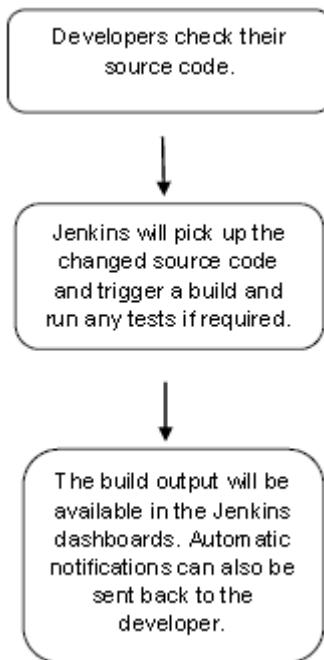


# Jenkins Tutorial

## 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. 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
```

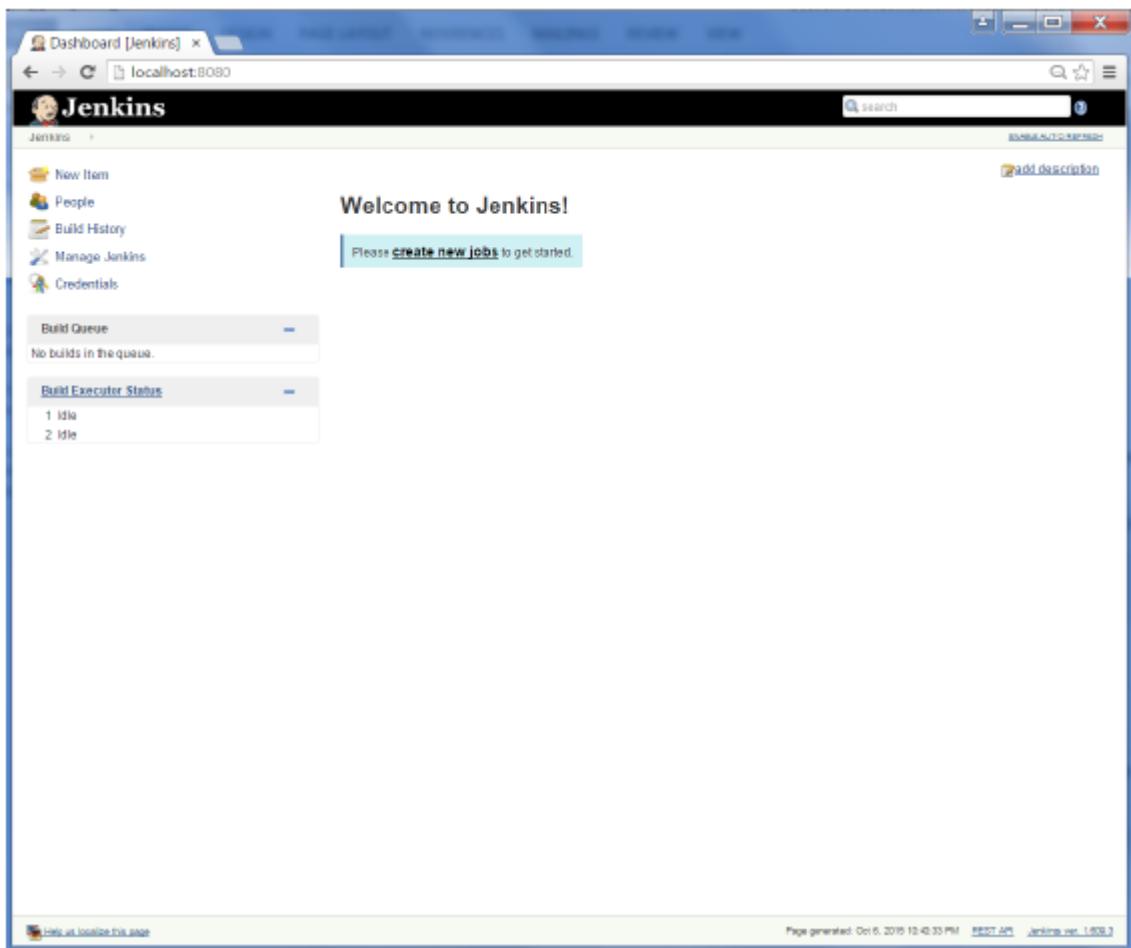
Once the processing is complete without major errors, the following line will come in the output of the command prompt.

```
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. The main navigation menu on the left includes links for Home, Taglibs, Maven Plugin, Download (with sub-links for Which version?, Tomcat 8.0, 7.0, 6.0, Connectors, Native, Archives), Documentation (with sub-links for Tomcat 8.0, 7.0, 6.0, Connectors, Native, Wiki, Migration Guide), Problems? (with sub-links for Security Reports, Find help, FAQ, Mailing Lists, Bug Database, IRC), and Get Involved (with sub-links for Overview, SVN Repositories, Buildbot, Reviewboard, Tools). The central content area features the Apache feather logo and a search bar. It displays two release announcements: "Tomcat 8.0.27 Released" (2015-10-01) and "Tomcat 7.0.64 Released" (2015-08-25). Each announcement includes a list of changes and a "Download" link.

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

The screenshot shows the "Tomcat 7 Downloads" page. The left sidebar is identical to the main homepage. The main content area has a "Tomcat 7 Downloads" header. It includes sections for "Quick Navigation" (with links to KEYS, 7.0.64, Browse, Archives), "Release Integrity" (warning about verifying file integrity using OpenPGP signatures), "Mirrors" (listing the current mirror as <http://www.us.apache.org/dist/> and providing a "Change" link), and a "7.0.64" section. The "7.0.64" section contains a note about reading the README file and a "Binary Distributions" section with a list of download links for various file formats (zip, tar.gz, etc.) and architectures (32-bit, 64-bit, Itanium Windows).

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

Then unzip the contents of the downloaded zip file.

## Step 4: Jenkins and Tomcat Setup

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

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

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

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

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

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

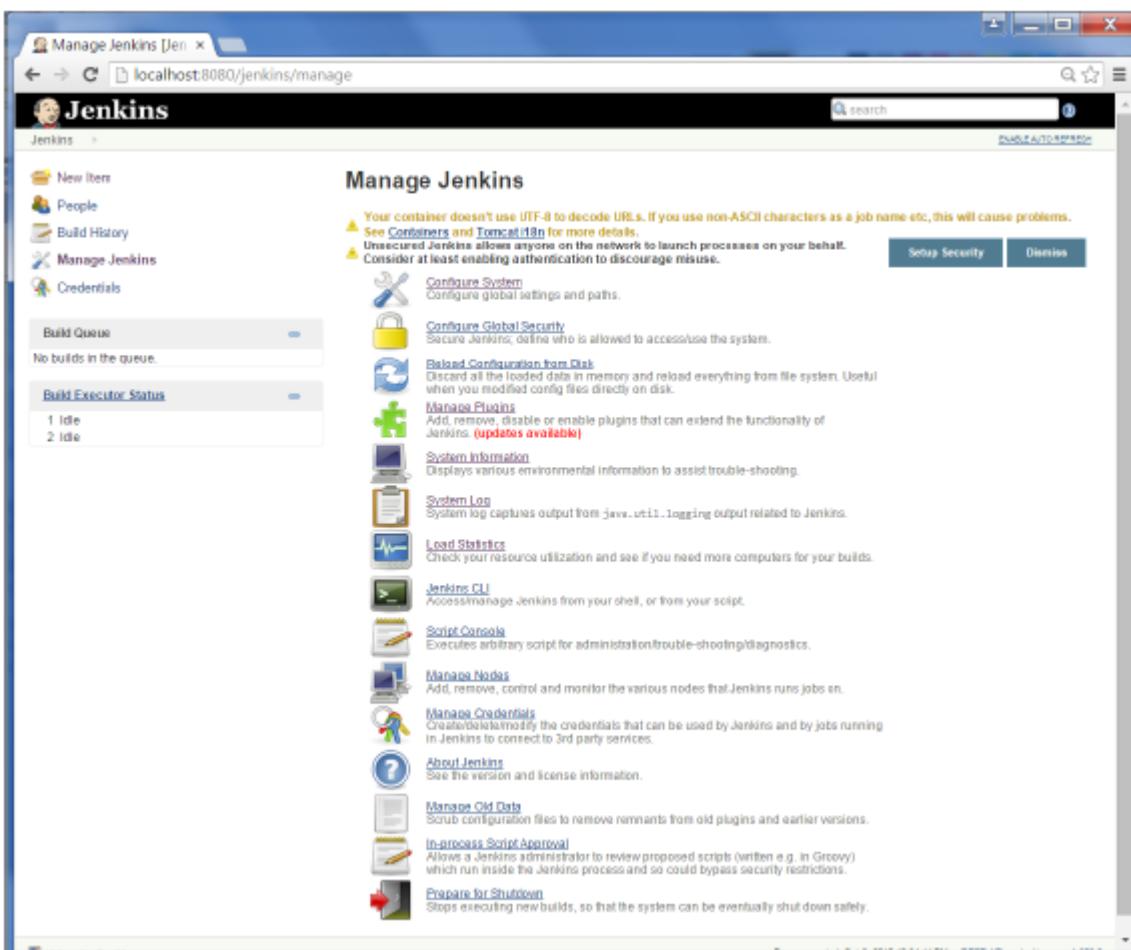


## Jenkins - Git Setup

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



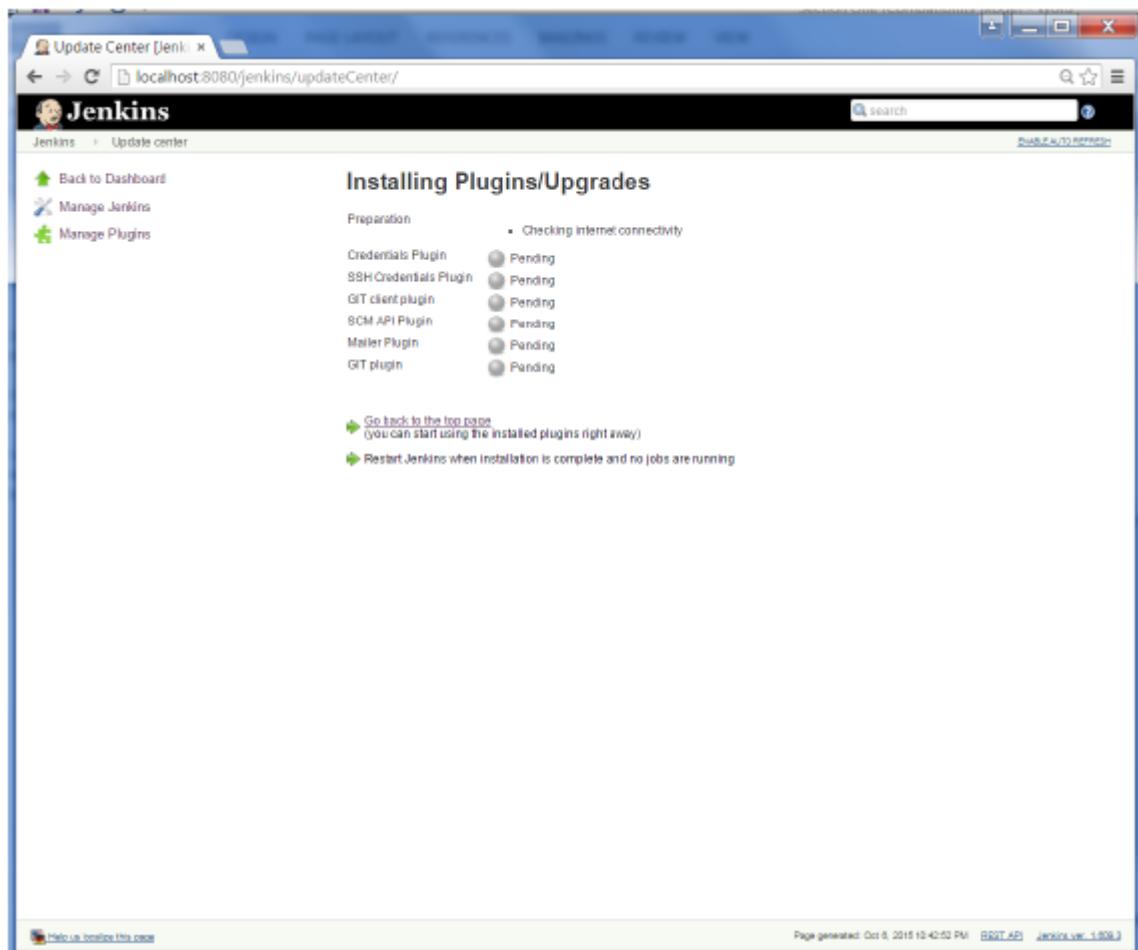
In the next screen, click the ‘Manage Plugins’ option.



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

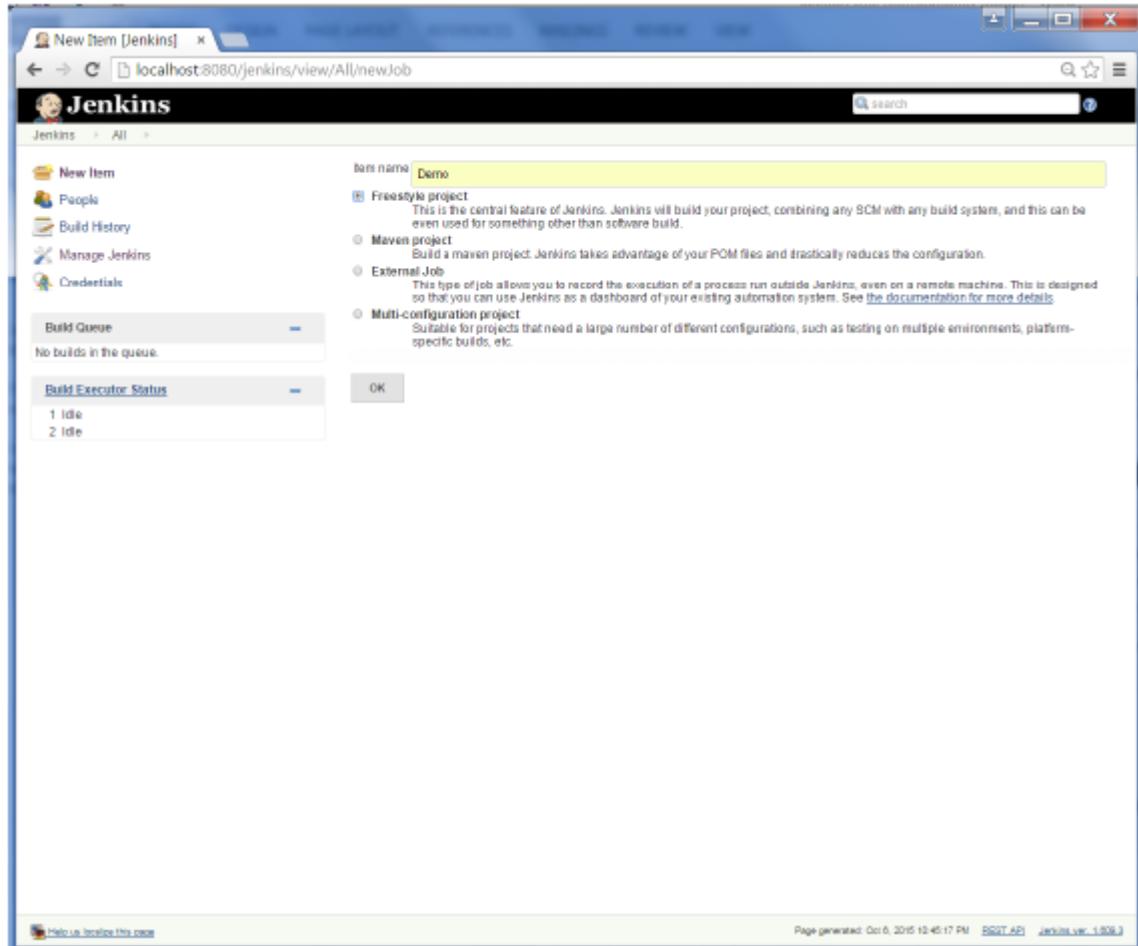
The list will then be filtered. Check the Git Plugin option and click on the button ‘Install without restart’

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

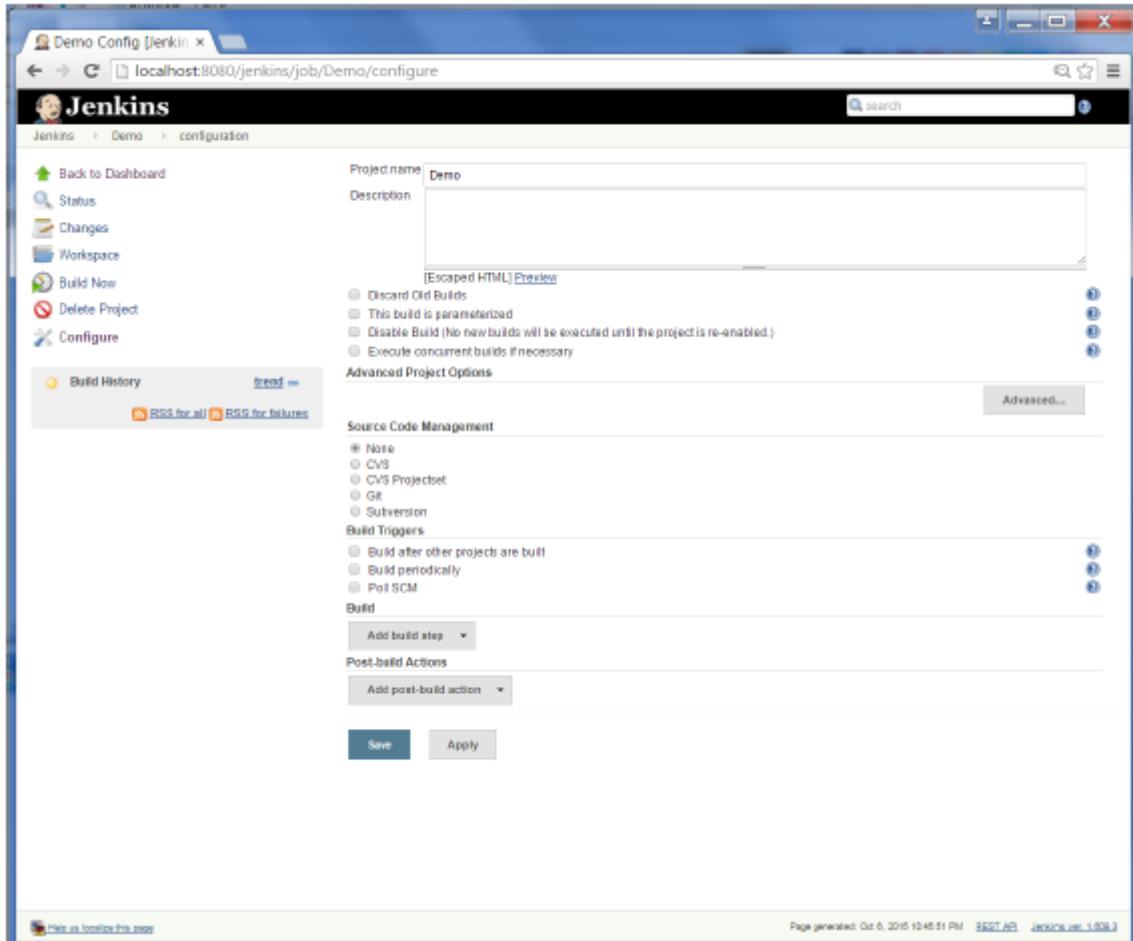


Once all installations are complete, restart Jenkins by issue the following command in the browser. **<http://localhost:8080/jenkins/restart>**

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



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



## Jenkins – Maven Setup

### Step 1: Downloading and Setting Up Maven

The official website for maven is Apache Maven. If you click the given link, you can get the home page of the maven official website as shown below.

**Apache Maven Project**  
http://maven.apache.org/

Apache / Maven / Download Apache Maven Last Published: 2015-09-24

**MAIN**

- Welcome
- License
- Download**
- Install
- Configure
- Run
- IDE Integration

**ABOUT MAVEN**

- What is Maven?
- Features
- FAQ
- Support and Training

**DOCUMENTATION**

- Maven Plugins
- Index (category)
- Running Maven

User Centre >  
Plugin Developer Centre  
Maven Repository Centre  
Maven Developer Centre  
Books and Resources

## Downloading Apache Maven 3.3.3

Apache Maven 3.3.3 is the latest release and recommended version for all users.

The currently selected download mirror is <http://www.us.apache.org/dist/>. If you encounter a problem with this mirror, please select another mirror. If all mirrors are failing, there are backup mirrors (at the end of the mirrors list) that should be available. You may also consult the [complete list of mirrors](#).

Other mirrors: <http://www.eu.apache.org/dist/> [Change](#)

### System Requirements

Java Development Kit (JDK)	Maven 3.3 requires JDK 1.7 or above to execute - it still allows you to build against 1.6 and other JDK versions by <a href="#">Using Toolchains</a>
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
--	------	----------	-----------

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

IxD

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

- [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

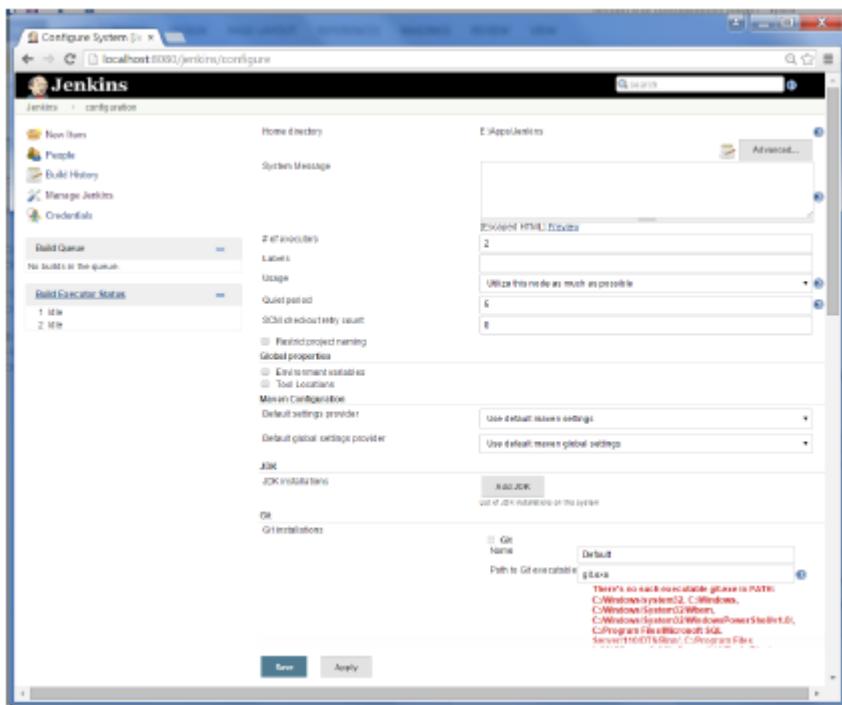
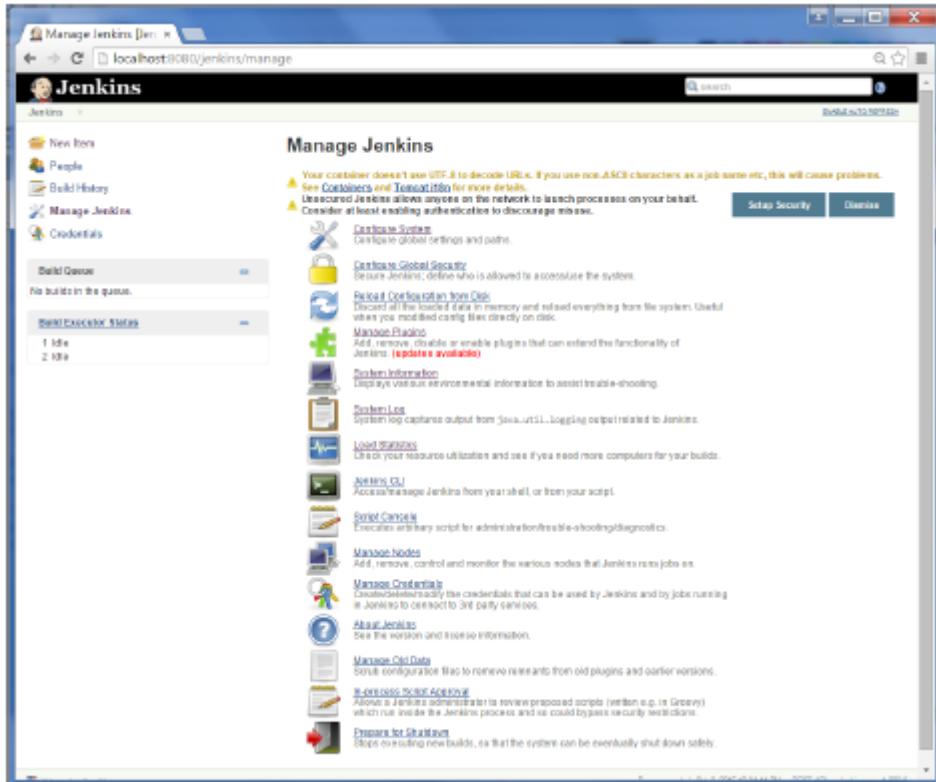
It is strongly recommended to use the latest release version of Apache Maven to take advantage of newest features and bug fixes. If you still want to use an old version you can find more information in the [Maven Releases History](#) and can download files from the [archives](#) for versions 3.0.4+ and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

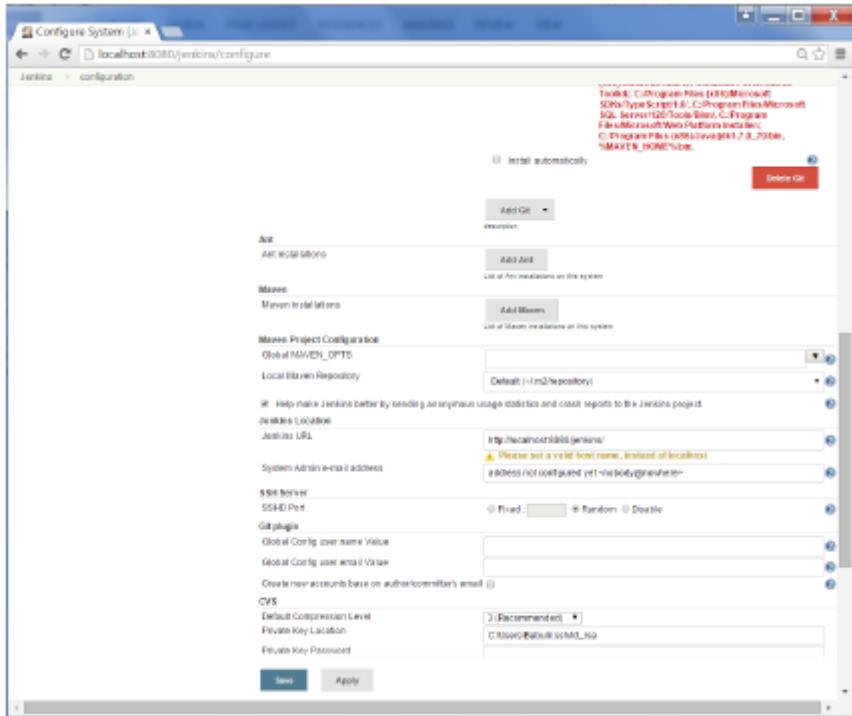
## Step 2: Setting up Jenkins and Maven

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.



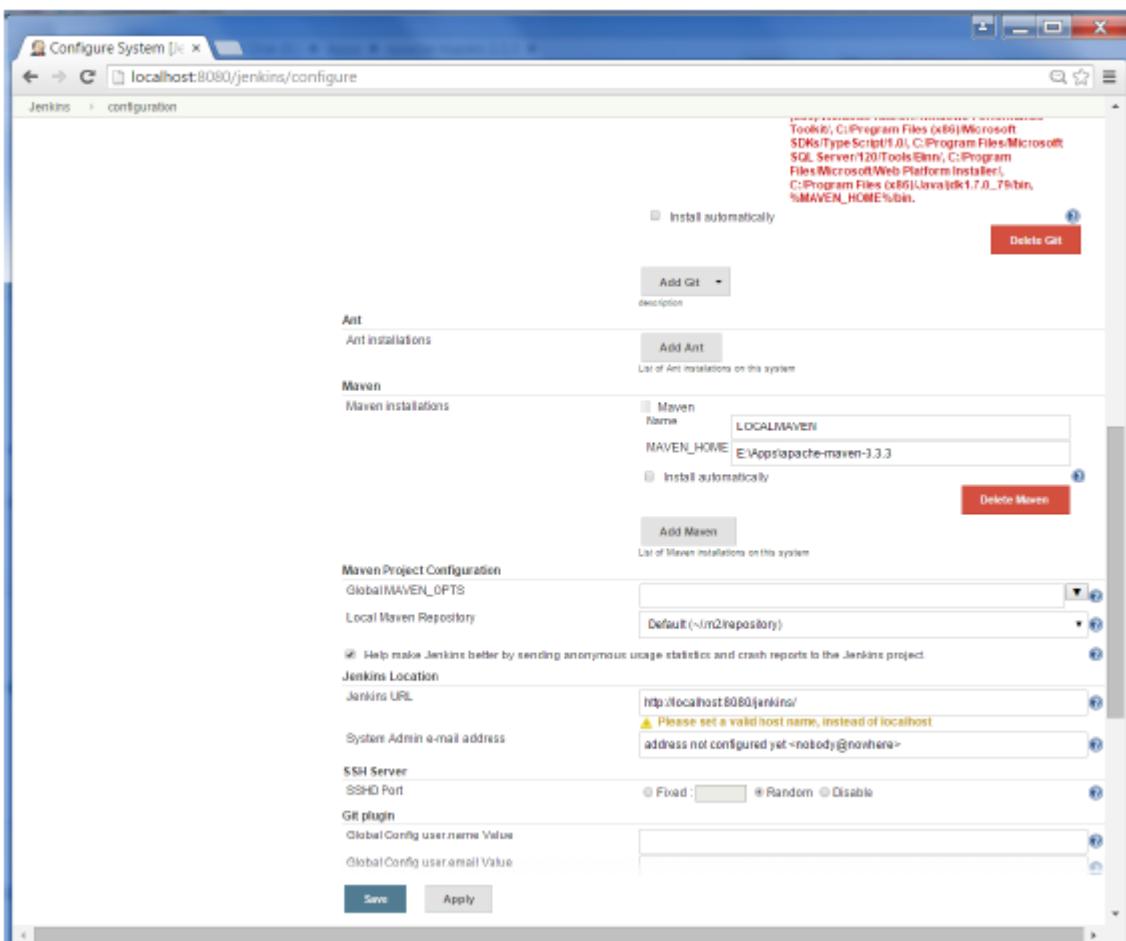
In the Configure system screen, scroll down till you see the Maven section and then click on the 'Add Maven' button.



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/ SEARCH ENABLE AUTO REFRESH

# Jenkins

New Item Add description

People

Build History

Manage Jenkins

Credentials

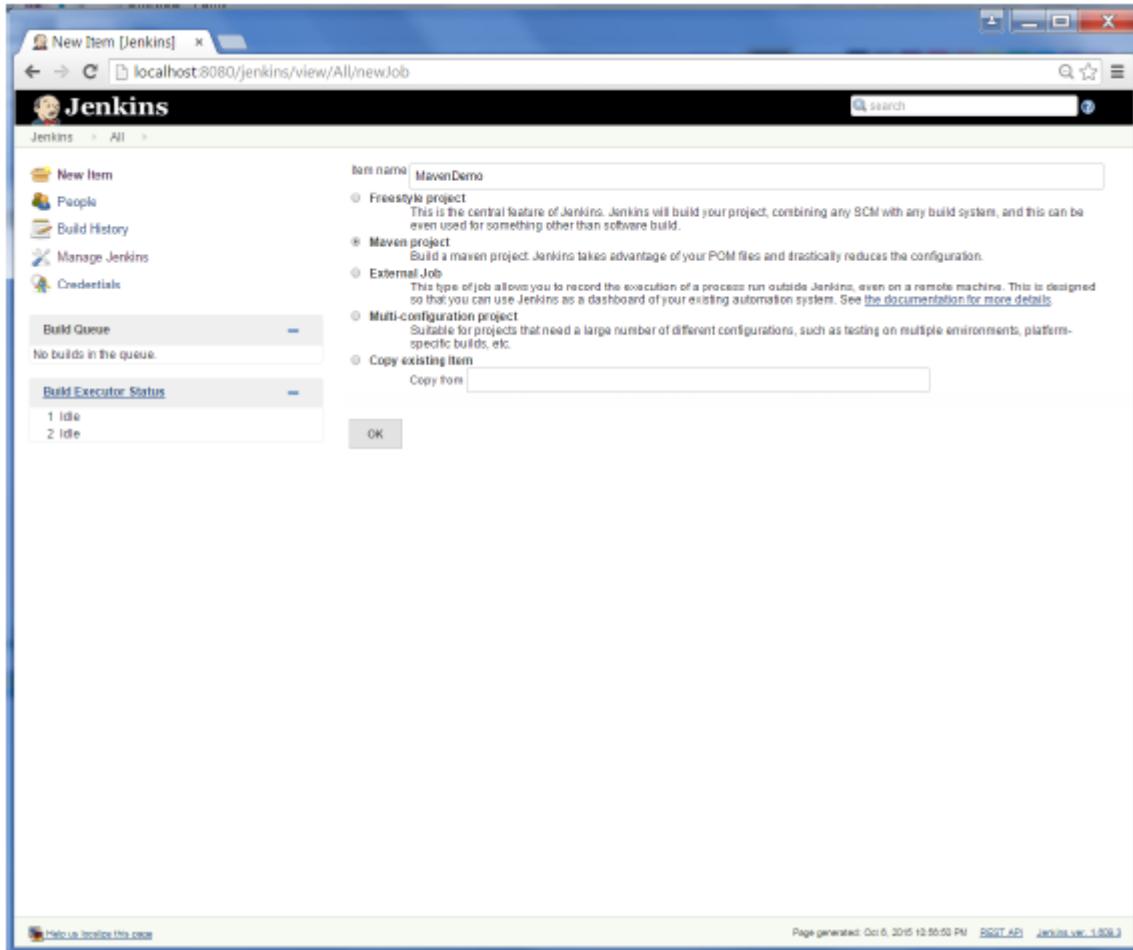
Build Queue —  
No builds in the queue.

Build Executor Status —  
1 Idle  
2 Idle

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

Kon: 토보 L Legend RSS for all RSS for failures RSS for just latest builds

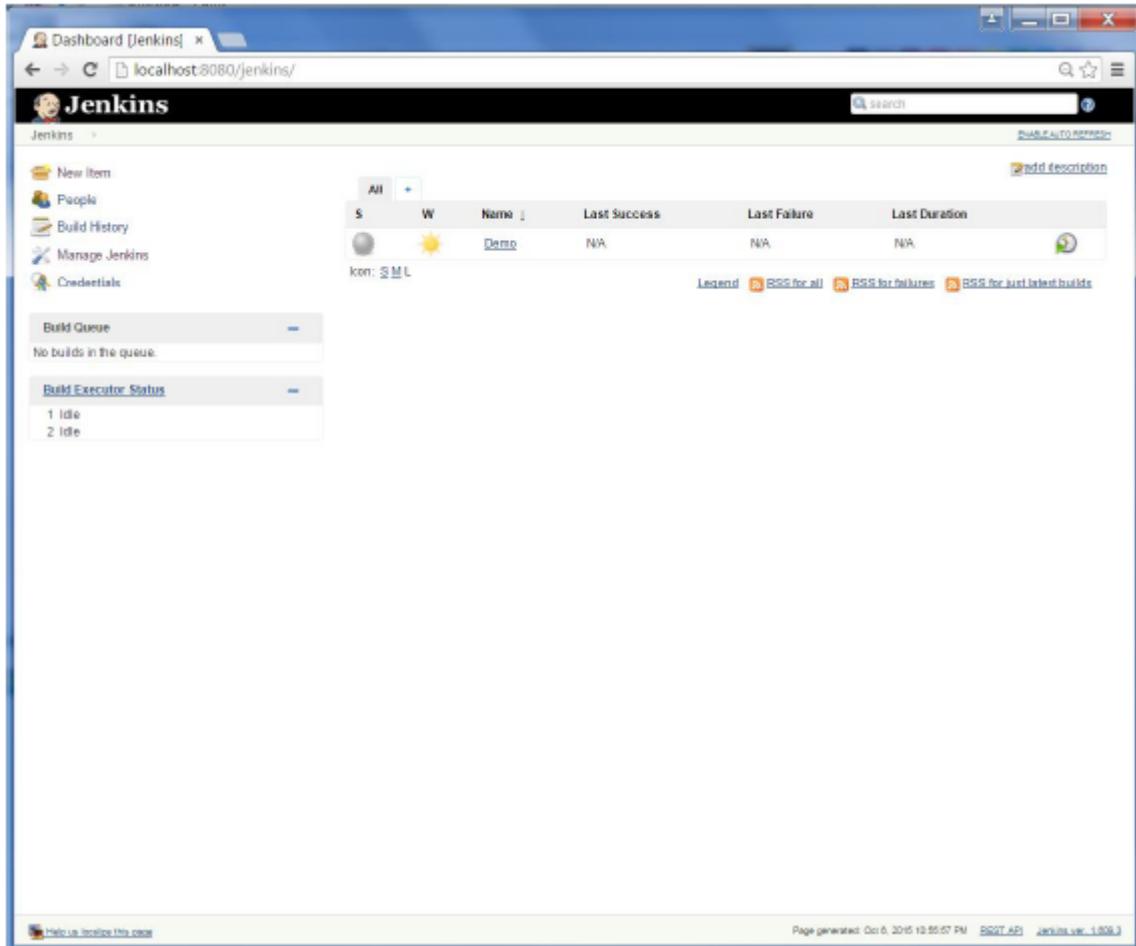
[Help us improve this page](#) Page generated: Oct 6, 2015 12:55:57 PM REST API Jenkins ver. 1.609.3



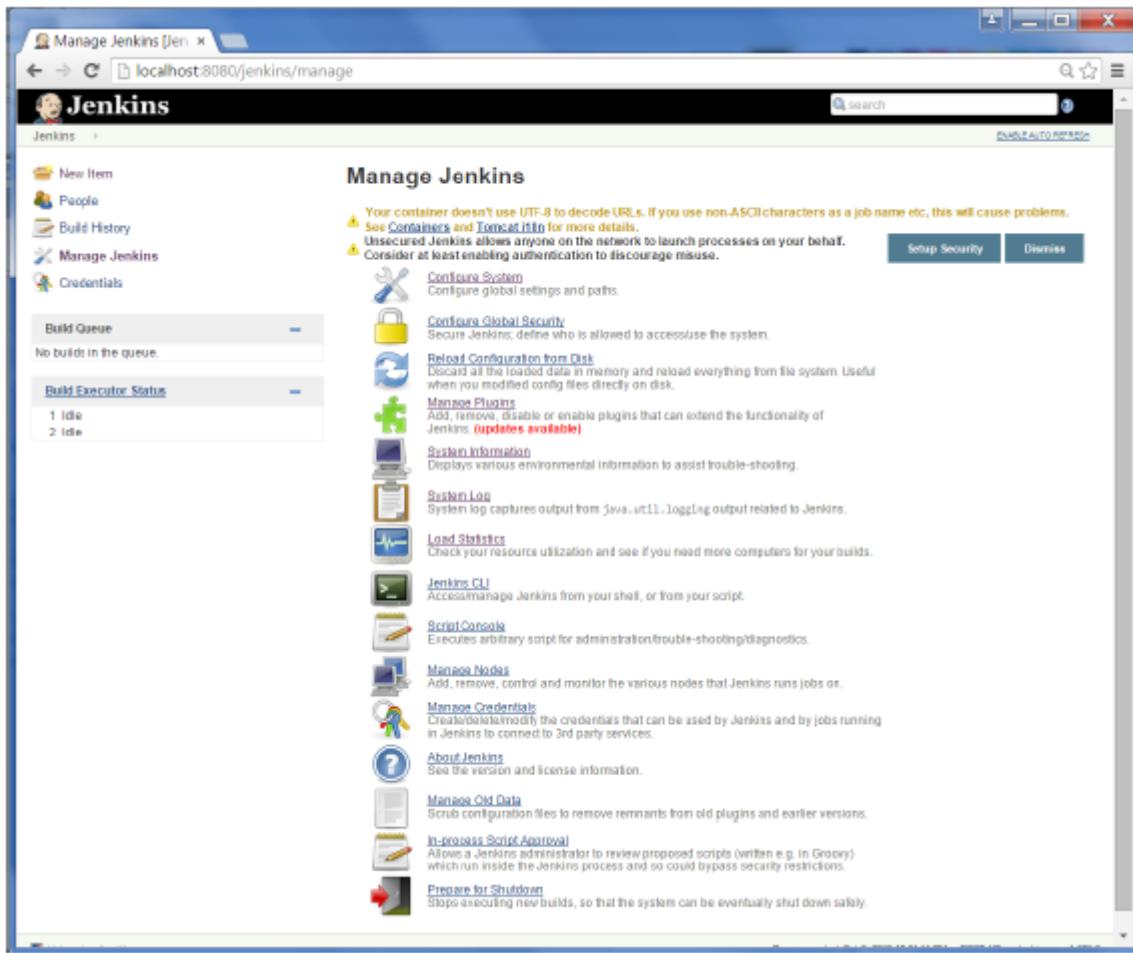
## Jenkins - Configuration

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.

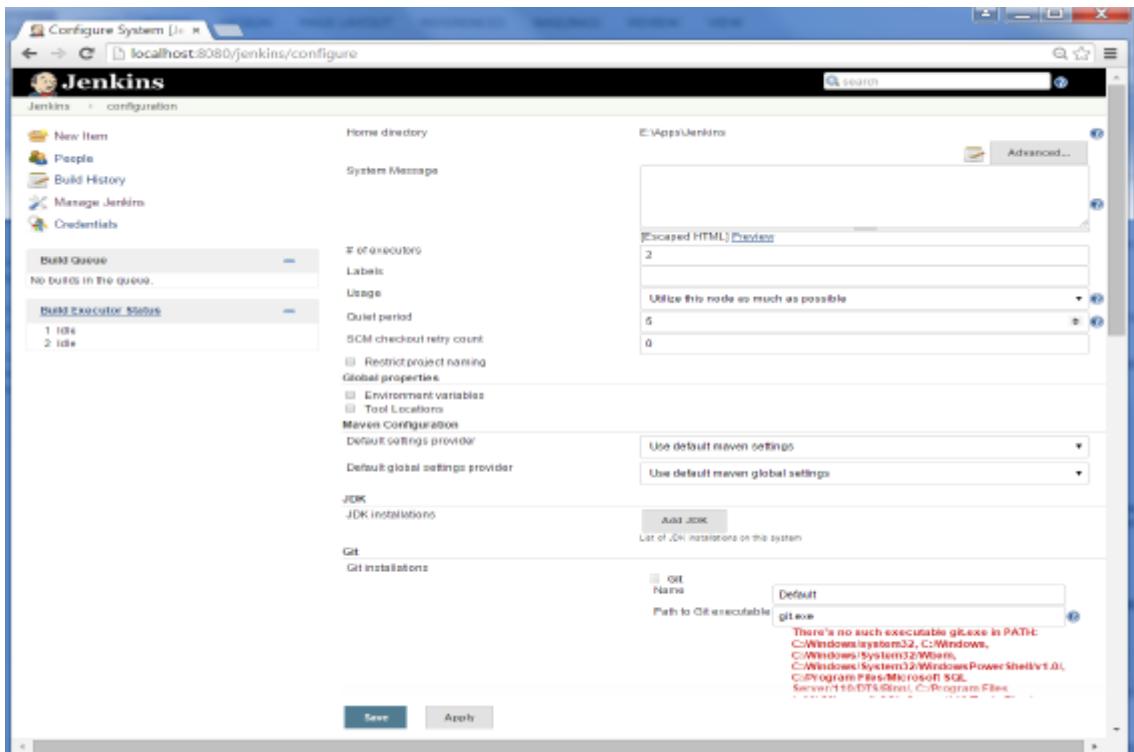
First create a new folder `E:\Apps\Jenkins`. Copy all the contents from the existing `~/.jenkins` to this new directory.

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}.

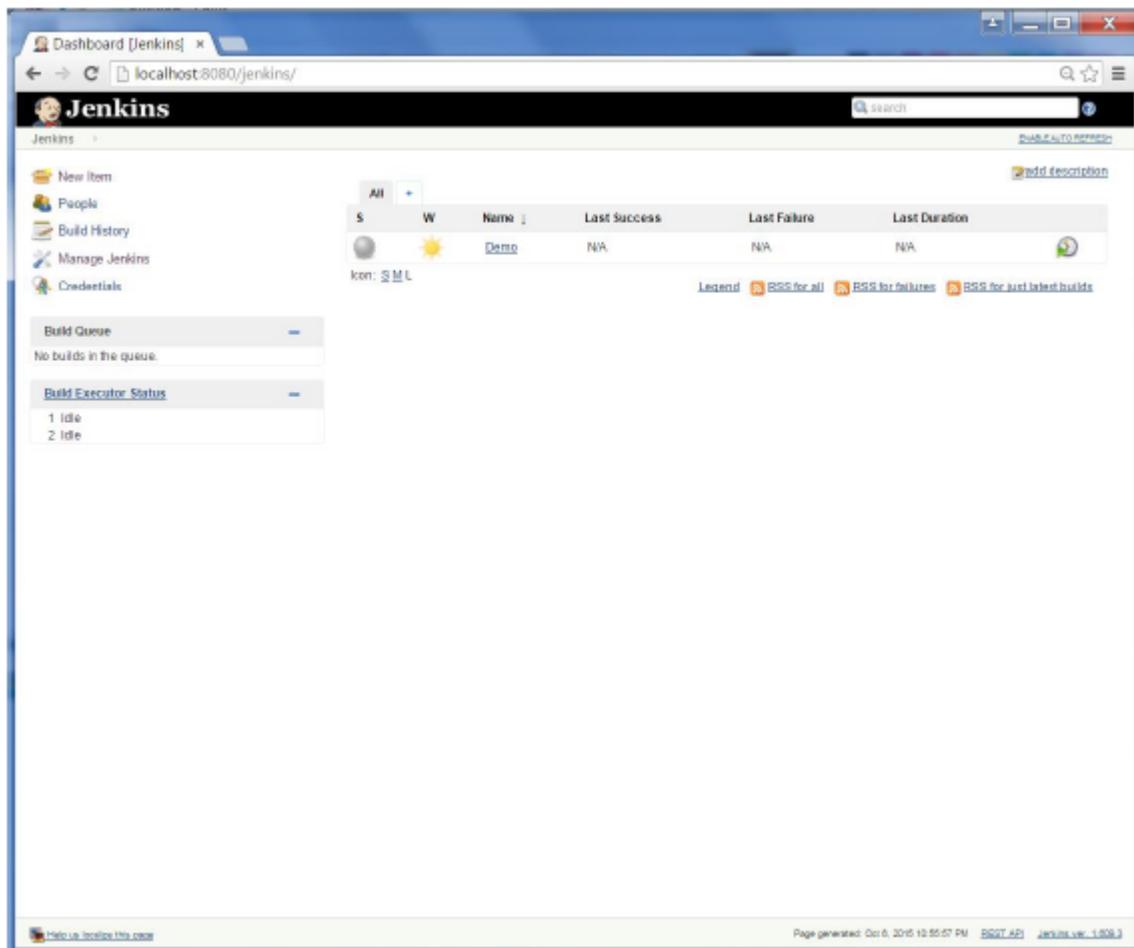
## Email Notification

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.

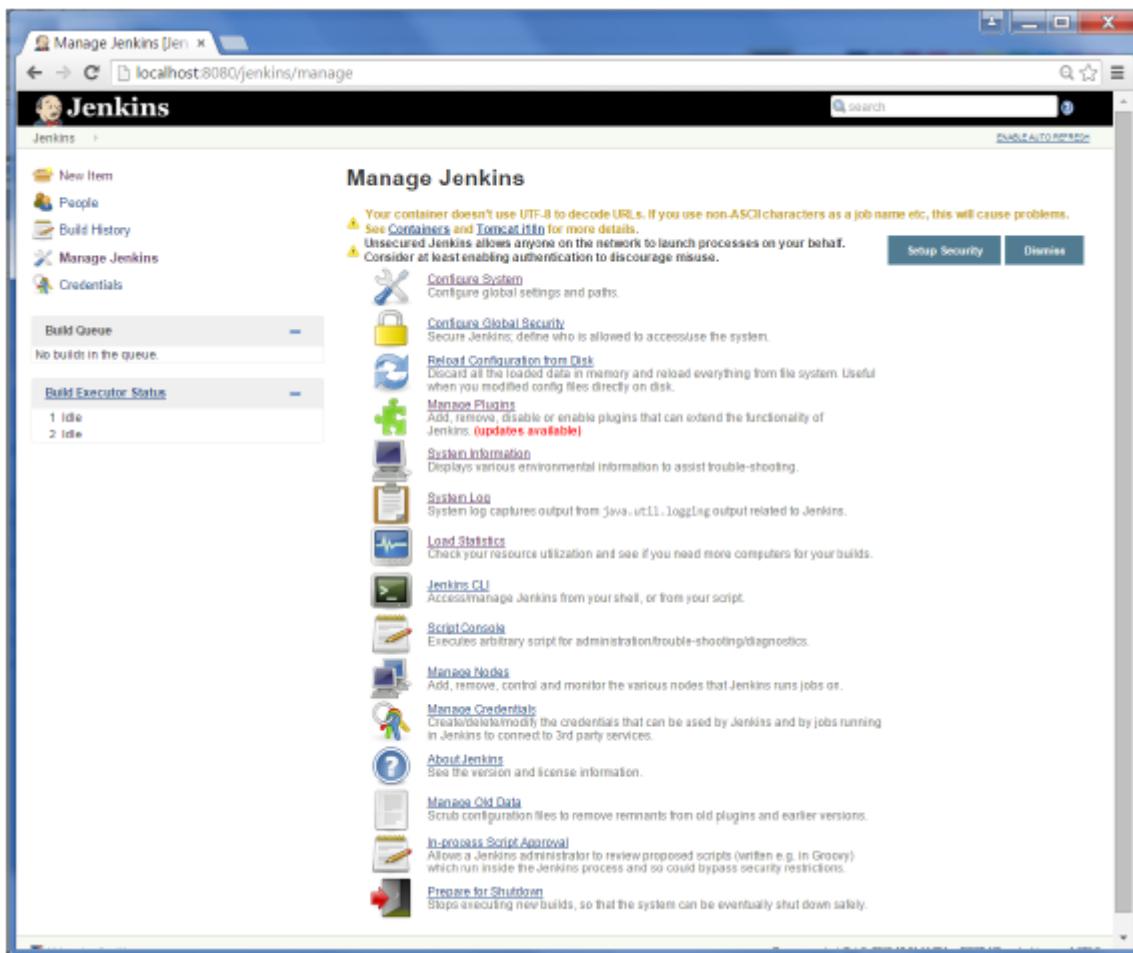
## Jenkins - Management

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

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



You will then be presented with the following screen –



Some of the management options are as follows –

## Configure System

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

## Reload Configuration from Disk

Jenkins stores all its system and build job configuration details as XML files which is stored in the Jenkins home directory. Here also all of the build history is stored. If you are migrating build jobs from one Jenkins instance to another, or archiving old build jobs, you will need to add or remove the corresponding build job directories to Jenkins's builds directory. You don't need to take Jenkins offline to do this—you can simply use the “Reload Configuration from Disk” option to reload the Jenkins system and build job configurations directly.

## Manage Plugin

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

The screenshot shows the Jenkins Update Center interface. At the top, there's a header with the Jenkins logo and a search bar. Below the header, a navigation bar has 'Back to Dashboard' and 'Manage Jenkins' links. The main content area is titled 'Plugin Manager' with tabs for 'Updates', 'Available', 'Installed', and 'Advanced'. A 'Filter' input field is at the top right of the table. The 'Updates' tab is selected, showing a table of available plugins:

Install	Name	Version	Installed
<a href="#">CVS Plug-in</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 2/3 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 Slaves plugin</a>	This plugin allows you to manage slaves running on Unix machines over SSH.	1.10	1.9
<a href="#">Subversion Plug-in</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 Slaves Plugin</a>	Allows you to connect to Windows machines and start slave agents on them.	1.1	1.0

Below the table, there are two buttons: 'Download now and install after restart' and 'Check now'. A note says 'Update information obtained: 1 hr 36 min ago'. At the bottom left, it says 'Select All, None' and 'This page lists updates to the plugins you currently use.' At the very bottom, there are links for 'Help us localize this page', 'Page generated: Oct 6 2015 11:08:25 PM', 'BROWSE API', and 'Jenkins ver. 1.609.3'.

## System Information

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

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

The screenshot shows the Jenkins System Information page at [localhost:8080/jenkins/systemInfo](http://localhost:8080/jenkins/systemInfo). The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. Under Build Queue, it says "No builds in the queue". Under Build Executor Status, there are 1 Idle and 2 Idle executors.

**System Properties**

Name	Value
awt.toolkit	sun.awt.windows.WToolkit
catalina.base	E:\Appstromcat7
catalina.home	E:\Appstromcat7
catalina.useNaming	true
common.loader	\$catalina.base\$/{catalina.base}/lib/*.jar,\$catalina.home\$/lib,\$catalina.home\$/{catalina.base}/lib/jar/*.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:\Appstromcat7\bin\bootstrap.jar;E:\Appstromcat7\bin\local-all.jar;51.0
java.class.version	E:\Appstromcat7\rendered
java.endorsed.dirs	C:\Program Files (x86)\Java\jdk1.7.0_79\jre\lib\endorsed;C:\Windows\Sun\Java\lib\endorsed
java.ext.dirs	C:\Program Files (x86)\Java\jdk1.7.0_79\jre
java.home	E:\Appstromcat7\temp
java.io.tmpdir	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\Windows;C:\Windows\System32\Windows\PowerShell\v1.0;C:\Program Files\Microsoft SQL Server\110\Tools\Binn;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\ManagementStudio;C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\PrivateAssemblies;C:\Program Files (x86)\Microsoft SQL Server\110\DT\Binn;C:\Program Files (x86)\Windows Kits\8.1\Windows Performance Toolkit;C:\Program Files (x86)\Microsoft SDKs\TypeScript\1.0;C:\Program Files\Microsoft SQL Server\120\Tools\Binn;C:\Program Files\Microsoft\Web Platform Installer;C:\Program Files (x86)\Java\jdk1.7.0_79\bin;%MAVEN_HOME%\bin;
java.naming.factory.initial	org.apache.naming.java.javaURLContextFactory
java.naming.factory.url.pkgs	org.apache.naming
java.runtime.name	Java(TM) SE Runtime Environment
java.runtime.version	1.7_0_79-b15
java.specification.name	Java Platform API Specification
java.specification.vendor	Oracle Corporation
java.specification.version	1.7
java.util.logging.config.file	E:\Appstromcat7\conf\logging.properties
java.util.logging.manager	org.apache.juli.ClassLoaderLogManager
java.vendor	Oracle Corporation
java.vendor.url	<a href="http://java.oracle.com/">http://java.oracle.com/</a>
java.vendor.url_bug	<a href="http://bugreport.sun.com/bugreport/">http://bugreport.sun.com/bugreport/</a>
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.

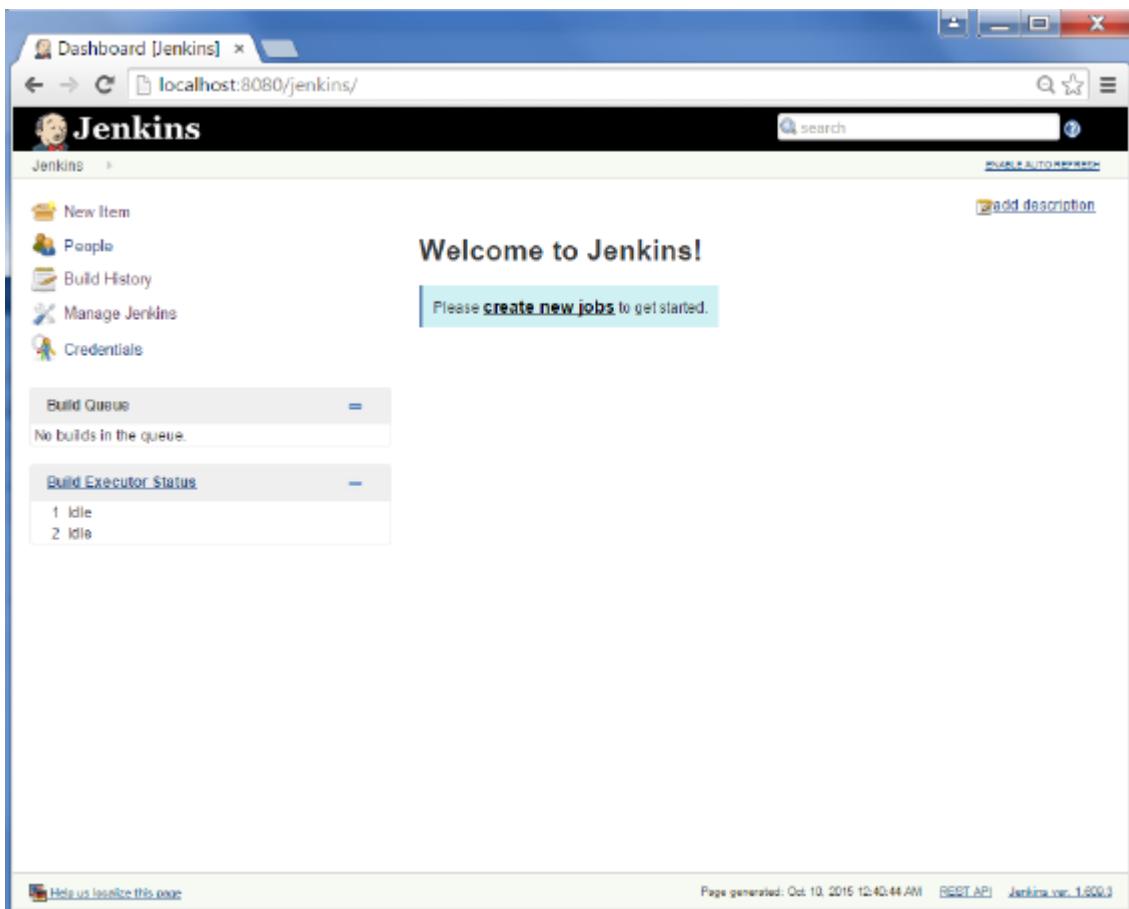
## Prepare for Shutdown

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.

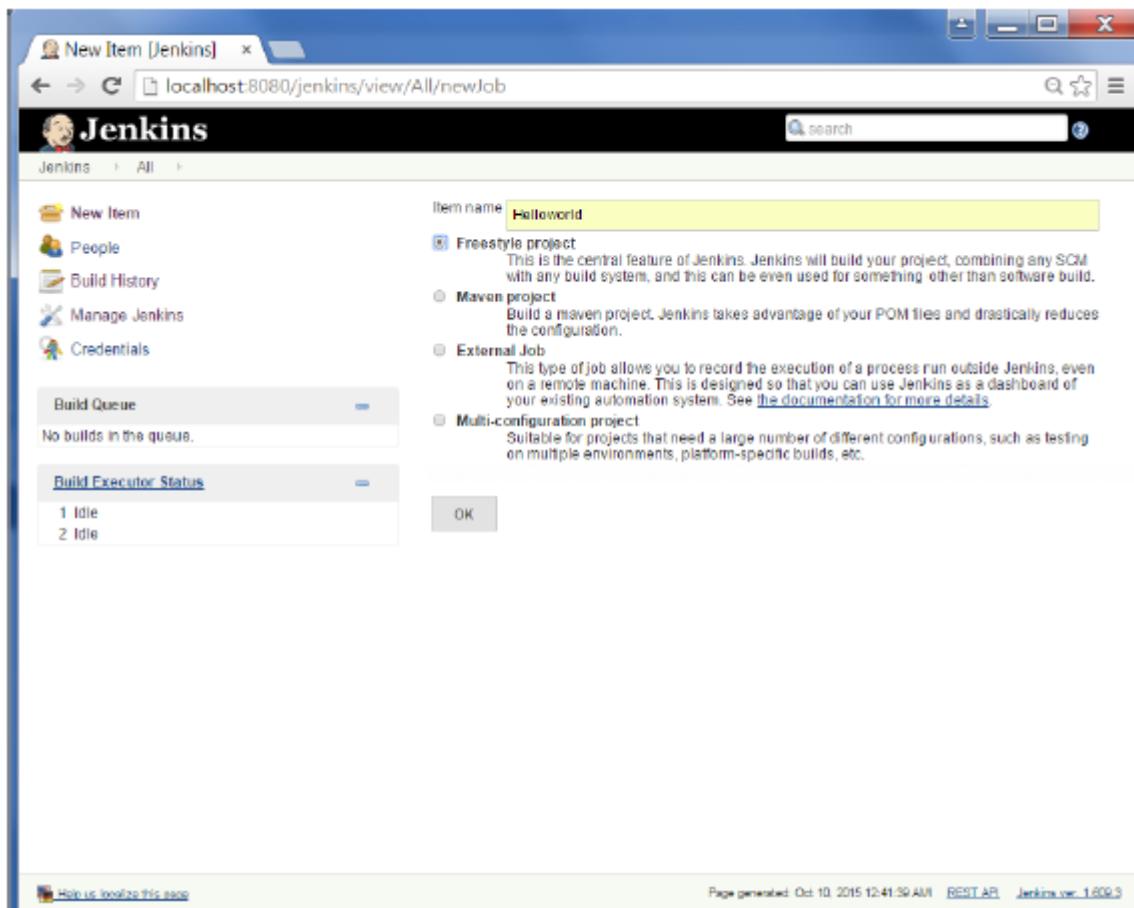
# Jenkins - Setup Build Jobs

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

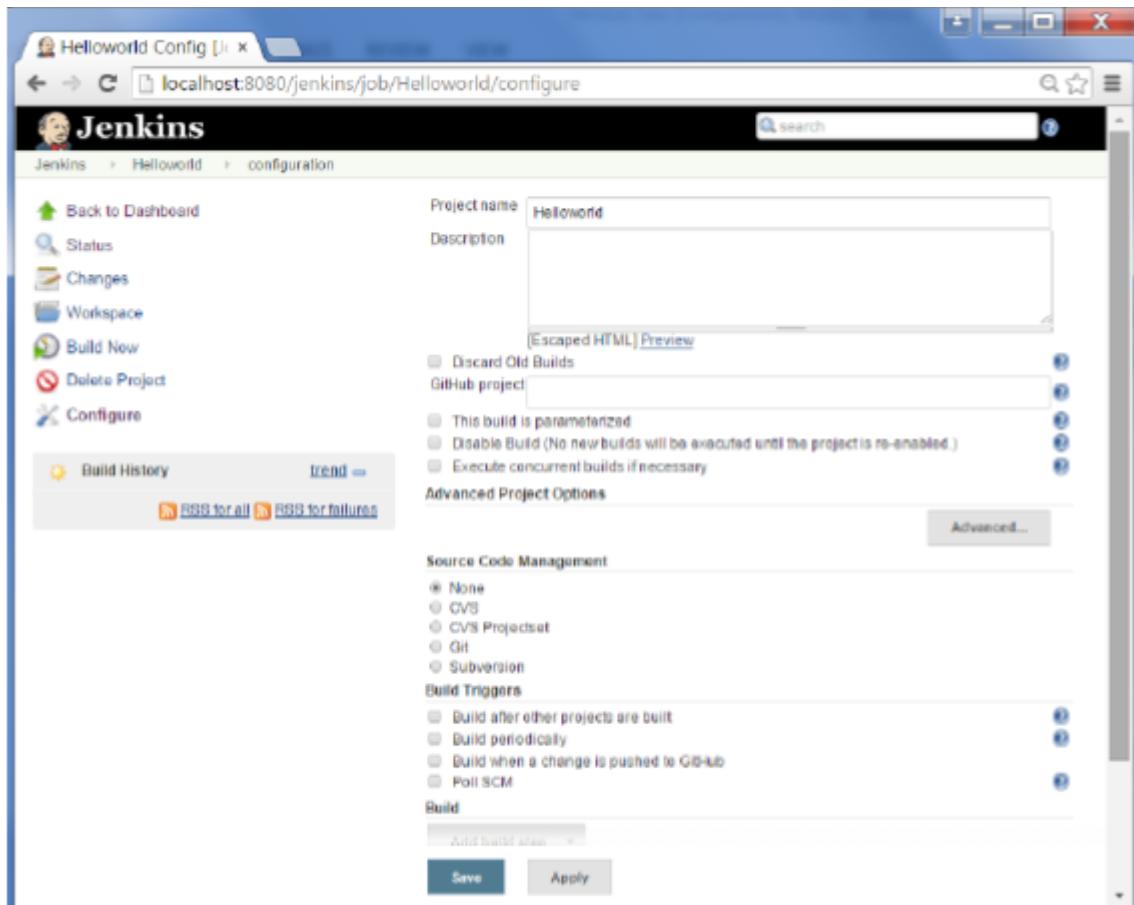
**Step 1** – Go to the Jenkins dashboard and Click on New Item



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

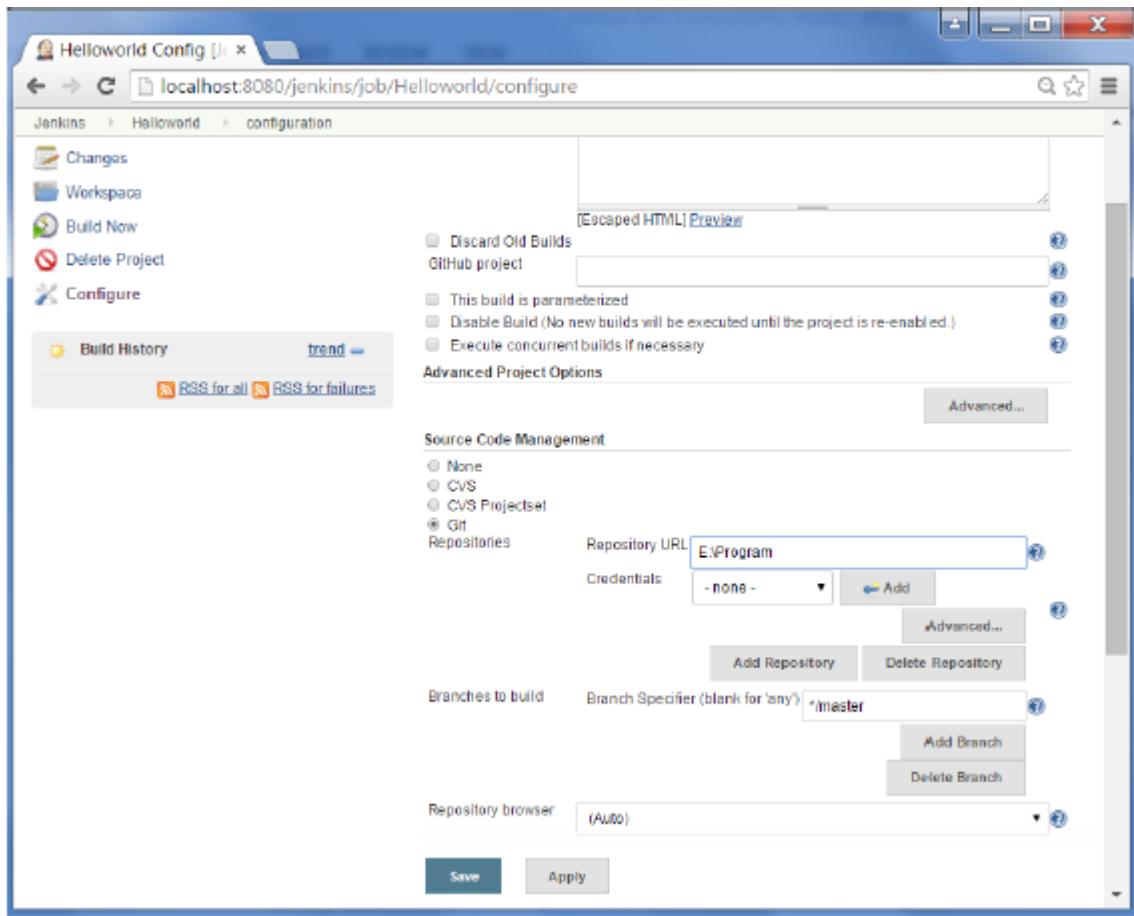


**Step 3** – The following screen will come up in which you can specify the details of the job.

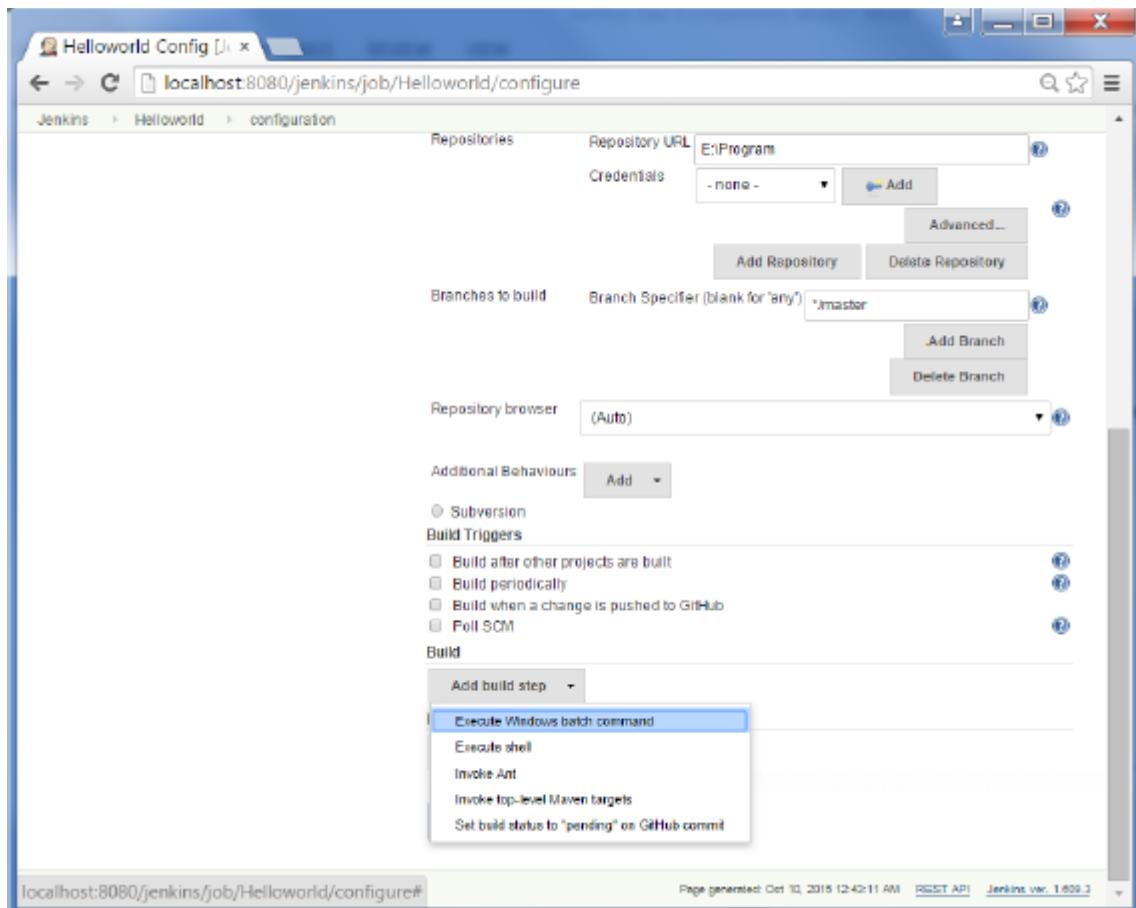


**Step 4** – We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a ‘HelloWorld.java’ file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

**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.

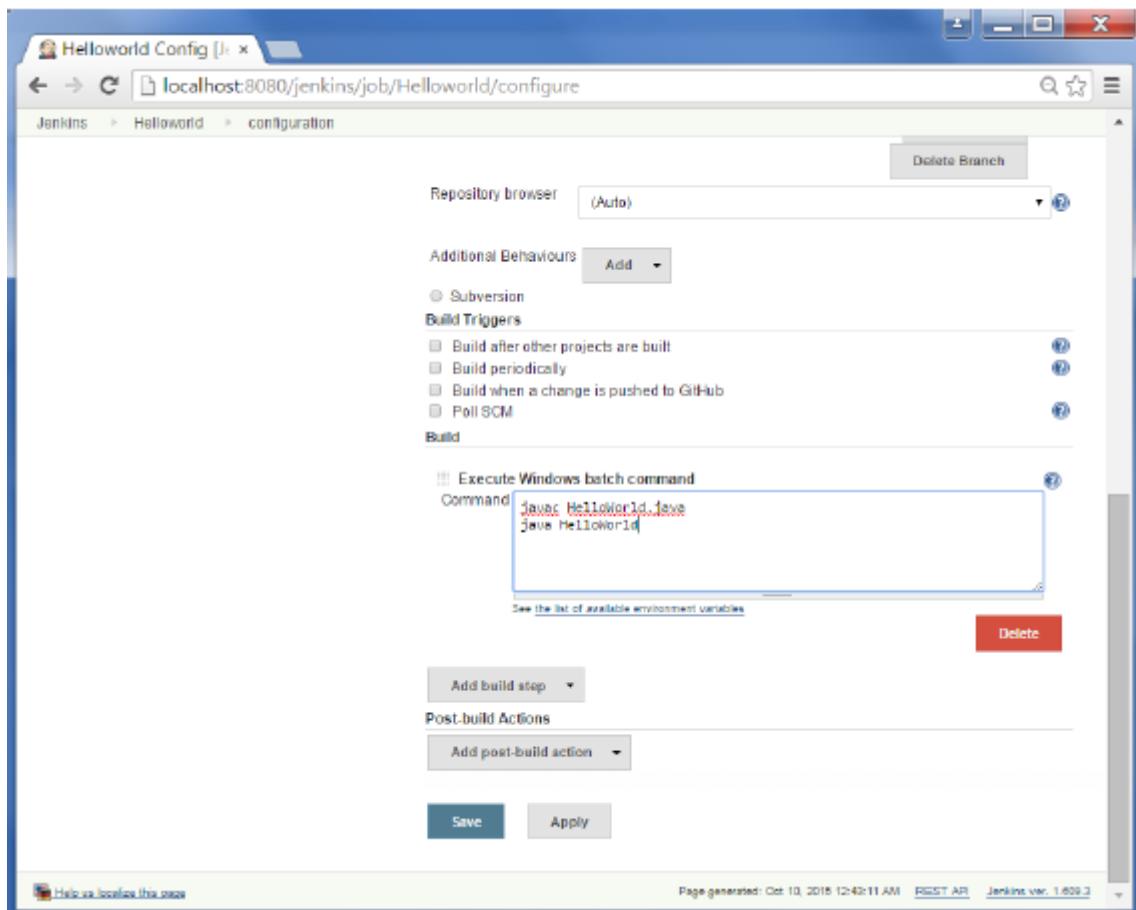


**Step 5** – Now go to the Build section and click on Add build step → Execute Windows batch command



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

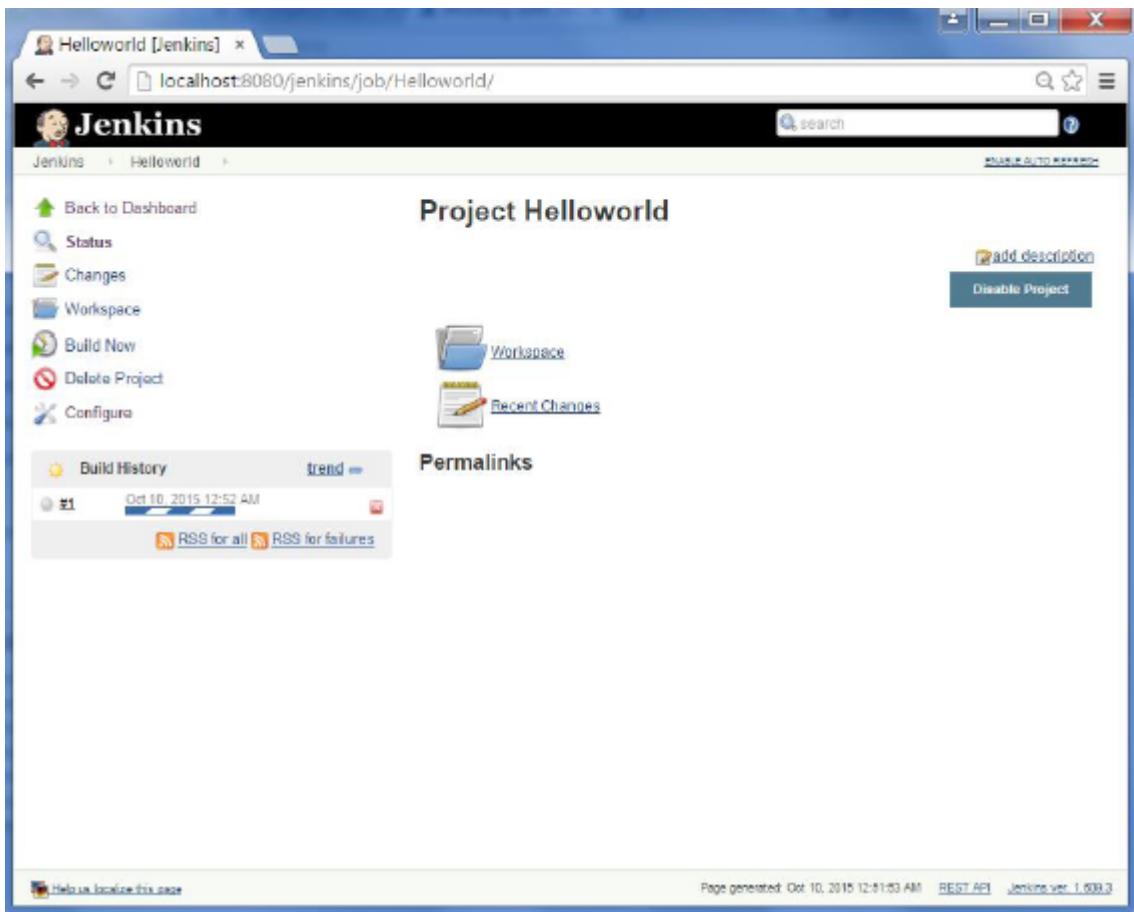
```
Javac HelloWorld.java  
Java HelloWorld
```



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

The screenshot shows the Jenkins interface for the 'Helloworld' project. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. Below this is a search bar and a 'Disable Project' button. A 'Project Helloworld' title is centered above a main content area. On the left, there are two buttons: 'Workspace' and 'Recent Changes'. On the right, there's a 'Permalinks' section with links for 'RSS for all' and 'RSS for failures'. At the bottom, there's a footer with a 'Help us localize this page' link, a timestamp 'Page generated: Oct 10, 2015 12:51:03 AM', and a Jenkins version 'Jenkins ver. 1.600.3'.

**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.

The screenshot shows the Jenkins interface for the 'Helloworld' project. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. Below this is a 'Build History' section showing one build from Oct 10, 2015, at 12:52 AM. There are links for 'RSS for all' and 'RSS for failures'. To the right of the history is a 'Project Helloworld' summary with sections for 'Workspace' and 'Recent Changes'. A 'Disable Project' button is visible in the top right corner. The bottom of the page includes a footer with links for 'Help us localize this page', 'Page generated: Oct 10, 2015 12:51:53 AM', 'REST API', and 'Jenkins ver. 1.609.3'.

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

Helloworld #1 [Jenkins] x

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

# Jenkins

Back to Project Status Changes Console Output Edit Build Information Delete Build Git Build Data No Tags

**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: 42f9a82fadd88fb5c3a9dfe40e731a90715c8f  
+ refs/remotes/origin/master

[Add description](#)

Help us localize this page Page generated: Oct 10, 2015 12:57:31 AM REST API Jenkins ver. 1.609.3

The screenshot shows a Jenkins job named "HelloWorld" with build number #12. The console output window displays the following log:

```
Started by user anonymous
Building in workspace E:\Jenkins\jobs\HelloWorld\workspace
> C:\Program Files\Git\bin\git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url E:\Program #
timeout=10
Fetching upstream changes from E:\Program
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> C:\Program Files\Git\bin\git.exe -c core.askpass=true fetch --tags --progress
E:\Program\refs/heads/*:refs/remotes/origin/*
> C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/master^{commit}" # timeout=10
> C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/origin^{commit}" # timeout=10
Checking out Revision 42f9a82ffadd86fb05c3a9dfeae40e731a907f5c8f
(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
42f9a82ffadd86fb05c3a9dfeae40e731a907f5c8f
> C:\Program Files\Git\bin\git.exe rev-list
42f9a82ffadd86fb05c3a9dfeae40e731a907f5c8f # timeout=10
[workspace] $ cmd / call E:\Apps\tomcat7\temp\hudson1928478766077504601.bat
E:\Jenkins\jobs\HelloWorld\workspace>javac HelloWorld.java
E:\Jenkins\jobs\HelloWorld\workspace>java HelloWorld
Hello World
E:\Jenkins\jobs\HelloWorld\workspace>exit 0
Finished: SUCCESS
```

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

## Jenkins - Unit Testing

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

xUnit Plugin - Jenkins

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

Dashboard > Jenkins > Plugins > xUnit Plugin

Browse Search

 xUnit Plugin

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

**Jenkins**

- Home
- Mailing lists
- Source code
- Bugtracker
- Security Advisories
- Events
- Donation
- Commercial Support
- Wiki Site Map

**Documents**

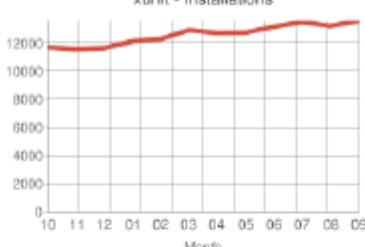
- Meet Jenkins
- Use Jenkins
- Extend Jenkins
- Plugins
- Servlet Container Notes

**Plugin Information**

Plugin ID	xunit	Changes	In Latest Release Since Latest Release
Latest Release	<a href="#">1.98 (archives)</a>	<a href="#">Source Code</a>	<a href="#">GitHub</a>
Latest Release Date	Oct 09, 2015	<a href="#">Issue Tracking</a>	<a href="#">Open Issues</a>
Required Core Dependencies	<a href="#">1.580.1</a> <a href="#">junit</a> (version: 1.6)	<a href="#">Pull Requests</a>	<a href="#">Pull Requests</a>
Maintainer(s)	<a href="#">Gregory Boissinot (id: gboissinot)</a>		

**Usage**

xunit - installations



Month	Installations
Oct 14	11692
Nov 14	11557
Dec 14	11631
Jan 15	12106
Feb 15	12262
Mar 15	12891
Apr 15	12694
May 15	12716
Jun 15	13143
Jul 15	13470
Aug 15	13192
Sep 15	13563

**Installations**

2014-Oct 11692  
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**

**Features**

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

**Supported tools**

**Embedded tools**

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

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

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

**External contributions**

## Example of a Junit Test in Jenkins

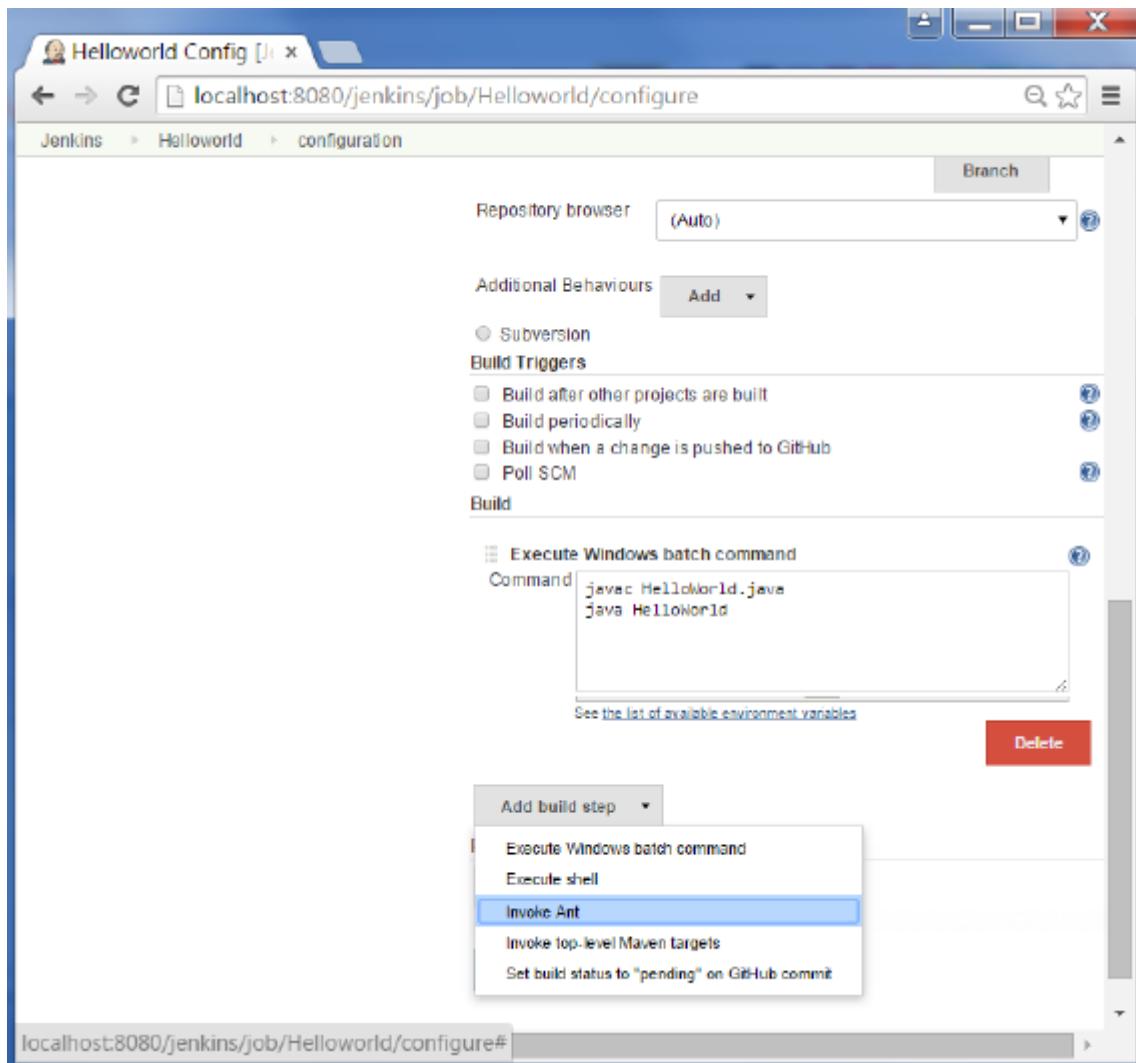
The following example will consider

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

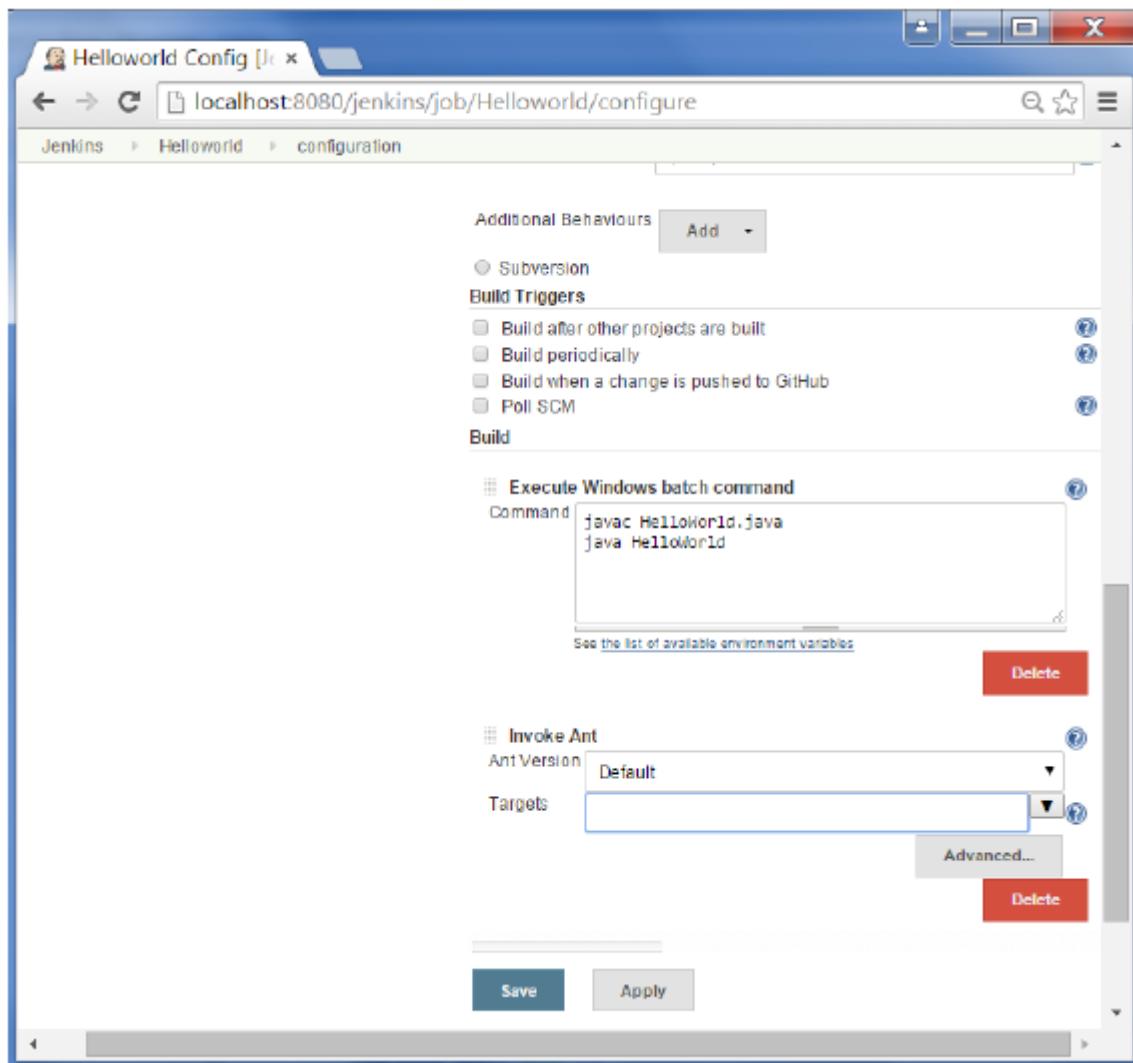
**Step 1** – Go to the Jenkins dashboard and Click on the existing HelloWorld project and choose the Configure option

The screenshot shows the Jenkins dashboard at [localhost:8080/jenkins/](http://localhost:8080/jenkins/). On the left, a sidebar lists navigation options: New Item, People, Build History, Manage Jenkins, and Credentials. Below these are sections for Build Queue (empty) and Build Executor Status, which shows one idle master and one offline build slave named 'build\_slave'. The main area displays the configuration for the 'Helloworld' job. The job's icon is a blue sphere, and its name is 'Helloworld'. It has a status of '6 sec - #11' (Last Success), '2 days 23 hr - #10' (Last Failure), and a duration of '3.2 sec'. Below the table are links for Changes, Workspace, Build Now, Delete Project, and Configure (which is highlighted with a blue border). At the bottom of the page, the URL is 'localhost:8080/jenkins/job/.../configure' and the footer indicates the page was generated on Oct 15, 2015 at 10:29:01 PM, with REST API and Jenkins ver. 1.809.9.

**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.

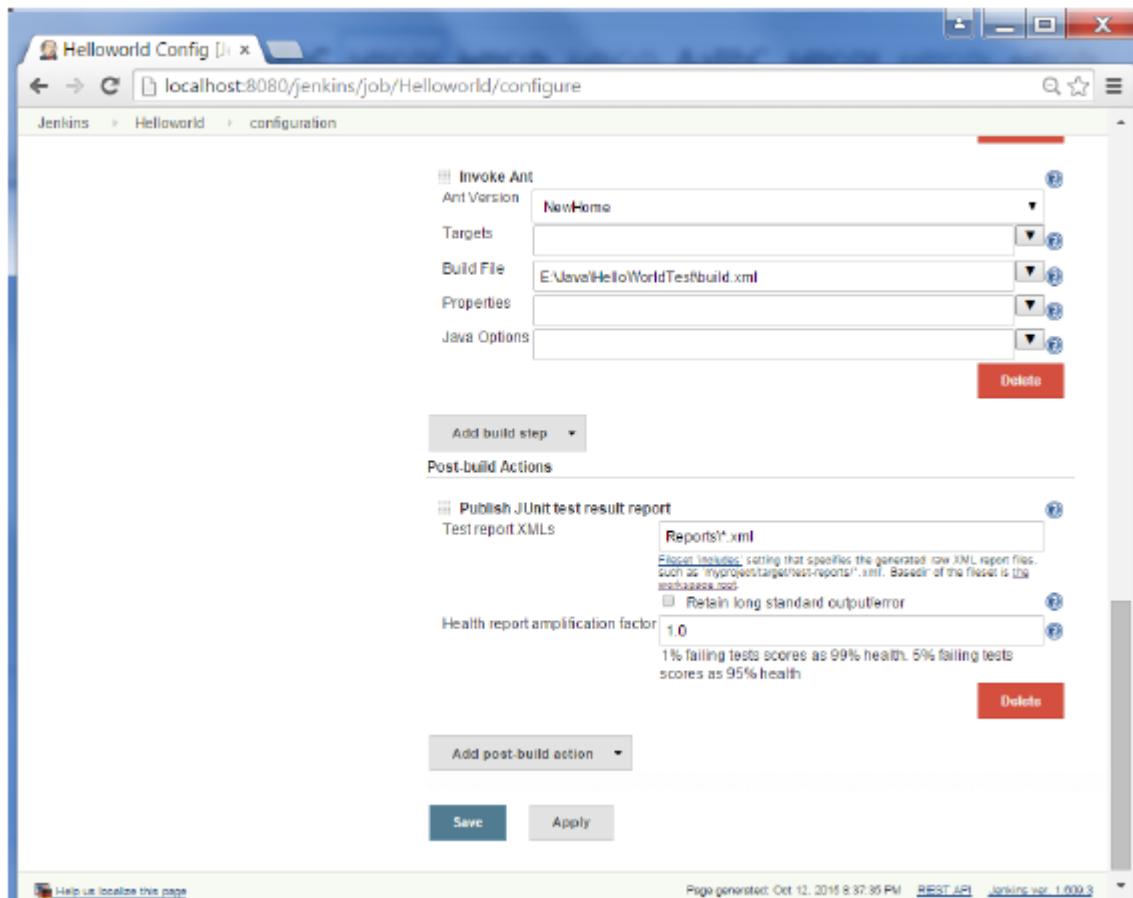
The screenshot shows the Jenkins configuration page for a job named "Helloworld". Under the "Build" section, there are two steps: "Execute Windows batch command" with the command "javac HelloWorld.java" and "java HelloWorld"; and "Invoke Ant" with the Ant Version set to "NewHome", Targets set to "", Build File set to "E:\JavaHelloWorldTest\build.xml", Properties set to "", and Java Options set to "".

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

The screenshot shows the Jenkins configuration page for the same job. In the "Post-build Actions" section, a context menu is open over the "Publish JUnit test result report" option. The menu lists several actions: Publish FindBugs analysis results, Publish combined analysis results, Aggregate downstream test results, Archive the artifacts, Build other projects, Publish JUnit test result report (which is highlighted with a blue selection bar), Publish Javadoc, Publish Selenium Report, Record fingerprints of files to track usage, Git Publisher, Deploy war/ear to a container, E-mail Notification, and Set build status on GitHub commit. At the bottom of the menu, there is an "Add post-build action" button.

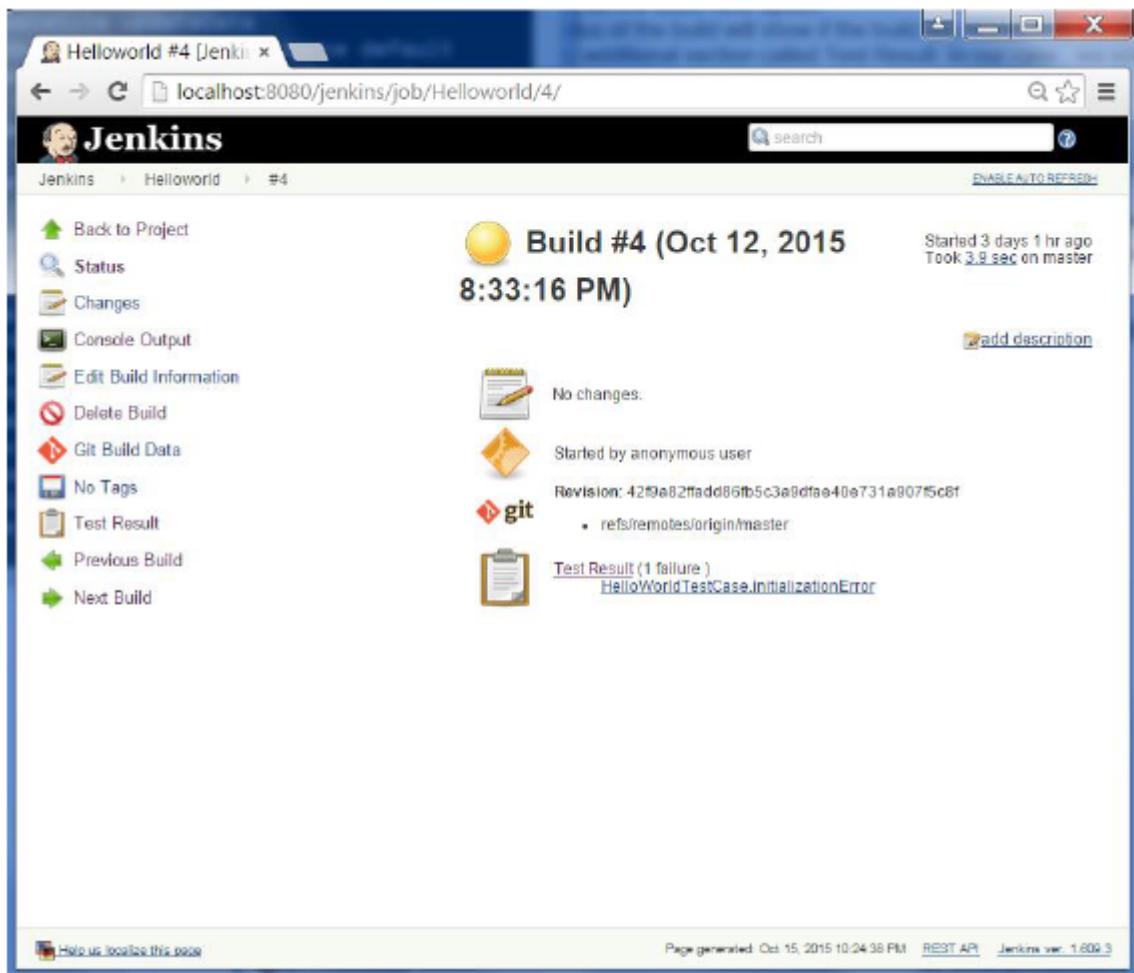
**Step 6** – In the Test reports XML's, enter the location as shown below. Ensure that Reports is a folder which is created in the HelloWorld project workspace. The “\*.xml” basically tells Jenkins to pick up the result xml files which are produced by the running of the Junit test cases. These xml files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



A screenshot of a web browser displaying a Jenkins job details page. The URL in the address bar is `localhost:8080/jenkins/job/Helloworld/4/`. The page title is "Helloworld #4 [Jenkins]". On the left, there's a sidebar with links like "Back to Project", "Status", "Changes", etc. The main content area shows "Build #4 (Oct 12, 2015 8:33:16 PM)". It includes a yellow circle icon, the build number, date, time, and duration ("Took 3.9 sec on master"). A "No changes." message with a notepad icon is shown. The build was started by an anonymous user. The revision is listed as `42f9a82ffadd86fb5c3a8dfeae40e731a807f5c8f`, with a "git" icon. Under "Test Result", it shows "Test Result (1 failure)" with a clipboard icon, linking to "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. 1.609.3".

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.

The screenshot shows the Jenkins Test Result page for a build named "Helloworld #4". The main title is "Test Result" with a subtitle "1 failures". A red bar indicates the failure count. Below it, a table lists "All Failed Tests" with one entry: "HelloWorldTestCase.InitializationError" which took 10 ms. Another table shows "All Tests" with a single row for "root" package, indicating 1 failure and 1 pass. The left sidebar contains links for Back to Project, Status, Changes, Console Output, Edit Build Information, History, Git Build Data, No Tags, Test Result, and Previous Build.

## Jenkins - Automated Testing

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

**Step 1** – Go to Manage Plugins.

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below that are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 idle). The main area is titled 'Manage Jenkins' and contains a warning about non-UTF-8 URLs. It lists various management tasks with icons:

- Configure System
- Configure Global Security
- Reload Configuration from Disk
- Manage Plugins
- System Information
- System Log
- Load Statistics
- Jenkins CLI
- Script Console
- Manage Nodes
- Manage Credentials
- About Jenkins

At the top right, there are 'Setup Security' and 'Dismiss' buttons.

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

The screenshot shows the Jenkins Plugin Manager. The 'Available' tab is selected. A search bar at the top right is set to 'Filter: selenium'. The 'Install' section lists several Selenium-related plugins:

Name	Version
Selenium Auto Exec Server(AES) plugin	0.5
Hudson SeleniumHTML plugin	0.4
Selenium HTML report	0.94
TestingBot plugin	1.11
TestLink Plugin	3.10
Nemvana Plugin for Jenkins	1.02.06
Source OnDemand plugin	1.141
Selenium Builder plugin	1.14
SeleniumRC plugin	1.0

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

### Step 3 – Go to Configure system.

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

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

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

The screenshot shows the Jenkins 'Configure System' page. It includes sections for CVS, Subversion, Selenium Remote Control, Shell, and E-mail Notification. The 'Selenium Remote Control' section has a 'htmlSuite Runner' field set to 'E:\App\seleium-server-standalone-2.48.2.jar'. At the bottom, there are 'Save' and 'Apply' buttons.

**Note** – The selenium jar file can be downloaded from the location SeleniumHQ

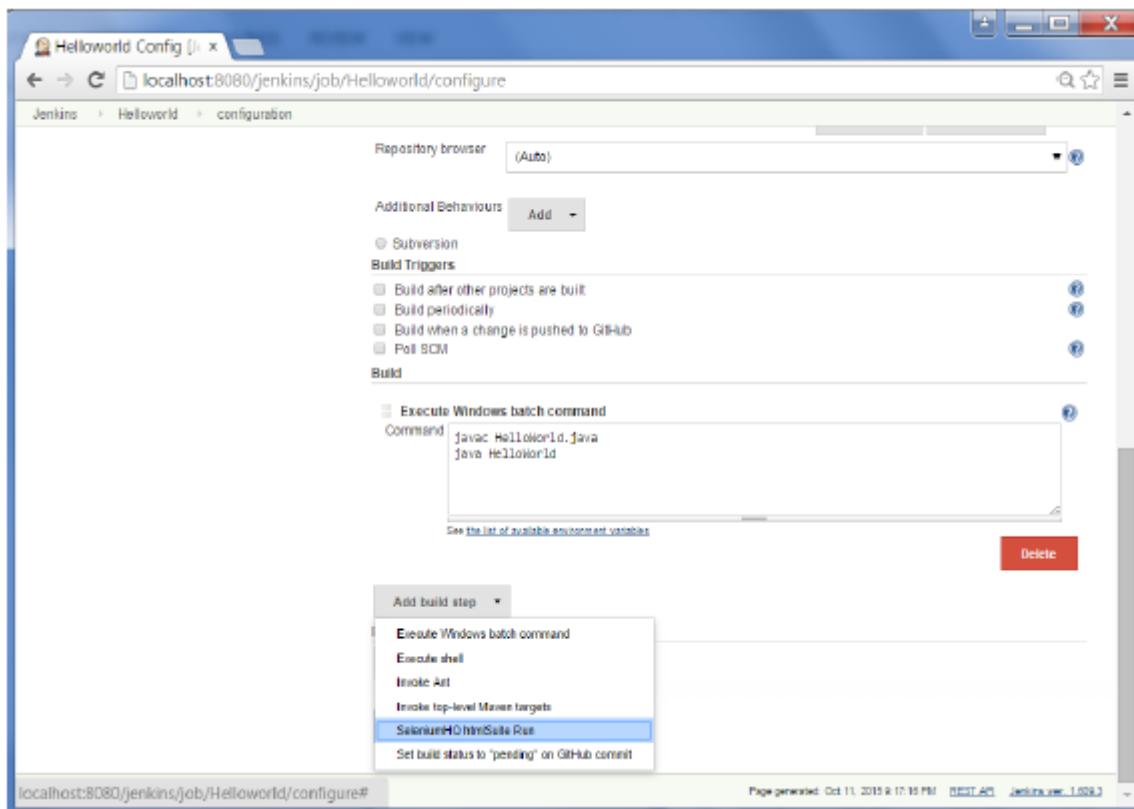
Click on the download for the Selenium standalone server.

The screenshot shows the SeleniumHQ website's 'Downloads' page. The main content area is titled 'Downloads' and contains information about the Selenium Standalone Server. It states that the server is needed for running Selenium RC style scripts or Remote WebDriver ones. A link to download version 2.48.2 is provided, along with instructions for using it in a Grid configuration and a link to the wiki page. There is also a section for the Internet Explorer Driver Server, which is required for using the WebDriver InternetExplorerDriver. A link to download version 2.48.0 for IE is given, along with a CHANGELOG link. The page includes a sidebar with links for Selenium Downloads, Latest Releases, Previous Releases, Source Code, and Maven Information. It also features a 'Donate to Selenium' section with a PayPal donation button and a 'Selenium Sponsors' section listing 'BrowserStack'.

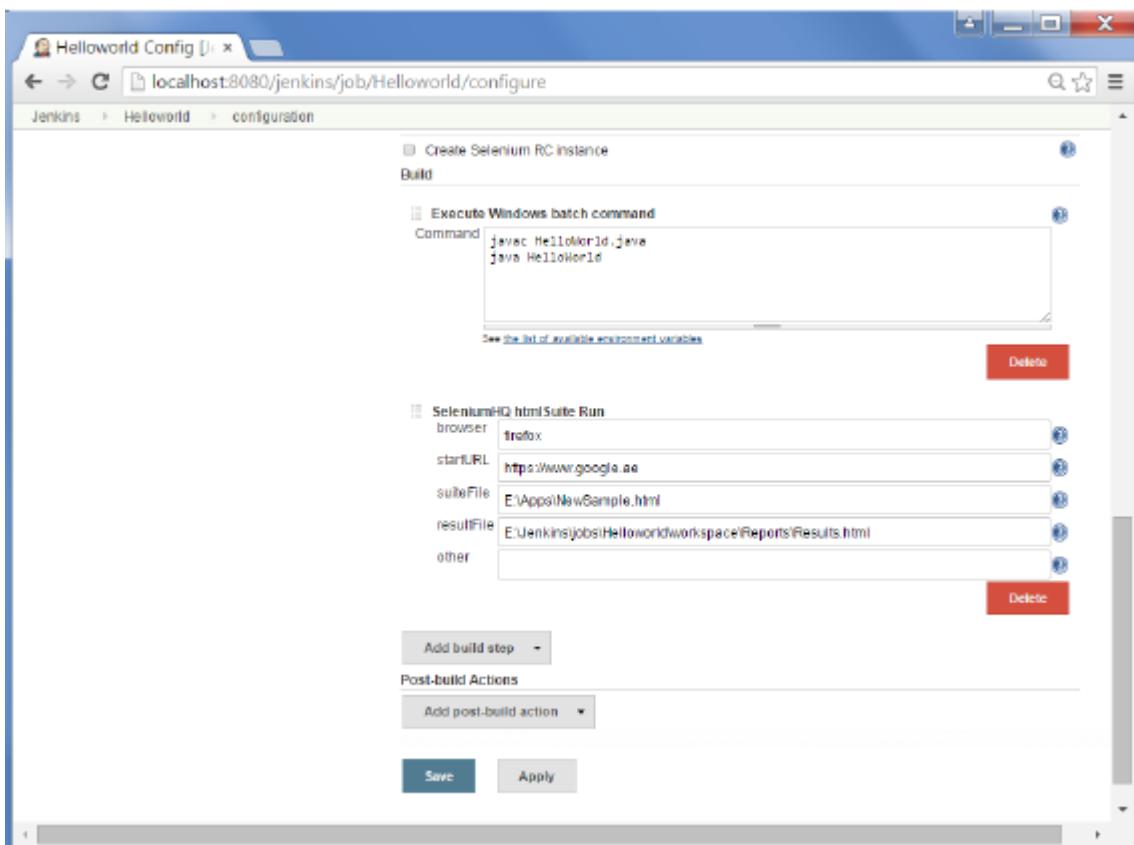
**Step 5** – Go back to your dashboard and click on the Configure option for the HelloWorld project.

The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with links for New Item, People, Build History, Manage Jenkins, and Credentials. Below that are sections for Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). The main area displays a table of projects. The 'HelloWorld' project is listed with its last success at 23 hr - #12, last failure at 23 hr - #10, and last duration at 3.7 sec. A context menu is open over the HelloWorld row, showing options: Changes, Workspace, Build Now, Delete Project, and Configure. The 'Configure' option is highlighted. At the bottom of the screen, the URL 'localhost:8080/jenkins/job/HelloWorld/configure' is visible in the address bar, along with page generation details: 'Page generated: Oct 11, 2015 @ 16:04 PM' and 'Build API'.

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



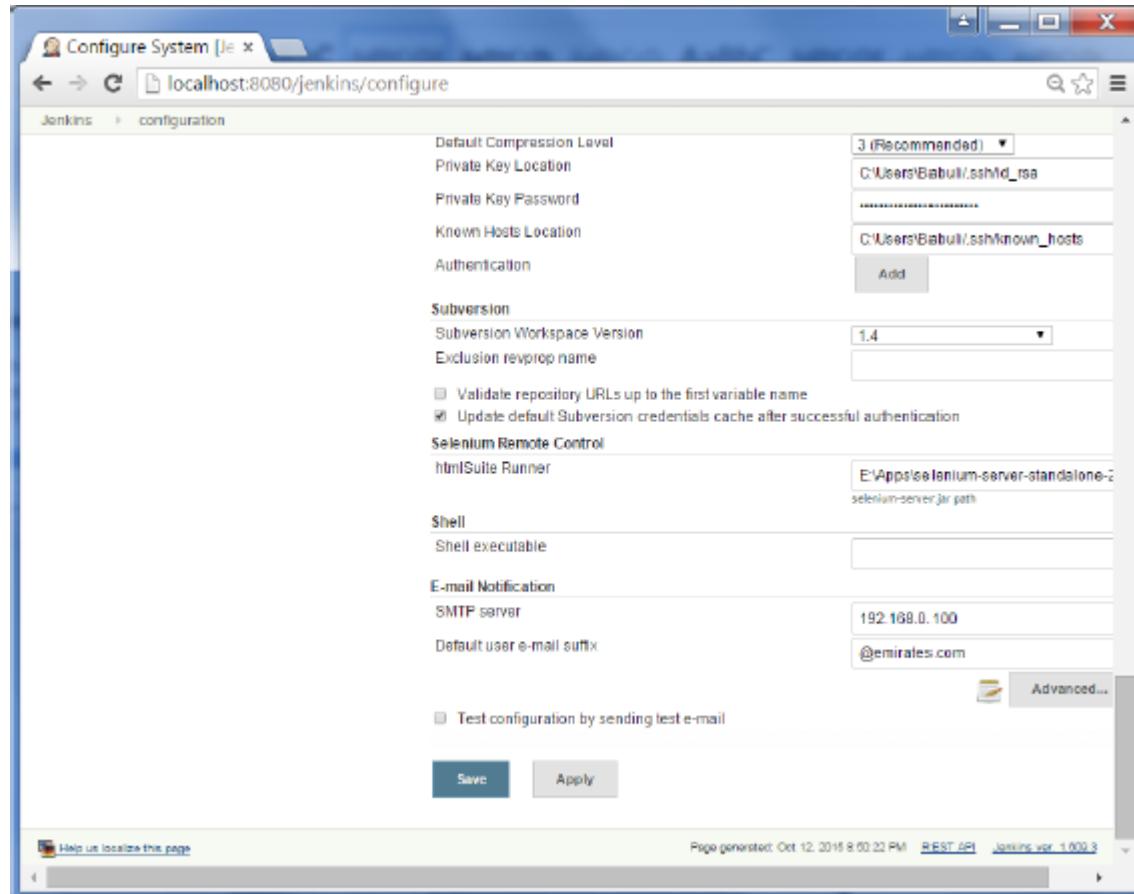
**Step 7 – Add the necessary details for the selenium test.** Here the suiteFile is the TestSuite generated by using the Selenium IDE. Click on Save and execute a build. Now the post build will launch the selenium driver, and execute the html test.



# 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'. Under 'Build Steps', there is a single entry: 'SeleniumHQ html Suite Run' with the browser set to 'ieexplorer', startURL to 'https://www.google.ae', suiteFile to 'E:\App\ Selenium\Sample.html', and resultFile to 'Result.html'. Under 'Post-build Actions', there is an 'E-mail Notification' action configured to send to '811233@domain.com' for every unstable build. Buttons at the bottom include 'Save' and 'Apply'.

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'. On the left sidebar, 'Configure' is selected. In the main area, under 'Job Notifications', there is a 'Notification Endpoints' section. A new endpoint is being configured with the following details: Format: 'JSON', Protocol: 'HTTP', Event: 'All Events', URL: 'http://dxbmrm30/1xe4567h', Timeout: '30000', and Log: '0'. There is an 'Add Endpoint' button and a list of checkboxes for project options: 'This build is parameterized', 'Disable Build (No new builds will be executed until the project is re-enabled.)', and 'Execute concurrent builds if necessary'. A 'Save' and 'Apply' button are at the bottom.

Here are the details of each option –

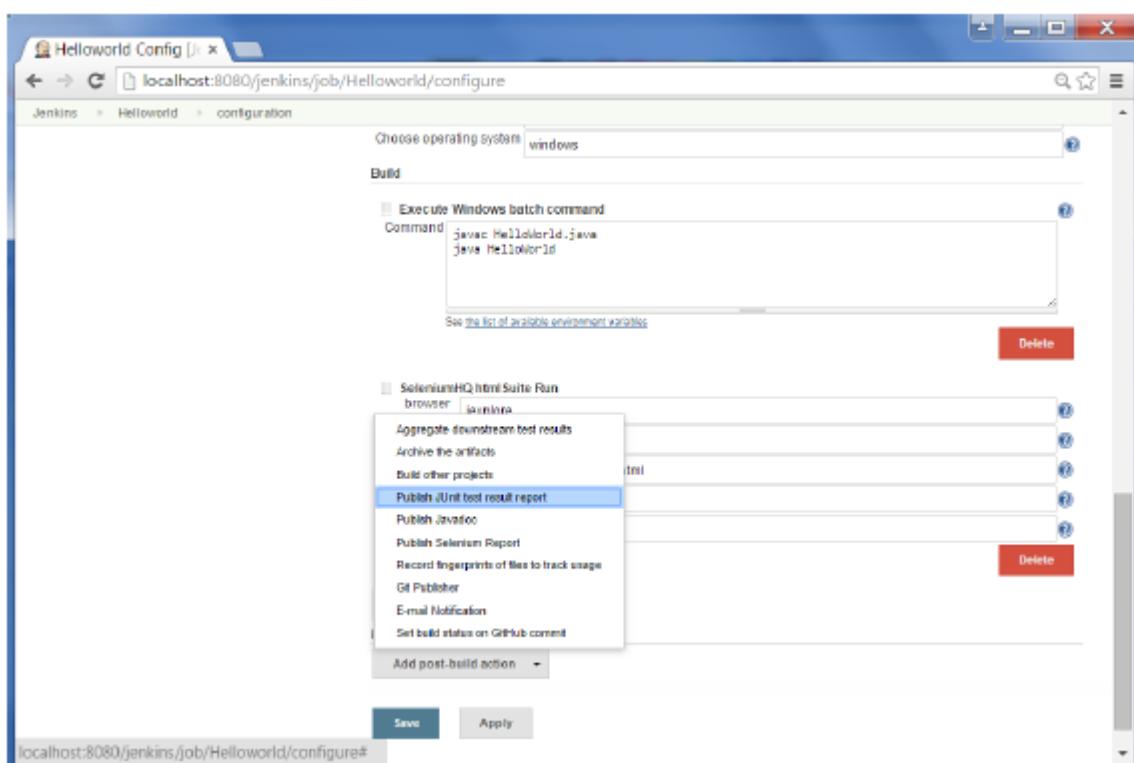
- **"Format"** – This is the notification payload format which can either be JSON or XML.
- **"Protocol"** – protocol to use for sending notification messages, HTTP, TCP or UDP.

- **"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. On the left, there's a sidebar with links like Home, Mailing Lists, Source code, Bugtracker, Security, Advisories, Events, Donation, Commercial Support, and Wiki Site Map. Under Documents, there are links to Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, Servlet Container Notes, and a note about waiting for wiki.jenkins-ci.org.

The main content area is titled "Static Code Analysis Plug-ins" and shows the "analysis-core" plugin. It includes a "Plugin Information" table with columns for Plugin ID, Changes, and Usage. The "Changes" section shows the latest release (1.74) was on Sep 07, 2015, and the source code is available on GitHub. The "Usage" section contains a line graph titled "analysis-core - installations" showing a steady increase from approximately 25,000 in October 2014 to over 30,000 in September 2015. The "Installations" section lists dates from 2014-Oct to 2015-Sep.

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

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

The plugins can provide information such as

- The total number of warnings in a job
- A showing of the new and fixed warnings of a build
- Trend Reports showing the number of warnings per build
- Overview of the found warnings per module, package, category, or type
- Detailed reports of the found warnings optionally filtered by severity (or new and fixed)

## Jenkins - Distributed Builds

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

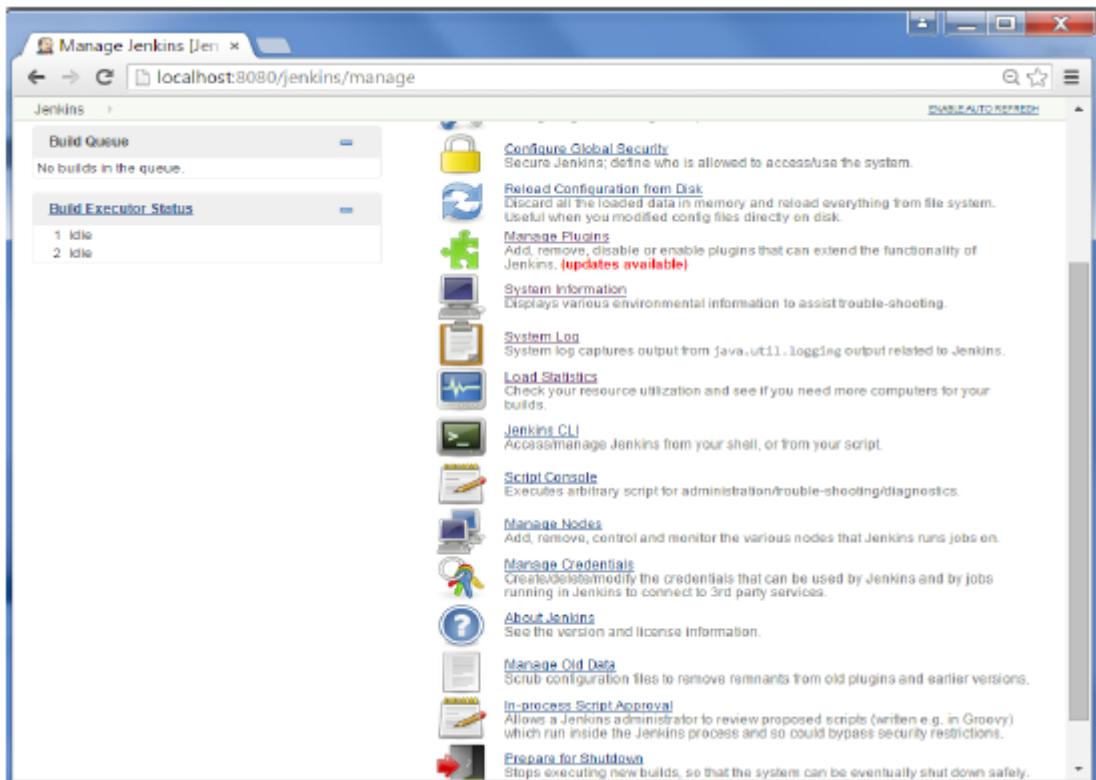
Sometimes you might also need several different environments to test your builds. In this case using a slave to represent each of your required environments is almost a must.

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

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

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

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



**Step 2** – Click on New Node

The screenshot shows the Jenkins 'Nodes' page at [localhost:8080/jenkins/computer/](http://localhost:8080/jenkins/computer/). The page title is 'Nodes [Jenkins]'. On the left sidebar, there are links: 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. The main content area displays a table of nodes:

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free
	master	Windows 7 (x86)	In sync	229.89 GB	12.13 GB	
		Data obtained	3 min 11 sec	3 min 12 sec	3 min 12 sec	3 min 12 sec

A 'Refresh status' button is located at the bottom right of the table. Below the table, there are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 12, 2015 9:22:44 PM', 'REST API', and 'Jenkins ver. 1.806.2'.

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

The screenshot shows the Jenkins 'New Node' configuration page at [localhost:8080/jenkins/computer/new](http://localhost:8080/jenkins/computer/new). The page title is 'Jenkins'. The left sidebar has the same links as the previous page: 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. The main form is for creating a new node:

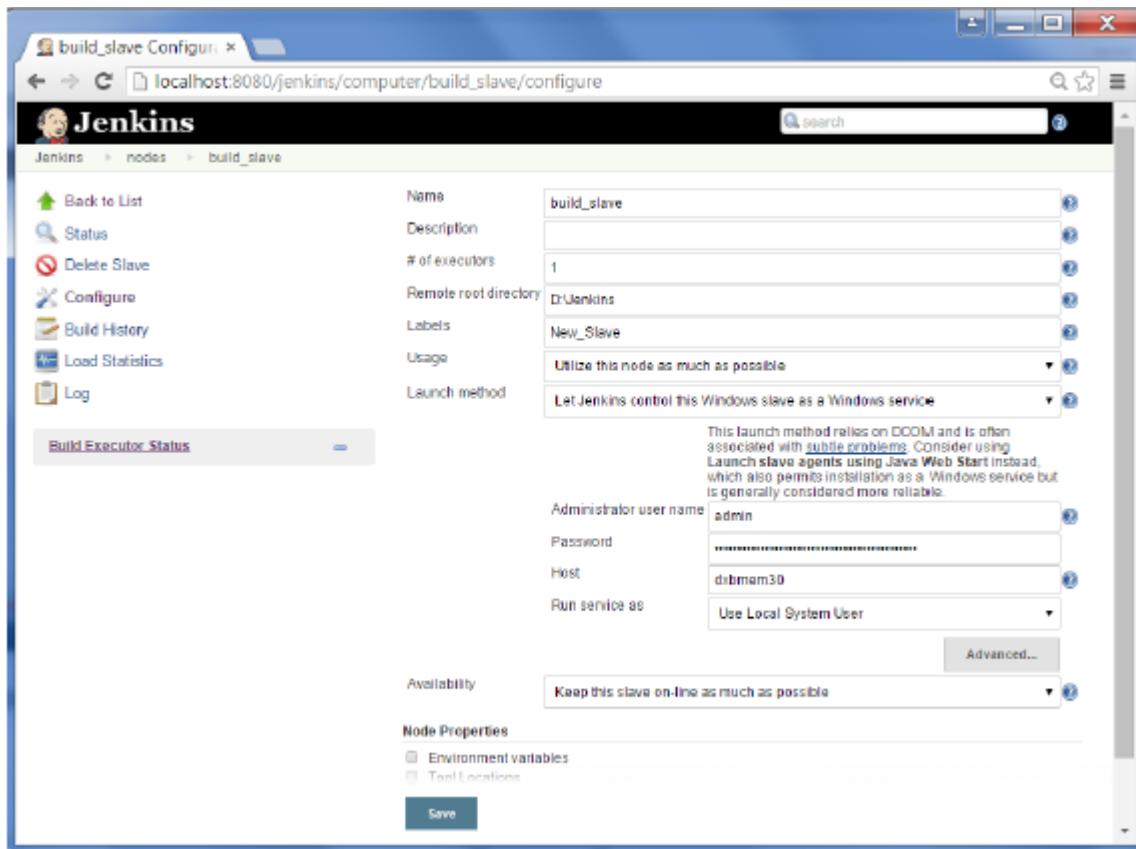
Node name:

Dumb Slave  
Adds a plain, dumb slave to Jenkins. This is called "dumb" because Jenkins doesn't provide higher level of integration with these slaves, such as dynamic provisioning. Select this type if no other slave types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

OK

Below the form, there are collapsed sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 12, 2015 9:25:11 PM', 'REST API', and 'Jenkins ver. 1.806.2'.

**Step 4** – Enter the details of the node slave machine. In the below example, we are considering the slave machine to be a windows machine, hence the option of “Let Jenkins control this Windows slave as a Windows service” was chosen as the launch method. We also need to add the necessary details of the slave node such as the node name and the login credentials for the node machine. Click the Save button. The Labels for which the name is entered as “New\_Slave” is what can be used to configure jobs to use this slave machine.



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

The screenshot shows the Jenkins 'Nodes' page. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. Below this is a search bar and a 'REFRESH AUTO REFRESH' button. The main content area has a table titled 'Nodes' with columns for Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, and Free Temp Space. It lists two nodes: 'build\_slave' (Windows 7 (x86), status N/A) and 'master' (Windows 7 (x86), status In sync). Below the table are sections for 'Build Queue' (empty) and 'Build Executor Status' (showing 1 idle and 2 idle executors for 'master' and 1 offline executor for 'build\_slave'). At the bottom, there are links for 'Help us localize this page', 'Page generated: Oct 12, 2016 9:31:43 PM', 'REST API', and 'Jenkins ver. 1.600.3'.

	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space
1	build_slave		N/A	N/A	N/A	
2	master	Windows 7 (x86)	In sync	229.89 GB	12.13 GB	229.89 GB

Build Queue: No builds in the queue. Refresh status.

Build Executor Status:

- master: 1 Idle, 2 Idle
- build\_slave: (offline)

## Jenkins - Automated Deployment

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

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

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [jenkins]". The address bar shows "localhost:8080/jenkins/pluginManager/available". The main content area is titled "Install 1" and lists several Jenkins plugins. One plugin, "Deploy to container Plugin", is highlighted with a blue selection bar at the bottom of its row. Below the table are three buttons: "Install without restart", "Download now and install after restart", and "Update information".

Install 1	Name	Version
<input type="checkbox"/>	<a href="#">Artifact Deployer Plug-in</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 AWS Elastic Beanstalk 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 5.6. 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 Deploy Plugin. 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

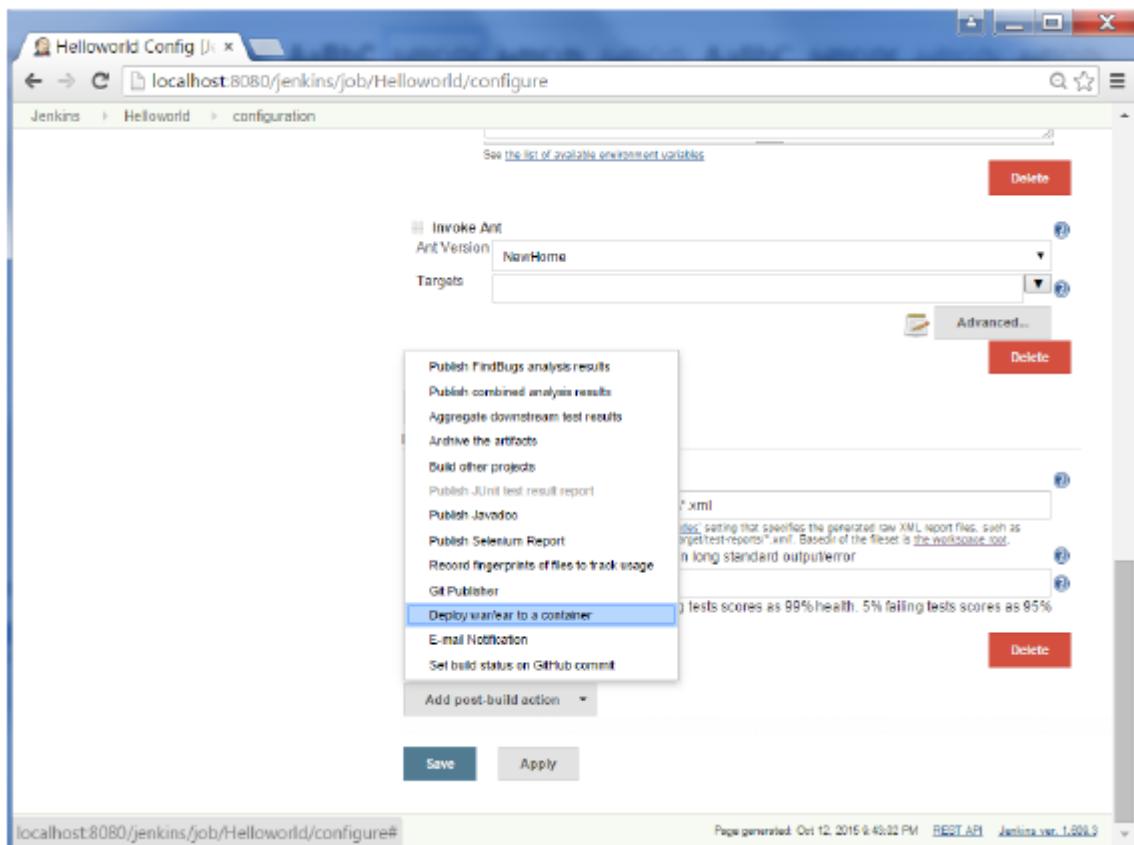
This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build.

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

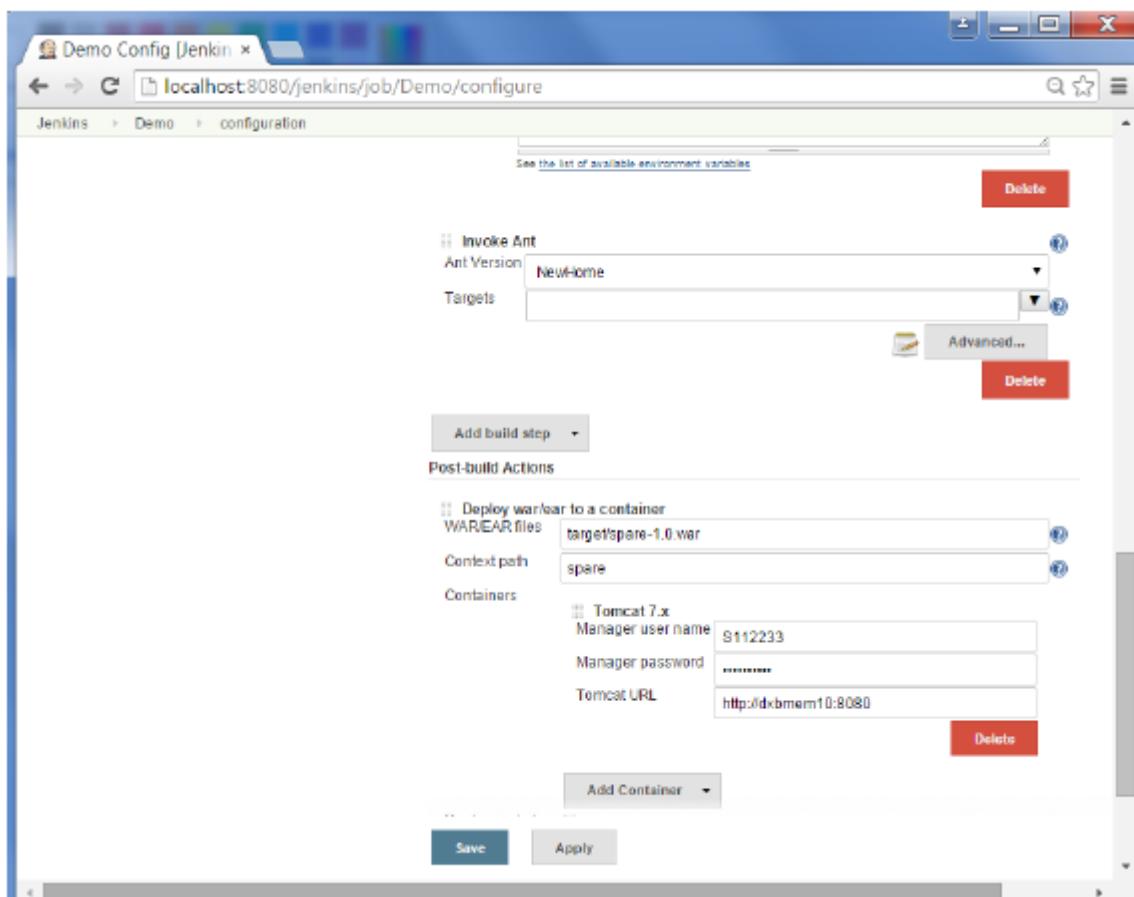
JBoss 3.x/4.x

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.



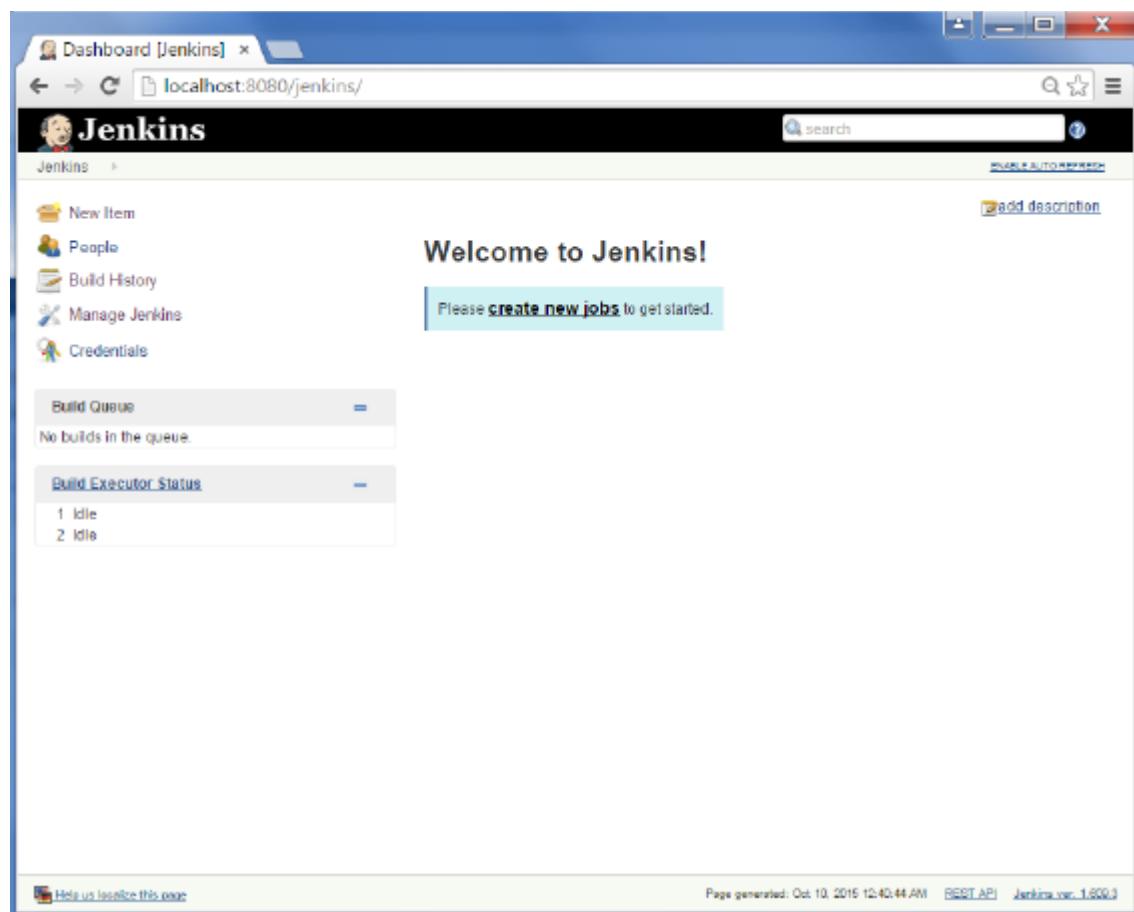
# Jenkins - Metrics & Trends

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.

Manage Jenkins [jen] ×

localhost:8080/jenkins/manage

# Jenkins

search

New Item

People

Build History

Manage Jenkins

Credentials

Build Queue

No builds in the queue.

Build Executor Status

1 Idle  
2 Idle

## Manage Jenkins

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 i18n](#) for more details.

Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse.

[Setup Security](#) [Dismiss](#)

 [Configure System](#)  
Configure global settings and paths.

 [Configure Global Security](#)  
Secure Jenkins; define who is allowed to access/use the system.

 [Reload Configuration from Disk](#)  
Discard all the loaded data in memory and reload everything from file system.  
Useful when you modified config files directly on disk.

 [Manage Plugins](#)  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins. [\(updates available\)](#)

 [System Information](#)  
Displays various environmental information to assist trouble-shooting.

 [System Log](#)  
System log captures output from java.util.logging output related to Jenkins.

 [Load Statistics](#)  
Check your resource utilization and see if you need more computers for your builds.

 [Jenkins CLI](#)  
Access/manage Jenkins from your shell, or from your script.

 [Script Console](#)  
Executes arbitrary script for administration/trouble-shooting/diagnostics.

 [Manage Nodes](#)  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

 [Manage Credentials](#)  
Create/delete/modify the credentials that can be used by Jenkins and by jobs running in Jenkins to connect to 3rd party services.

 [About Jenkins](#)  
Our full copyright and license information.

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

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Jenkins]". 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 columns are "Install", "Name", and "Version". Below the table are two buttons: "Install without restart" and "Download now and install after restart". A status message says "Update information obtained: 2 mi".

**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 page. The title bar says "Update Center [Jenkins]". The address bar shows "localhost:8080/jenkins/updateCenter/". The main content area has a heading "Installing Plugins/Upgrades". It shows the "Build History Metrics plugin" was installed successfully. There are two green checkmark icons with instructions: "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".

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

The screenshot shows the Jenkins Project Helloworld dashboard. On the left, there is a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. Below this is a table titled 'Build History' showing builds from Oct 24 to Oct 11. To the right of the sidebar are three tables: MTTR, MTTF, and Standard Deviation, each with rows for Last 7 Days, Last 30 Days, and All Time. At the bottom, there is a section titled 'Permalinks' with a list of links to specific builds.

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

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

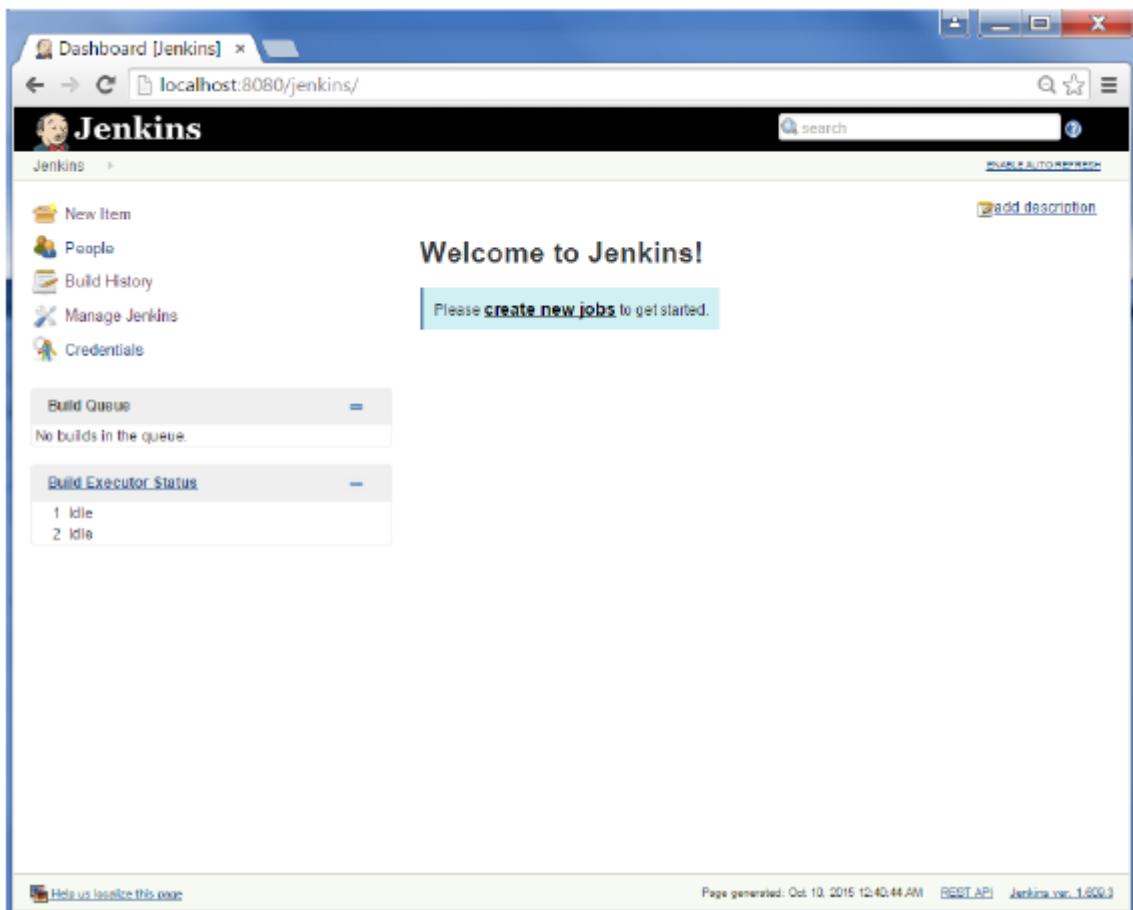
	Last 7 Days	Last 30 Days	All Time
<b>MTTR</b>	0 ms	23 hr	23 hr
<b>MTTF</b>	0 ms	2 days 4 hr	2 days 4 hr
<b>Standard Deviation</b>	0 ms	52 sec	52 sec

**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

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below these are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 Idle). The main area is titled 'Manage Jenkins' and contains a list of available plugins:

- Configure System
- Configure Global Security
- Reload Configuration from Disk
- Manage Plugins
- System Information
- System Log
- Load Statistics
- Jenkins CLI
- Script Console
- Manage Nodes
- Manage Credentials
- About Jenkins

At the top right, there are two buttons: 'Setup Security' and 'Dismiss'. 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](#) for more details.' Another warning below it says: 'Unsecured Jenkins allows anyone on the network to launch processes on your behalf. Consider at least enabling authentication to discourage misuse.'

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

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Jenkins]". The address bar shows "localhost:8080/jenkins/pluginManager/available". The main header has "Jenkins" and "Plugin Manager". On the left, there are links for "Back to Dashboard" and "Manage Jenkins". The central area has a search bar with "global-build-stats" and tabs for "Updates", "Available", "Installed", and "Advanced". The "Available" tab is selected. A table lists the "Hudson global-build-stats plugin" with version 1.3. The table columns are "Install", "Name", and "Version". Below the table are buttons for "Install without restart", "Download now and install after restart", and "Update information".

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

The screenshot shows the Jenkins Update Center interface. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Manage Jenkins', and 'Manage Plugins'. The main title is 'Installing Plugins/Upgrades'. Below the title, under 'Preparation', there's a bulleted list: 'Checking internet connectivity', 'Checking update center connectivity', and 'Success'. A message indicates that the 'Hudson global-build-stats plugin' has been installed successfully. There are two green checkmark icons with associated instructions: one pointing to 'Go back to the top page (you can start using the installed plugins right away)' and another pointing to 'Restart Jenkins when installation is complete and no jobs are running'. At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 4:01:04 PM', 'REST API', and 'Jenkins ver. 1.809.3'.

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

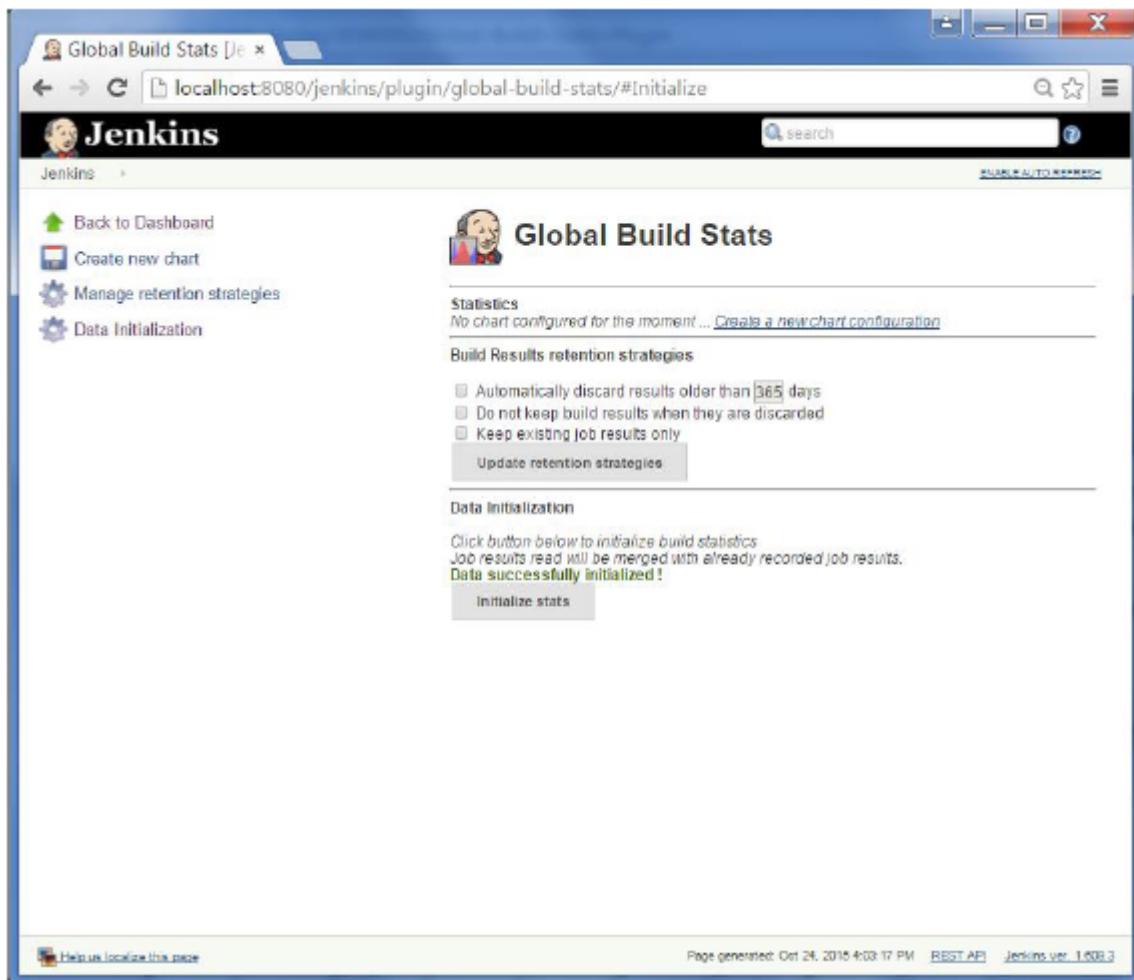
**Step 5** – Go to the Jenkins dashboard and click on Manage Jenkins. In the Manage Jenkins screen, scroll down and now you will now see an option called ‘Global Build Stats’. Click on this link.



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

The screenshot shows the Jenkins Global Build Stats plugin page. At the top left is the Jenkins logo and the title "Global Build Stats". The address bar shows the URL "localhost:8080/jenkins/plugin/global-build-stats/". On the left sidebar, there are links: "Back to Dashboard", "Create new chart", "Manage retention strategies", and "Data Initialization". The main content area has a section titled "Global Build Stats" with a small icon of a person. Below it is a "Statistics" section with the message "No chart configured for the moment ... [Create a new chart configuration](#)". Underneath is a "Build Results retention strategies" section with three checkboxes: "Automatically discard results older than 365 days" (unchecked), "Do not keep build results when they are discarded" (unchecked), and "Keep existing job results only" (unchecked). A "Update retention strategies" button is below these checkboxes. Another section titled "Data Initialization" contains the instruction "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 are links for "Help us localize this page", "Page generated: Oct 24, 2015 4:03:17 PM", "REST API", and "Jenkins ver. 1.80.3".

**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](#)

Build Results retention strategies

**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 page. At the top, there are navigation links: 'Back to Dashboard' and 'Back to Global Build Stats'. Below that is the title 'Global Build Search'. There are two tabs: 'Build History' (selected) and 'Build Log'. A 'Search' input field is present. Underneath, there are several filter options: 'Job Filtering' (radio buttons for 'All Jobs', 'Job name regex' with a text input), 'Node Filtering' (radio buttons for 'All Nodes', 'Master Only', 'Node name regex' with a text input), 'Launcher Filtering' (radio buttons for 'All Launchers', 'System only', 'Username regex' with a text input), and 'Statuses taken into account' (checkboxes for 'Success', 'Failure', 'Unstable', 'Aborted', and 'Not Started'). A 'Search' button is located below the filters. The main area is titled 'Search results' and contains a table with three rows of build data:

Status	Job name	#	Date	Duration	Node name	Launched by
Failure	Hellosword	13	Oct 11, 2015 11:04:57 PM	5.6 sec	master	SYSTEM
Failure	Hellosword	12	Oct 11, 2015 18:51:37 PM	4.6 sec	master	SYSTEM
Failure	Hellosword	11	Oct 11, 2015 18:48:11 PM	4.2 sec	master	SYSTEM

At the bottom right of the page, there is a footer note: 'Page generated: Oct 24, 2015 4:30:40 PM - JENKINS API - Jenkins ver. 1.603.3'.

## 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.

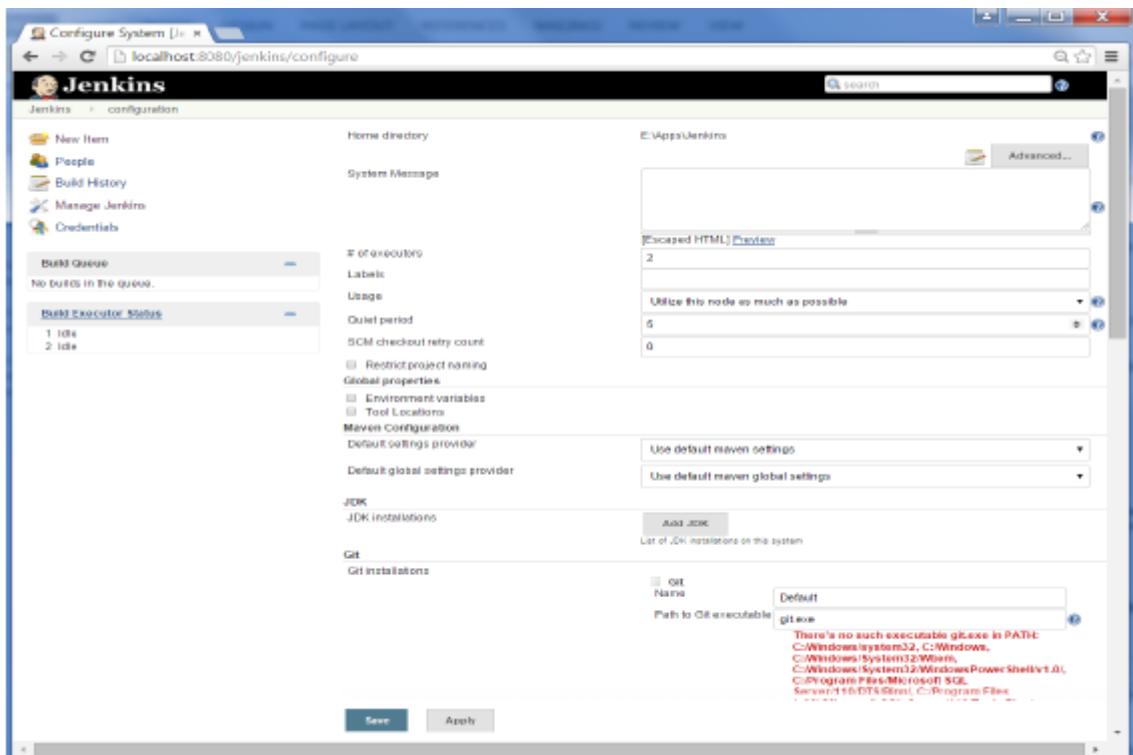
**http://localhost:8080/jenkins/exit** – shutdown jenkins

**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.

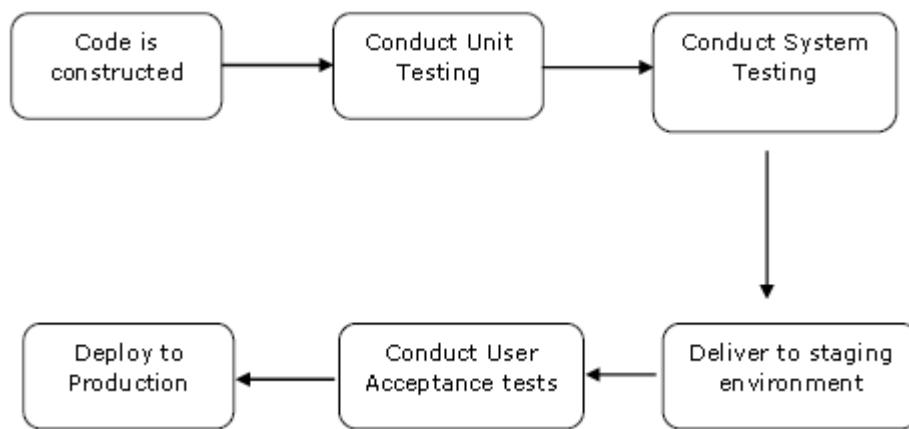


Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined and doing continuous builds, always ensure that Jenkins is setup on a drive that has enough hard disk space. If you hard disk runs out of space, then all builds on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

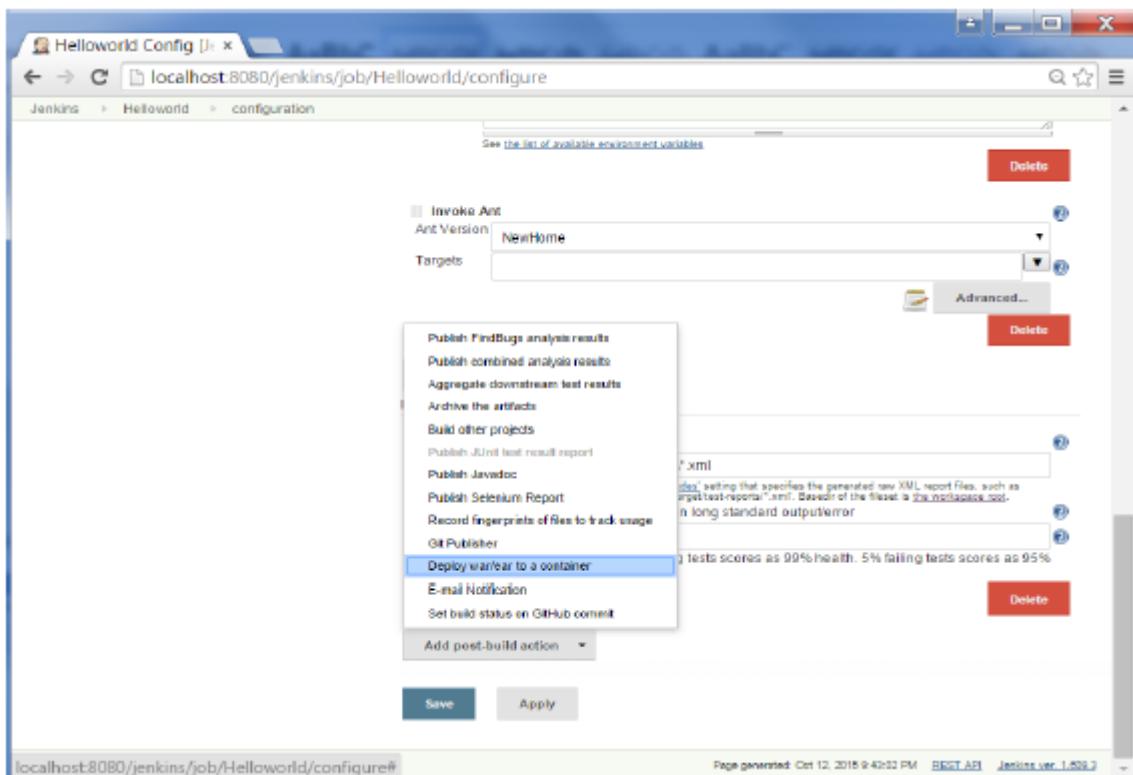
Another best practice is to write cron jobs or maintenance tasks that can carry out clean-up operations to avoid the disk where Jenkins is setup from becoming full.

## 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.



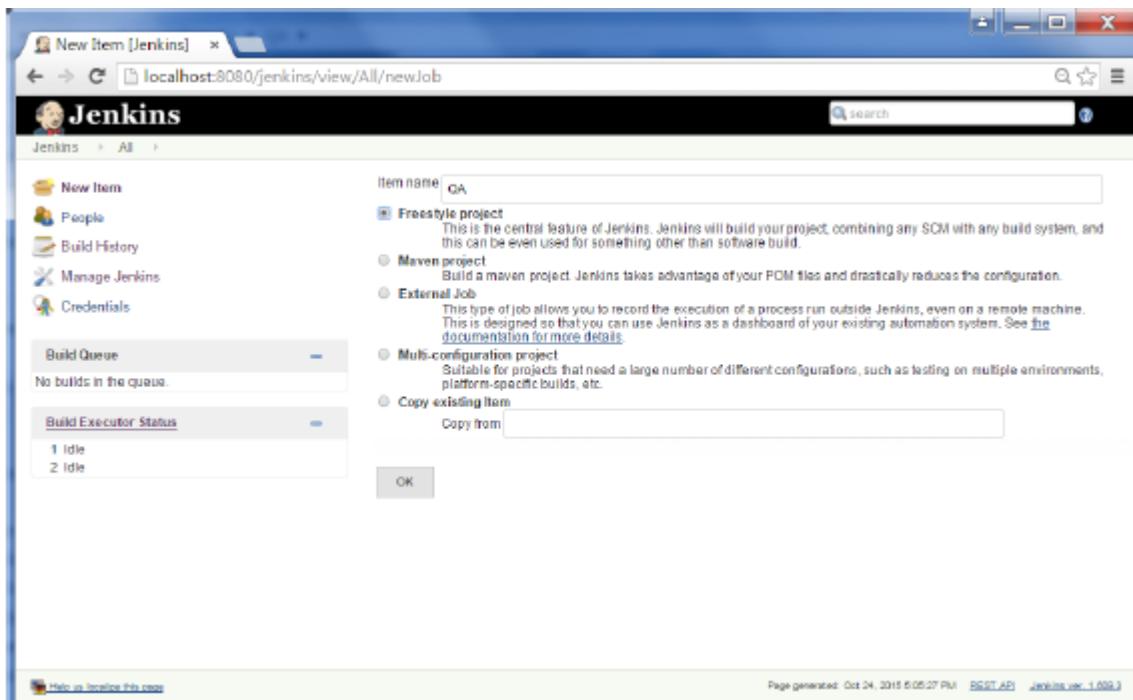
The main part of Continuous deployment is to ensure that the entire process which is shown above is automated. Jenkins achieves all of this via various plugins, one of them being the “Deploy to container Plugin” which was seen in the earlier lessons.



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.

The screenshot shows the Jenkins configuration interface for a job named 'QA'. The 'Build' section is expanded, showing a 'Command' field containing the following Java code:

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

Below the command, there is a link 'See the list of available environment variables' and a red 'Delete' button. At the bottom of the build section, there are 'Add build step' and 'PostBuild Actions' buttons, along with 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins dashboard for the 'QA' project. On the left, there is a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main area is titled 'Project QA' and contains the following sections:

- Workspace:** A link to view workspace details.
- Recent Changes:** A link to view recent changes.
- Build History:** A table showing the last four builds:

Build	Date
Oct 24, 2015 5:29 PM	Oct 24, 2015 5:29 PM
Oct 24, 2015 5:19 PM	Oct 24, 2015 5:19 PM
Oct 24, 2015 5:18 PM	Oct 24, 2015 5:18 PM
Oct 24, 2015 5:17 PM	Oct 24, 2015 5:17 PM

Links for RSS feeds for all and failures are provided.
- MTTR:** A table showing metrics for MTTR:

Timeframe	Mean Time To Recovery (MTTR)
Last 7 Days	2 min 28 sec
Last 30 Days	2 min 28 sec
All Time	2 min 28 sec
- MTTF:** A table showing metrics for MTTF:

Timeframe	Mean Time To Failure (MTTF)
Last 7 Days	0 ms
Last 30 Days	0 ms
All Time	0 ms
- Standard Deviation:** A table showing metrics for Standard Deviation:

Timeframe	Standard Deviation
Last 7 Days	75 ms
Last 30 Days	75 ms
All Time	75 ms

At the bottom, there is a 'Permalinks' section with two links:

- Last build (#4), 6.4 sec ago
- Last stable build (#4), 6.4 sec ago

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

The screenshot shows the Jenkins dashboard at [localhost:8080/jenkins/](http://localhost:8080/jenkins/). On the left sidebar, there are links for New Item, People, Build History, Manage Jenkins, and Credentials. Below these are sections for Build Queue (No builds in the queue) and Build Executor Status (1 idle). The main area displays a table of projects. The 'Helloworld' project is listed with the following details:

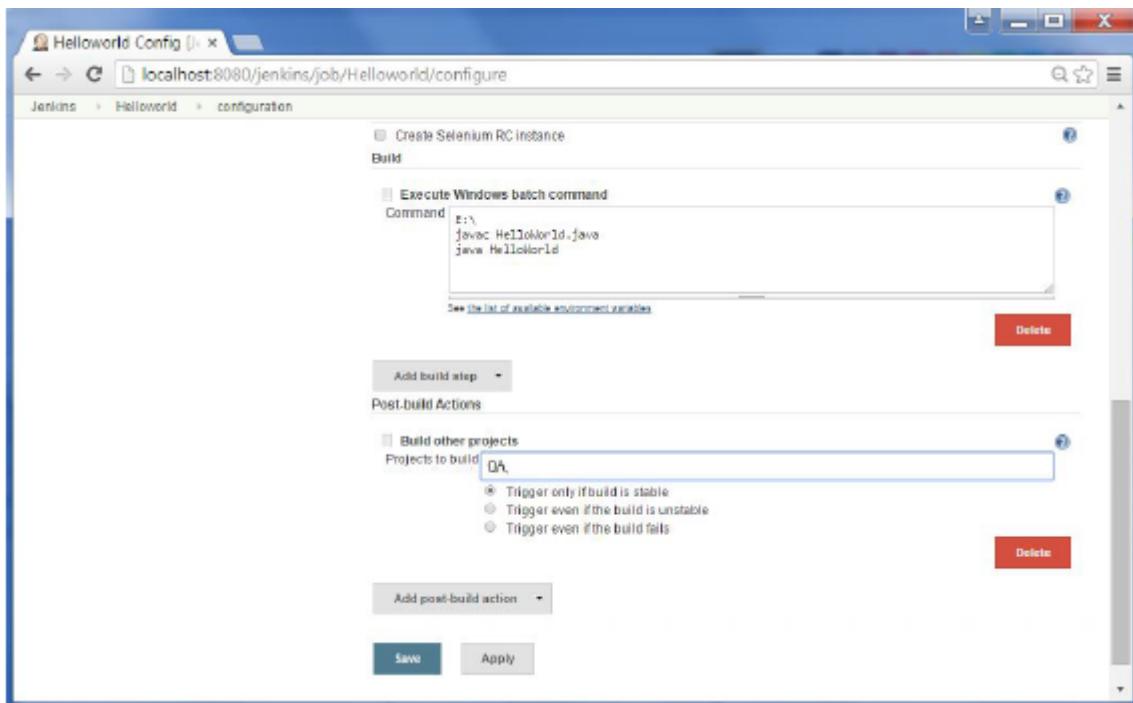
S	W	Name	Last Success	Last Failure	Last Duration
●	●	Helloworld	12 days - #15	#14	6.6 sec
●	●			N/A	N/A

Below the table are links for Changes, Workspace, Build Now, and Delete Project. A 'Configure' button is highlighted with a blue box. At the bottom of the page, the URL [localhost:8080/jenkins/job/Helloworld/configure](http://localhost:8080/jenkins/job/Helloworld/configure) is shown.

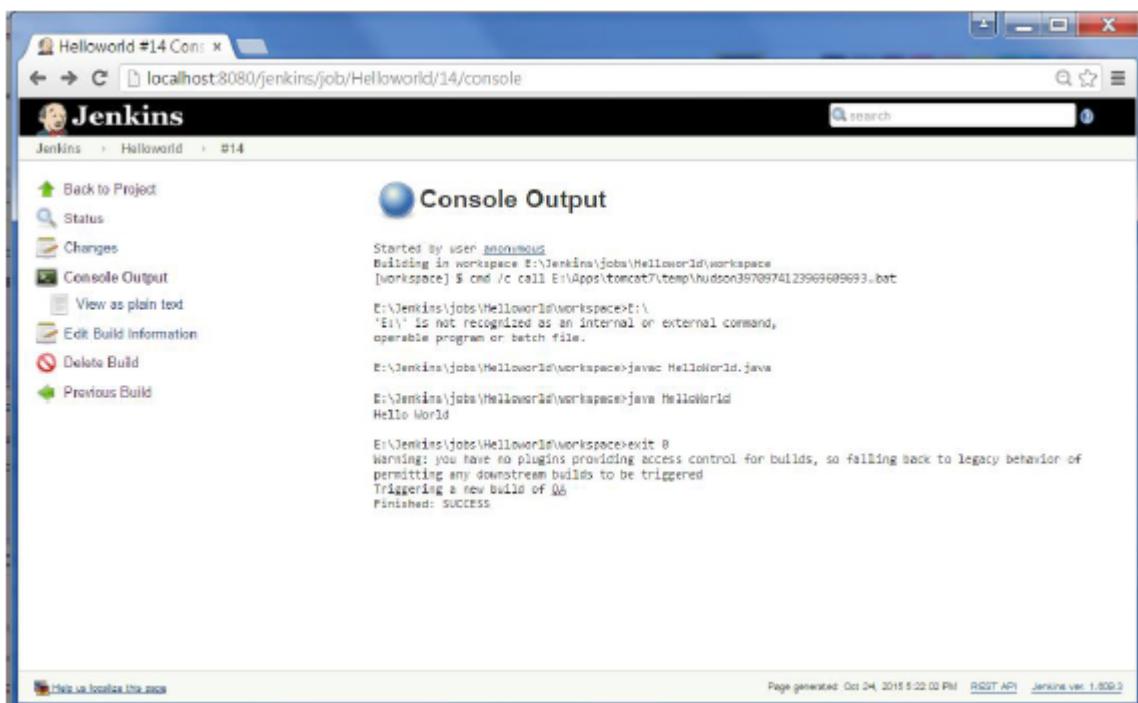
**Step 4** – In the project configuration, choose the ‘Add post-build action’ and choose ‘Build other projects’

The screenshot shows the configuration page for the 'Helloworld' project at [localhost:8080/jenkins/job/Helloworld/configure](http://localhost:8080/jenkins/job/Helloworld/configure). The 'Build other projects' option is highlighted with a blue box in the list of post-build actions. The configuration page includes tabs for 'Jenkins', 'Helloworld', and 'configuration'. At the bottom are 'Save' and 'Apply' buttons, and a footer with links for Help, Page generated: Oct 25, 2015 9:58:29 AM, REST API, and Jenkins ver. 1.609.3.

**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. At the top, there's a search bar and a list of available plugins. The list includes:

- ontrack Jenkins plug-in
- Fail The Build Plugin
- Runscape plugin
- Build Graph View Plugin
- DeploymentSphere
- CloudBees Docker Hub Notification
- Seed Jenkins plug-in
- Delivery Pipeline Plugin

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

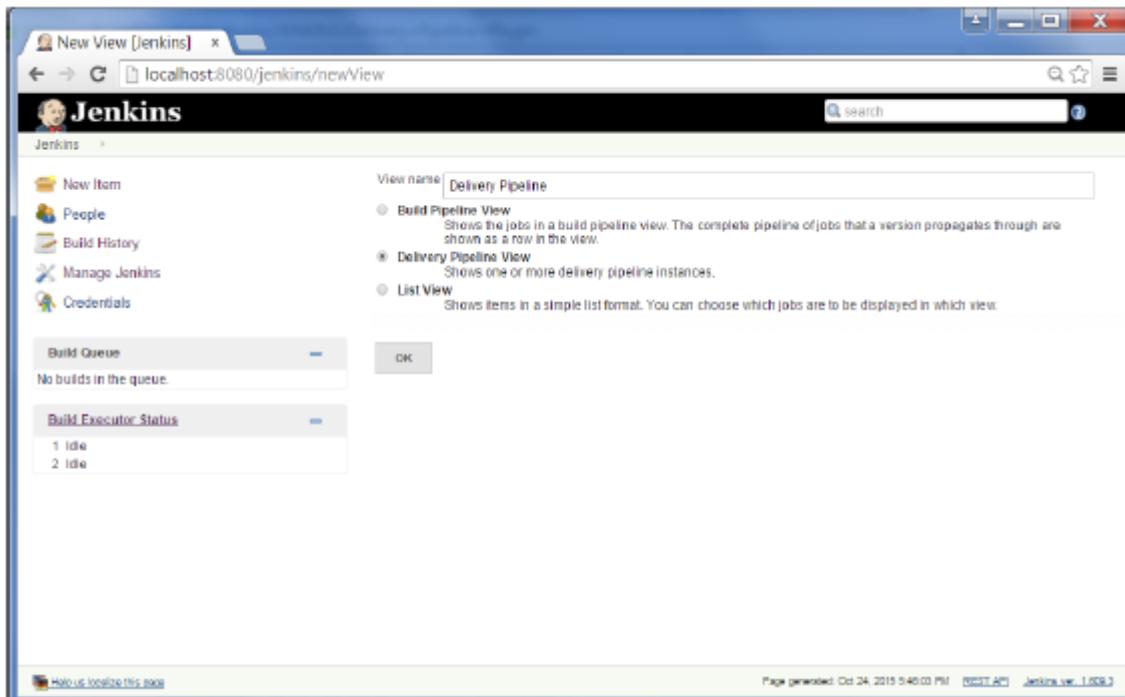
**Step 8** – To see the Delivery pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the ‘All’ Tab.

The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with links like New Item, People, Build History, Manage Jenkins, and Credentials. The main area has tabs for All, Pipelines, and Delivery Pipeline View. The Pipelines tab is selected, showing a table with two items:

S	W	Name	Last Success	Last Failure	Last Duration
1	1	Helloworld	25 min - E14	1 hr 40 min - E12	1.4 sec
2	2	QA	25 min - E5	28 min - E2	1.4 sec

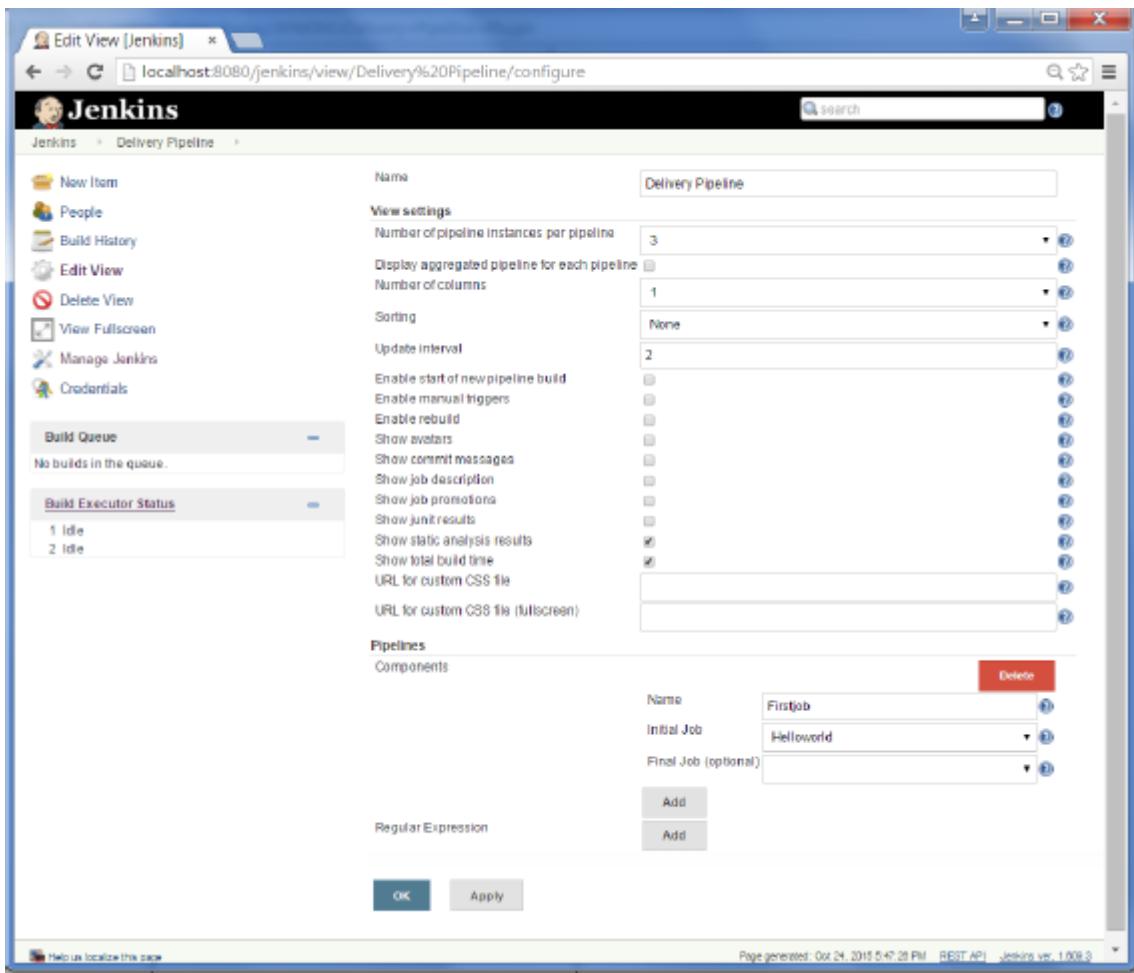
Below the table, there are links for Legend, RSS for all, RSS for failures, and RSS for just latest builds. The bottom of the page shows a footer with "Page generated: Oct 24, 2015 5:48:52 PM" and "Jenkins ver. 1.609.3".

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

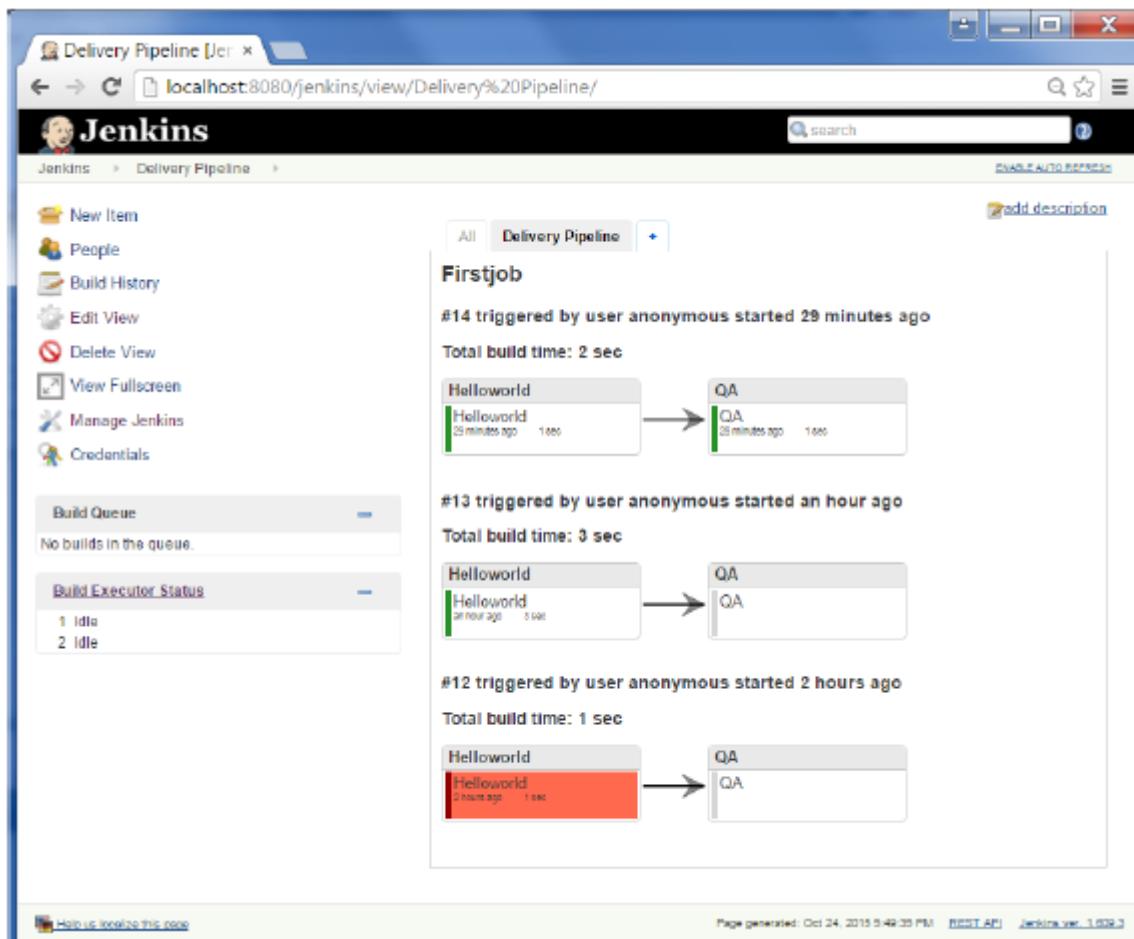


**Step 10** – In the next screen, you can leave the default options. One can change the following settings –

- 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 Plugins. In the available tab, search for ‘Build Pipeline Plugin’. Click On Install without Restart. Once done, restart the Jenkins instance.

The screenshot shows the Jenkins Update Center interface at [localhost:8080/jenkins/pluginManager/available](http://localhost:8080/jenkins/pluginManager/available). The 'Available' tab is selected. A search bar at the top right contains the text 'Build pipeline'. The main area displays a list of plugins:

Install	Name	Version
<a href="#">Build Pipeline Plugin</a>	This plugin provides a _Build Pipeline View_ of upstream and downstream connected jobs that typically form a build pipeline.	1.4.8
<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
<a href="#">Runscope plugin</a>	This plugin allows you to add a <a href="#">Runscope API</a> test as a build step into your Jenkins build pipeline. The plugin will trigger a specific API test (via <a href="#">Trigger URL</a> ) and wait for the test to complete. If the test passes, the build steps will continue. However, if it fails, the build will be marked as failed.	1.44
<a href="#">Build Graph View Plugin</a>	Shows a graph of builds that relates together (aka "build pipeline").	1.1.1
<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

Buttons at the bottom include 'Install without restart', 'Download now and install after restart', and 'Update info'.

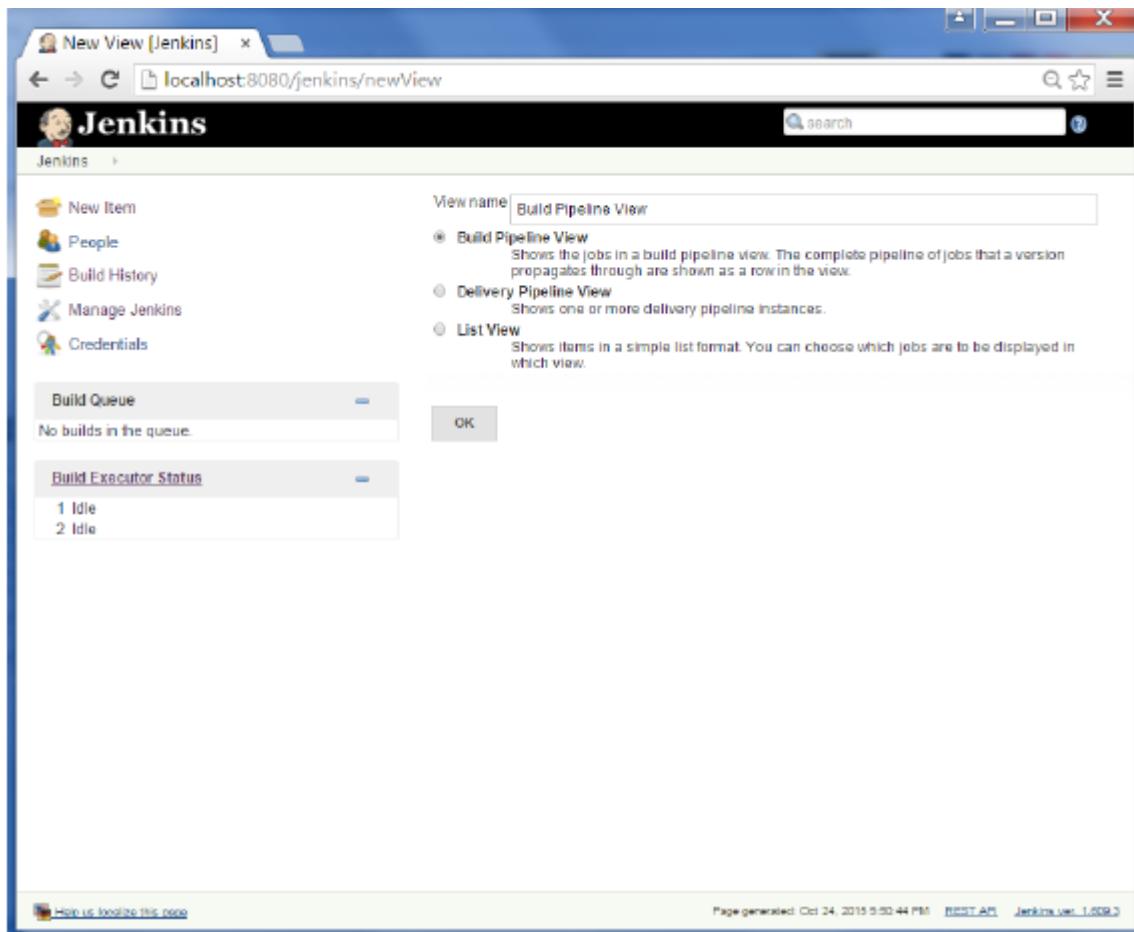
**Step 2** – To see the Build pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the ‘All’ Tab.

The screenshot shows the Jenkins Dashboard at [localhost:8080/jenkins/](http://localhost:8080/jenkins/). On the left, the sidebar includes links like 'New item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. The main area displays a table of build items:

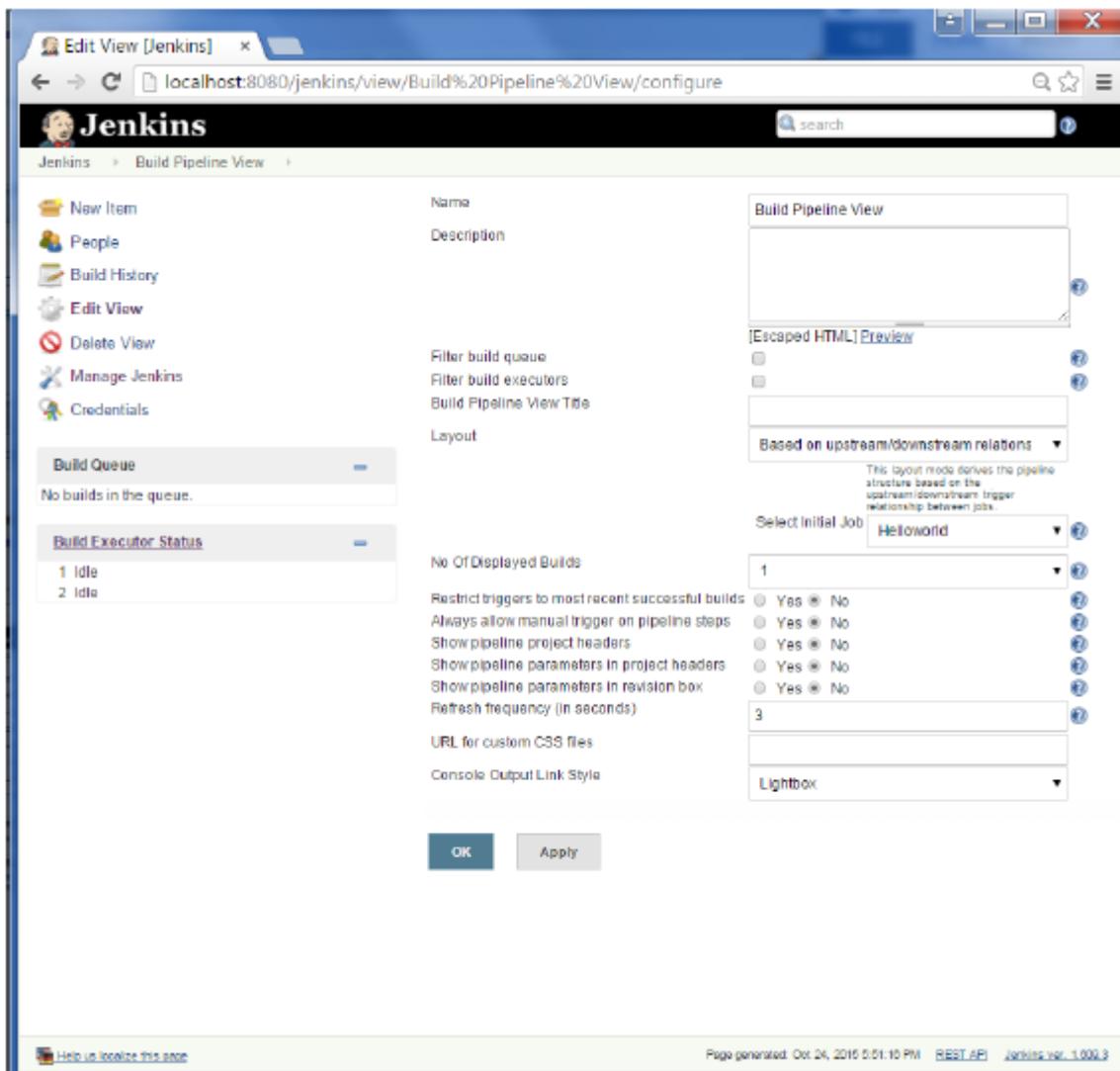
All	W	Name	Last Success	Last Failure	Last Duration
5	W	HelloWorld	25 min - #14	1 hr 49 min - #12	1.4 sec
	W	QA	25 min - #5	28 min - #2	1.4 sec

Below the table, there's a legend for RSS feeds: 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'. At the bottom, it says 'Page generated: Oct 24, 2015 4:48:09 PM REST API Jenkins ver. 1.809.3'.

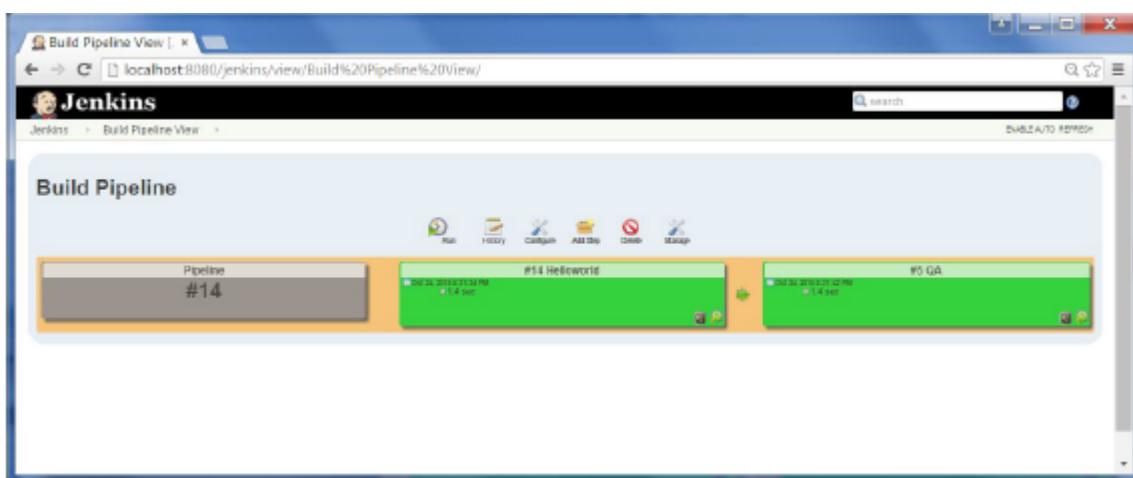
**Step 3** – Enter any name for the View name and choose the option ‘Build Pipeline View’.



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



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



## Jenkins - Managing Plugins

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

The screenshot shows the Jenkins Plugins page at <https://wiki.jenkins-ci.org/display/JENKINS/Plugins>. The left sidebar includes links for Home, Mailing lists, Source code, Bugtracker, Security Advisories, Events, Donation, Commercial Support, and Wiki Site Map. Under the 'Documents' section, there are links for Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, and Servlet Container Notes. The main content area displays a hierarchical list of documentation sections:

- 1 How to install plugins
  - 1.1 Using the interface
    - 1.1.1 Installing the newest version
    - 1.1.2 Installing a specific version
  - 1.2 By hand
- 2 Getting notified of plugin releases
- 3 Developers
- 4 Plugins by topic
  - 4.1 Source code management
  - 4.2 Build triggers
  - 4.3 Build tools
  - 4.4 Build wrappers
  - 4.5 Build notifiers
  - 4.6 Slave launchers and controllers
  - 4.7 Build reports
  - 4.8 Artifact updaters
  - 4.9 Other post-build actions
  - 4.10 External site/tool integrations
  - 4.11 UI plugins
  - 4.12 List View column plugins
  - 4.13 Page decorators
  - 4.14 Authentication and user management
  - 4.15 Cluster management and distributed build
  - 4.16 CLI extensions
  - 4.17 Maven
  - 4.18 Parameters
  - 4.19 iOS development
  - 4.20 .NET development
  - 4.21 Android development
  - 4.22 Ruby development

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

## Uninstalling Plugins

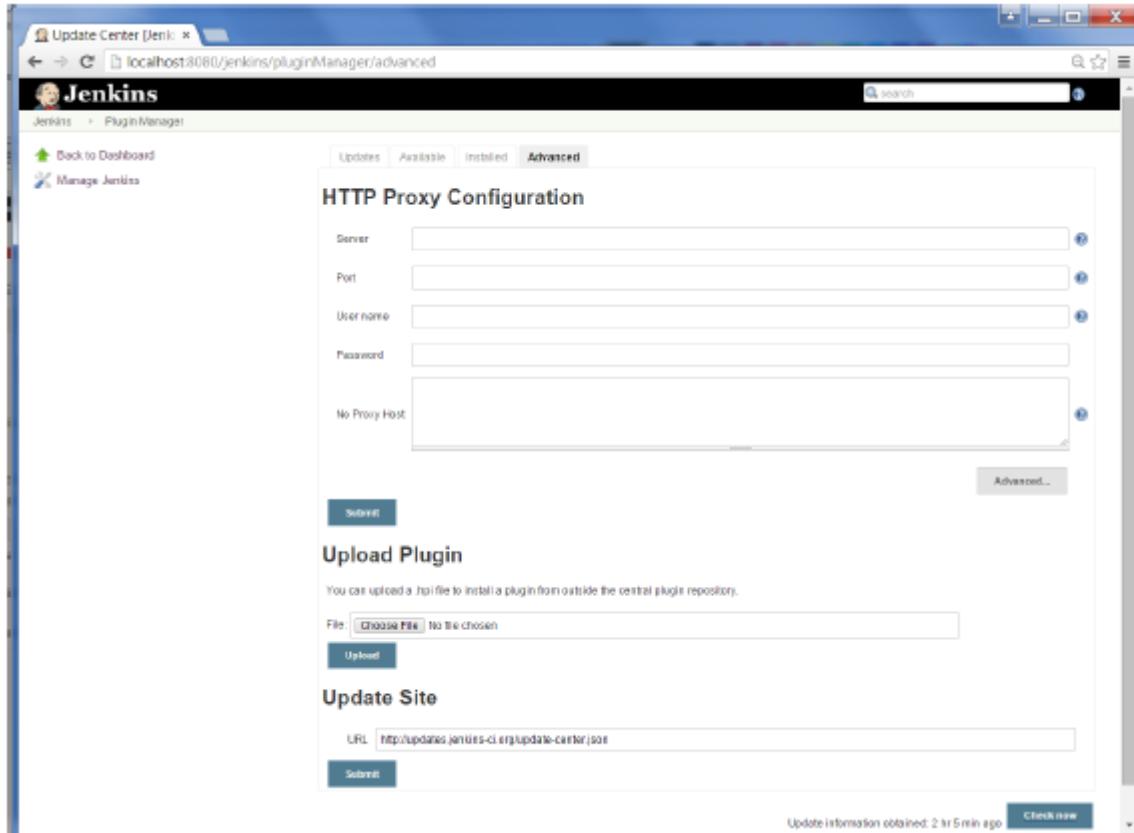
To uninstall a plugin, Go to Manage Jenkins → Manage plugins. Click on the Installed tab. Some of the plugins will have the Uninstall option. You can click these buttons to uninstall the plugins. Ensure to restart your Jenkins instance after the uninstallation.

The screenshot shows the Jenkins Plugin Manager at <http://localhost:8080/jenkins/pluginManager/installed>. The interface includes a sidebar with 'Manage Jenkins' and 'Plugin Manager'. The main area has tabs for 'Updates', 'Available', 'Installed' (which is selected), and 'Advanced'. The 'Installed' tab displays a table of currently enabled plugins:

Enabled	Name	Version	Previously Installed Version	Pinned	Uninstall
<input checked="" type="checkbox"/>	Ant Plugin This plugin adds Apache Ant support to Jenkins.	1.2			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	Build History Matrix Plugin This plugin is used to provide reliability metrics for a job.	1.2			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	Build Pipeline Plugin This plugin masters upstream and downstream connected jobs that typically run sequentially. It adds the ability to define parallel triggers for jobs that require information prior to execution, e.g. an approval process outside of Jenkins.	1.4.8			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	Credentials Plugin This plugin allows you to store credentials in Jenkins.	1.20	Downgrade to 1.18	<a href="#">Unpin</a>	<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	CVS Plugin Integrates Jenkins with CVS version control system using a modified version of the NetBeans client.	2.15			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	Delivery Pipeline Plugin This plugin visualizes Delivery Pipelines (Jobs with upstream/downstream dependencies).	0.8.7			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	Deploy To Container Plugin This plugin allows you to deploy a war to a container after a successful build. Deploy to a remote deployment.	1.10			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	External Monitor Job Type Plugin Add the ability to monitor the result of externally executed jobs.	1.6			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	Extreme Notification Plugin This plugin is a sample to explain how to write a Jenkins plugin.	1.1			<a href="#">Uninstall</a>
<input checked="" type="checkbox"/>	FindBugs Plug-in This plug-in collects the <a href="#">FindBugs</a> analysis results of the project modules and visualizes the found warnings.	4.62			<a href="#">Uninstall</a>

## Installing another Version of a Plugin

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

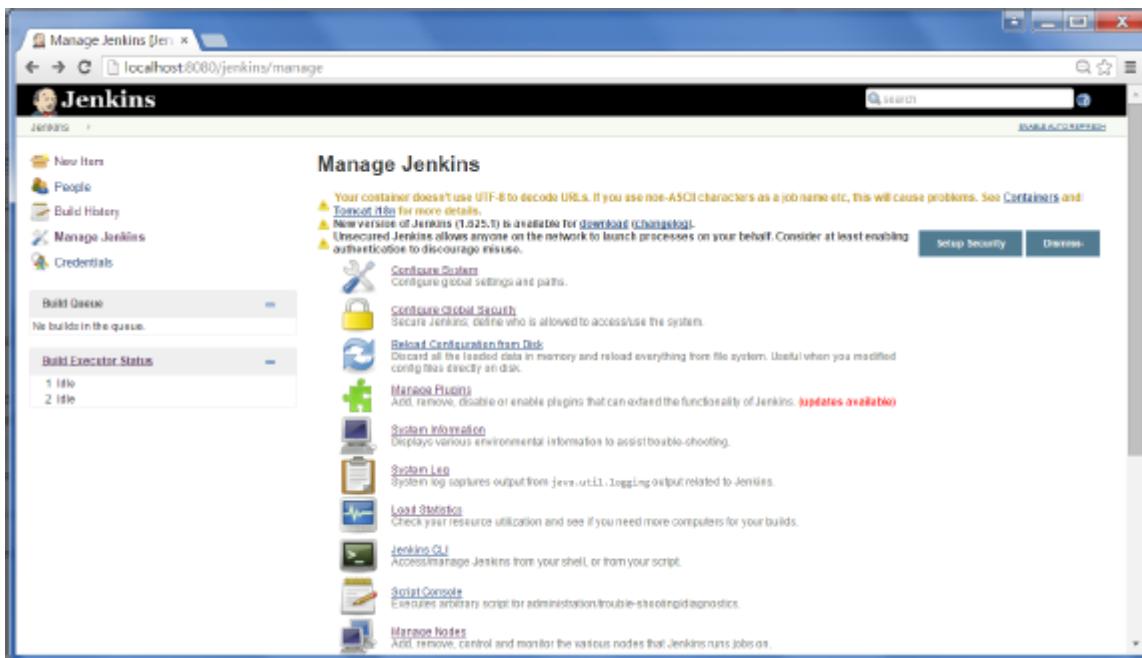


## Jenkins - Security

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

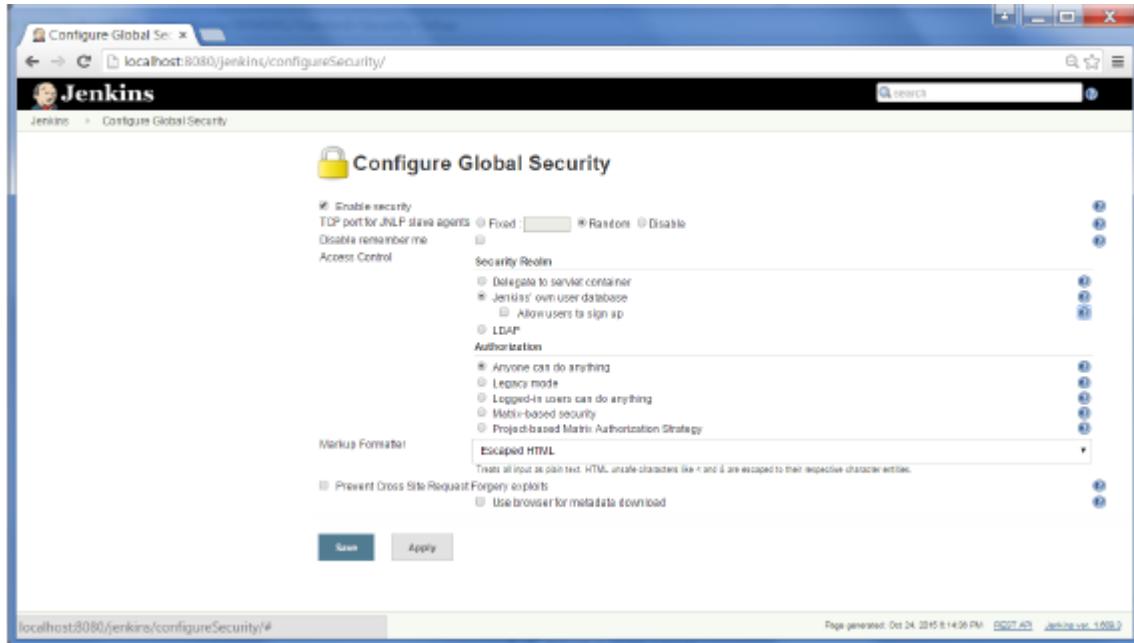
To configure Security in Jenkins, follow the steps given below.

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



**Step 2** – Click on Enable Security option. As an example, let's assume that we want Jenkins to maintain its 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 Jenkins' own user database

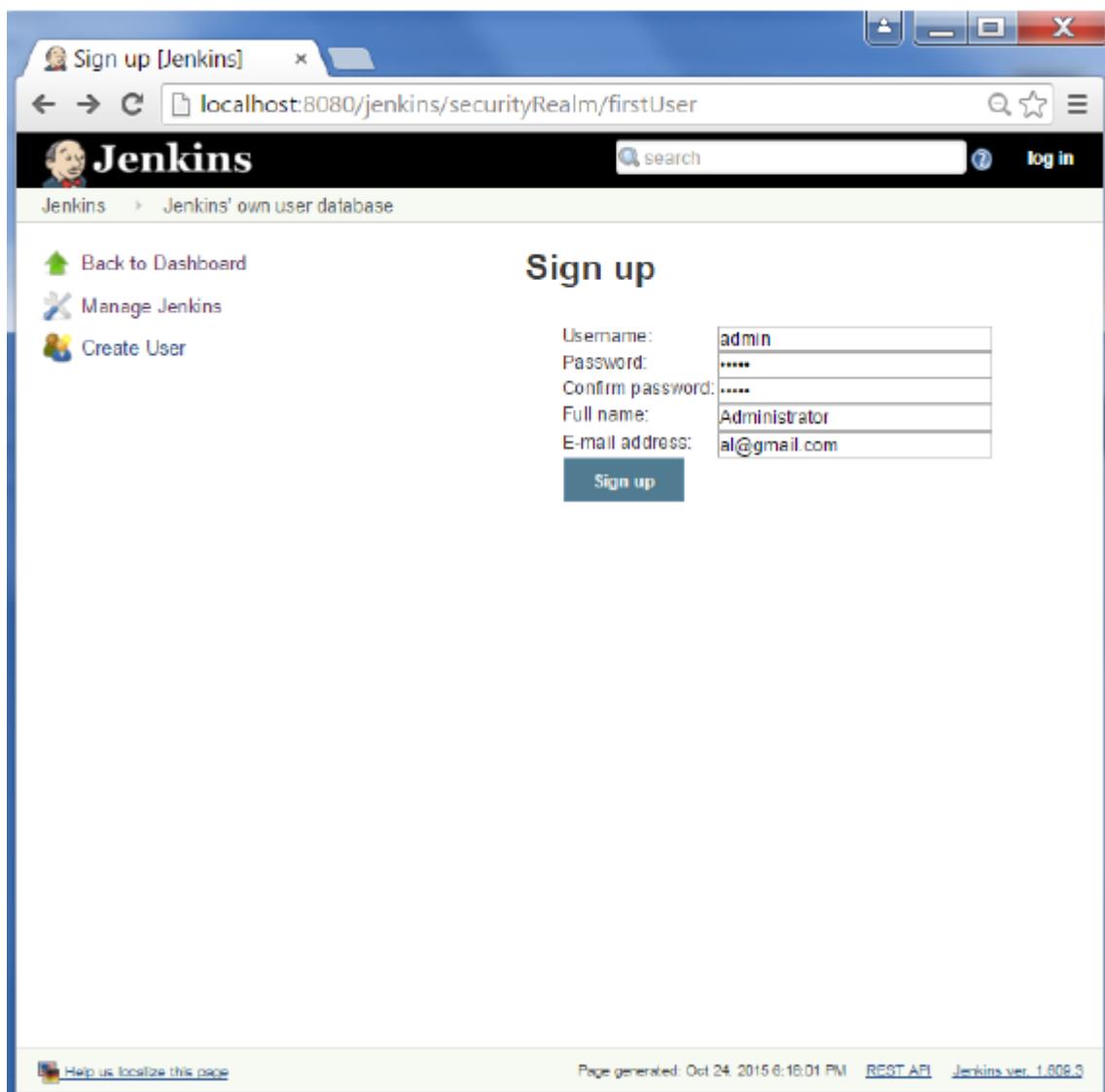
Back to Dashboard Manage Jenkins Create User

## Sign up

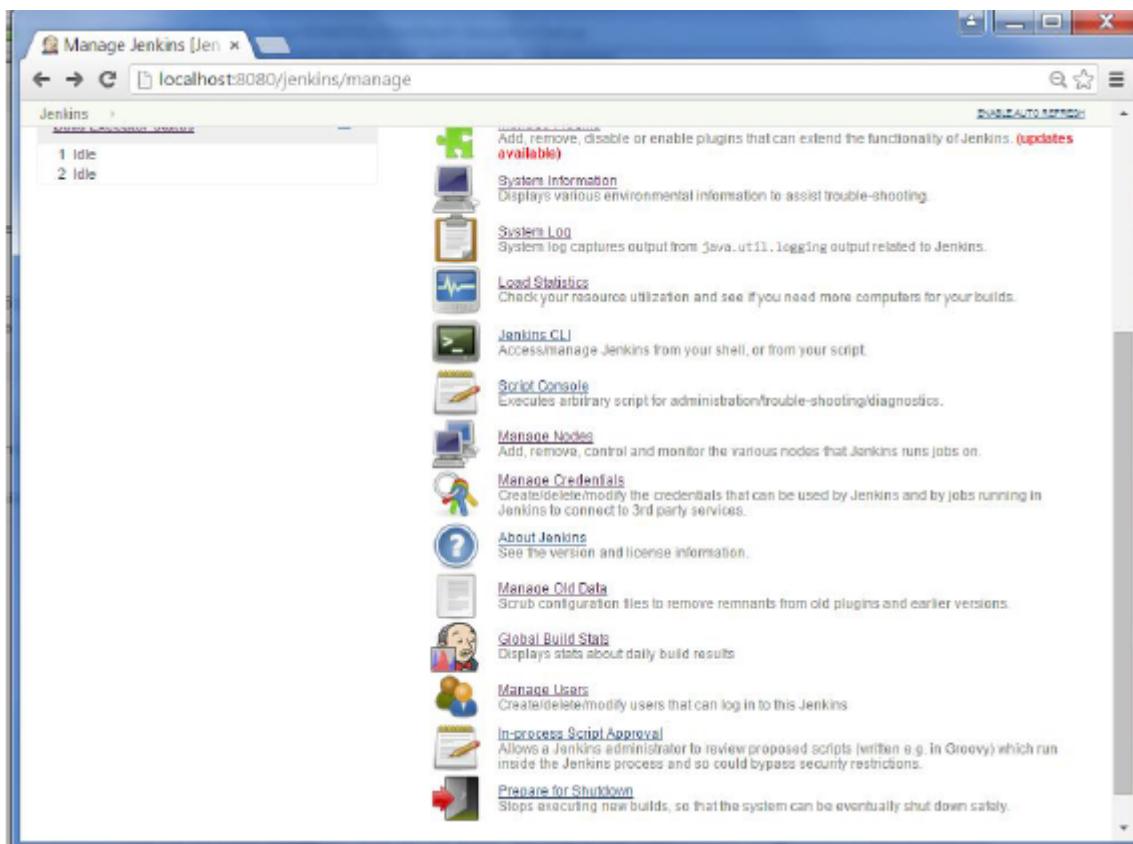
Username:	admin
Password:	.....
Confirm password:	.....
Full name:	Administrator
E-mail address:	al@gmail.com

**Sign up**

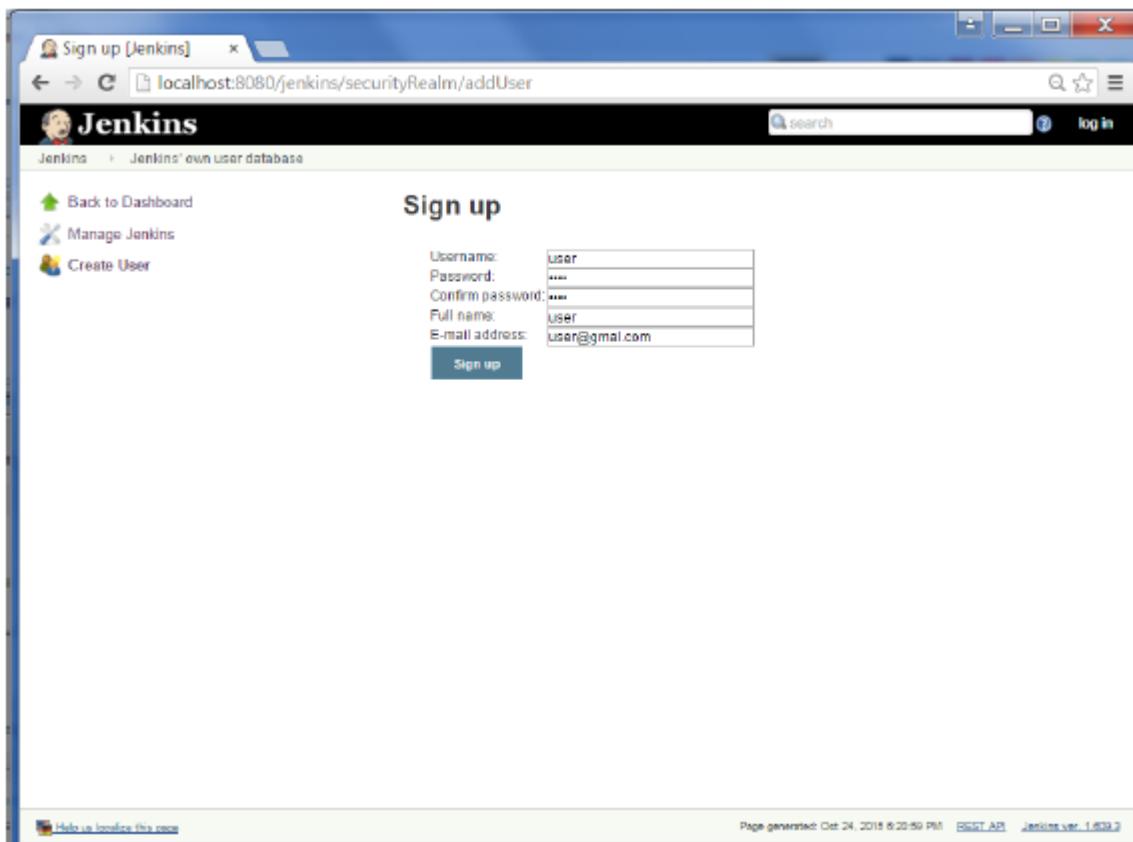
 Page generated: Oct 24, 2015 6:19:01 PM REST API Jenkins ver. 1.608.3



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

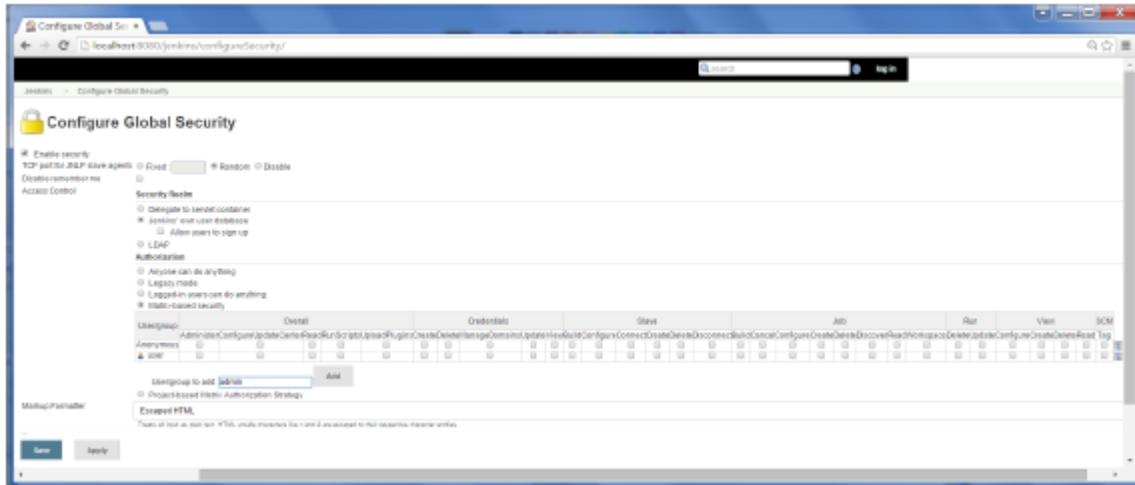


**Step 5** – Just like you defined your admin user, start creating other users for the system. As an example, we are just creating another user called ‘user’.



**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.

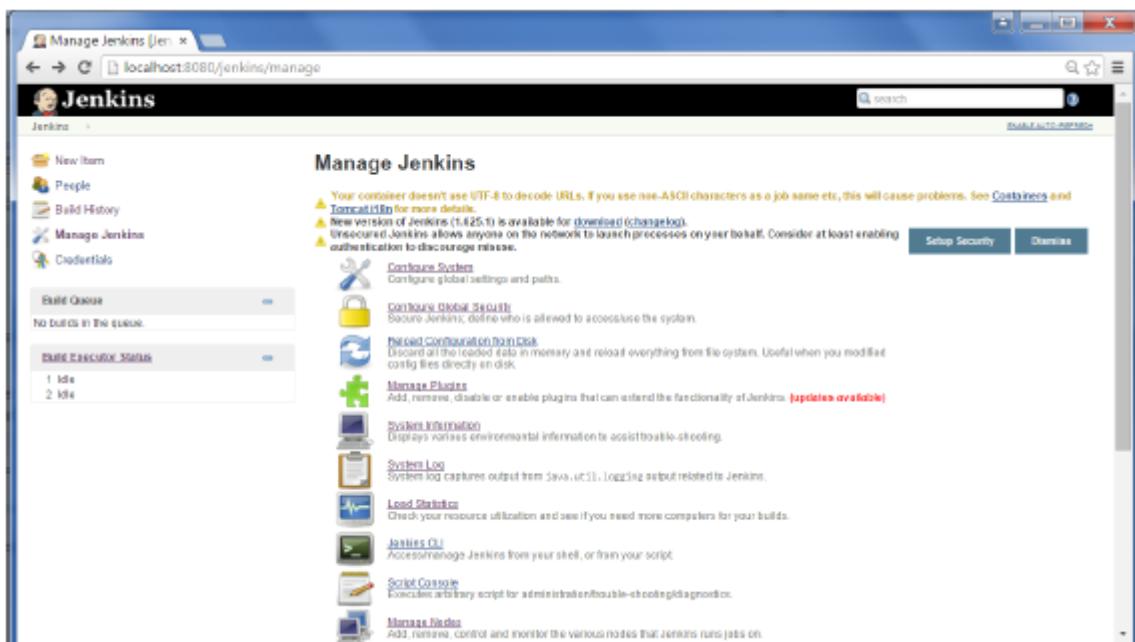
Your Jenkins security is now setup.

**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 used to backup critical configuration settings related to Jenkins. Follow the steps given below to have a backup in place.

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



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

The screenshot shows the Jenkins Update Center interface. The 'Available' tab is selected, and the search bar contains the text 'backup'. A single plugin, 'Backup plugin', is listed with a version of 1.8.1. Below the plugin description, there is a note about CloudBees Jenkins Enterprise. At the bottom of the list, there are three buttons: 'Install without restart' (highlighted in blue), 'Download now and install after restart', and 'Update information of'.

The screenshot shows the Jenkins Update Center interface with the title 'Installing Plugins/Upgrades'. It displays the 'Preparation' section with a bulleted list: 'Checking internal connectivity', 'Checking update center connectivity', and 'Success'. Below this, it shows the 'Backup plugin' status as 'Success'. There are two green checkmark icons with accompanying 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'. At the bottom of the page, there is footer text: 'Help us localize this page' and 'Page generated: Oct 26, 2015 9:26:30 PM REST API Jenkins ver. 1.691.3'.

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

Manage Jenkins [Jen] x

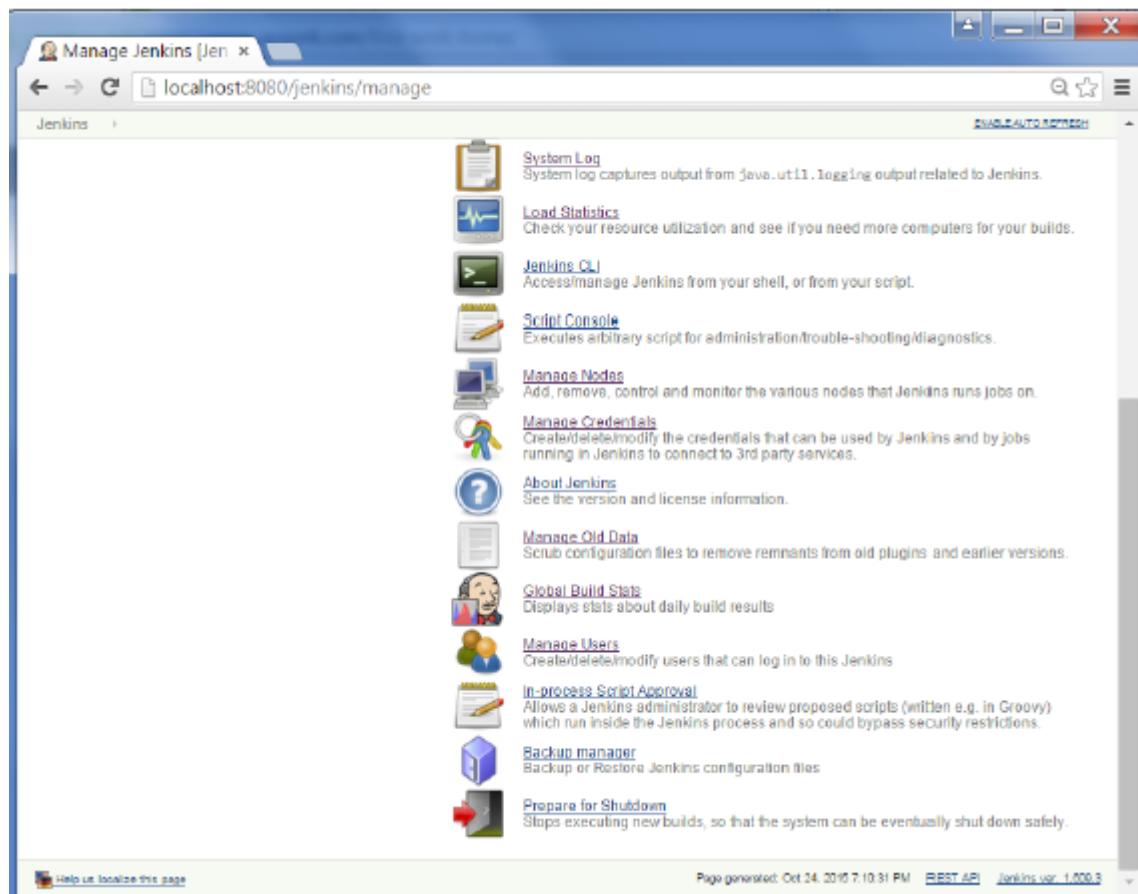
localhost:8080/jenkins/manage

Jenkins > Manage Jenkins

ENABLE AUTO REFRESH

System Log  
Load Statistics  
Jenkins CLI  
Script Console  
Manage Nodes  
Manage Credentials  
About Jenkins  
Manage Old Data  
Global Build Stats  
Manage Users  
In-process Script Approval  
Backup manager  
Prepare for Shutdown

Help us localize this page Page generated: Oct 24, 2015 7:10:31 PM REST API Jenkins ver. 1.600.3



#### Step 4 – Click on Setup.

Jenkins x

localhost:8080/jenkins/backup/

Jenkins > Backup manager

search log in

New Item  
People  
Build History  
Manage Jenkins  
Credentials

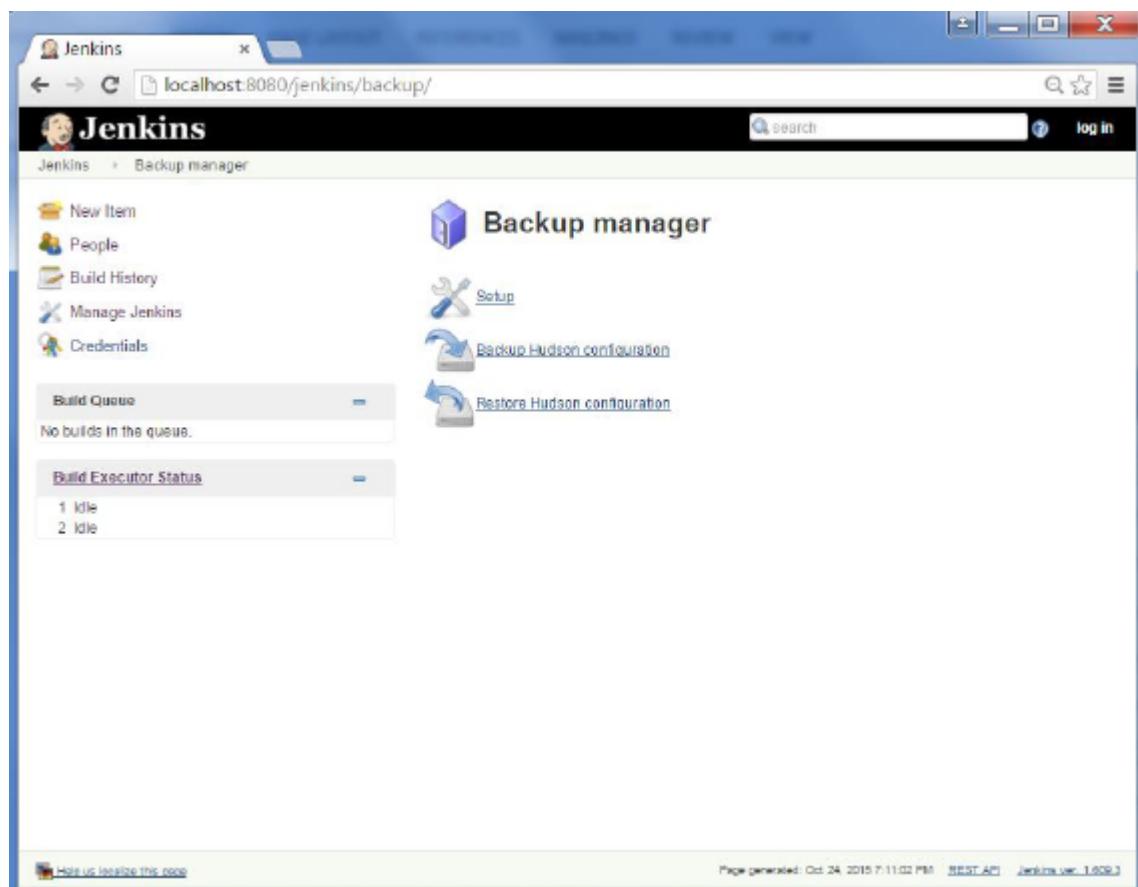
Build Queue  
No builds in the queue.

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 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 interface for managing backups. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below that are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (2 idle). The main content area is titled 'Backup config files' and contains a 'Backup configuration' section. It includes fields for 'Hudson root directory E:\Jenkins', 'Backup directory D:\Backup' (with a help icon), 'Format Zip' (with a dropdown menu), 'File name template backup\_@date@.@(extension@)' (with a help icon), and 'Custom exclusions' (with a help icon). Under 'Backup content', there are four checkboxes: 'Verbose mode', 'Configuration files (.xml) only', 'No shutdown', and 'Backup job workspace', 'Backup builds history', 'Backup maven artifacts archives', and 'Backup fingerprints'. A 'Save' button is at the bottom right of this section. At the very bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2016 7:17:46 PM', 'REST API', and 'Jenkins ver. 1.639.3'.

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

The screenshot shows the Jenkins Backup manager interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below that are two expandable sections: 'Build Queue' (which says 'No builds in the queue.') and 'Build Executor Status' (which shows '1 Idle' and '2 Idle'). The main area is titled 'Backup manager' and contains three buttons: 'Setup', 'Backup Hudson configuration', and 'Restore Hudson configuration'. At the bottom, there's a link 'Help us localize this page' and some footer text: 'Page generated: Oct 24, 2015 7:11:02 PM REST API Jenkins ver. 1.609.3'.

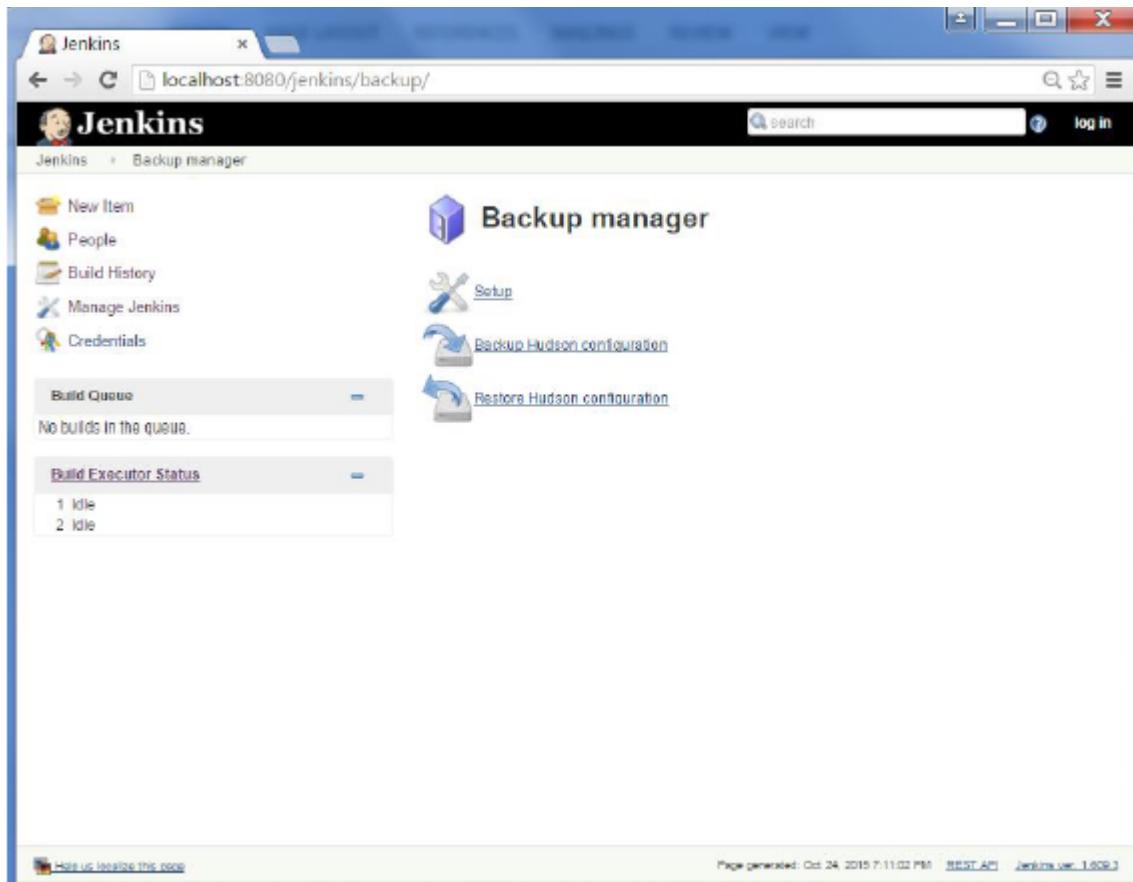
The next screen will show the status of the backup

This screenshot shows the Jenkins Backup manager log page. The interface is similar to the previous one, with a sidebar and sections for build queue and executor status. A prominent red banner at the top right states 'Jenkins is going to shut down'. Below this, a log window displays the following text:

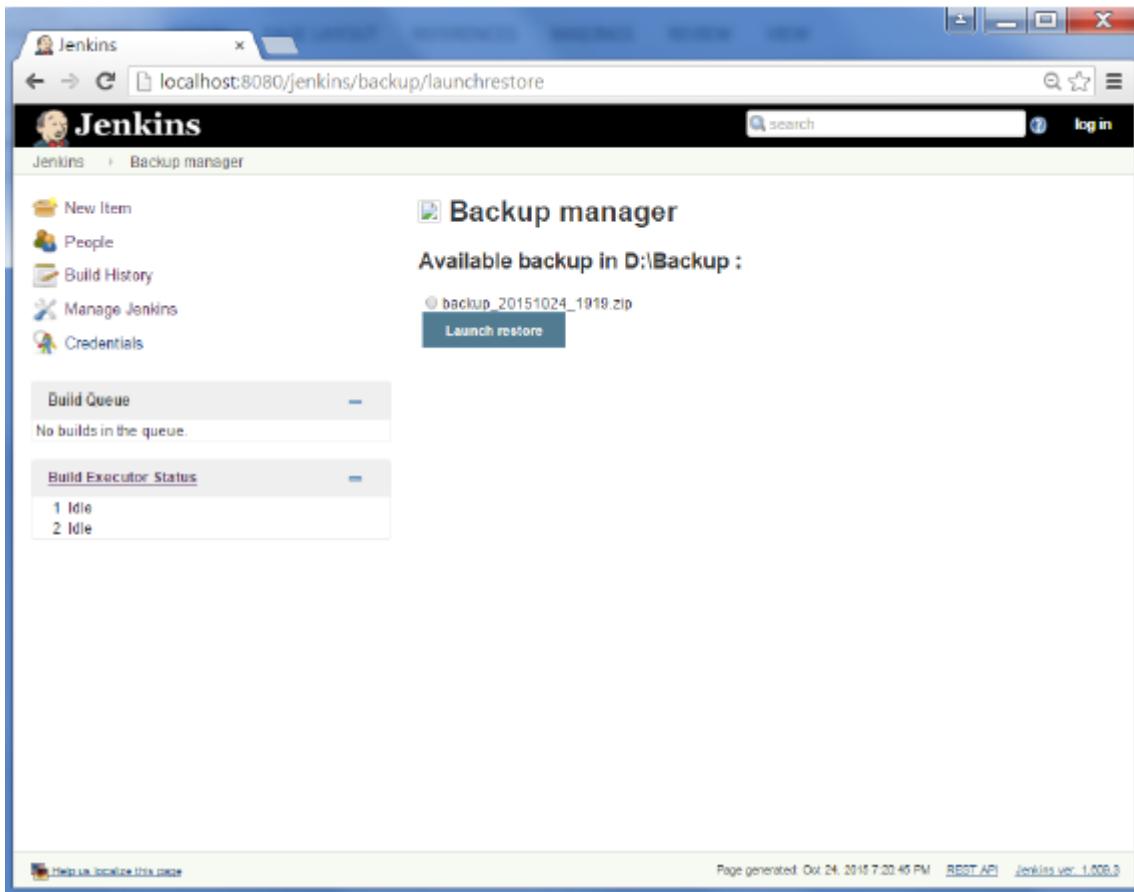
```
[ INFO] Backup started at [18/24/15 19:19:31]
[ INFO] Setting hudson in shutdown mode to avoid files corruptions.
[ INFO] Waiting all jobs end...
[ INFO] Number of running jobs detected : 0
[ INFO] All jobs finished.
[ INFO] Full backup file name : D:\Backup\backup_20151024_1919.zip
[ INFO] Saved files : 011
[ INFO] Number of errors : 0
[ INFO] Cancel hudson shutdown mode
[ INFO] Backup end at [18/24/15 19:19:38]
[ INFO] [19.524s]
```

At the bottom, there's a link 'Help us localize this page' and footer text: 'Page generated: Oct 24, 2015 7:10:31 PM REST API Jenkins ver. 1.609.3'.

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



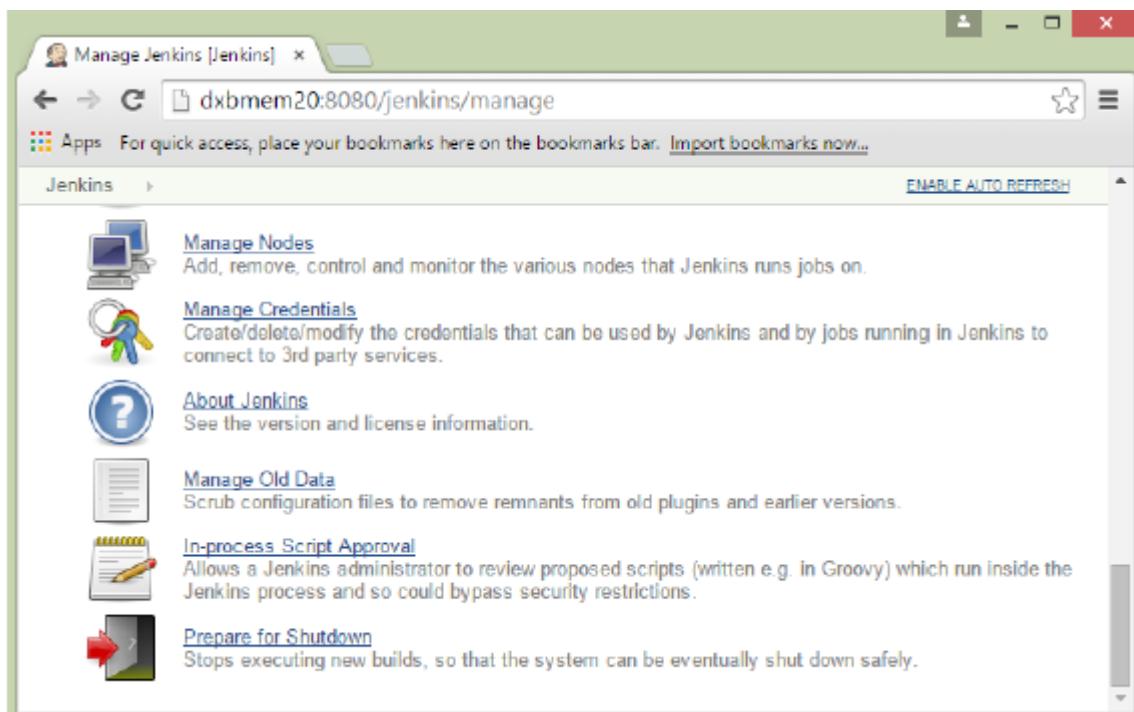
The list of backup's will be shown, click on the appropriate one to click on Launch Restore to begin the restoration of the backup.



## Jenkins - 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. Go 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. It displays two nodes:

- idle**: 2 Idle
- DXBMEM30**: 1 Idle

A detailed table provides information for each node:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space
1	DXBMEM30	Windows Server 2012 (x86)	In sync	112.79 GB	4.54 GB	13 min 112.79
2	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. At the top, there are links for 'Jenkins', 'nodes', and '2 Idle'. Below this, a table lists two nodes: 'DXBMEM30' and another node whose details are partially visible. The 'DXBMEM30' row has a context menu open, with 'Configure' highlighted in blue. Other options in the menu include 'Delete Slave' and 'Build History'. A 'Refresh status' button is located at the bottom right of the table area.

### Step 3 – Ensure the launch method is put as ‘Launch slave agents via Java Web Start’

The screenshot shows the 'DXBMEM30 Configuration' page. The 'Name' field is set to 'DXBMEM30'. Under the 'Labels' section, there is a dropdown menu with the option 'Utilize this node as much as possible'. In the 'Launch method' section, the dropdown is set to 'Launch slave agents via Java Web Start'. A 'Save' button is located at the bottom left of the form.

**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

The screenshot shows the Jenkins dashboard at <http://dxbmemp30:8080/jenkins/>. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. Below the sidebar are sections for Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). A 'Build description' link is located at the bottom right of the status section.

Step 5 – Click on the DXBMEMP30 instance.

The screenshot shows the Jenkins Nodes page at <http://dxbmemp30:8080/jenkins/computer/>. It displays the 'Build Executor Status' for the 'master' node (1 Idle, 2 Idle) and the 'DXBMEMP30' node (offline). Below this is a table of nodes:

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space
	<a href="#">DXBMEMP30</a>	Windows Server 2012 (x86)	In sync	112.79 GB	4.54 GB
	<a href="#">master</a>	Windows Server 2012 (x86)	In sync	94.20 GB	3.85 GB
	Data obtained	45 min	45 min	45 min	45 min

A 'Refresh status' button is located at the bottom right of the table.

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

The screenshot shows a web browser window titled "DXBMEM30 [Jenkins]". The URL in the address bar is "dxbmem20:8080/jenkins/computer/DXBMEM30/". The page content includes:

- A Jenkins navigation bar with links for "Jenkins", "nodes", and "DXBMEM30".
- An "ENABLE AUTO REFRESH" link.
- A stack trace starting with "at org.jenkinsci.remoting.nio.NioChannelHub.run(NioChannelHub.java:561)" followed by "... 6 more".
- Instructions for connecting a slave to Jenkins:
  - A large orange "Launch" button with a gear icon.
  - A list item: "Run from slave command line: javaws http://localhost:8080/jenkins/computer/DXBMEM30/slave-agent.jnlp".
  - A list item: "Or if the slave is headless: java -jar slave.jar -jnlpUrl http://localhost:8080/jenkins/computer/DXBMEM30/slave-agent.jnlp".
- A note: "Created by anonymous user".
- A section titled "Projects tied to DXBMEM30" with a table:

S	W	Name ↓	Last Success	Last Failure	Last Duration
		<a href="#">HelloWorld</a>	43 min - #12	41 min - #13	7.3 sec

Below the table are icons for "Icon: S M L" and links for "Legend", "RSS for all", "RSS for failures", and "RSS for just latest builds".

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



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.

The screenshot shows the Jenkins configuration page for the 'HelloWorld' job. The 'Label Expression' is set to 'DXBMEM30'. Under 'Advanced Project Options', there is a 'Source Code Management' section. At the bottom, there are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins configuration page for the 'HelloWorld' job. A 'SeleniumHQ htmlSuite Run' build step is selected. The configuration includes:

- browser: firefox
- startURL: http://localhost:8080
- suiteFile: C:\Selenium\Sample.html
- resultFile: C:\Users\administrator.EMIRATES\jenkins\jobs\HelloWorld\workspace\Reports\Results.html
- other: (empty)

At the bottom, there is a 'Delete' button, an 'Add build step' dropdown menu, and 'Save' and 'Apply' buttons.

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.

