">Ant Plugin</a> This plugin adds <a href="#">Apache Ant</a> support to Jenkins.	1.2			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	<a href="#">Build History Metrics Plugin</a> This plugin is used to provide reliability metrics for a job.	1.2			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	<a href="#">Build Pipeline Plugin</a> This plugin renders 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			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	<a href="#">Credentialed Plugin</a> This plugin allows you to store credentials in Jenkins.	1.22	<a href="#">Downgrade to 1.19</a>	<a href="#">Unpin</a>	<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	<a href="#">CVS Plugin</a> Integrates Jenkins with CVS version control system using a modified version of the NetBeans client.	2.1.1			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	<a href="#">Delivery Pipeline Plugin</a> This plugin visualizes Delivery Pipelines (Jobs with upstream/downstream dependencies).	0.8.7			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	<a href="#">Deploy to container Plugin</a> This plugin allows you to deploy a war to a container after a successful build.	1.10			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	<a href="#">External Monitor Job Type Plugin</a> Allows the ability to monitor the result of externally executed jobs.	1.4			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	<a href="#">Extreme Notification Plugin</a> This plugin is a sample to explain how to write a Jenkins plugin.	1.1			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	<a href="#">FindBugs Plugin</a> This plugin collects the <a href="#">FindBugs</a> analytic results of the project modules and visualizes the found warnings.	4.6.0			<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.



**HTTP Proxy Configuration**

Server:   
Port:   
User name:   
Password:   
No Proxy Host:

**Upload Plugin**

You can upload a JAR file to install a plugin from outside the central plugin repository.

File:  No file chosen  
[Upload](#)

**Update Site**

URL:   
[Submit](#)

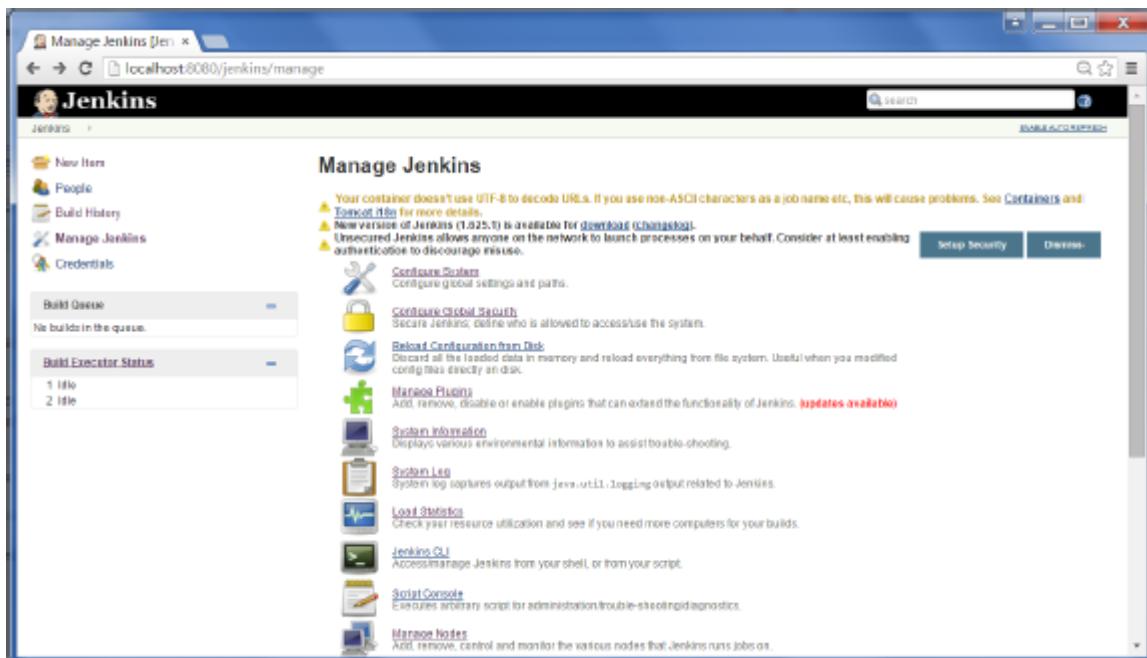
Check now

# 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.

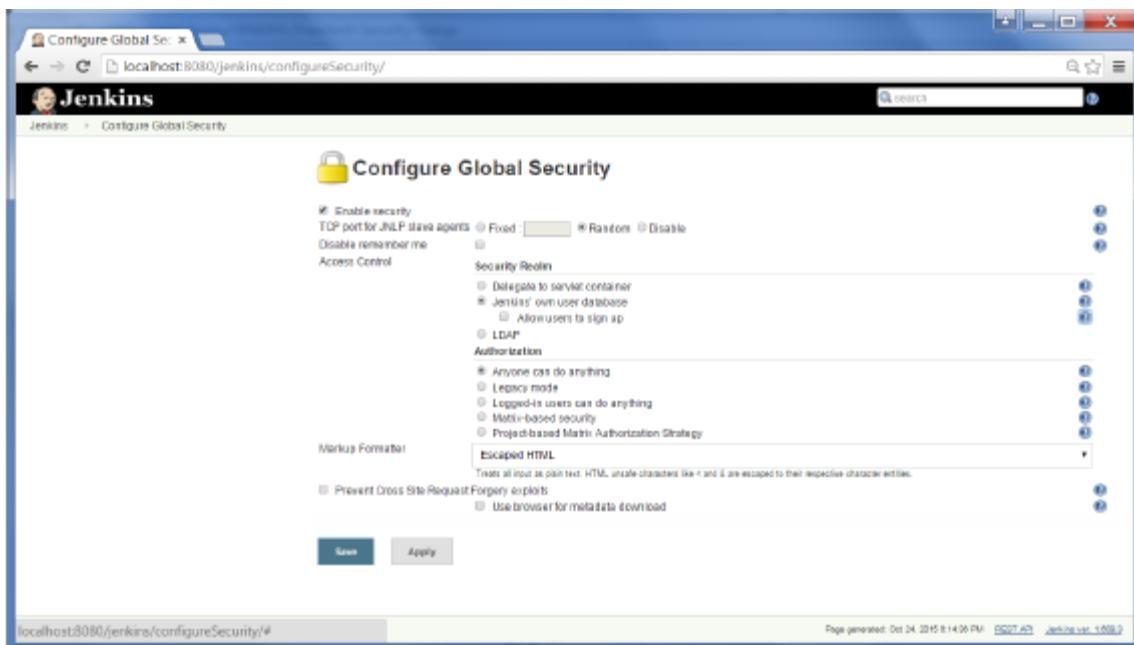
To configure Security in Jenkins, follow the steps given below.

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

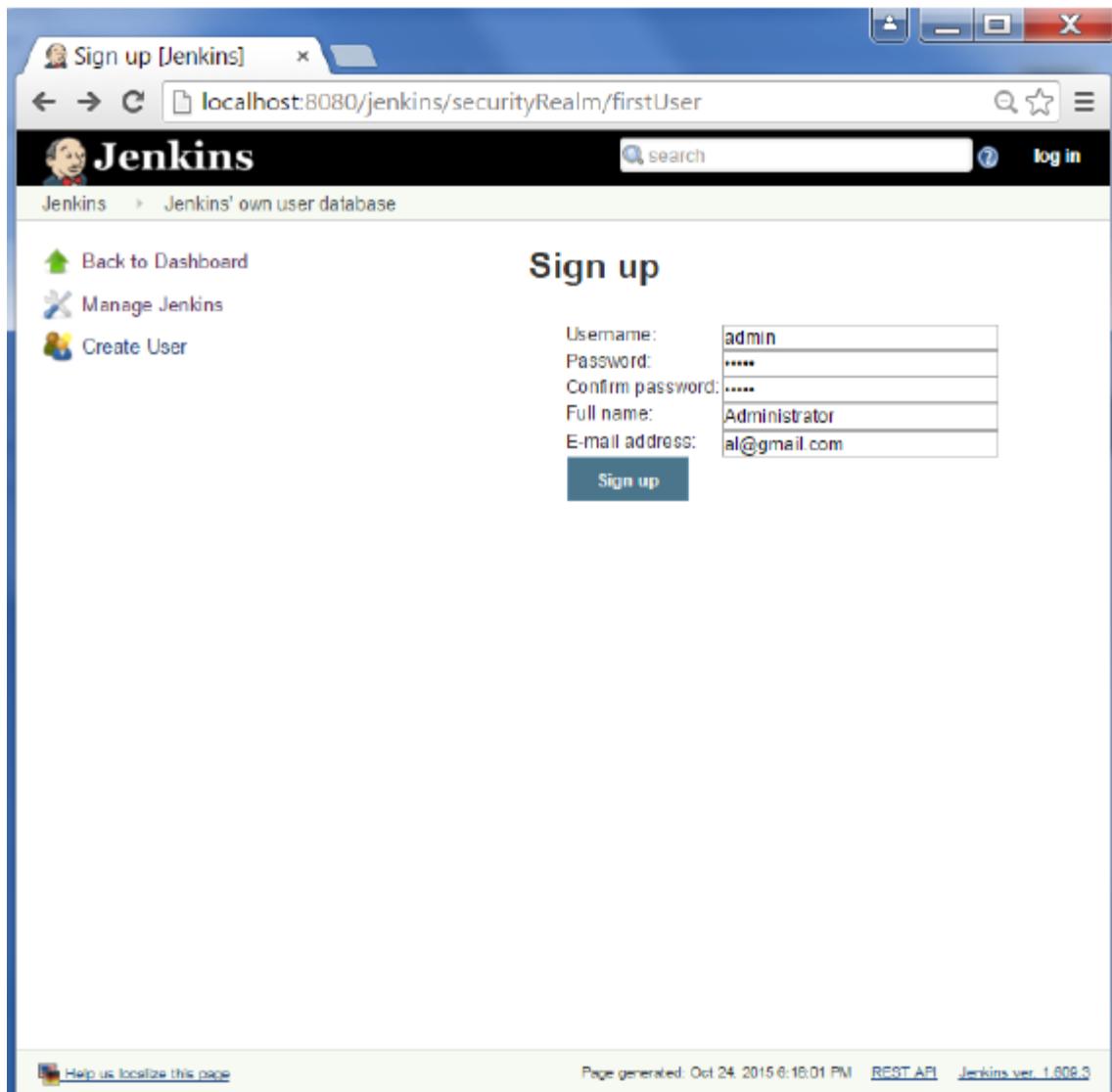


**Step 2** – Click on Enable Security option. As an example, let’s assume that we want Jenkins to maintain it’s own database of users, so in the Security Realm, choose the option of ‘Jenkins’ own user database’.

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.



Sign up [Jenkins] ×

localhost:8080/jenkins/securityRealm/firstUser

**Jenkins** search log in

Back to Dashboard Manage Jenkins Create User

## Sign up

Username:

Password:

Confirm password:

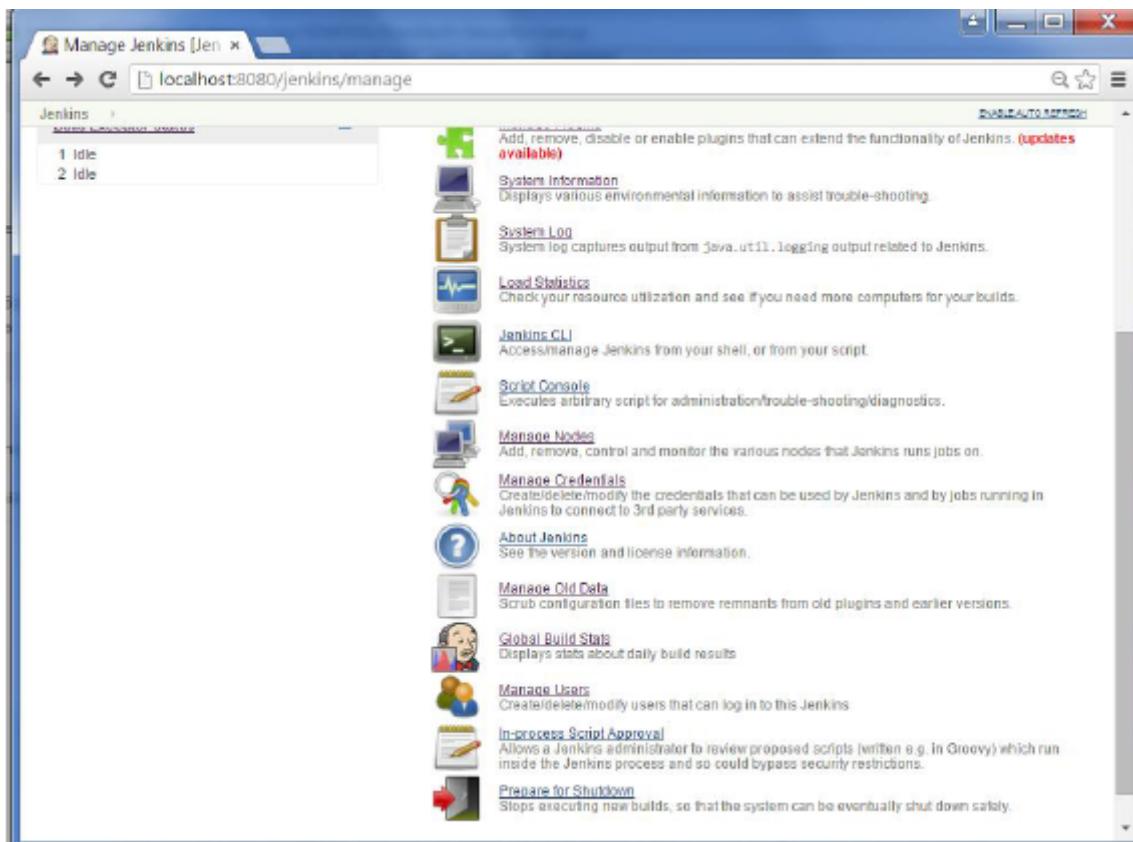
Full name:

E-mail address:

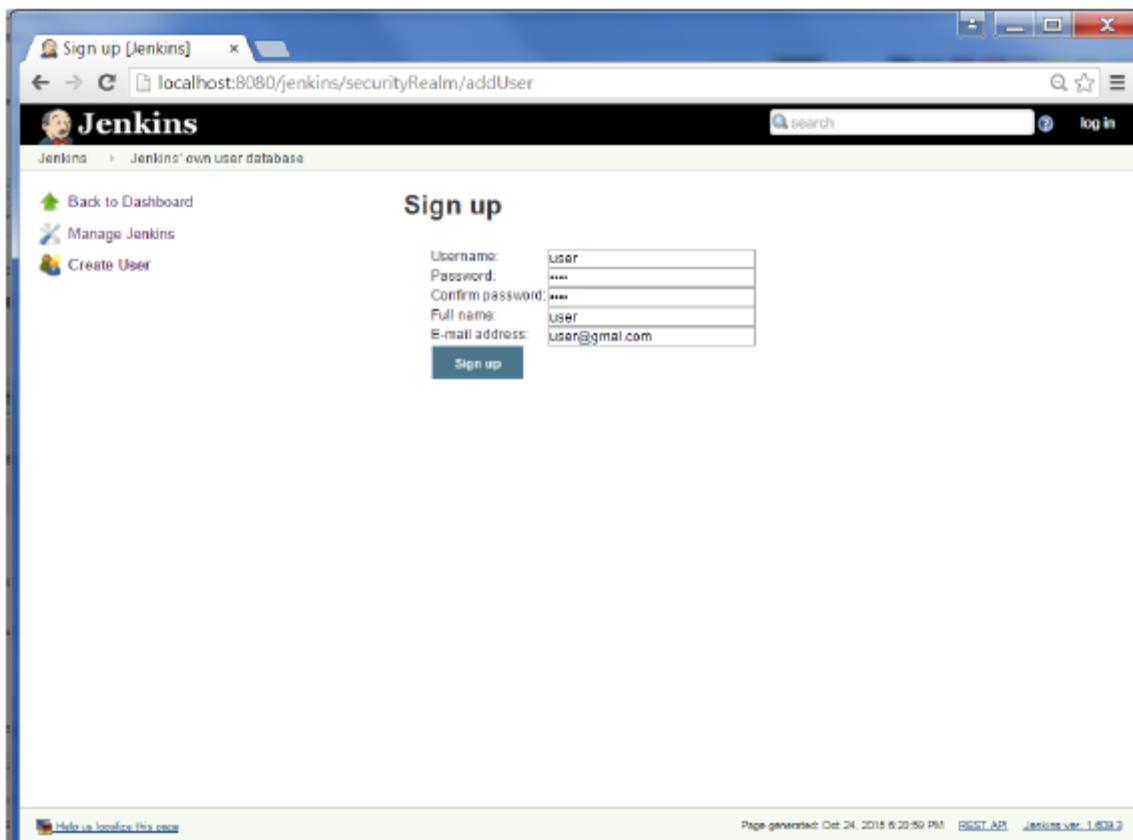
**Sign up**

Help us localize this page Page generated: Oct 24, 2015 6:16:01 PM REST API Jenkins ver. 1.808.3

**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.

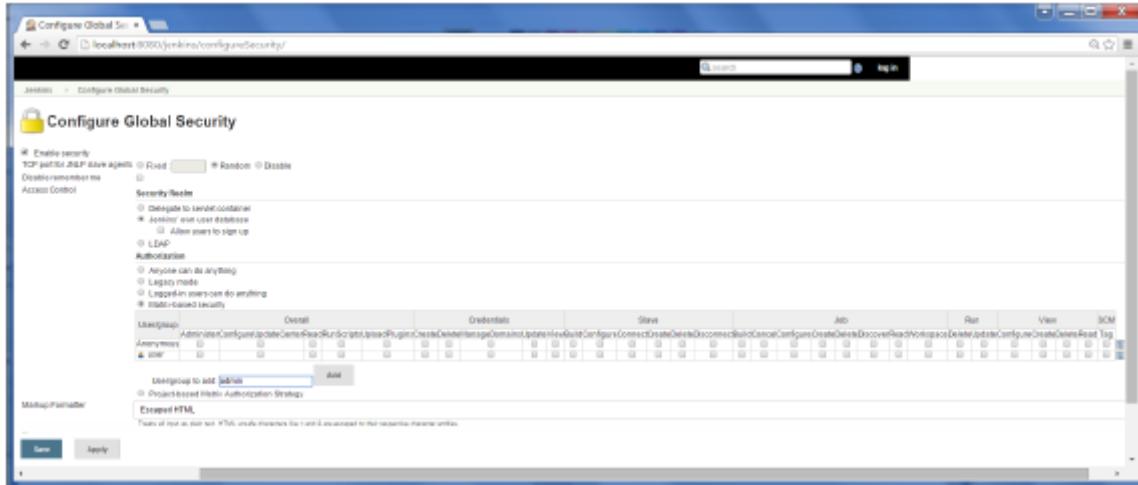


**Step 5** – Just like you defined your admin user, start creating other users for the system. As an example, we are just creating another user called 'user'.



**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.

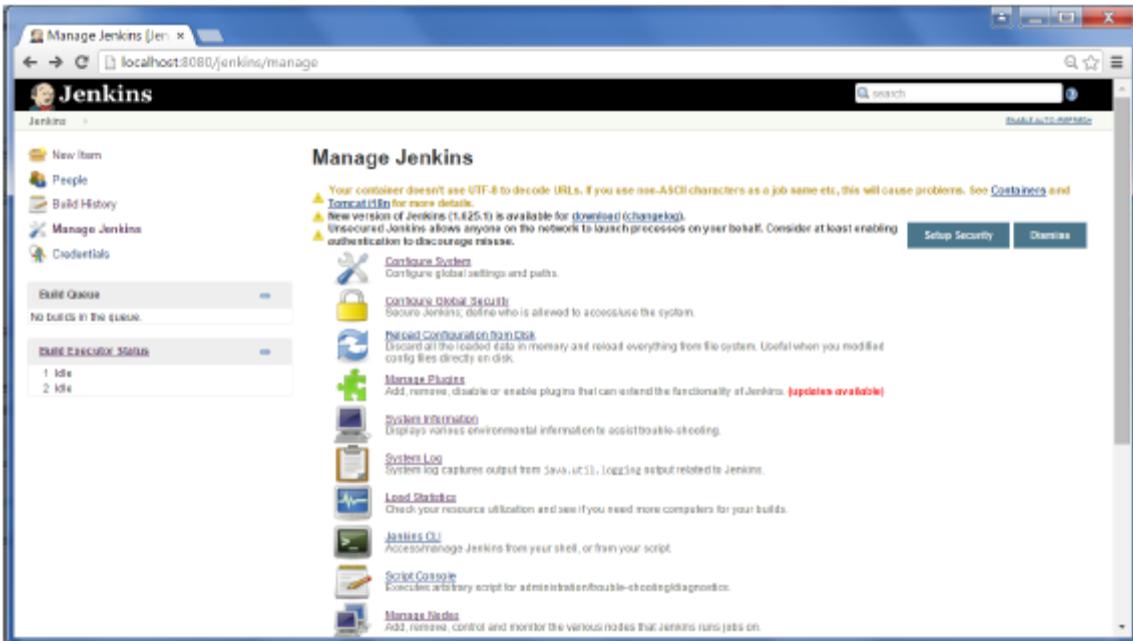
Your Jenkins security is now setup.

**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

Update Center Jenkins > Jenkins > Plugin Manager

Available

Filter: backup

Install	Name	Version
<input checked="" type="checkbox"/>	Backup plugin	1.8.1
<input type="checkbox"/>	Backup and interrupt job plugin	1.0
<input type="checkbox"/>	CloudBees Jenkins Enterprise	15.06.1
<input type="checkbox"/>	CloudBees Free Enterprise Plugins	5.0
<input type="checkbox"/>	Periodic Backup	1.3
<input type="checkbox"/>	ThinBackup	1.7.4

Install without restart   Download now and install after restart   Update information of

Update Center Jenkins > Jenkins > Update center

## Installing Plugins/Upgrades

Preparation

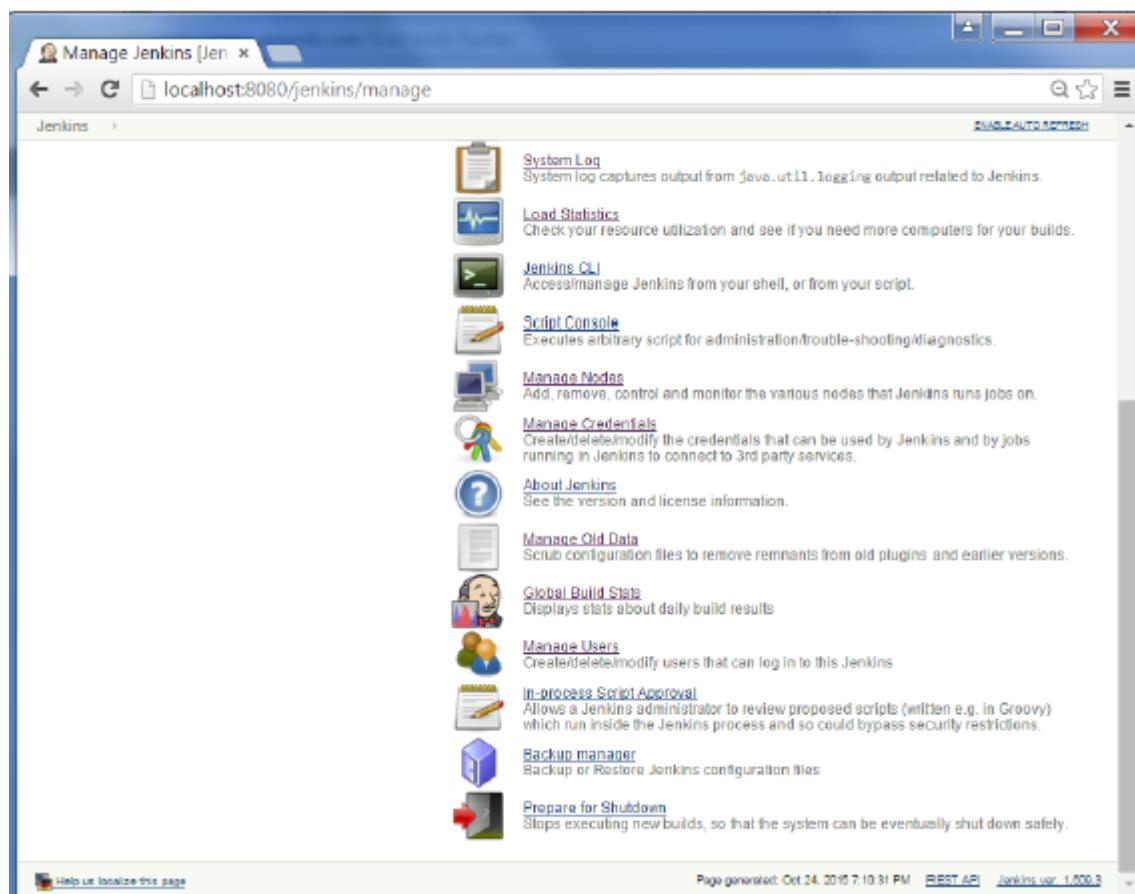
- Checking internet connectivity
- Checking update center connectivity
- Success

Backup plugin   Success

Go back to the top page (you can start using the installed plugin right away)

Restart Jenkins when installation is complete and no jobs are running

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

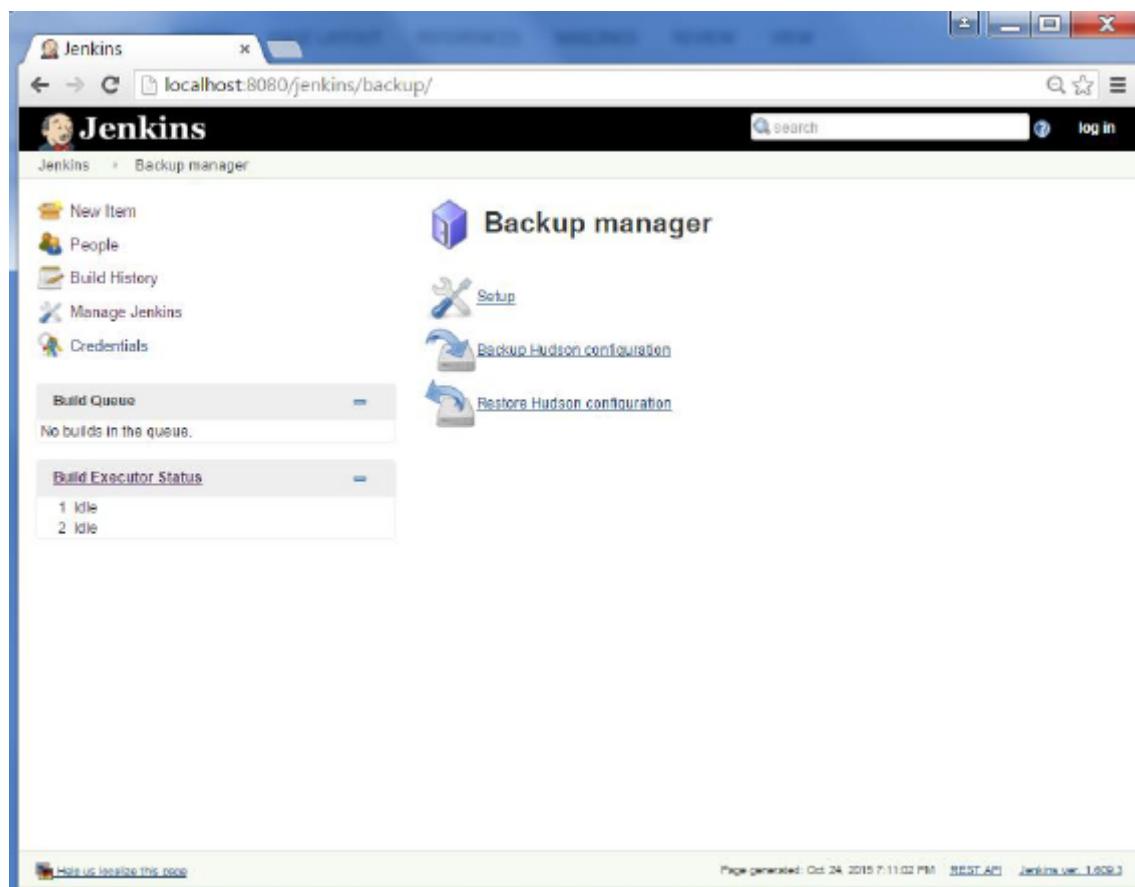


The screenshot shows the Jenkins Manage Jenkins page at [localhost:8080/jenkins/manage](http://localhost:8080/jenkins/manage). The page lists several management options with icons:

- System Log**: System log captures output from `java.util.logging` related to Jenkins.
- Load Statistics**: Check your resource utilization and see if you need more computers for your builds.
- Jenkins CLI**: Access 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.
- Manage Old Data**: Scrub configuration files to remove remnants from old plugins and earlier versions.
- Global Build Stats**: Displays stats about daily build results.
- Manage Users**: Create/delete/modify users that can log in to this Jenkins.
- 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.
- Backup manager**: Backup or Restore Jenkins configuration files.
- Prepare for Shutdown**: Stops executing new builds, so that the system can eventually shut down safely.

Page generated: Oct 24, 2015 7:10:31 PM | REST API | Jenkins ver. 1.600.3

#### Step 4 – Click on Setup.



The screenshot shows the Jenkins Backup manager page at [localhost:8080/jenkins/backup/](http://localhost:8080/jenkins/backup/). The page has a sidebar with links:

- New Item
- People
- Build History
- Manage Jenkins
- Credentials

Below the sidebar are two sections:

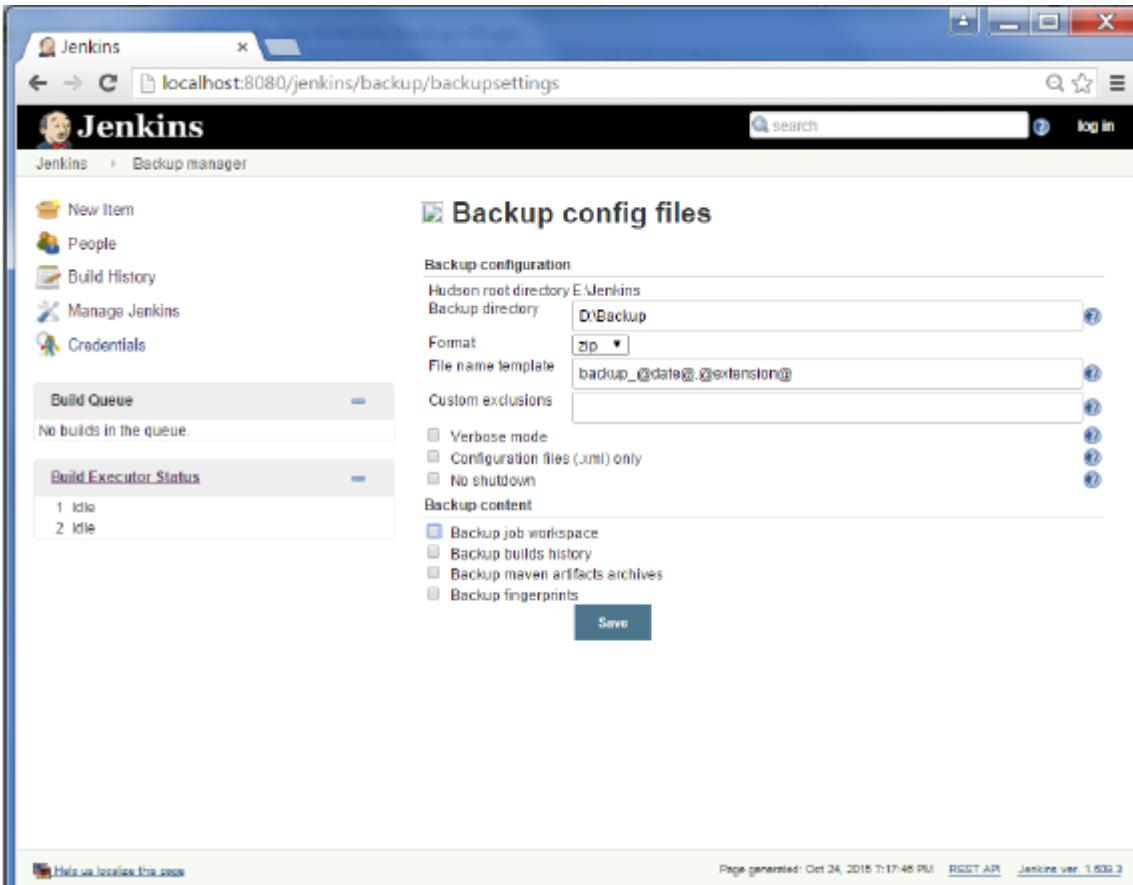
- Build Queue**: No builds in the queue.
- Build Executor Status**: 1 Idle, 2 Idle.

The main content area is titled **Backup manager** and contains the following options:

- Setup**: An icon of a wrench and screwdriver.
- Backup Hudson configuration**: An icon of a blue arrow pointing up.
- Restore Hudson configuration**: An icon of a blue arrow pointing down.

Page generated: Oct 24, 2015 7:11:02 PM | REST API | Jenkins ver. 1.600.3

**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.



The screenshot shows the Jenkins Backup config files configuration page. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. The main content area is titled 'Backup config files' and contains 'Backup configuration' and 'Backup content' sections. In the 'Backup configuration' section, the 'Hudson root directory' is set to 'E:\Jenkins' and the 'Backup directory' is set to 'D:\Backup'. The 'Format' is set to 'zip' and the 'File name template' is 'backup\_@date@.@(extension@)'. Under 'Custom exclusions', there are three checkboxes: 'Verbose mode', 'Configuration files (.xml) only', and 'No shutdown'. In the 'Backup content' section, there are four checkboxes: 'Backup job workspace', 'Backup builds history', 'Backup maven artifacts archives', and 'Backup fingerprints'. A 'Save' button is located at the bottom of the configuration section. The bottom of the page includes a 'Help us localize this page' link, a page footer with 'Page generated: Oct 24, 2015 7:17:48 PM', 'REST API', and 'Jenkins ver. 1.603.2'.

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

Jenkins

localhost:8080/jenkins/backup/

**Jenkins**

Backup manager

New Item

People

Build History

Manage Jenkins

Credentials

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Backup manager

Setup

Backup Hudson configuration

Restore Hudson configuration

Help us localize this page

Page generated: Oct 24, 2015 7:11:02 PM

JENKINS API

Jenkins ver. 1.609.3

The next screen will show the status of the backup

Backup manager log

localhost:8080/jenkins/backup/backup

**Jenkins**

Backup manager

New Item

People

Build History

Manage Jenkins

Credentials

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

Jenkins is going to shut down

```
[ INFO] Backup started at [10/24/15 19:19:31]
[ INFO] Setting hudson in shutdown mode to avoid files corruptions.
[ INFO] Waiting all jobs end...
[ INFO] Number of running jobs detected : 0
[ INFO] All jobs finished.
[ INFO] Full backup file name : D:\Backup\backup_20151024_1919.zip
[ INFO] Saved files : 911
[ INFO] Number of errors : 0
[ INFO] Cancel hudson shutdown mode
[ INFO] Backup end at [10/24/15 19:19:58]
[ INFO] [19.524s]
```

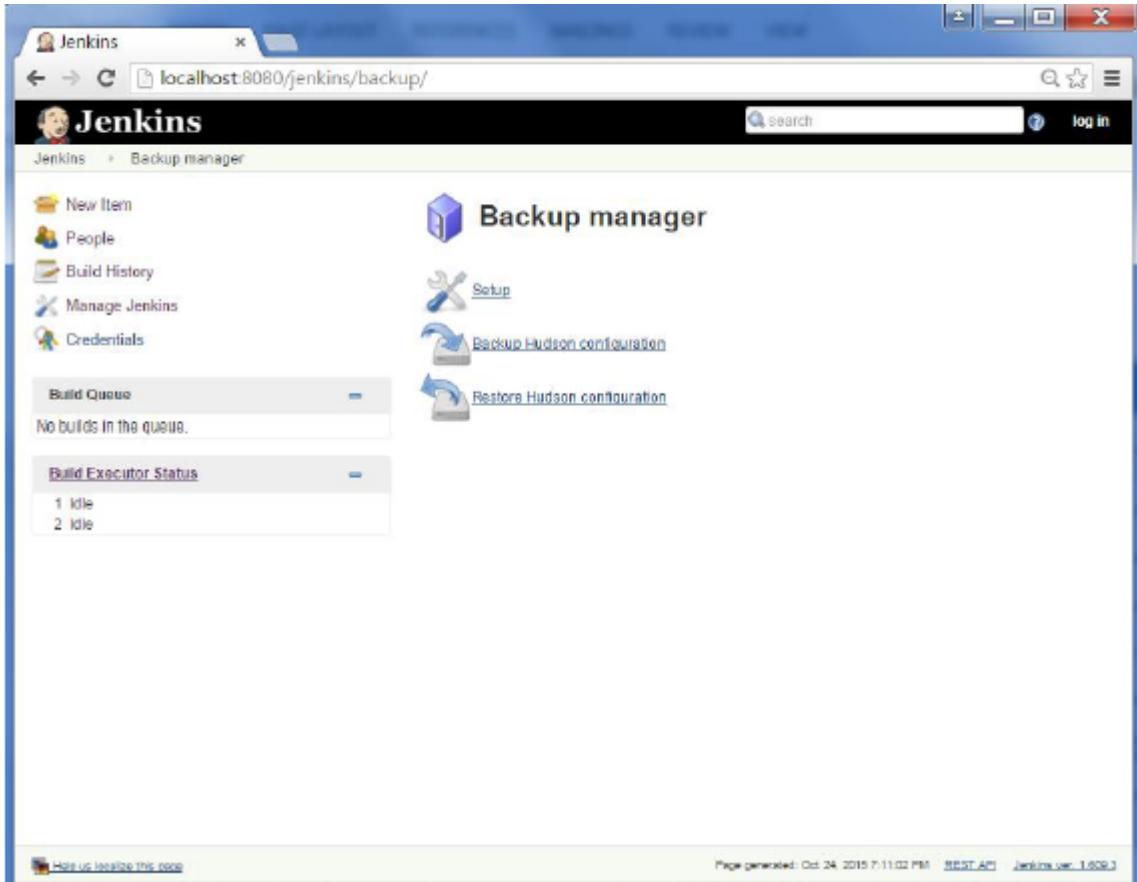
Help us localize this page

Page generated: Oct 24, 2015 7:12:31 PM

JENKINS API

Jenkins ver. 1.609.3

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

localhost:8080/jenkins/backup/launchrestore

## Backup manager

Available backup in D:\Backup :

backup\_20151024\_1919.zip

Launch restore

New Item

People

Build History

Manage Jenkins

Credentials

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

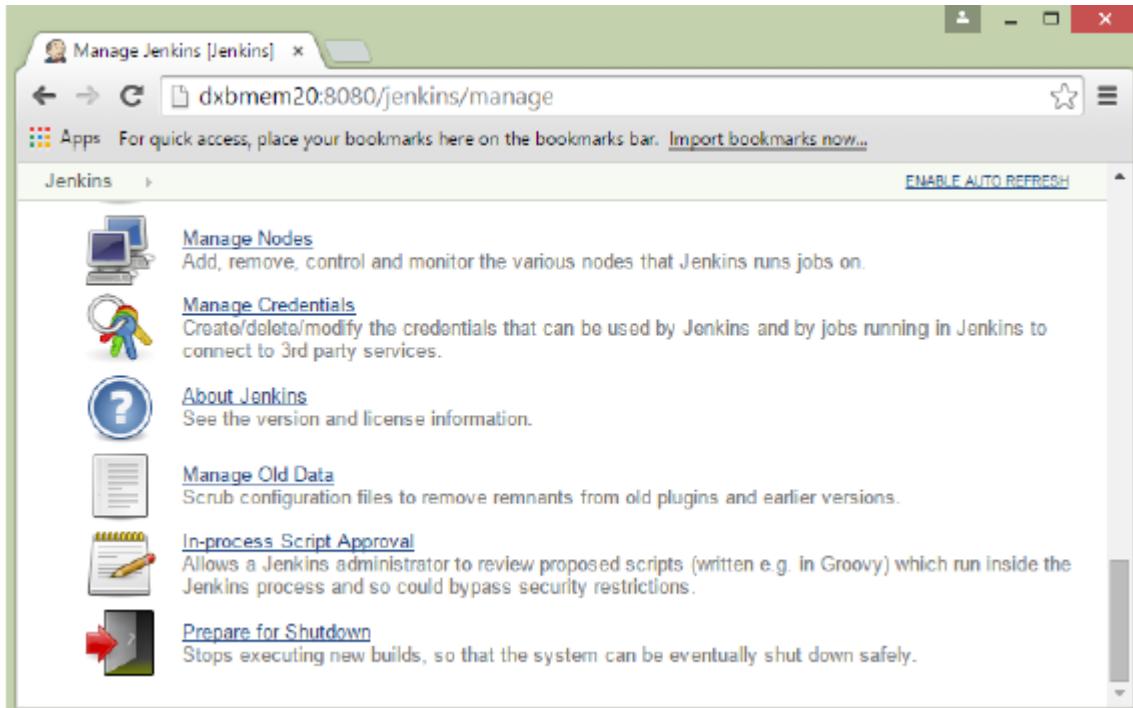
Help us localize this page

Page generated: Oct 24, 2015 7:20:45 PM REST API Jenkins ver. 1.600.3

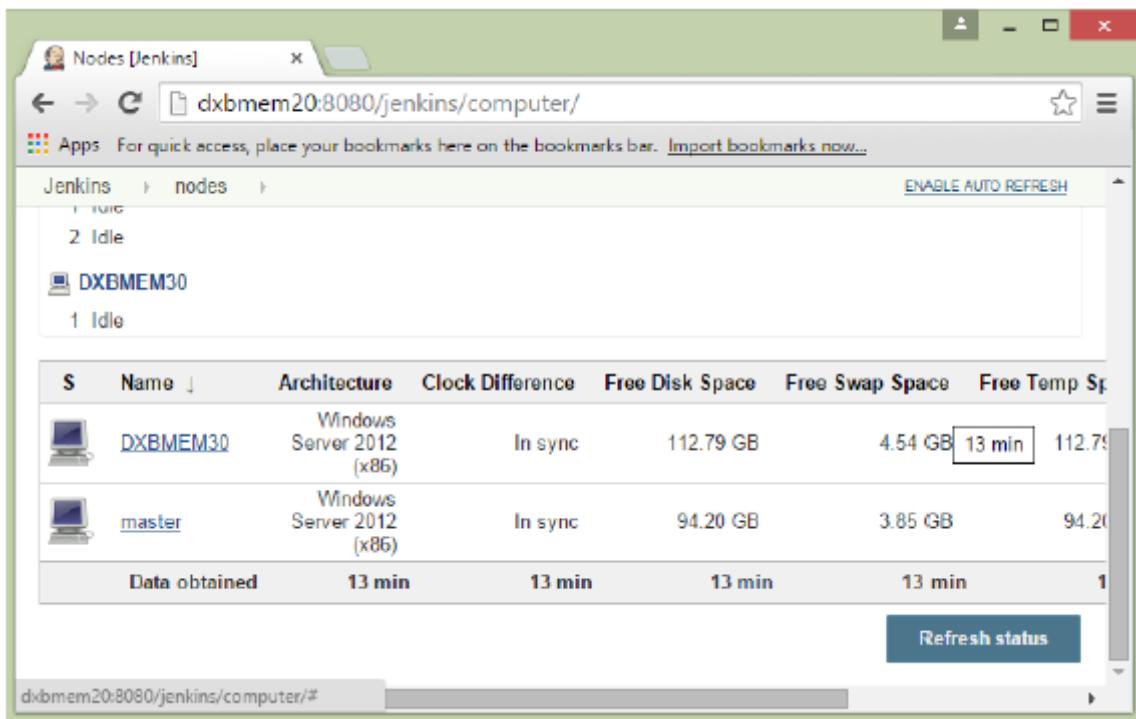
# Jenkins - Remote Testing

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** – Ensuring your master slave configuration is in place. Got to your master Jenkins server. Go to Manage Jenkins → Manage Nodes.



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

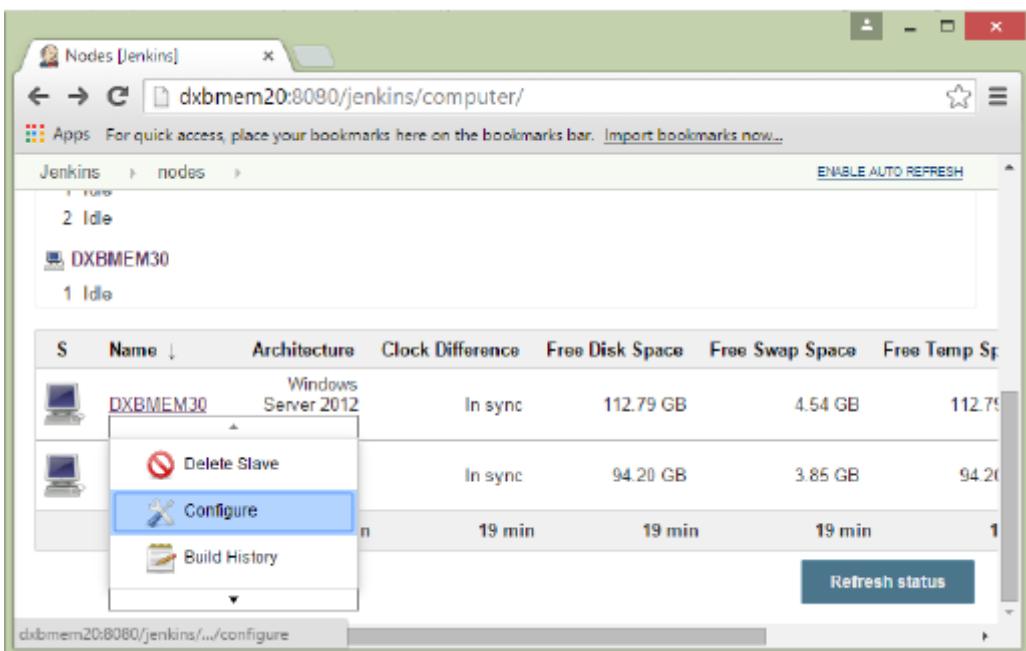


The screenshot shows the Jenkins 'Nodes' page. At the top, there are two nodes listed: 'DXBMEM30' (1 Idle) and 'master' (1 Idle). Below this is a table with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, and Free Temp Sp. The table data is as follows:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Sp
	DXBMEM30	Windows Server 2012 (x86)	In sync	112.79 GB	4.54 GB	13 min 112.79
	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

At the bottom right of the table is a 'Refresh status' button.

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

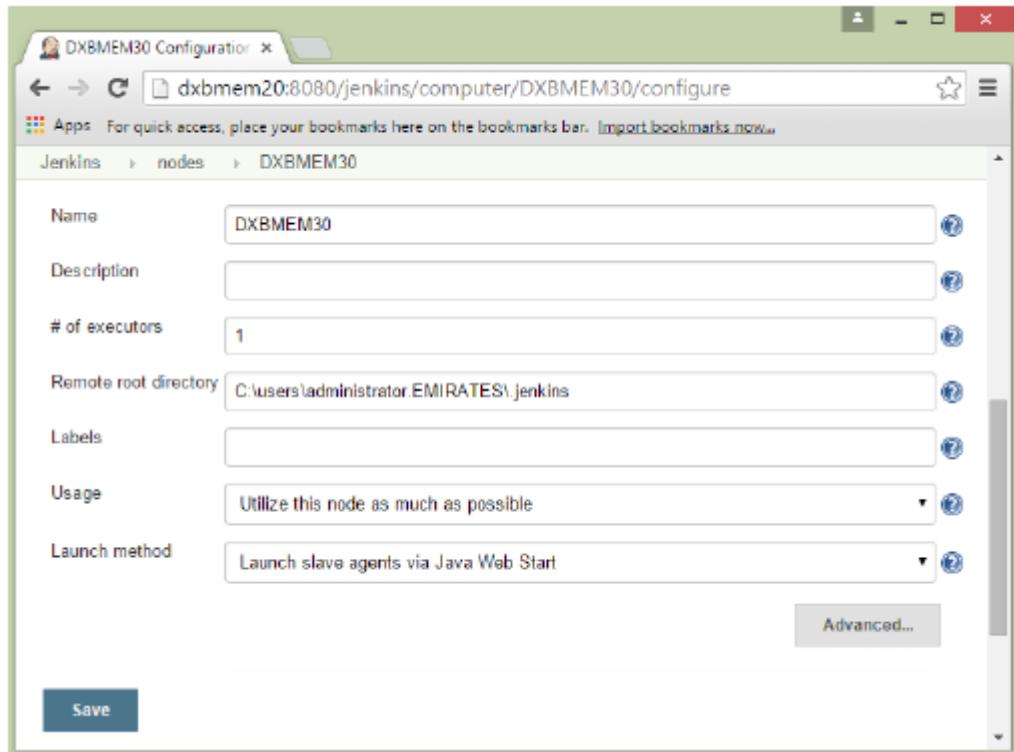


The screenshot shows the Jenkins 'Nodes' page for the 'DXBMEM30' node. A context menu is open over the 'DXBMEM30' row, with the 'Configure' option highlighted. The menu also includes 'Delete Slave' and 'Build History' options. The table data for DXBMEM30 is as follows:

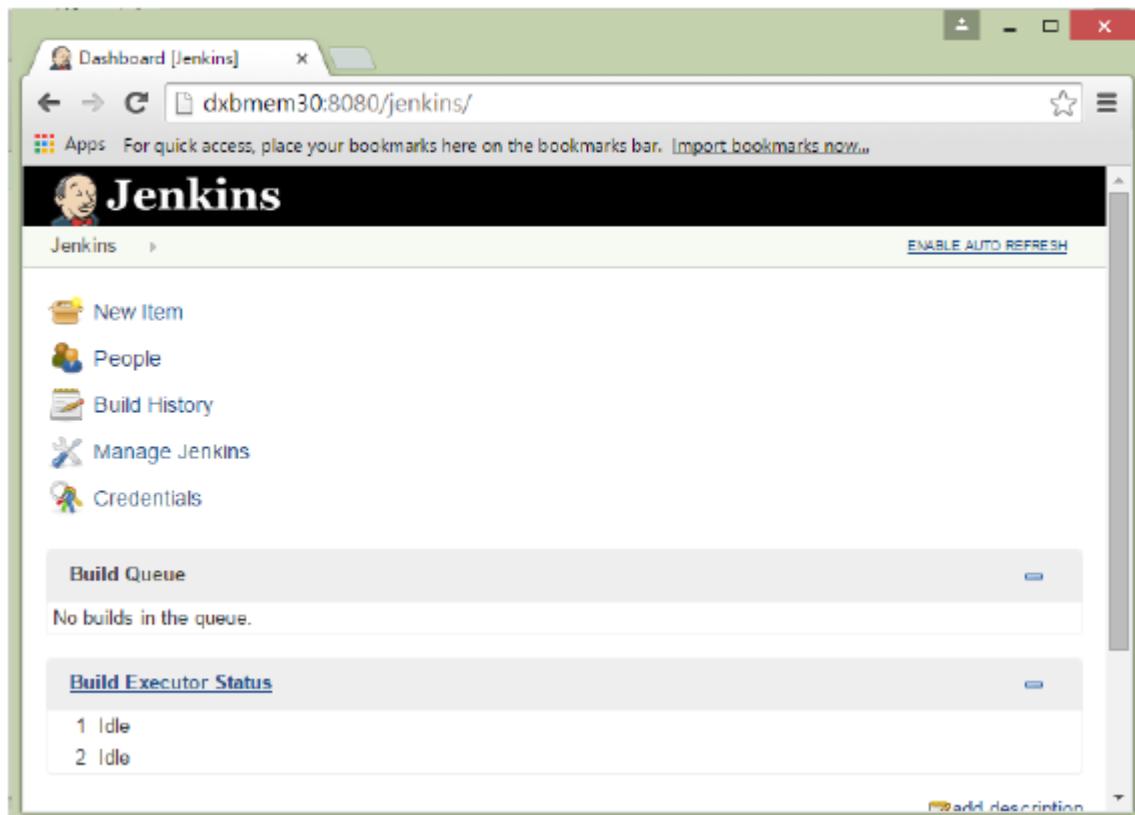
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Sp
	DXBMEM30	Windows Server 2012	In sync	112.79 GB	4.54 GB	112.79
	master	Windows Server 2012 (x86)	In sync	94.20 GB	3.85 GB	94.20
	Data obtained	19 min	19 min	19 min	19 min	1

At the bottom right of the table is a 'Refresh status' button.

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



**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.

Nodes [Jenkins]

dxbmem20:8080/jenkins/computer/

Jenkins > nodes >

ENABLE AUTO REFRESH

No builds in the queue.

Build Executor Status

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space
	DXBMEM30	Windows Server 2012 (x86)	In sync	112.79 GB	4.54 GB
	master	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'

DXBMEM30 [Jenkins] x

dxbmemp20:8080/jenkins/computer/DXBMEM30/

Jenkins > nodes > DXBMEM30 [ENABLE AUTO REFRESH](#)

at org.jenkinsci.remoting.nio.NioChannelHub.run(NioChannelHub.java:561)  
... 6 more

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
		<a href="#">HelloWorld</a>	43 min - #12	41 min - #13	7.3 sec 

Icon:  [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.

Security Warning

**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.

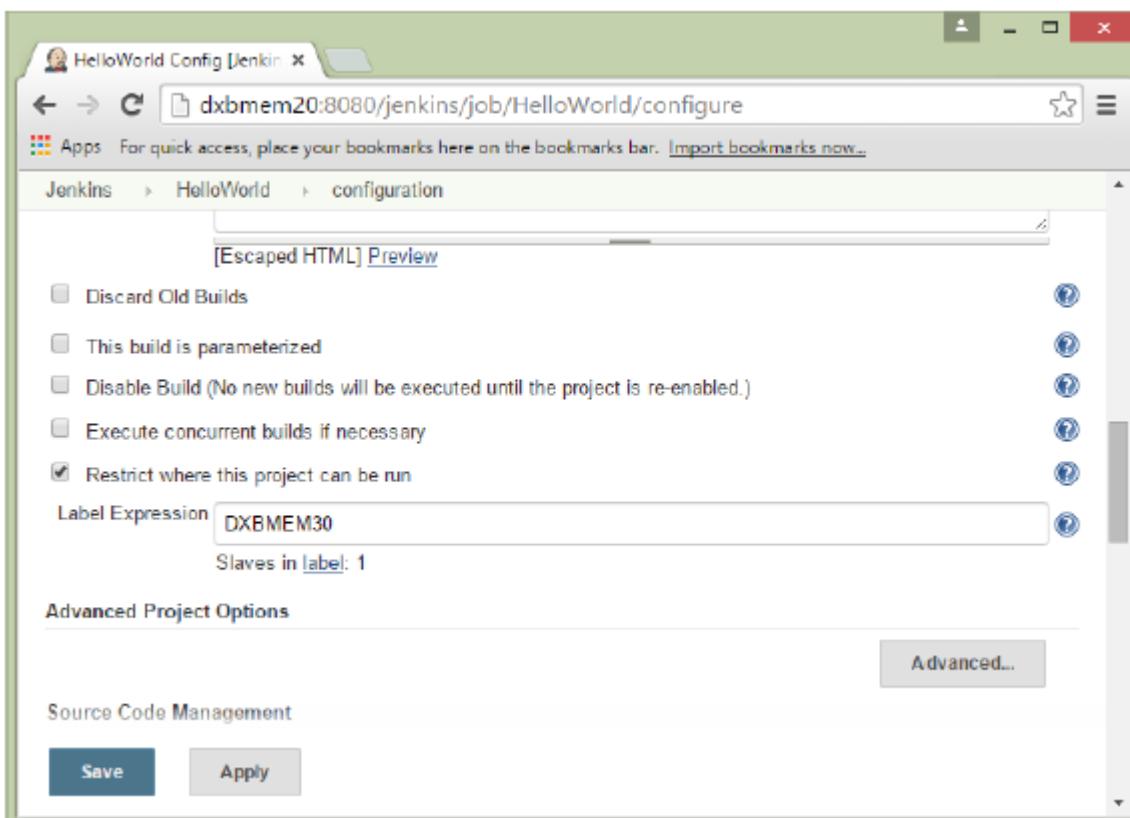
[Run](#) [Cancel](#)

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

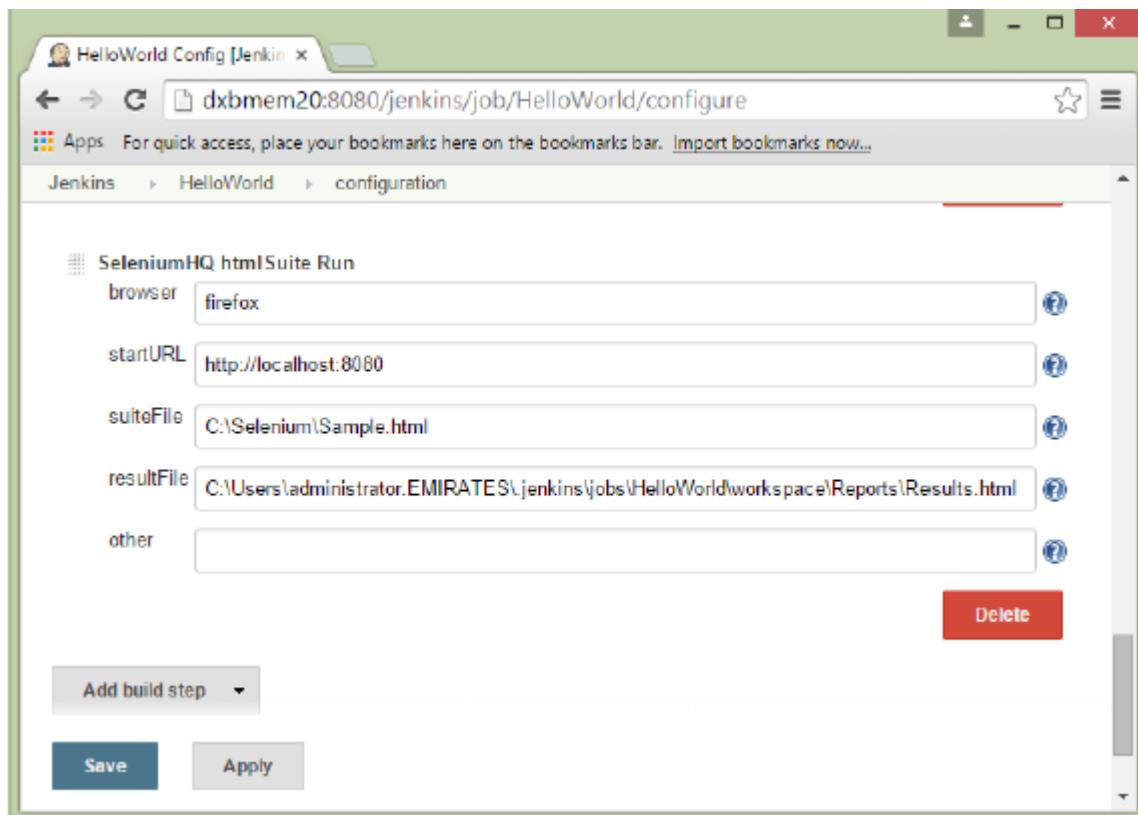


**Step 8** – Configuring your tests to run on the slave. Here, you have to ensure that the job being created is meant specifically to only run the selenium tests.

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.



**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.