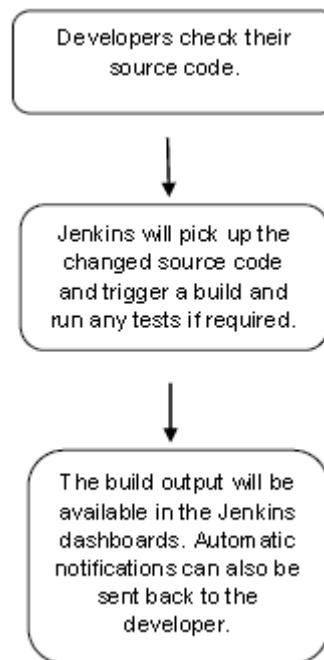


Jenkins - Quick Guide

Jenkins - Overview

Why Jenkins?

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



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

What is Continuous Integration?

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

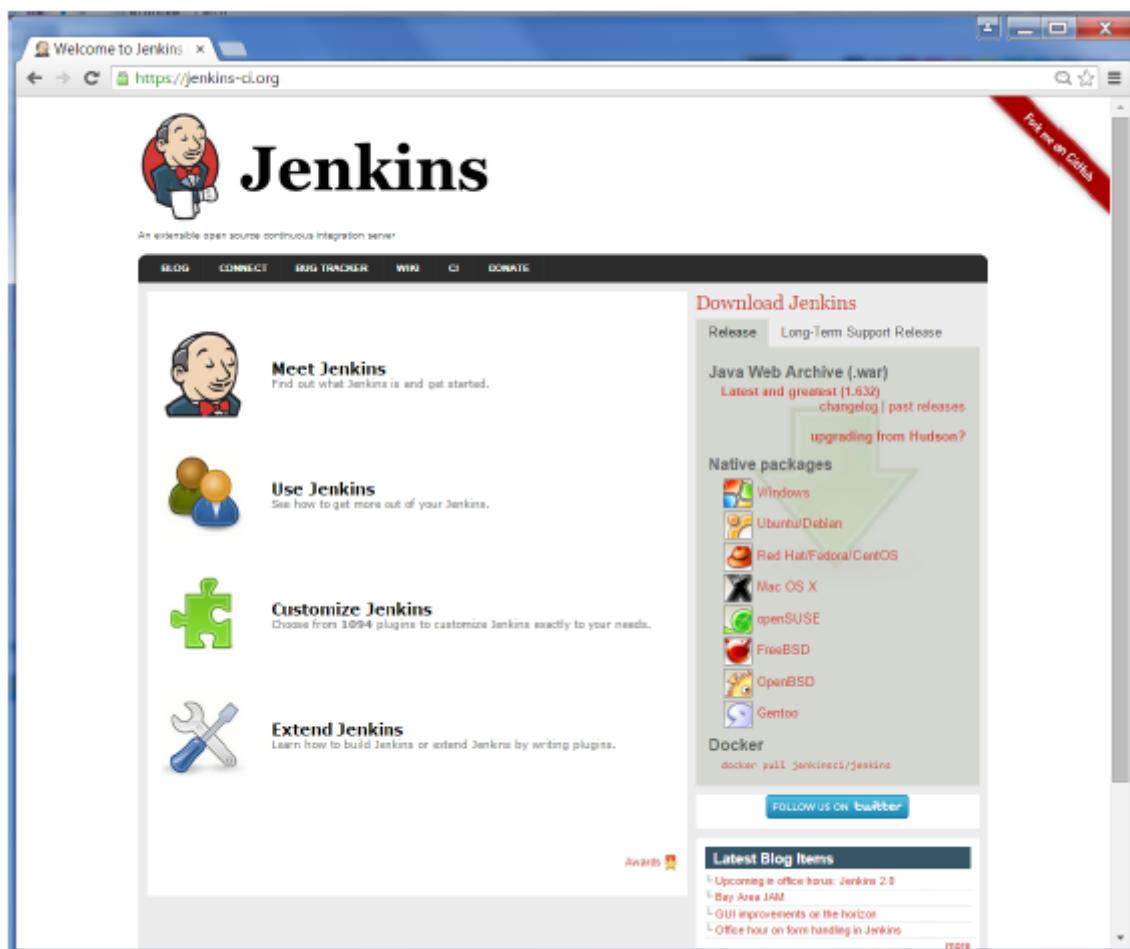
System Requirements

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

Jenkins - Installation

Download Jenkins

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



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



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

Starting Jenkins

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

```
D:\>Java -jar Jenkins.war
```

After the command is run, various tasks will run, one of which is the extraction of the war file which is done by an embedded webserver called winstone.

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

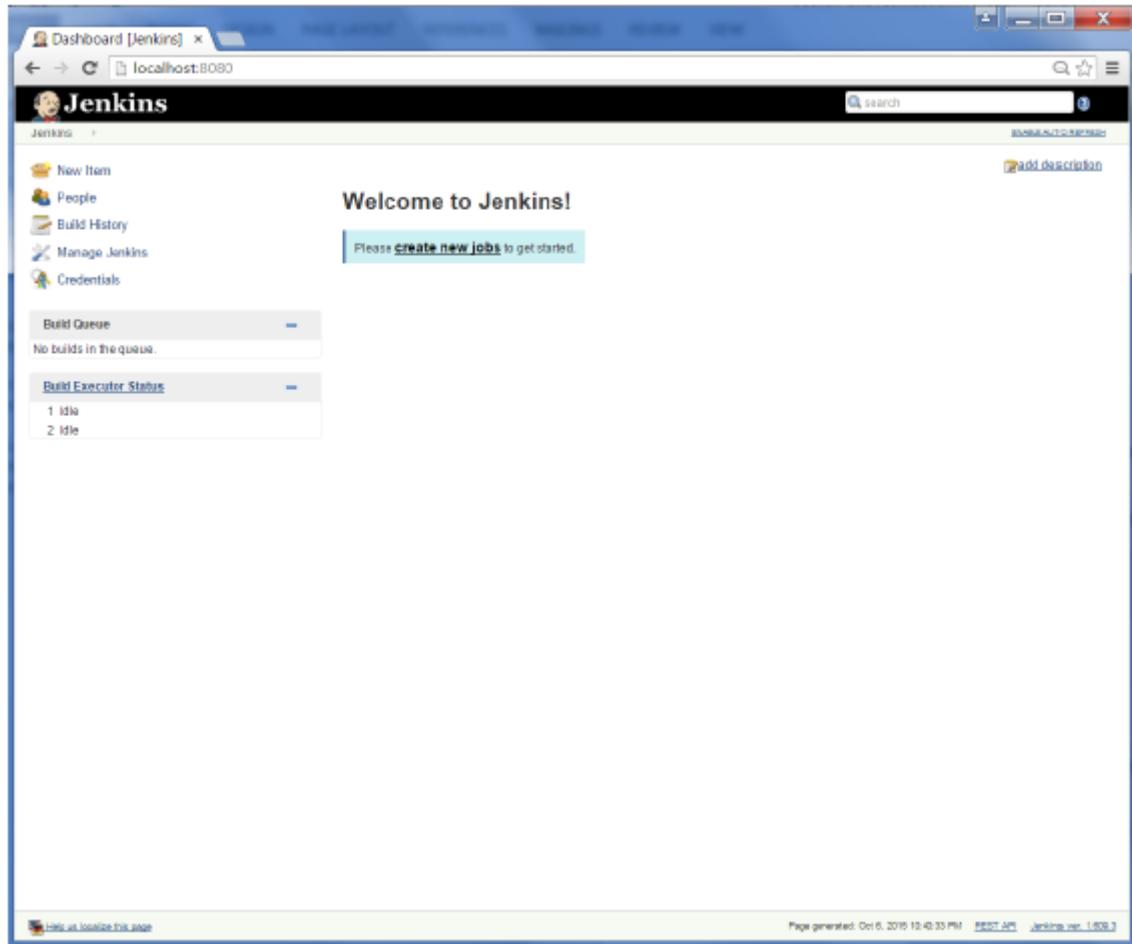
Once the processing is complete without major errors, the following line will come in the output of the command prompt.

INFO: Jenkins is fully up and running

Accessing Jenkins

Once Jenkins is up and running, one can access Jenkins from the link – <http://localhost:8080>

This link will bring up the Jenkins dashboard.



Jenkins – Tomcat Setup

The following prerequisites must be met for Jenkins Tomcat setup.

Step 1: Verifying Java Installation

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

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

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

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

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

In case you do not have Java JDK, you can download it from the link [Oracle](#)

Step 2: Verifying Java Installation

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

OS	Output
Windows	Set Environmental variable JAVA_HOME to C:\ProgramFiles\java\jdk1.7.0_60
Linux	export JAVA_HOME=/usr/local/java-current

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

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

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

Step 3: Download Tomcat

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

The screenshot shows the Apache Tomcat website homepage. The main content area displays the "Tomcat 8.0.27 Released" section, dated 2015-10-01. It highlights several fixes and improvements, including correctly handling ISO vs US escaping in JSP and EL, fixing issues with NIO + SSL + sendfile, various TLD parsing fixes, and fixing multiple concurrency issues. A link to the full changelog is provided.

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

The screenshot shows the Apache Tomcat website's download page for Tomcat 7.0.64. It includes sections for "Tomcat 7 Downloads", "Quick Navigation" (with links to KEYS, 7.0.64, Browse, and Archives), "Release Integrity" (instructions for verifying file integrity using OpenPGP signatures), "Mirrors" (a list of mirrors for download, with the current one being http://www.us.apache.org/dist/), and "7.0.64" (a note to see the README file for packaging information). The "Binary Distributions" section lists various download links for Core distributions, including 32-bit and 64-bit Windows zip files, tar.gz files, and an installer.

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

Then unzip the contents of the downloaded zip file.

Step 4: Jenkins and Tomcat Setup

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

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

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

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

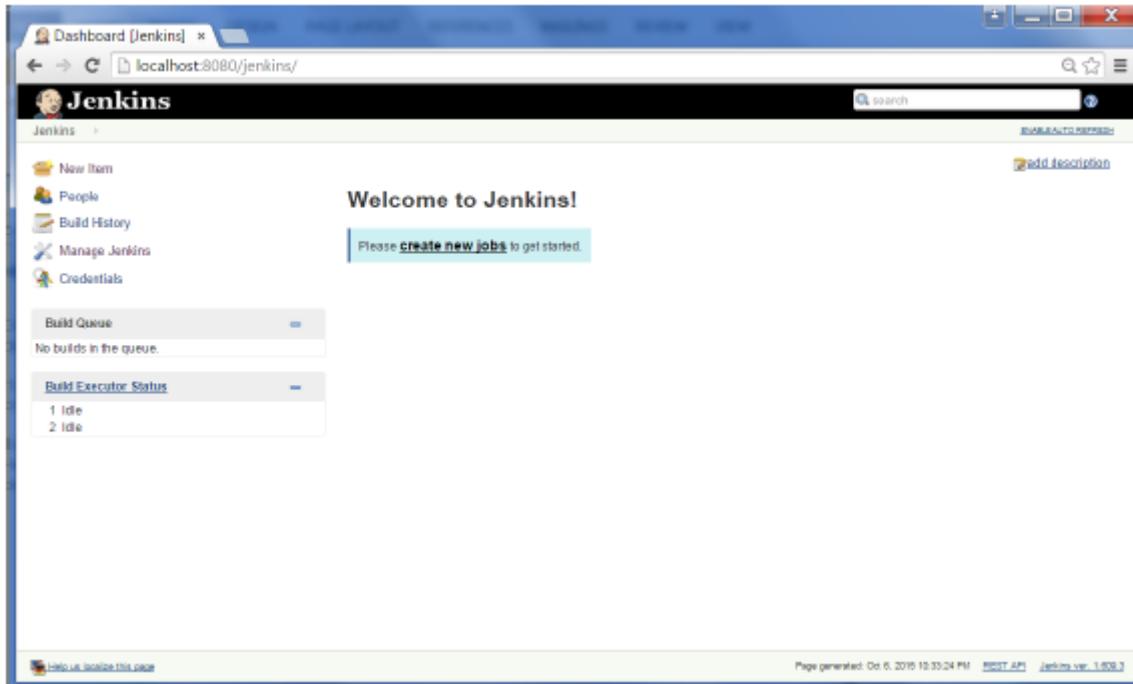
```
INFO: Server startup in 1302 ms
```

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

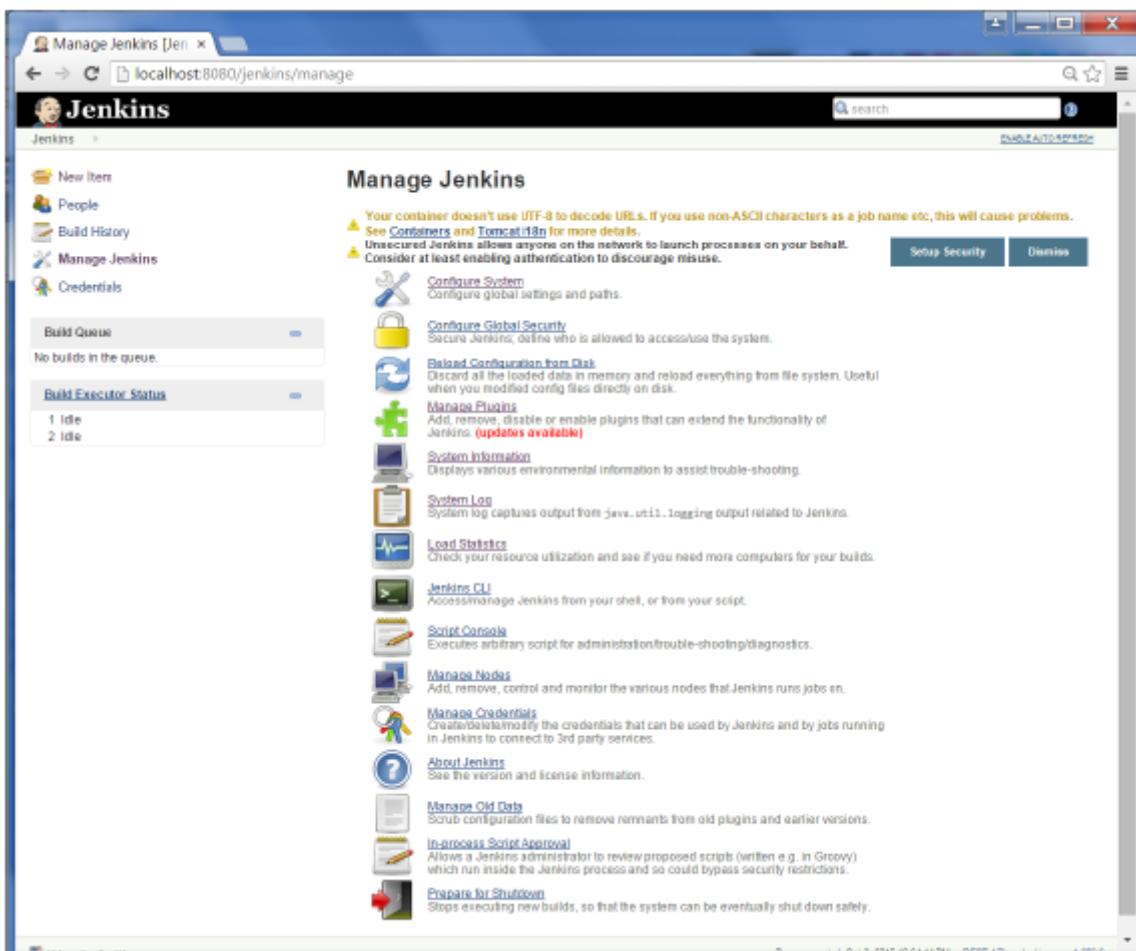


Jenkins - Git Setup

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



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



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

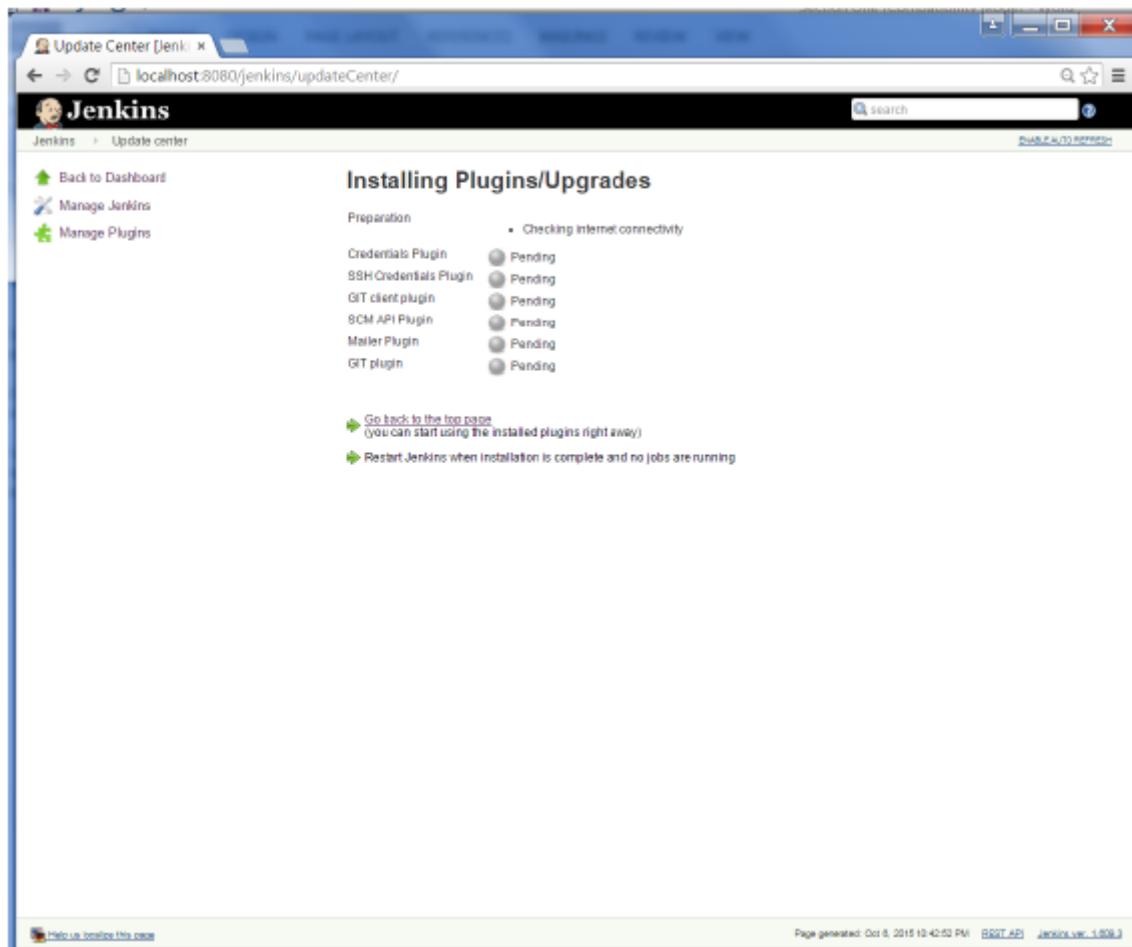
The screenshot shows the Jenkins Update Center interface. The 'Available' tab is selected. A search bar at the top right contains the text 'Git plugin'. Below the search bar, a table lists several Jenkins plugins. The 'GIT plugin' row is highlighted with a blue background, indicating it is selected for installation. The table has columns for Name, Version, and a brief description. At the bottom of the page, there are three buttons: 'Install without restart', 'Download now and install after restart', and 'Check now'.

Install	Name	Version
Git Parameter Plugin	0.4.0	This plugin allows you to choose between Git tags or sha1 of your SCM repository so Git Plugin installed is required.
UserContent in Git plugin	1.4	This plugin exposes \$JENKINS_HOME/userContent as Git repository.
Alternative build chooser	1.1	An alternative build chooser plugin for the Jenkins git plugin.
Team Concert Git Plugin	1.0.10	Integrates Jenkins with Rational Team Concert for Jenkins Builds which use Git as source control. This plugin will create traceability links from a Jenkins build to Rational Team Concert Work Items and build results. This plugin adds traceability links from a Jenkins build to an RTC build result. It also publishes links to work items and annotates the change log generated by Jenkins with links to RTC Work Items. It leverages the current RTC features and workflows that users are already familiar with such as, emails, toaster popups, reporting, dashboards, etc.
Tracking Git Plugin	1.0	Lets one project track the Git revisions that are built for another project.
GIT plugin	2.4.0	This plugin allows use of Git as a build SCM. A recent Git runtime is required (1.7.9 minimum, 1.8.x recommended). Plugin is only tested on official git client . Use exotic installations at your own risks.

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

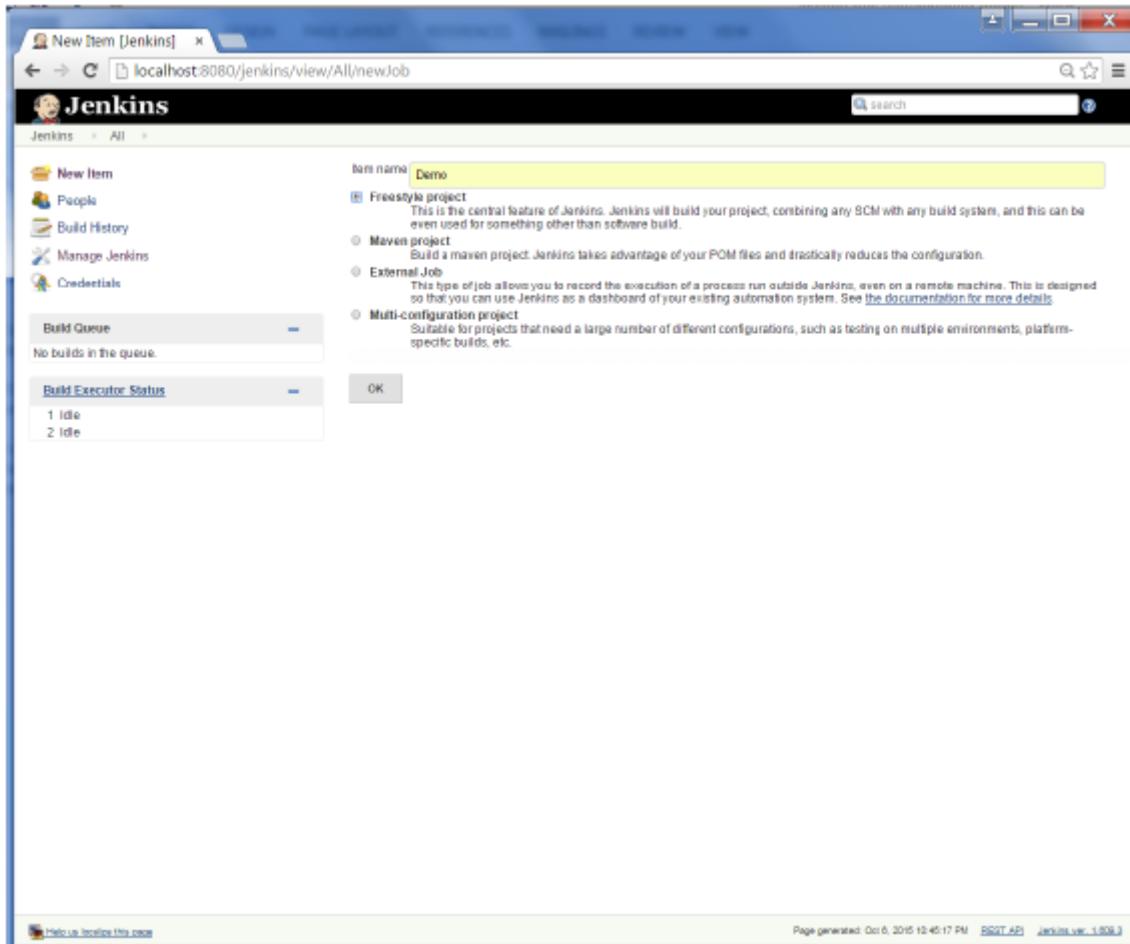
The screenshot shows the Jenkins Update Center interface, identical to the previous one but with a slight difference in the 'GIT plugin' row. The 'GIT plugin' row now has a checked checkbox next to it, indicating it is selected for installation. The rest of the interface, including the table, buttons, and footer, remains the same.

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

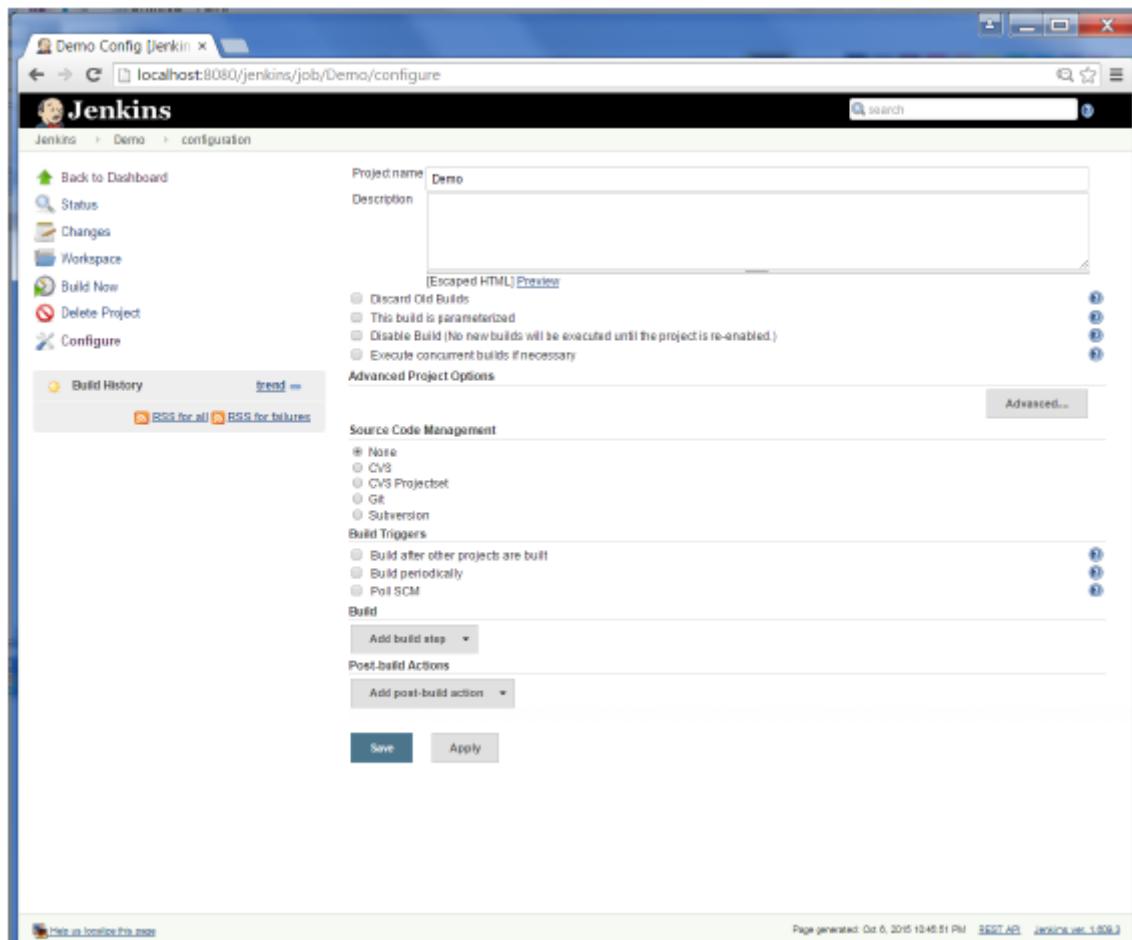


Once all installations are complete, restart Jenkins by issue the following command in the browser. **<http://localhost:8080/jenkins/restart>**

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



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



Jenkins – Maven Setup

Step 1: Downloading and Setting Up Maven

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

The screenshot shows the Apache Maven Project download page. The left sidebar has a 'Download' link highlighted. The main content area is titled 'Downloading Apache Maven 3.3.3'. It includes a note about the selected download mirror, a 'Change' button for mirrors, and a 'System Requirements' table. The 'Files' section is also visible.

	Link	Checksum	Signature
Binary tar.gz archive	apache-maven-3.3.3-bin.tar.gz	apache-maven-3.3.3-bin.tar.gz.md5	apache-maven-3.3.3-bin.tar.gz.asc
Binary zip archive	apache-maven-3.3.3-bin.zip	apache-maven-3.3.3-bin.zip.md5	apache-maven-3.3.3-bin.zip.asc
Source tar.gz archive	apache-maven-3.3.3-src.tar.gz	apache-maven-3.3.3-src.tar.gz.md5	apache-maven-3.3.3-src.tar.gz.asc
Source zip archive	apache-maven-3.3.3-src.zip	apache-maven-3.3.3-src.zip.md5	apache-maven-3.3.3-src.zip.asc

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

The screenshot shows the Apache Maven Project download page. The left sidebar has a 'Support and Training' link highlighted. The main content area is titled 'Files'. It includes a note about Maven formats and download instructions, followed by a table of download links for different archive types.

	Link	Checksum	Signature
Binary tar.gz archive	apache-maven-3.3.3-bin.tar.gz	apache-maven-3.3.3-bin.tar.gz.md5	apache-maven-3.3.3-bin.tar.gz.asc
Binary zip archive	apache-maven-3.3.3-bin.zip	apache-maven-3.3.3-bin.zip.md5	apache-maven-3.3.3-bin.zip.asc
Source tar.gz archive	apache-maven-3.3.3-src.tar.gz	apache-maven-3.3.3-src.tar.gz.md5	apache-maven-3.3.3-src.tar.gz.asc
Source zip archive	apache-maven-3.3.3-src.zip	apache-maven-3.3.3-src.zip.md5	apache-maven-3.3.3-src.zip.asc

Below the table, there is a list of links:

- 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.

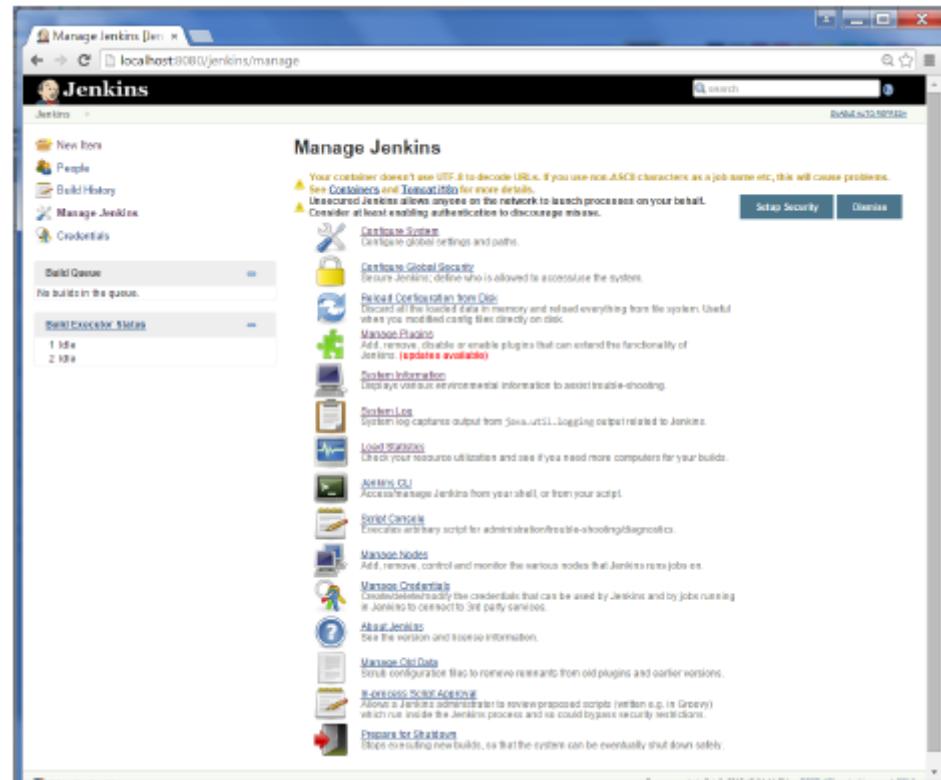
Once the file is downloaded, extract the files to the relevant application folder. For this purpose, the maven files will be placed in E:\Apps\apache-maven-3.3.3.

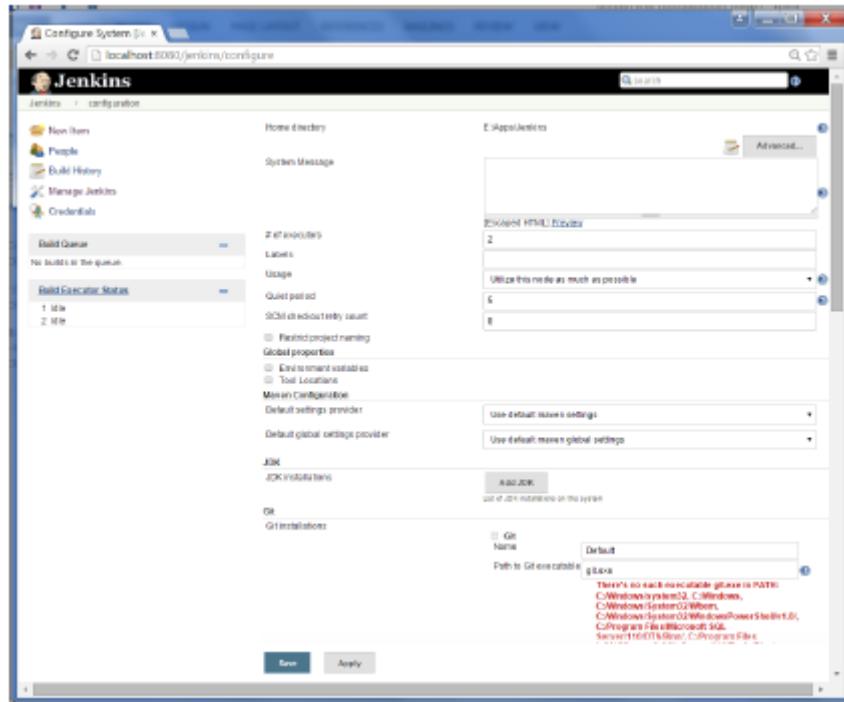
Step 2: Setting up Jenkins and Maven

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

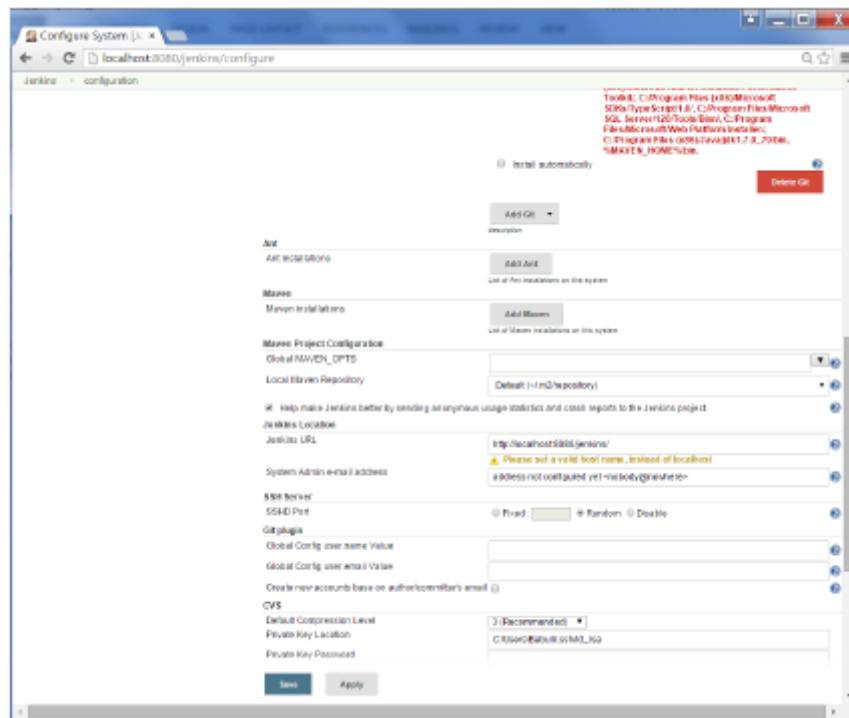


Then, click on 'Configure System' from the right hand side.





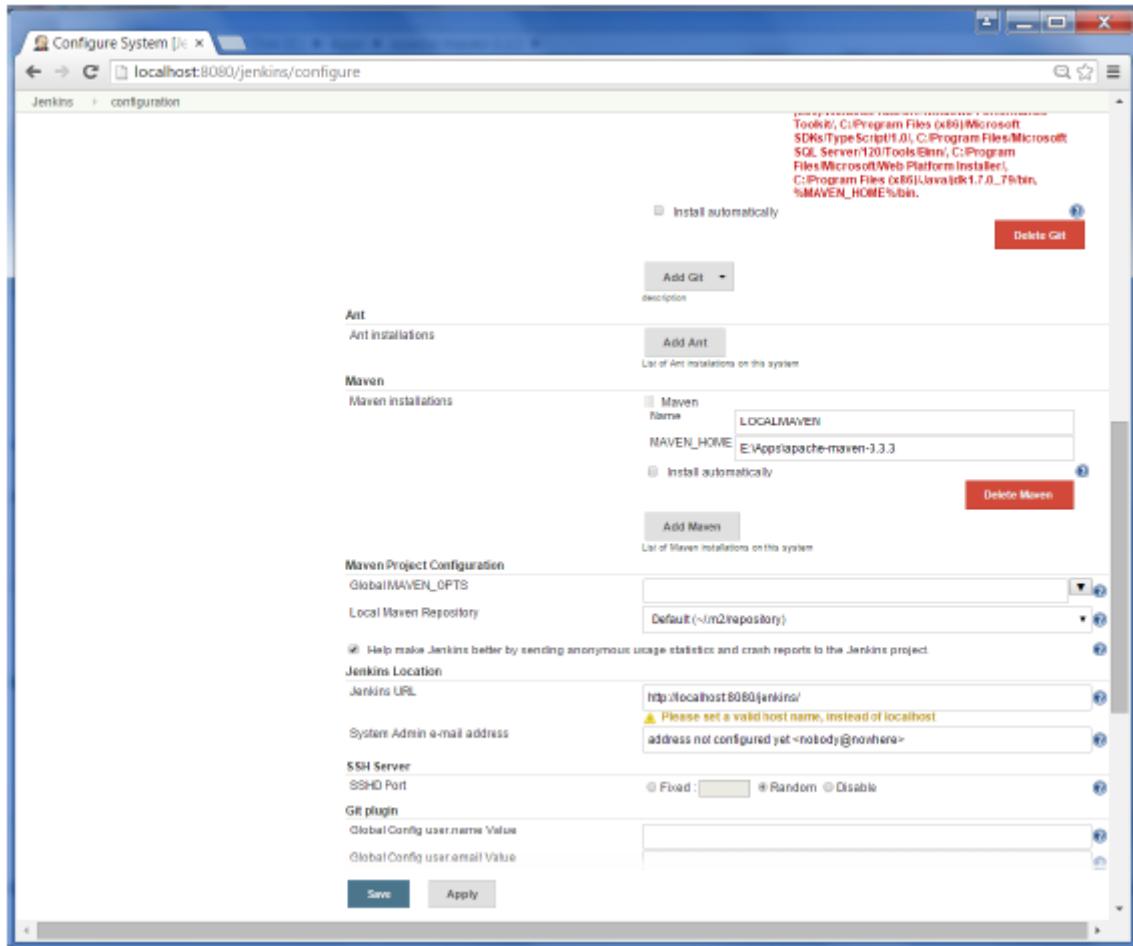
In the Configure system screen, scroll down till you see the Maven section and then click on the 'Add Maven' button.



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.

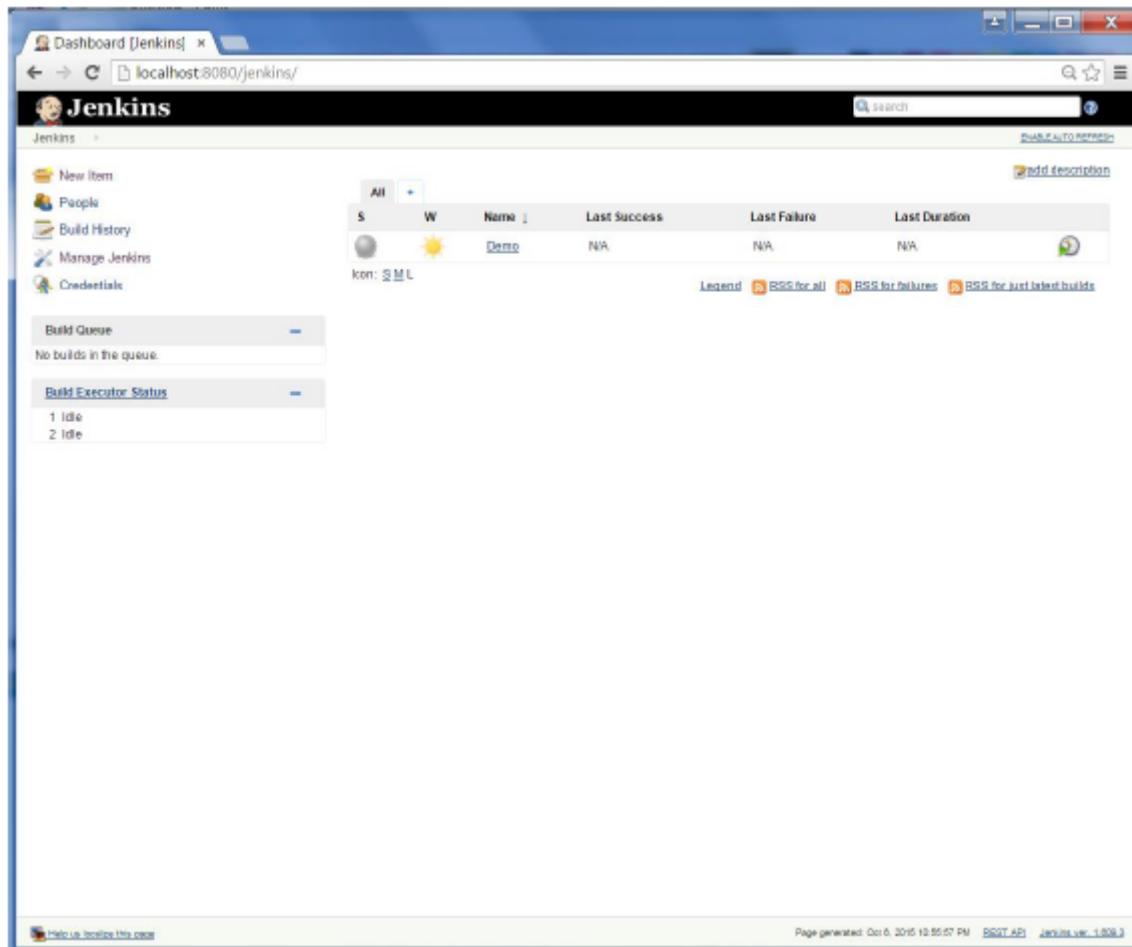
The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. The main area displays the 'Build Queue' and 'Build Executor Status'. The 'Build Queue' section shows 'No builds in the queue.' The 'Build Executor Status' section shows '1 Idle' and '2 Idle' executors.

The screenshot shows the 'New Item' dialog at localhost:8080/jenkins/view/All/newJob. The 'Job name' field is set to 'MavenDemo'. The 'Freestyle project' radio button is selected, with a note explaining it's the central feature of Jenkins. Other options shown include 'Maven project', 'External Job', 'Multi-configuration project', and 'Copy existing item'. A 'Copy from' dropdown is present. At the bottom is an 'OK' button.

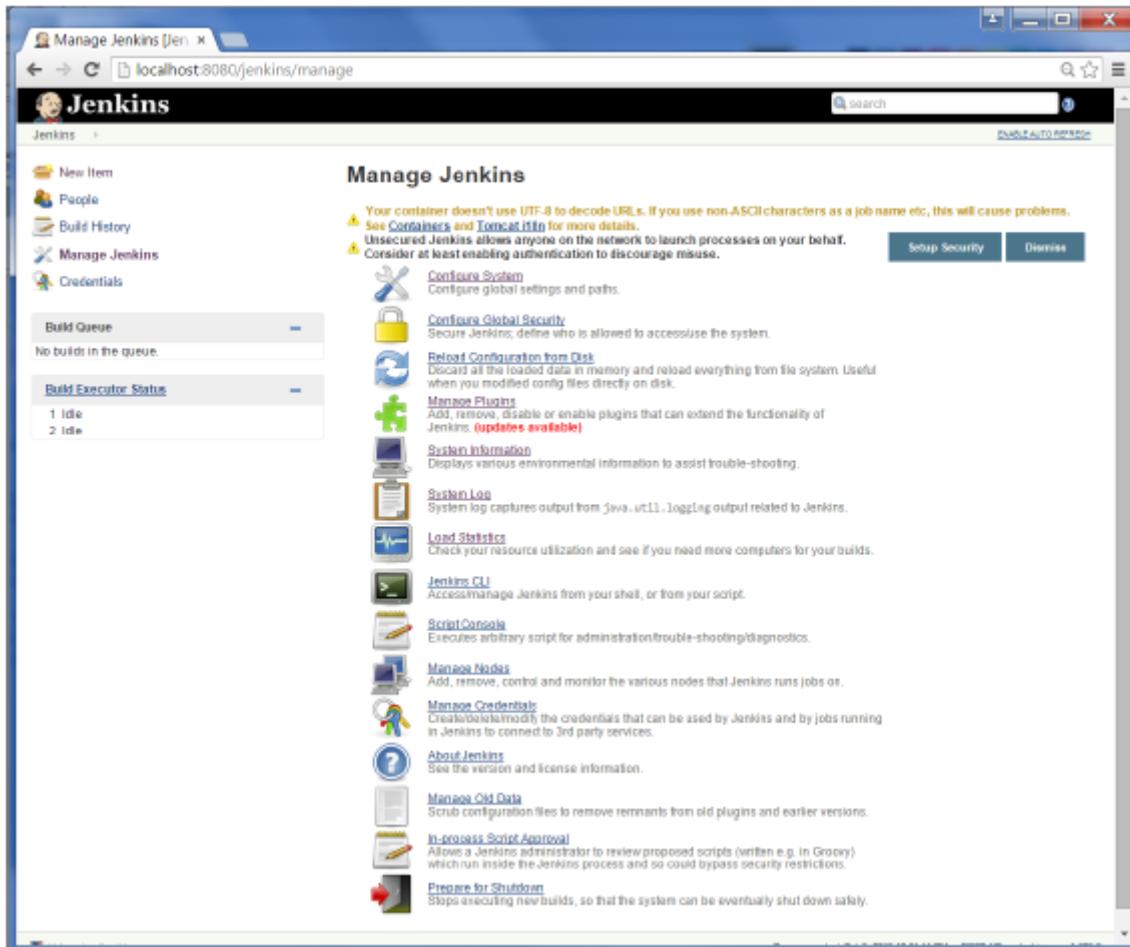
Jenkins - Configuration

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.

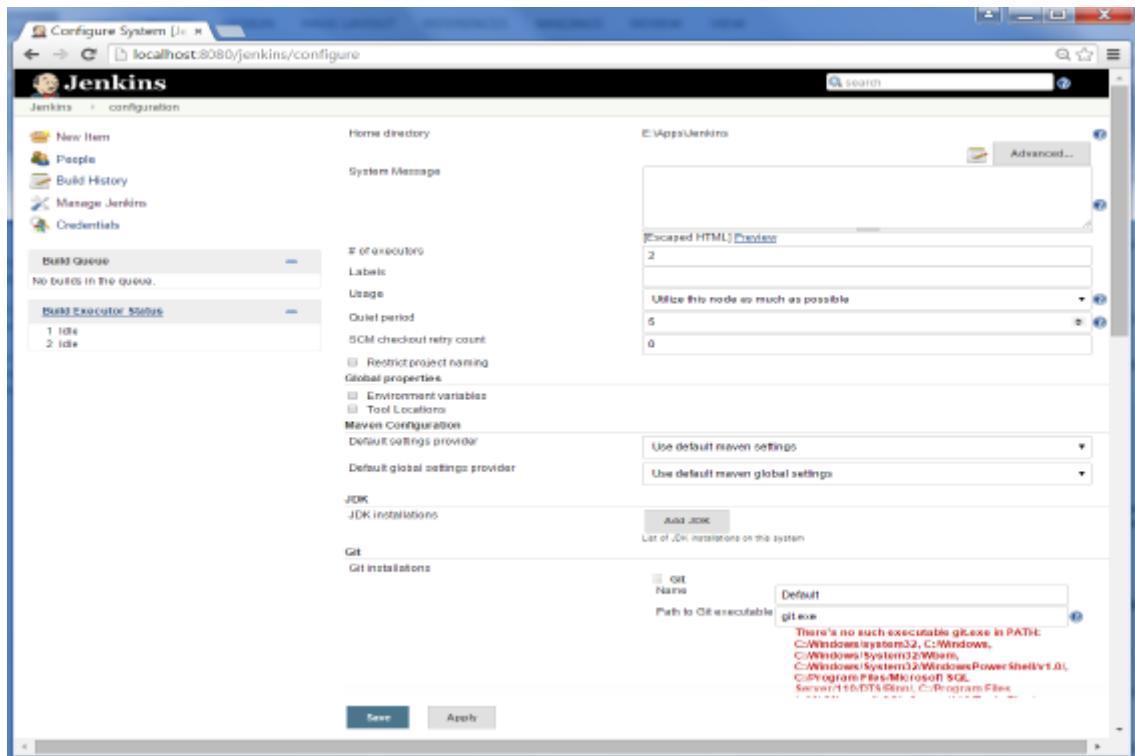
First create a new folder `E:\Apps\Jenkins`. Copy all the contents from the existing `~/jenkins` to this new directory.

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}.

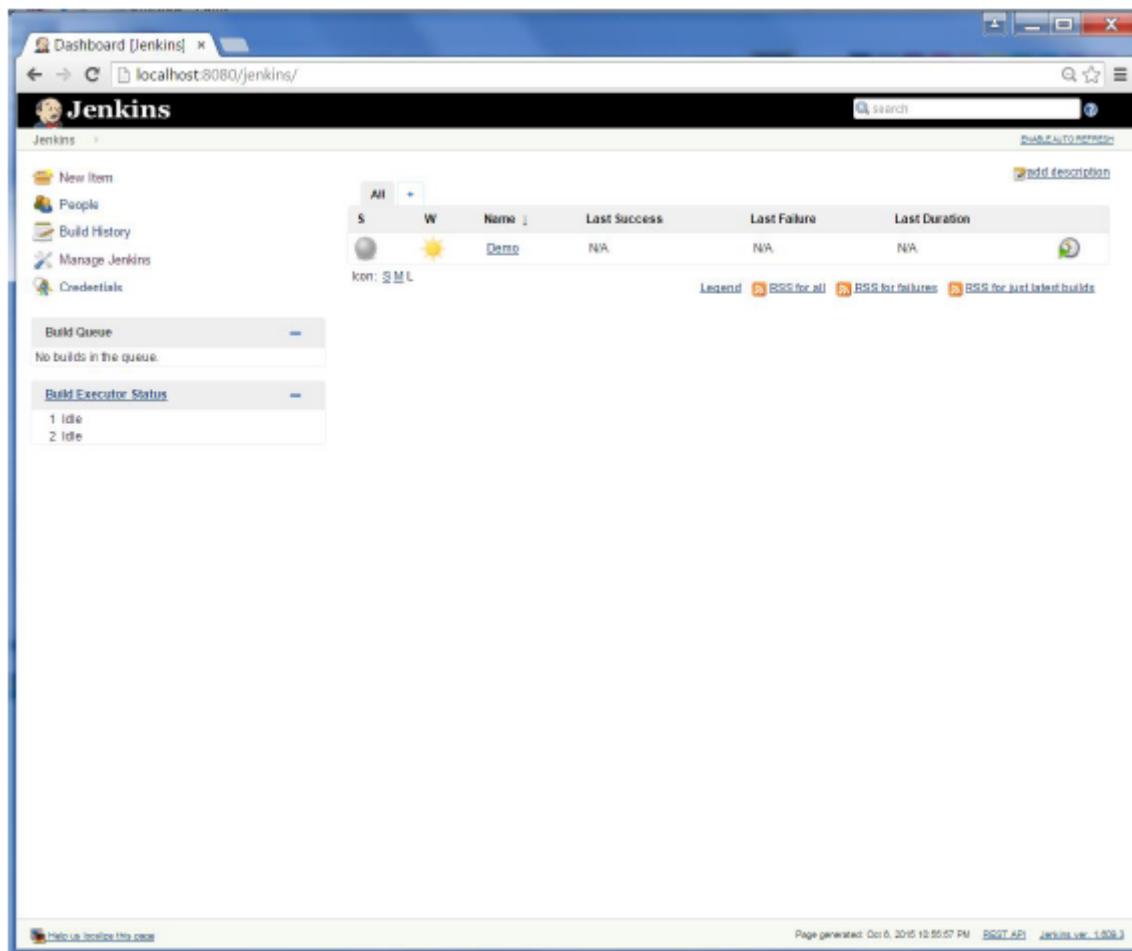
Email Notification

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.

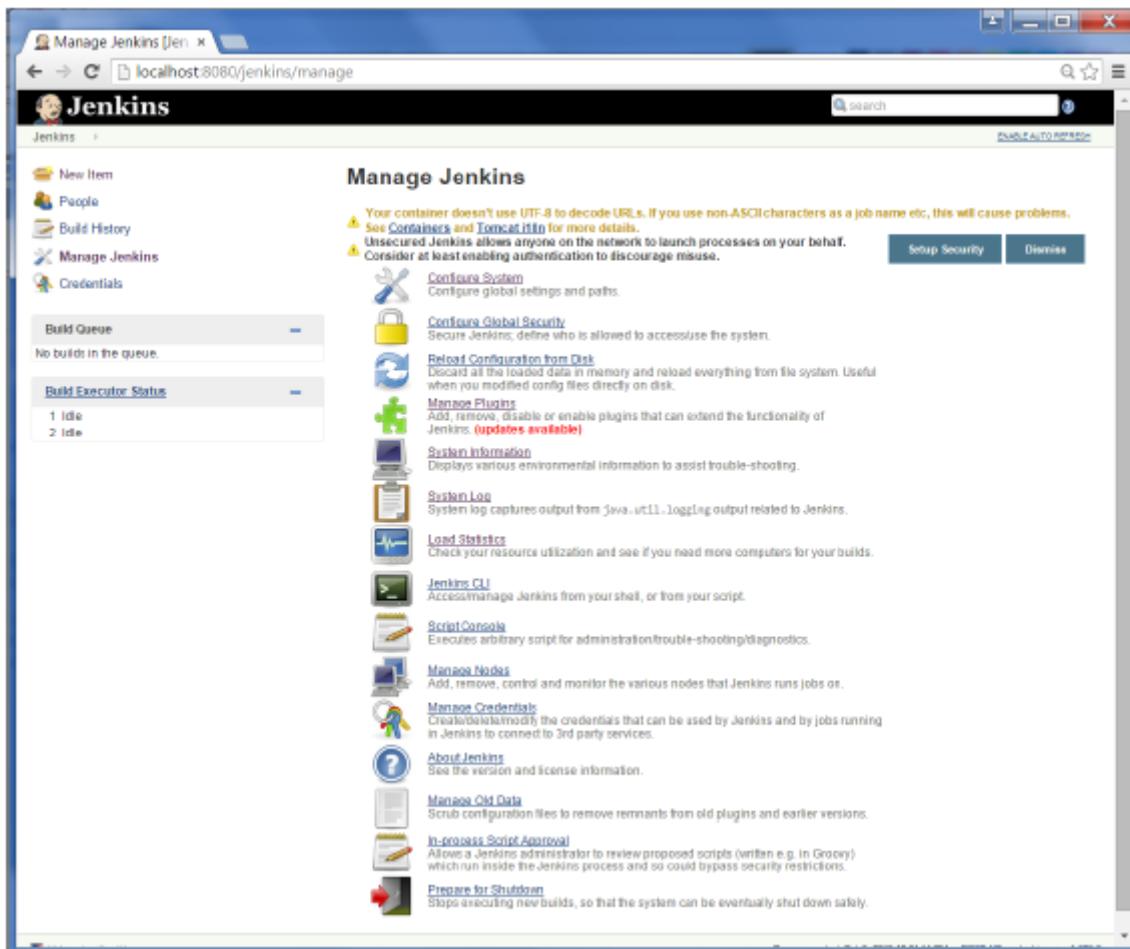
Jenkins - Management

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 Plugin Manager interface. At the top, there's a navigation bar with links for 'Back to Dashboard' and 'Manage Jenkins'. Below this is a search bar and a filter input field. The main area has tabs for 'Updates', 'Available', 'Installed', and 'Advanced'. The 'Updates' tab is selected, displaying a list of plugins with their names, descriptions, current version, and installed version. Plugins listed include CVS Plugin, Javadoc Plugin, JUnit Plugin, Matrix Authorization Strategy Plugin, Matrix Project Plugin, Maven Integration plugin, OWASP Markup Formatter Plugin, PAM Authentication plugin, Script Security Plugin, SSH Slaves plugin, Subversion Plugin, Translation Assistance plugin, and Windows Slaves Plugin. A prominent blue button at the bottom left says 'Download now and install after restart'. To its right, a message says 'Update information obtained: 1 hr 36 min ago' and a 'Check now' button. Below these buttons, there's a link 'Select All None' and a note stating 'This page lists updates to the plugins you currently use.' At the very bottom of the page, there are footer links for 'Help us localize this page', 'Page generated: Oct 6, 2015 11:08:05 PM', 'REST API', and 'Jenkins ver. 1.809'.

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

System Information

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

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

System Properties	
Name	Value
awt.toolkit	sun.awt.windows.WToolkit
catalina.base	E:\Appstomcat7
catalina.home	E:\Appstomcat7
catalina.useNaming	true
common.loader	\$catalina.base\$lib\$\${catalina.base}lib*.jar,\$catalina.home\$lib\$\${catalina.home}lib*.jar
file.encoding	Cp1252
file.encoding.pkg	sun.io
file.separator	\
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.path	E:\Appstomcat7\bin\bootstrap.jar;E:\Appstomcat7\bin\tomcat-juli.jar
java.class.version	51.0
java.endorsed.dirs	E:\Appstomcat7\endorsed
java.ext.dirs	C:\Program Files (x86)\Java\jdk1.7.0_79\jre\lib\ext;C:\Windows\SunJava\lib\ext
java.home	C:\Program Files (x86)\Java\jdk1.7.0_79\jre
java.io.tmpdir	E:\Appstomcat7\tmp
java.library.path	C:\Program Files (x86)\Java\jdk1.7.0_79\bin;C:\Windows\SunJava\bin;C:\Windows\System32;C:\Windows\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\VSShell\10.0\ManagementStudio\;C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\PrivateAssemblies\;C:\Program Files (x86)\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files (x86)\Windows Performance Toolkit\1.0\;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.URLContextFactory
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:\Appstomcat7\conf\logging.properties
java.util.logging.manager	org.apache.juli.ClassLoaderLogManager
java.vendor	Oracle Corporation
java.vendor.url	http://java.oracle.com/
java.vendor.url_bug	http://bugreport.sun.com/bugreport/
java.version	1.7.0_79
java.vm.info	mixed mode, sharing

System Log

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

Load Statistics

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

Script Console

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

Manage nodes

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

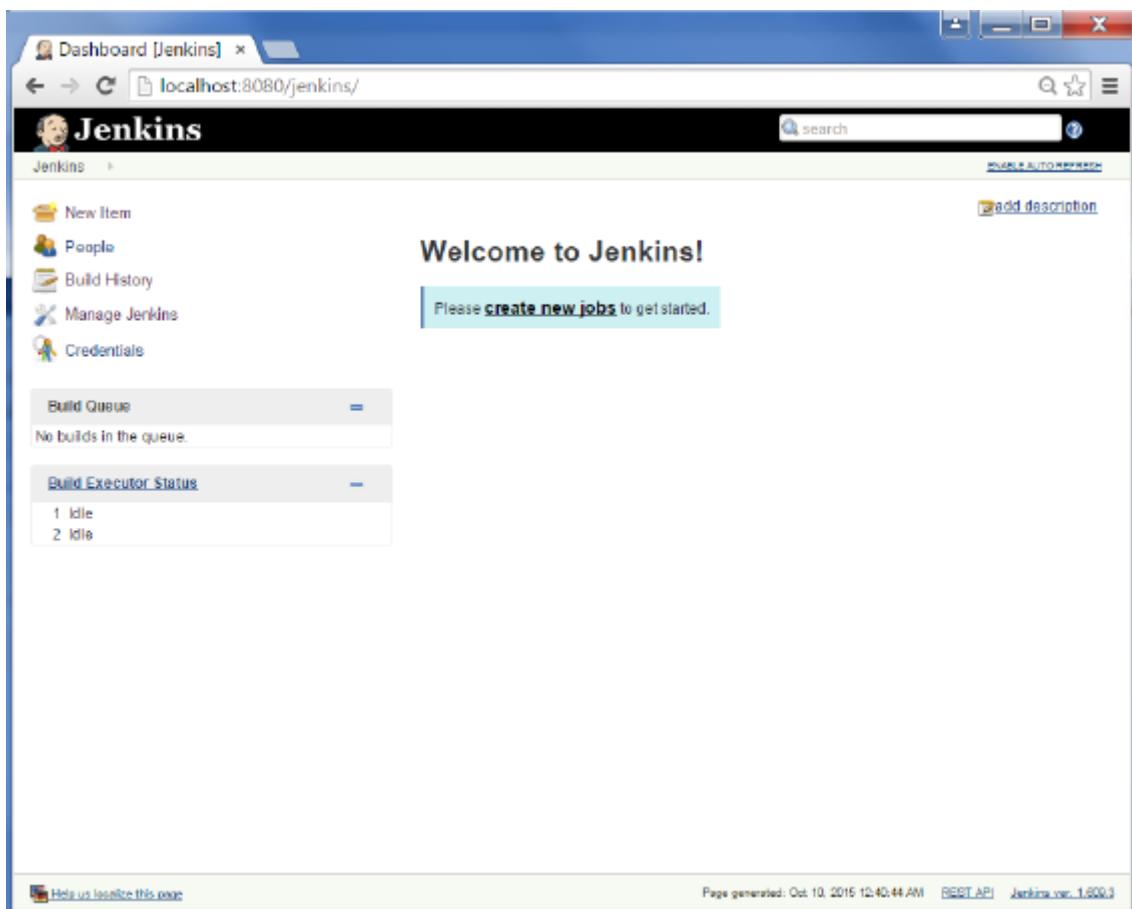
Prepare for Shutdown

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

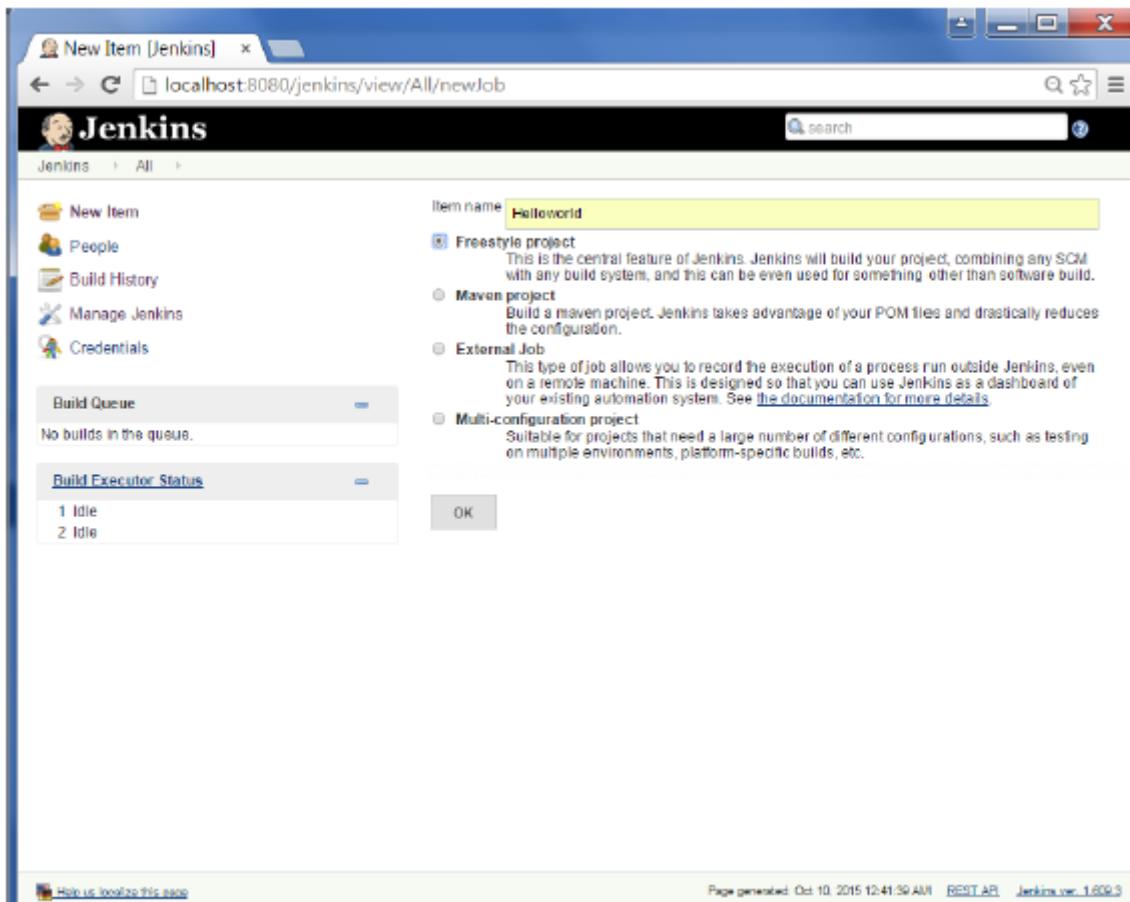
Jenkins - Setup Build Jobs

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

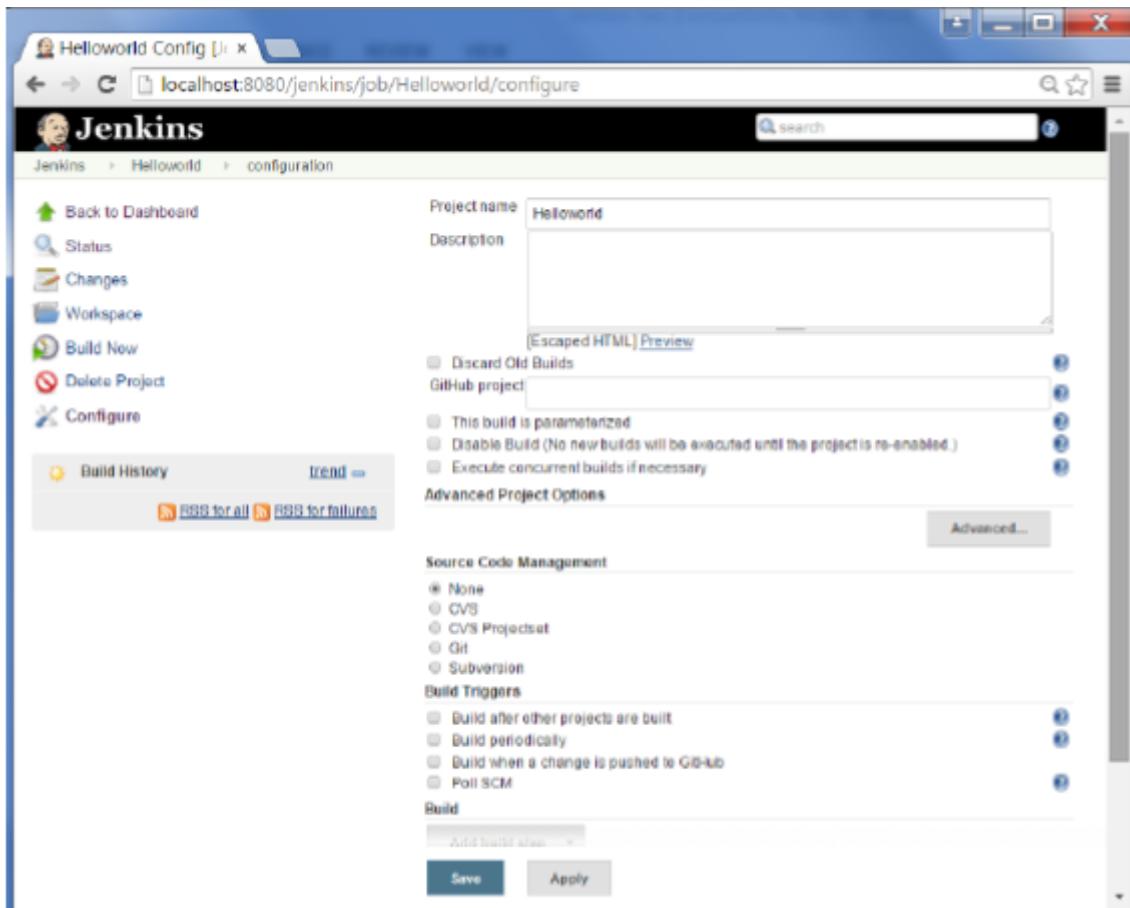
Step 1 – Go to the Jenkins dashboard and Click on New Item



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



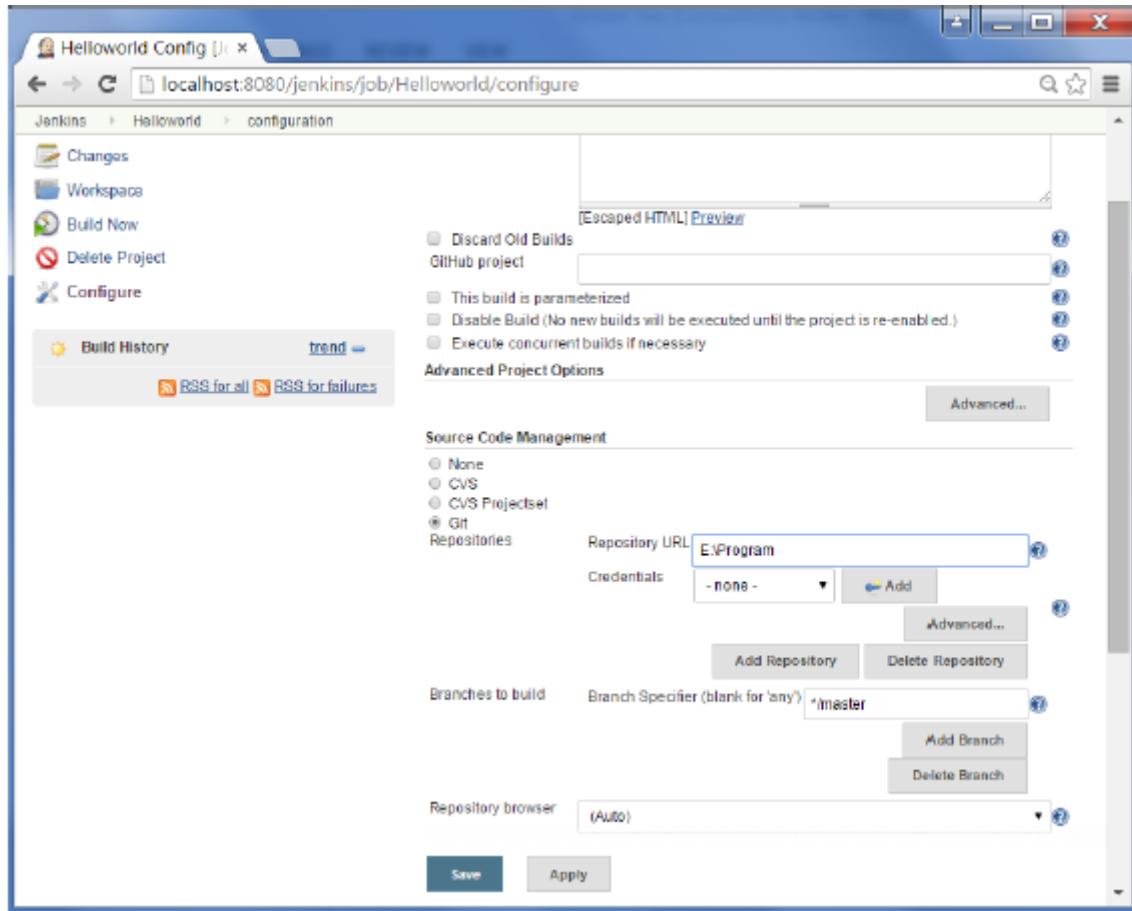
Step 3 – The following screen will come up in which you can specify the details of the job.



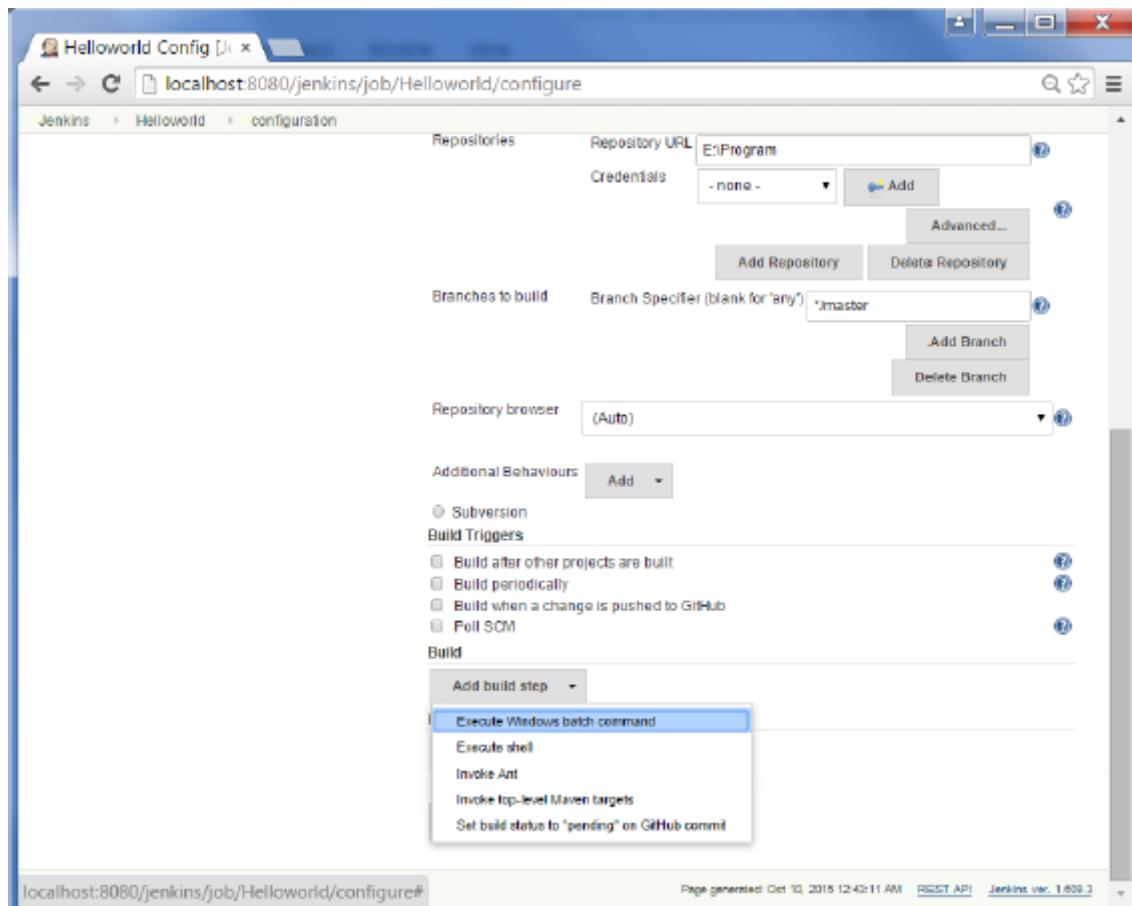
Step 4 – We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a

'HelloWorld.java' file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

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

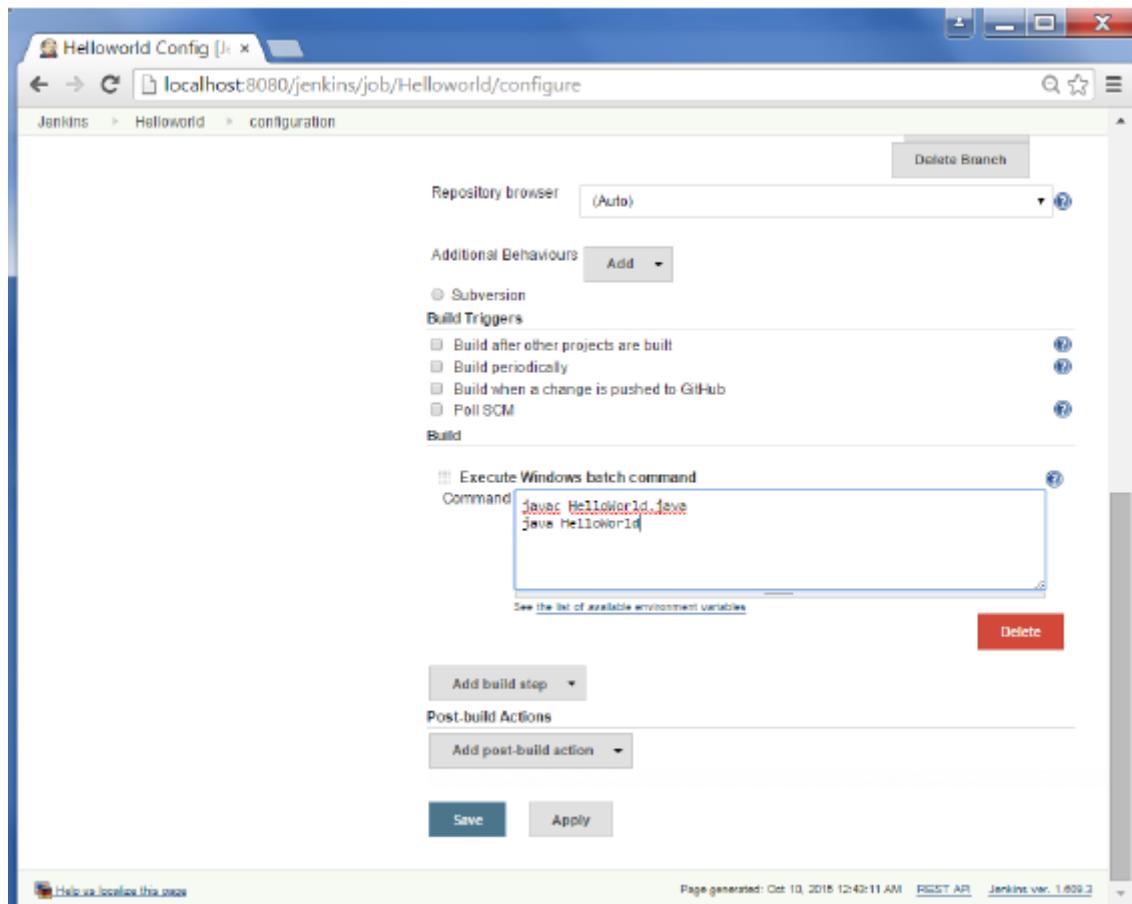


Step 5 – Now go to the Build section and click on Add build step → Execute Windows batch command



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

```
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 project page for 'Project Helloworld'. The 'Recent Changes' link is highlighted with a red box.

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

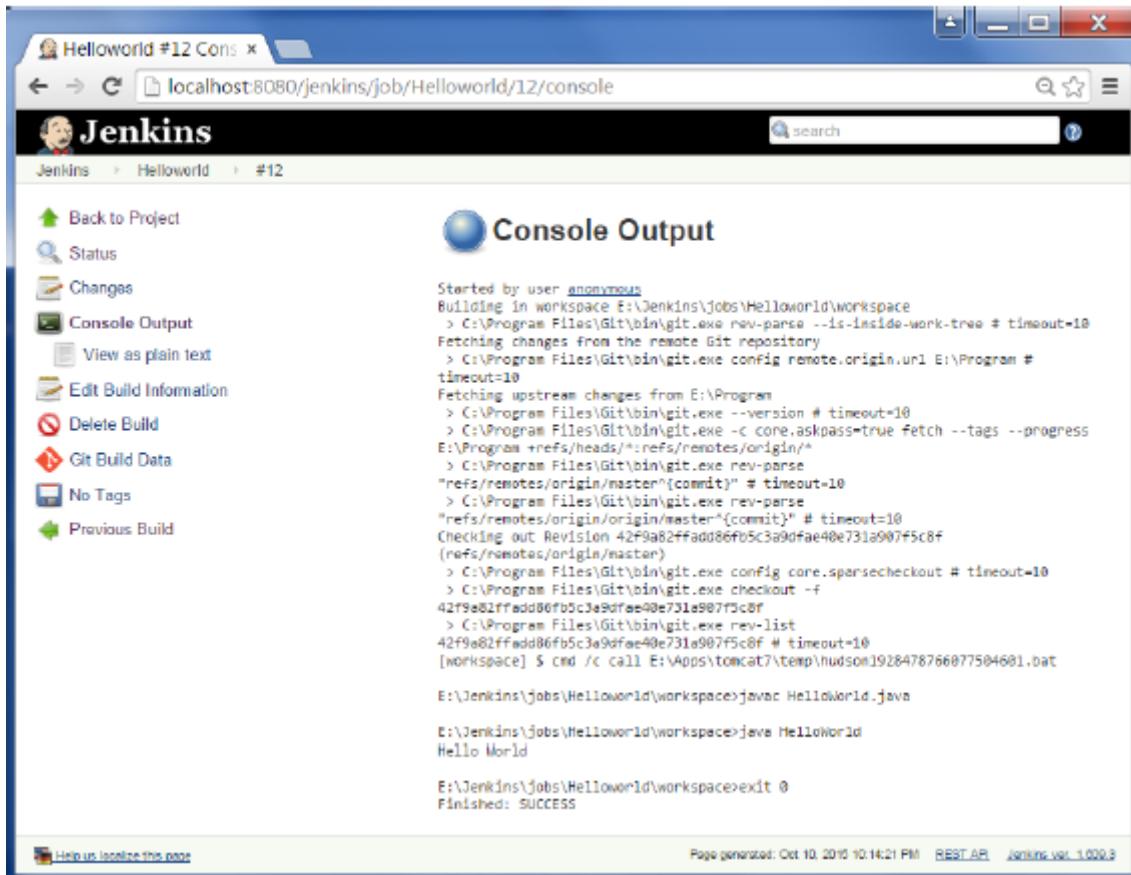
The screenshot shows the Jenkins interface for the 'Helloworld' project. At the top, there's a navigation bar with links like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. To the right of the navigation is a search bar and a 'Project Help' link. Below the navigation, the title 'Project Helloworld' is displayed. On the left, there's a sidebar with icons for 'Workspace' and 'Recent Changes'. In the center, under the heading 'Permalinks', there's a 'Build History' section. It shows one build entry: '#1 Oct 10, 2015 12:52 AM'. Below this entry are links for 'RSS for all' and 'RSS for failures'. At the bottom of the page, there's a footer with links for 'Help us localize this page', 'Page generated: Oct 10, 2015 12:51:53 AM', 'REST API', and 'Jenkins ver. 1.603.3'.

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

The screenshot shows the Jenkins interface for the 'Helloworld' project. The left sidebar contains links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main area is titled 'Project Helloworld'. It features a 'Workspace' section with a folder icon and a 'Recent Changes' section with a document icon. Below these are 'Permalinks' and build history. The build history table has one entry: 'Oct 10, 2015 12:52 AM'. At the bottom right are 'Add description' and 'Disable Project' buttons.

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

The screenshot shows the Jenkins interface for the first build of the 'Helloworld' project. The title is 'Build #1 (Oct 10, 2015 12:52:50 AM)'. It indicates the build started 4 min 40 sec ago and took 4.7 sec. The sidebar includes links: Back to Project, Status, Changes, Console Output (which is selected), Edit Build Information, Delete Build, Git Build Data, and No Tags. The main content area displays build details: 'No changes.', 'Started by anonymous user', and 'Revision: 42f0a82ffadd06fb5c3a9dfae40e731a907f5c0f * ref/remotes/origin/master'. At the bottom right are 'Add description' and 'Disable Project' buttons.



The screenshot shows the Jenkins interface for a build named "Helloworld #12". The left sidebar lists various build-related actions: Back to Project, Status, Changes, Console Output (selected), View as plain text, Edit Build Information, Delete Build, Git Build Data, No Tags, and Previous Build. The main content area is titled "Console Output" and displays the command-line log of the build process. The log shows the following steps:

```

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/master^{commit}" # timeout=10
Checking out Revision 42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
(refs/remotes/origin/master)
> C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -f
42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
> C:\Program Files\Git\bin\git.exe rev-list
42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f # timeout=10
[workspace] $ cmd /c call E:\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

```

At the bottom of the page, there are links for "Help us localize this page", "Page generated: Oct 10, 2010 10:14:21 PM", "RESTART", and "Jenkins ver. 1.009.3".

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

Jenkins - Unit Testing

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

xUnit Plugin - Jenkins

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

Dashboard > Jenkins > Plugins > xUnit Plugin

xUnit Plugin

Added by Gregory Boissinot, last edited by Gregory Boissinot on Oct 08, 2015. (view change)

Plugin Information

Plugin ID	xunit	Changes	In Latest Release Since Latest Release																									
Latest Release	1.98 (archives)																											
Latest Release Date	Oct 09, 2015																											
Required Core Dependencies	JUnit (version: 1.6)																											
Usage	<p>xunit - installations</p> <table border="1"> <caption>Estimated Data for xunit - installations</caption> <thead> <tr> <th>Month</th> <th>Installations</th> </tr> </thead> <tbody> <tr><td>Oct 14</td><td>12000</td></tr> <tr><td>Nov 14</td><td>11500</td></tr> <tr><td>Dec 14</td><td>11631</td></tr> <tr><td>Jan 15</td><td>12106</td></tr> <tr><td>Feb 15</td><td>12262</td></tr> <tr><td>Mar 15</td><td>12891</td></tr> <tr><td>Apr 15</td><td>12894</td></tr> <tr><td>May 15</td><td>12716</td></tr> <tr><td>Jun 15</td><td>13143</td></tr> <tr><td>Jul 15</td><td>13470</td></tr> <tr><td>Aug 15</td><td>13192</td></tr> <tr><td>Sep 15</td><td>13563</td></tr> </tbody> </table>	Month	Installations	Oct 14	12000	Nov 14	11500	Dec 14	11631	Jan 15	12106	Feb 15	12262	Mar 15	12891	Apr 15	12894	May 15	12716	Jun 15	13143	Jul 15	13470	Aug 15	13192	Sep 15	13563	Source Code Issue Tracking Pull Requests Maintainer(s) GitHub Open Issues Pull Requests Gregory Boissinot (id: gboissinot)
Month	Installations																											
Oct 14	12000																											
Nov 14	11500																											
Dec 14	11631																											
Jan 15	12106																											
Feb 15	12262																											
Mar 15	12891																											
Apr 15	12894																											
May 15	12716																											
Jun 15	13143																											
Jul 15	13470																											
Aug 15	13192																											
Sep 15	13563																											

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

CppUnit output

xUnit Plugin - Jenkins

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

Features

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

Supported tools

Embedded tools

- * JUnit itself
- * AUnit
- * MSTest (imported from MSTest Plugin)
- * NUnit (imported from NUnit Plugin)
- * UnitTest++
- * Boost Test Library
- * PHPUnit
- * Free Pascal Unit
- * CppUnit
- * MiniUnit
- * GoogleTest
- * EmbUnit
- * gtest/glib
- * QTestLib

Other plugins as an extension of the xUnit plugin:

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

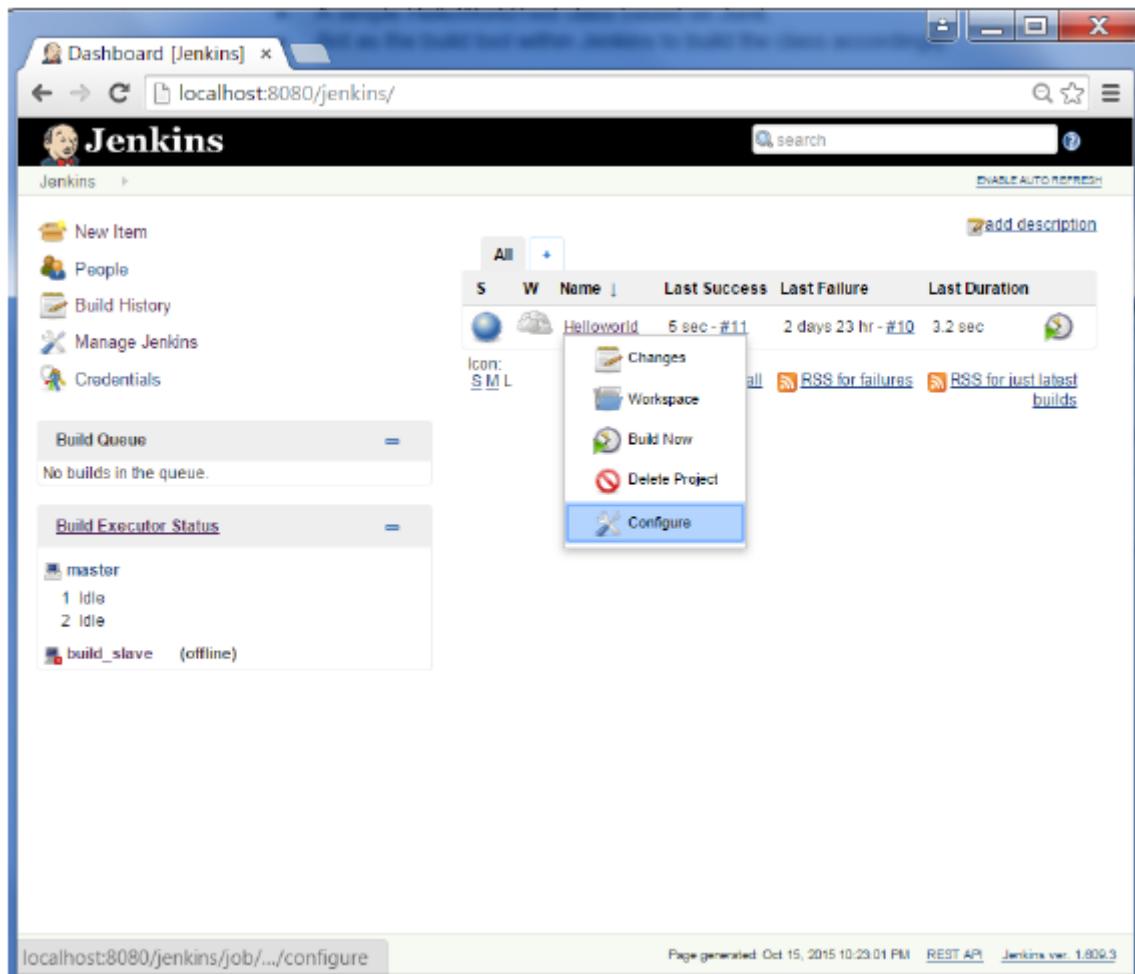
External contributions

Example of a JUnit Test in Jenkins

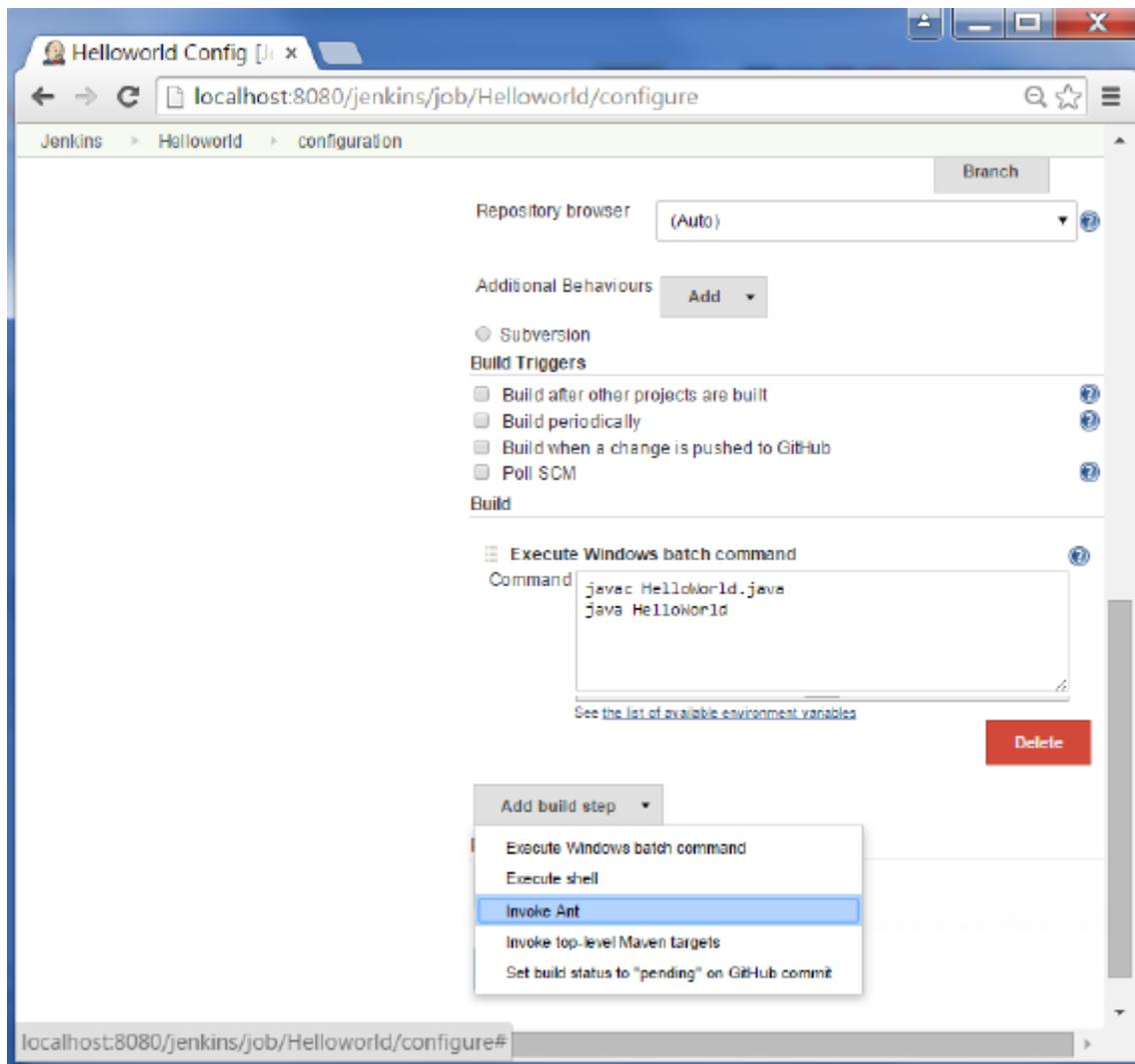
The following example will consider

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

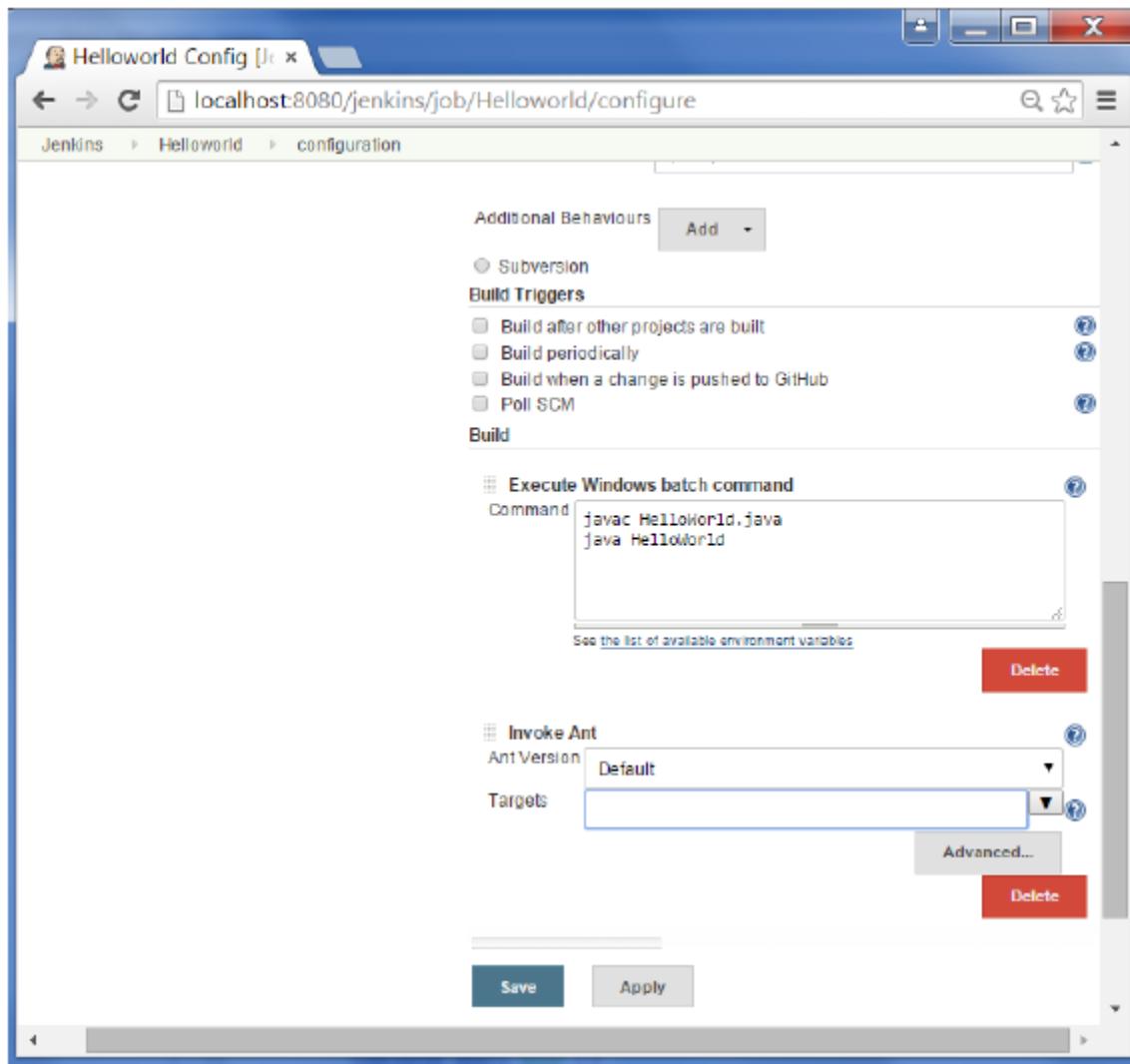
Step 1 – Go to the Jenkins dashboard and Click on the existing HelloWorld project and choose the Configure option



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



Step 3 – Click on the Advanced button.



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

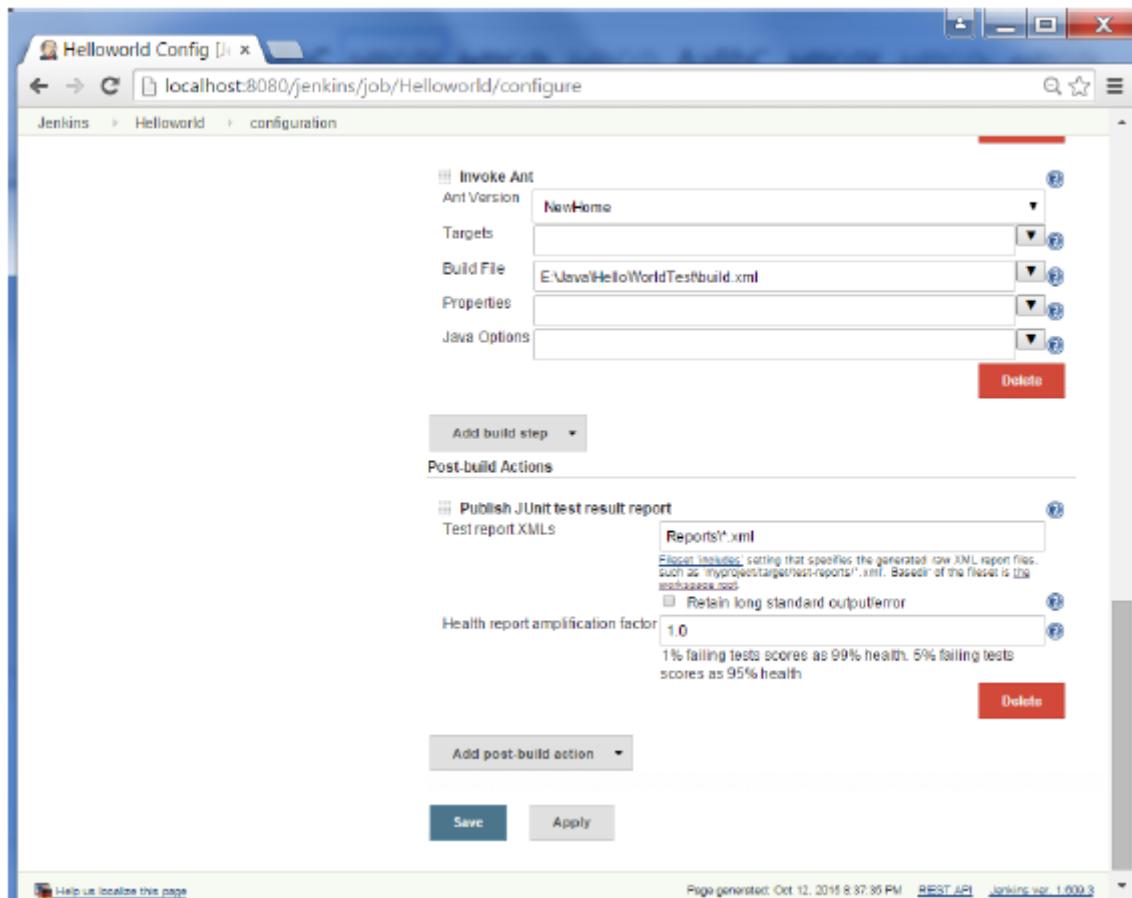
The screenshot shows the Jenkins configuration page for a job named "Helloworld". Under the "Build" section, there is a "Execute Windows batch command" step with the command "javac HelloWorld.java" and "java HelloWorld". Below it, there is an "Invoke Ant" step with "Ant Version" set to "NewHome", "Targets" set to "", "Build File" set to "E:\Java\HelloWorldTest\build.xml", and "Properties" and "Java Options" both empty. Under "Post-build Actions", there is a "Publish JUnit test result report" step with "Test report XMLs" set to "Report1.xml". At the bottom are "Save" and "Apply" buttons.

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

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

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

Once done, click the Save option at the end.



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

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

The screenshot shows the Jenkins interface for a job named 'Helloworld'. The main title is 'Build #4 (Oct 12, 2015) 8:33:16 PM'. It indicates the build started 3 days 1 hr ago and took 3.9 sec on master. On the left, there's a sidebar with links like 'Back to Project', 'Status', 'Changes', etc. The central area shows a summary of the build: 'No changes.', 'Started by anonymous user', 'Revision: 42f9a82ffadd86fb5c3a0d1ae40e731a807fc81', and 'Test Result (1 failure) HelloWorldTestCase.InitializationError'. At the bottom, it says 'Page generated: Oct 15, 2015 10:24:38 PM REST API 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 'Test Results' page for build #4. The title is 'Test Result' with '1 failures'. It lists 'All Failed Tests' with one entry: 'HelloWorldTestCase InitializationError'. Below that is 'All Tests' with a table showing the count of fails, skips, and passes. The table has columns for Package, Duration, Fail, Skip, Pass, Total, and Overall. The data shows 10 ms duration, 1 fail, 0 skip, 0 pass, 0 total, and 1 overall. At the bottom, it says 'Page generated: Oct 12, 2015 8:45:49 PM REST API Jenkins ver. 1.609.3'.

Jenkins - Automated Testing

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

Step 1 – Go to Manage Plugins.

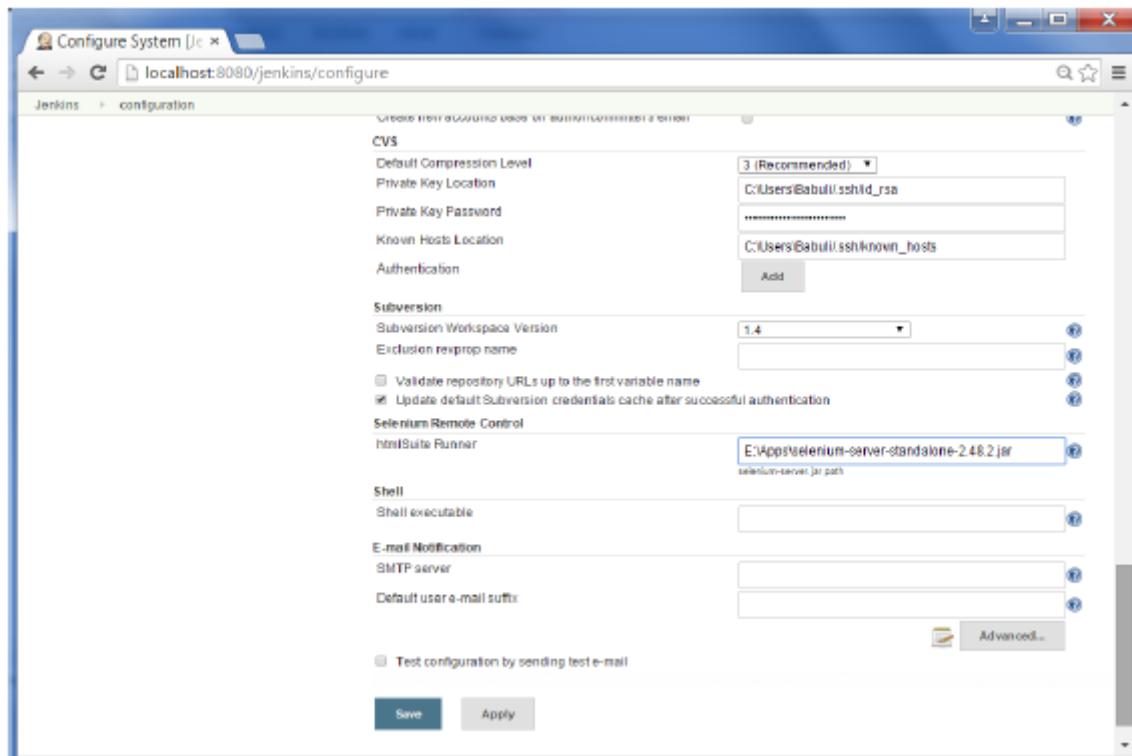
The screenshot shows the Jenkins Manage Jenkins page. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below that are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 Idle). The main area is titled 'Manage Jenkins' and contains several configuration options. One section, 'Manage Plugins', is highlighted with a red border. It lists the 'Hudson Selenium' plugin as 'updates available'. Other listed items include 'Configure System', 'Configure Global Security', 'Reload Configuration from Disk', 'System Information', 'System Log', 'Load Statistics', 'Jenkins CLI', 'Script Console', 'Manage Nodes', 'Manage Credentials', and 'About Jenkins'. A warning message at the top right says: '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 Tomcat18n for more details.' Buttons for 'Setup Security' and 'Dismiss' are also present.

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

Name	Version
Selenium Auto Exec Server(AES) plugin	0.5
Hudson Seleniumhq plugin	0.4
Selenium HTML report	0.94
TestingBot plugin	1.11
TestLink Plugin	3.10
Nervana Plugin for Jenkins	1.02.08
Sauce OnDemand plugin	1.141
Selenium Builder plugin	1.14
SeleniumRC plugin	1.12

Step 3 – Go to Configure system.

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



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

Click on the download for the Selenium standalone server.

The screenshot shows the SeleniumHQ website's 'Downloads' page. The 'Selenium Standalone Server' section is the focal point, with a link to 'Download version 2.48.2'. Other sections visible include 'Selenium Downloads' (Latest Releases, Previous Releases, Source Code, Maven Information), 'Donate to Selenium' (with PayPal and a 'Donate' button), 'through sponsorship' (with a note about sponsoring the project), 'Selenium Sponsors' (listing 'See who supports the Selenium project'), and 'BrowserStack' (with a note about language bindings). The URL in the address bar is www.seleniumhq.org/download/.

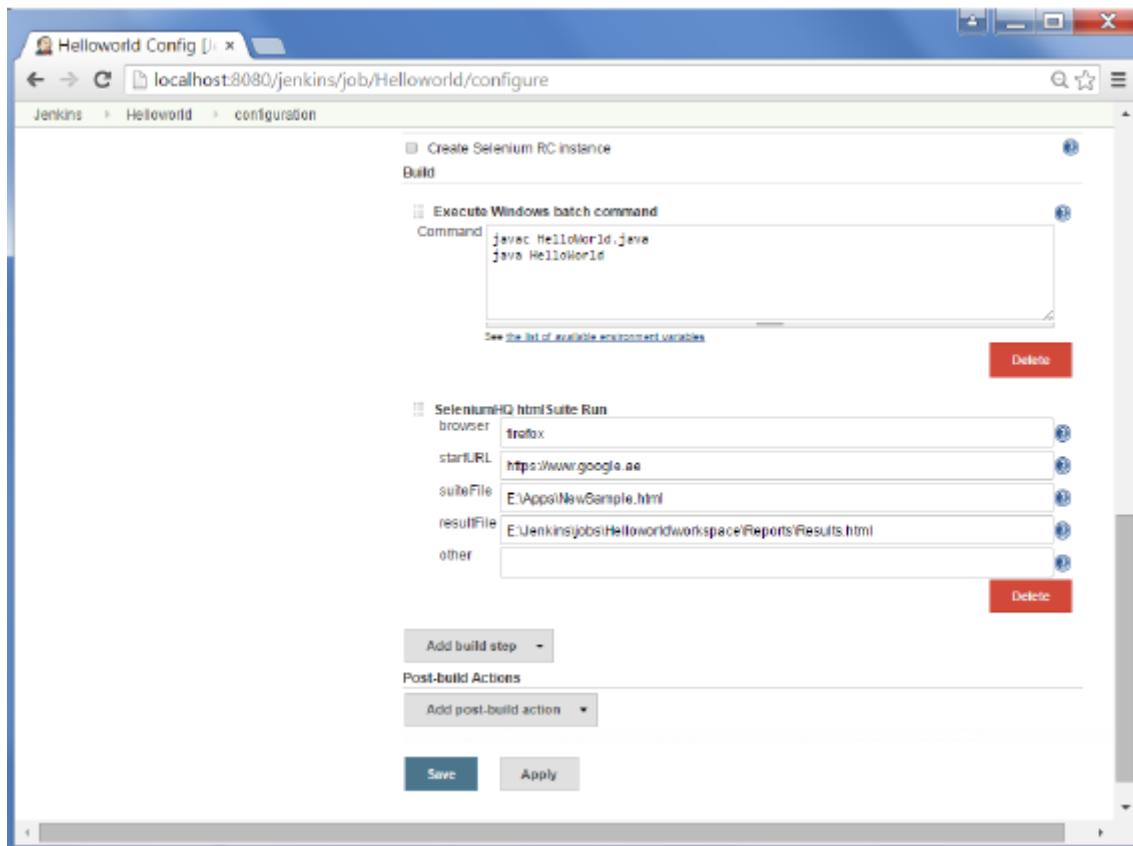
Step 5 – Go back to your dashboard and click on the Configure option for the HelloWorld project.

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. 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, 2 idle). The main area displays a table of jobs. One job, 'Helloworld', is selected, showing its status: Last Success was 23 hr - #12, Last Failure was 23 hr - #10, and Last Duration was 3.7 sec. A context menu is open over the 'Helloworld' row, with 'Configure' highlighted in blue. The URL in the address bar is localhost:8080/jenkins/job/Helloworld/configure.

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

The screenshot shows the configuration page for the 'Helloworld' job at localhost:8080/jenkins/job/Helloworld/configure. The page includes sections for Repository browser (Auto), Additional Behaviours (Subversion selected), Build Triggers (Build after other projects are built, Build periodically, Build when a change is pushed to GitHub, Poll SCM), and Build (Execute Windows batch command: Command: `javac HelloWorld.java`, `java HelloWorld`). At the bottom, there is a 'Delete' button and a 'Add build step' dropdown menu. The 'SeleniumHQ htmlSuite Run' option is highlighted in blue. The URL in the address bar is localhost:8080/jenkins/job/Helloworld/configure#.

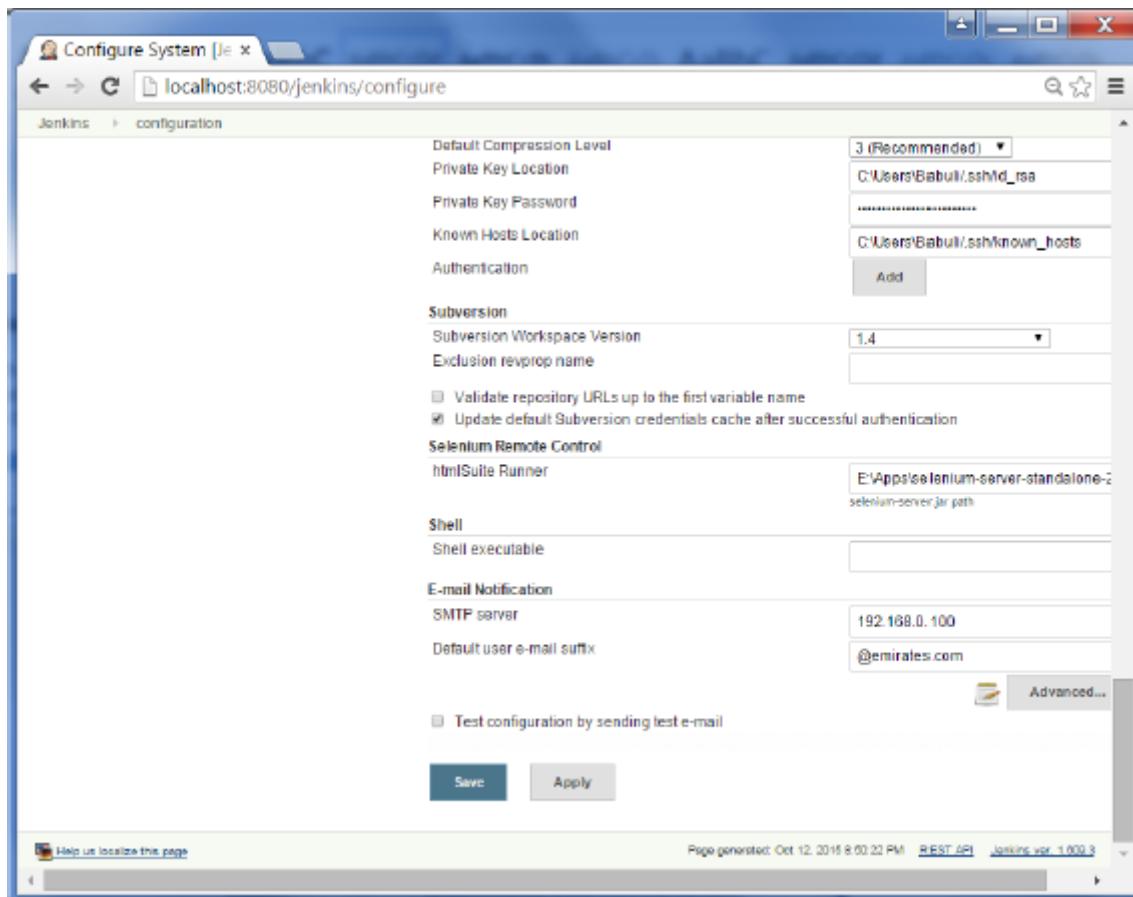
Step 7 – Add the necessary details for the selenium test. Here the suiteFile is the TestSuite generated by using the Selenium IDE. Click on Save and execute a build. Now the post build will launch the selenium driver, and execute the html test.



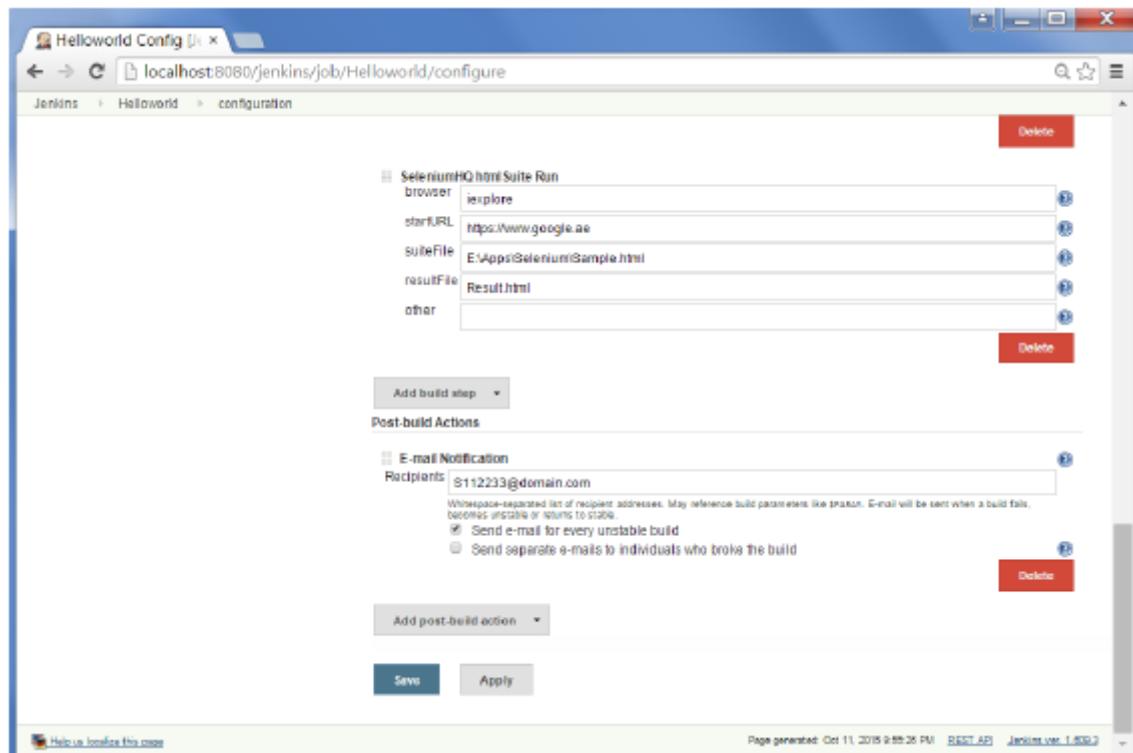
Jenkins - Notification

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

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



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



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

The screenshot shows the Jenkins configuration interface for the 'Helloworld' job. In the left sidebar, under 'Job Notifications', there is a 'Notification Endpoints' section. A red box highlights the 'Add Endpoint' button. Below it, a configuration card is open for a new endpoint:

- Format:** JSON
- Protocol:** HTTP
- Event:** All Events
- URL:** http://tcbmem301xe4507h
- Timeout:** 30000
- Log:** 0

At the bottom of the configuration card, there are 'Save' and 'Apply' buttons.

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.

The screenshot shows the Jenkins job configuration page for 'Helloworld'. In the 'Build' section, there is a 'Command' step containing the command 'java -jar HelloWorld.jar'. Below this, a context menu is open over the 'Publish JUnit test result report' step, with the option 'Publish JUnit test result report' highlighted. Other options in the menu include 'Aggregate downstream test results', 'Archive the artifacts', 'Build other projects', 'Publish JavaDoc', 'Publish Selenium Report', 'Record fingerprints of files to track usage', 'Git Publisher', 'E-mail Notification', and 'Set build status on GitHub commit'. At the bottom of the configuration page, there are 'Save' and 'Apply' buttons.

Jenkins - Code Analysis

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

The screenshot shows the Jenkins Static Code Analysis Plugins page. On the left, there is a sidebar with links like Home, Mailing lists, Source code, Bugtracker, Security, Advisories, Events, Donation, Commercial Support, and Wiki Site Map. The main content area is titled 'Static Code Analysis Plug-ins' and shows the 'analysis-core' plugin. It provides information such as the Plugin ID (analysis-core), Latest Release (1.74), Latest Release Date (Sep 07, 2015), Required Core Dependencies (ant, token-macro, maven-plugin, matrix-project, dashboard-view), and Source Code Issue Tracking Pull Requests Maintainer(s) (GitHub, Open Issues, Pull Requests, Ulli Hafner). Below this, there is a chart titled 'analysis-core - installations' showing the number of installations over time, and a table of installations 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.

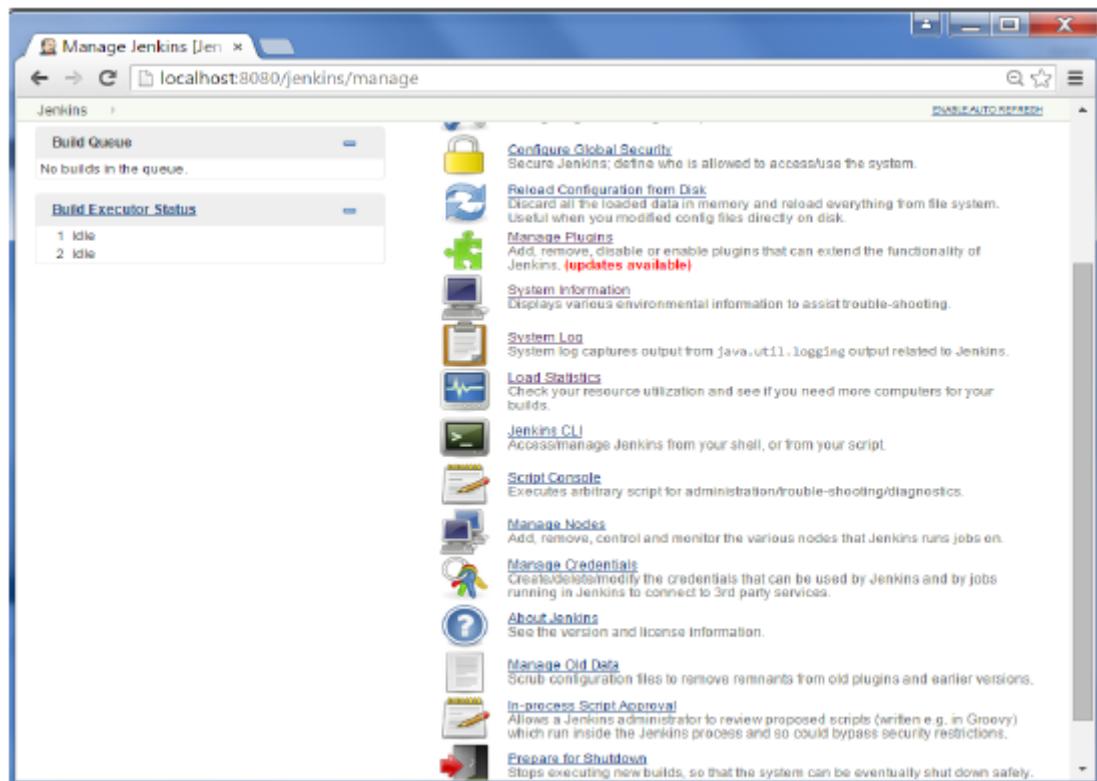
Sometimes you might also need several different environments to test your builds. In this case using a slave to represent each of your required environments is almost a must.

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.



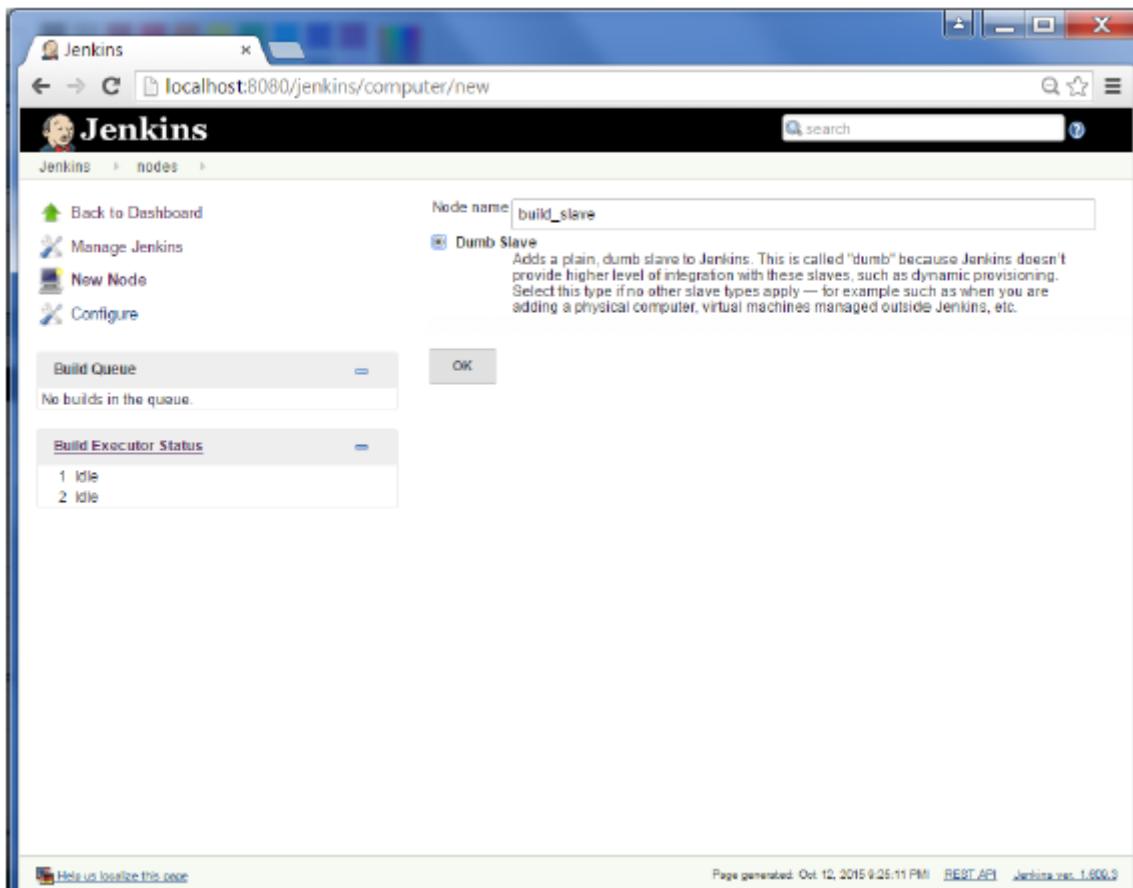
Step 2 – Click on New Node

The screenshot shows the Jenkins Nodes page at localhost:8080/jenkins/computer/. The left sidebar includes 'Back to Dashboard', 'Manage Jenkins', 'New Node' (which is highlighted in blue), and 'Configure'. The main area displays a table of nodes:

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Fre...
1	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	...

At the bottom, there is a 'Refresh status' button and footer links for Help, REST API, and Jenkins version.

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



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

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

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp S
	build_slave		N/A	N/A	N/A	
	master	Windows 7 (x86)	In sync	229.89 GB	12.13 GB	229.8
		Data obtained	3 ms	2 ms	1 ms	11 min

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]" and the address bar shows "localhost:8080/jenkins/pluginManager/available". The main area is titled "Available" and lists several plugins. The "Deploy to container Plugin" is highlighted with a blue selection bar at the bottom. Below the list are three buttons: "Install without restart", "Download now and install after restart", and "Update information".

	Name	Version
Artifact Deployer Plugin	0.33	
AWS Lambda Plugin	0.3.1	
AWS Elastic Beanstalk Deployment Plugin	0.0.3	
Capistrano Plugin	0.1.0	
AWS CodeDeploy Plugin for Jenkins	1.7	
CRX Content Package Deployer Plugin	1.3.2	
Deploy to container Plugin	1.10	
Deploy to WebSphere container Plugin	1.0	
Xebialabs XL Deploy Plugin	5.0.0	

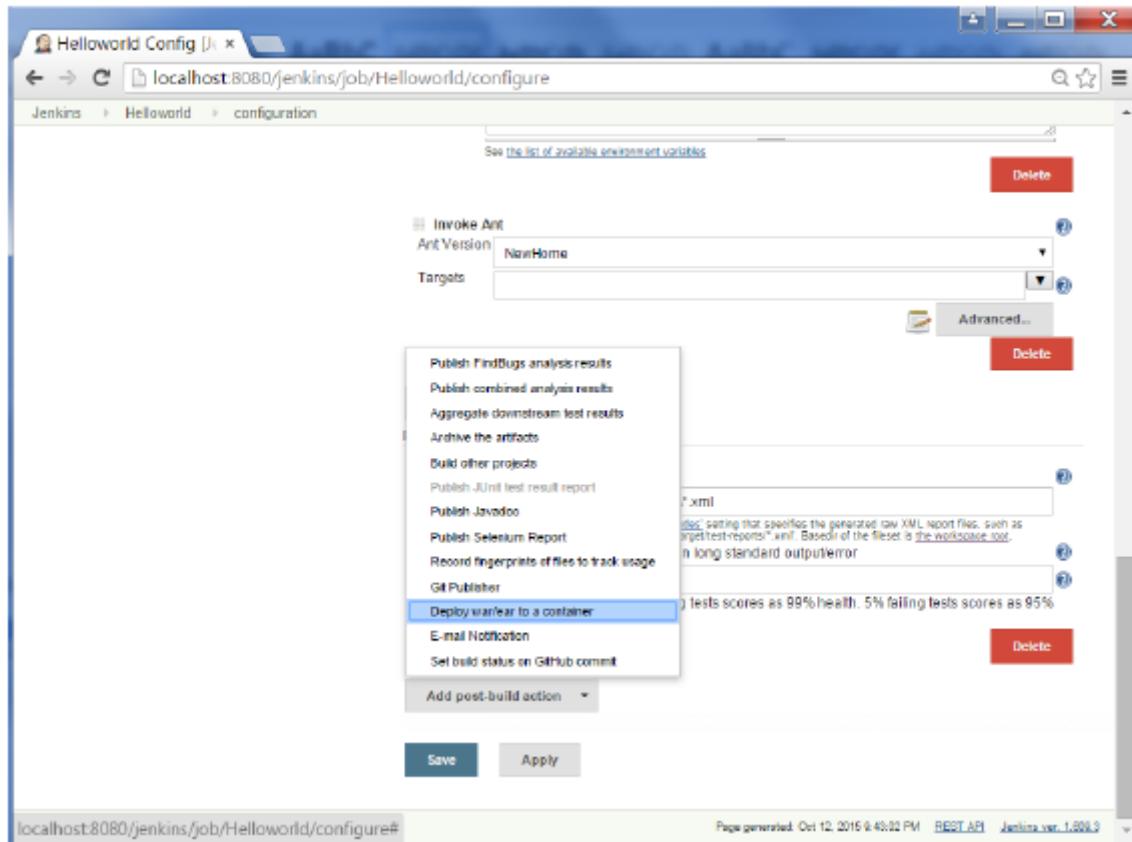
This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build.

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

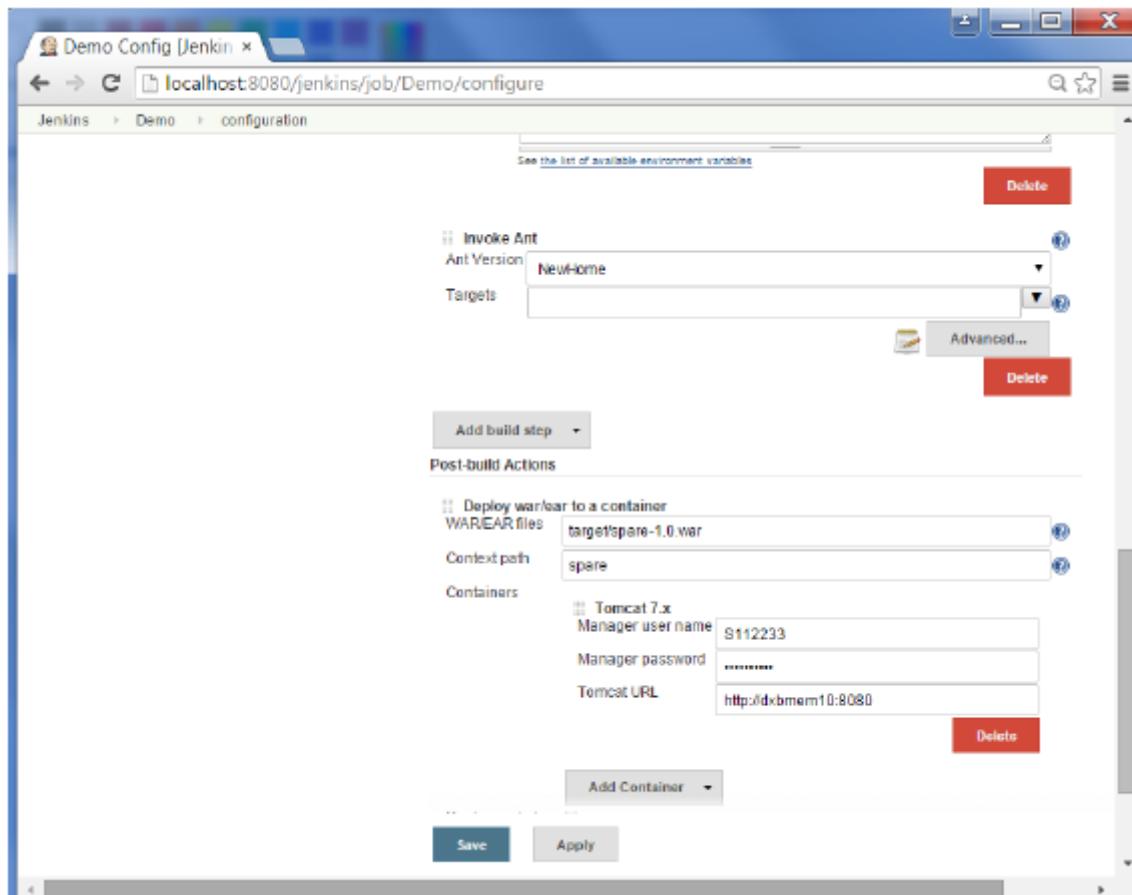
JBoss 3.x/4.x

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.



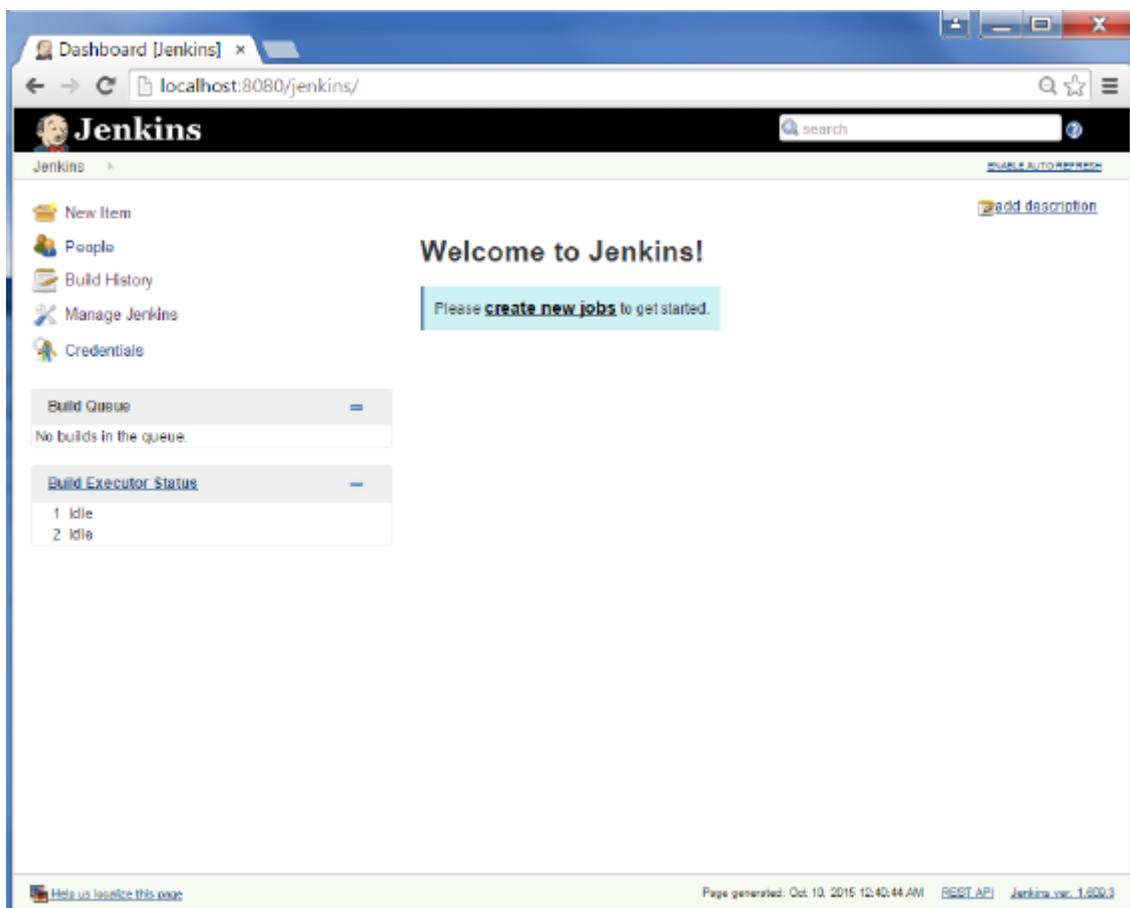
Jenkins - Metrics & Trends

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.

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

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

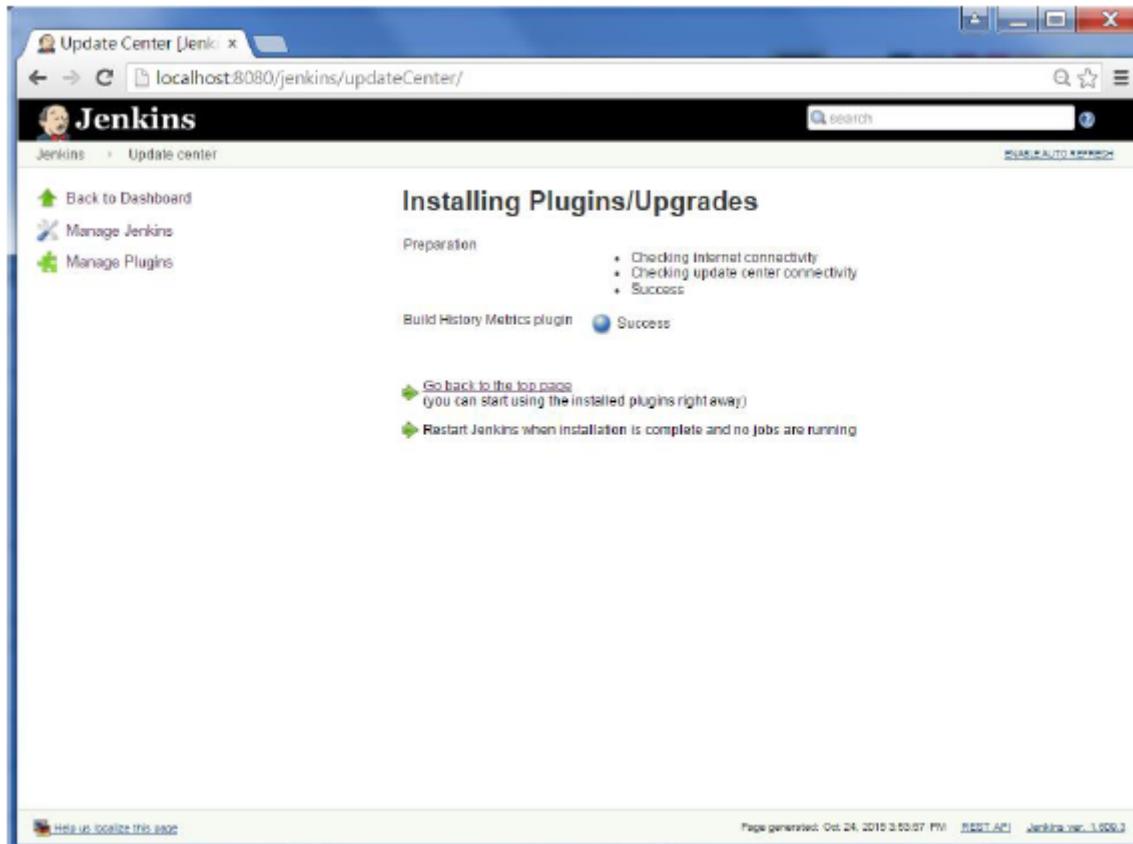
Step 3 – Go to the Available tab and search for the plugin ‘Build History Metrics plugin’ and choose to ‘install without restart’.

The screenshot shows the Jenkins Plugin Manager on the 'Available' tab. A search bar at the top right is set to 'build-history-metrics-plugin'. The results table has columns for Name, Version, and Install. One result is shown:

Name	Version
Build History Metrics plugin Provides build metrics that encompass the history of all the runs	1.2

Below the table are two buttons: 'Install without restart' (highlighted in blue) and 'Download now and install after restart'. At the bottom right, it says 'Update information obtained: 2 mi'.

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



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

The screenshot shows the Jenkins Project Helloworld dashboard. On the left, there's a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. Below that is the Build History table:

	trend
	#12 Oct 24, 2015 3:57 PM
	#11 Oct 15, 2015 10:22 PM
	#10 Oct 12, 2015 10:50 PM
	#9 Oct 12, 2015 10:48 PM
	#8 Oct 12, 2015 10:40 PM
	#7 Oct 12, 2015 10:39 PM
	#6 Oct 12, 2015 10:31 PM
	#5 Oct 12, 2015 10:29 PM
	#4 Oct 12, 2015 8:33 PM
	#3 Oct 11, 2015 11:04 PM
	#2 Oct 11, 2015 10:51 PM
	#1 Oct 11, 2015 10:48 PM

At the bottom of the history table are two RSS feed links: RSS for all and RSS for failures.

On the right side, there are three tables for MTTR, MTTF, and Standard Deviation:

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

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

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

Below the tables is a section titled "Permalinks" with a list of links:

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

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

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

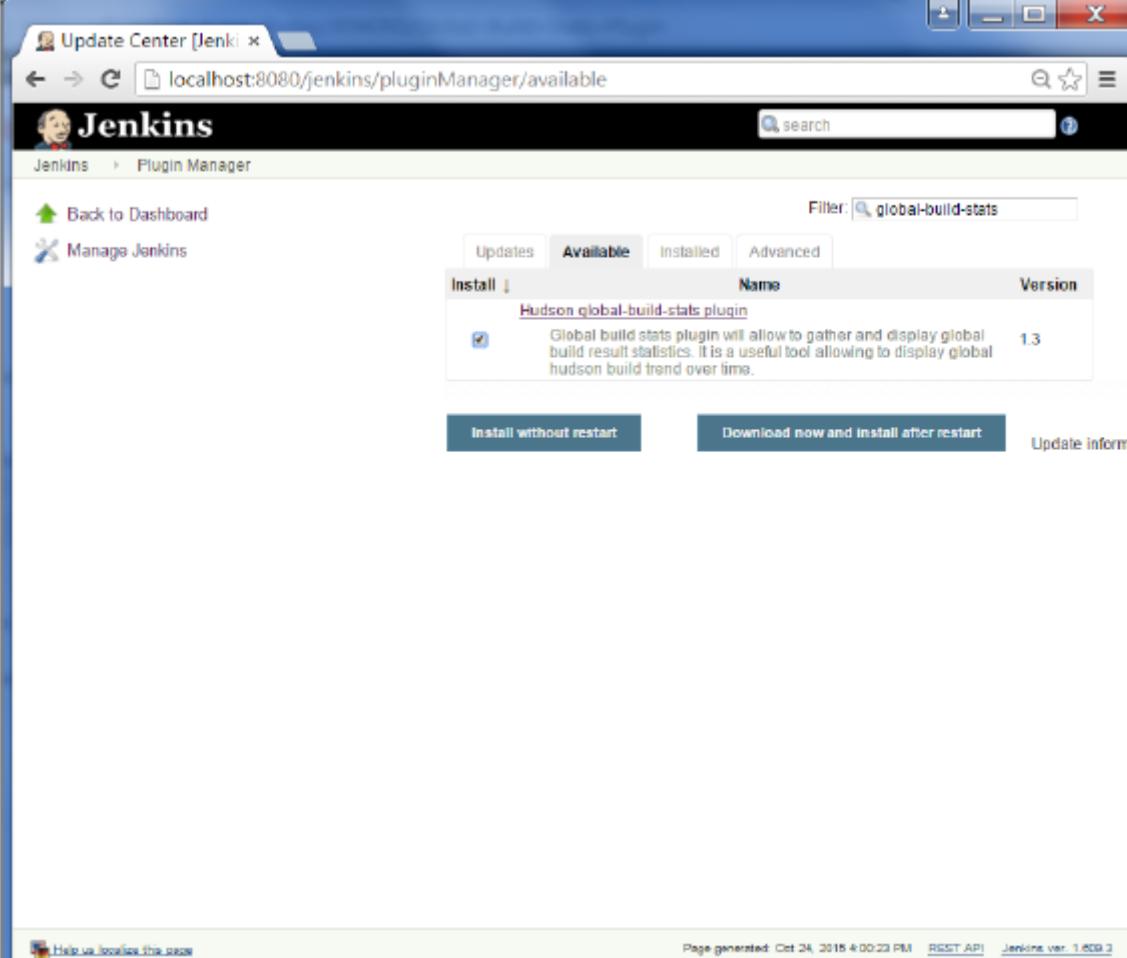
Step 1 – Go to the Jenkins dashboard and click on Manage Jenkins

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The main header says "Dashboard [Jenkins]". The left sidebar includes links for "New Item", "People", "Build History", "Manage Jenkins", and "Credentials". The central area features a large "Welcome to Jenkins!" message with a sub-instruction "Please [create new jobs](#) to get started." Below this are two summary boxes: "Build Queue" (No builds in the queue) and "Build Executor Status" (1 Idle, 2 Idle). At the bottom, there's a link to "Help us license this page" and footer text indicating the page was generated on Oct 10, 2015, at 12:40:44 AM.

Step 2 – Go to the Manage Plugins option

The screenshot shows the "Manage Jenkins" screen at localhost:8080/jenkins/manage. The left sidebar is identical to the dashboard. The main content area is titled "Manage Jenkins" and lists several configuration options: "Configure System", "Configure Global Security", "Reload Configuration from Disk", "Manage Plugins" (which is highlighted in red), "System Information", "System Log", "Load Statistics", "Jenkins CLI", "Script Console", "Manage Nodes", "Manage Credentials", and "About Jenkins". 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 i18n](#) for more details." There are "Setup Security" and "Dismiss" buttons next to the warning.

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 tabs for "Updates", "Available" (which is selected), "Installed", and "Advanced". A search bar at the top right contains the text "global-build-stats". Below the tabs is a table with columns "Name" and "Version". One row in the table is highlighted, showing the "Hudson global-build-stats plugin" with version 1.3. A description below the table states: "Global build stats plugin will allow to gather and display global build result statistics. It is a useful tool allowing to display global hudson build trend over time." At the bottom of the table are three buttons: "Install without restart" (highlighted in blue), "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'. Below this is a main title 'Installing Plugins/Upgrades'. Underneath the title, there's a section titled 'Preparation' with 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. At the bottom of the page, there are two links: 'Go back to the top page (you can start using the installed plugins right away)' and 'Restart Jenkins when installation is complete and no jobs are running'. The footer of the page includes a link to 'Help us localize this page', the page generation date 'Page generated: Oct 24, 2015 4:01:04 PM', and the Jenkins version 'Jenkins ver. 1.802.3'.

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 a 'Back to Dashboard' link. To the right is a search bar and an 'ENABLE AUTO REFRESH' checkbox. The main title is 'Global Build Stats' with a small icon. Below it is a 'Statistics' section stating 'No chart configured for the moment ... [Create a new chart configuration](#)'. Underneath is a 'Build Results retention strategies' section with three options: 'Automatically discard results older than 30 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. A 'Data initialization' section follows, containing the text 'Click button below to initialize build statistics. Job results read will be merged with already recorded job results.' and a 'Initialize stats' button. At the bottom left is a 'Help us localize this page' link, and at the bottom right are links for 'Page generated: Oct 24, 2015 4:03:17 PM', 'REST API', and 'Jenkins ver. 1.809.3'.

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

The screenshot shows the Jenkins Global Build Stats page at localhost:8080/jenkins/plugin/global-build-stats/#Initialize. The left sidebar has links for Back to Dashboard, Create new chart, Manage retention strategies, and Data Initialization. The main content area is titled "Global Build Stats". Under "Statistics", it says "No chart configured for the moment ... [Create a new chart configuration](#)". Under "Build Results retention strategies", there are 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. Under "Data initialization", it says "Click button below to initialize build statistics. Job results read will be merged with already recorded job results." followed by "Data successfully initialized!". A "Initialize stats" button is shown. At the bottom, there's a link to "Help us localize this page" and footer text "Page generated: Oct 24, 2015 4:03:17 PM REST API Jenkins ver. 1.603.3".

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

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

Adding new chart

Title : Demo

Chart Width * Height : 800 * 600

Chart time scale : Daily

Chart time length : 30 days

Filters :

- Job filtering : ALL Jobs Job name regex: []
- Node filtering : ALL Nodes Master only Node name regex: []
- Launcher filtering : ALL Users System only Username regex: []
- Statuses taken into account: Success Failures Unstables Aborted Not Build

Elements displayed on chart :

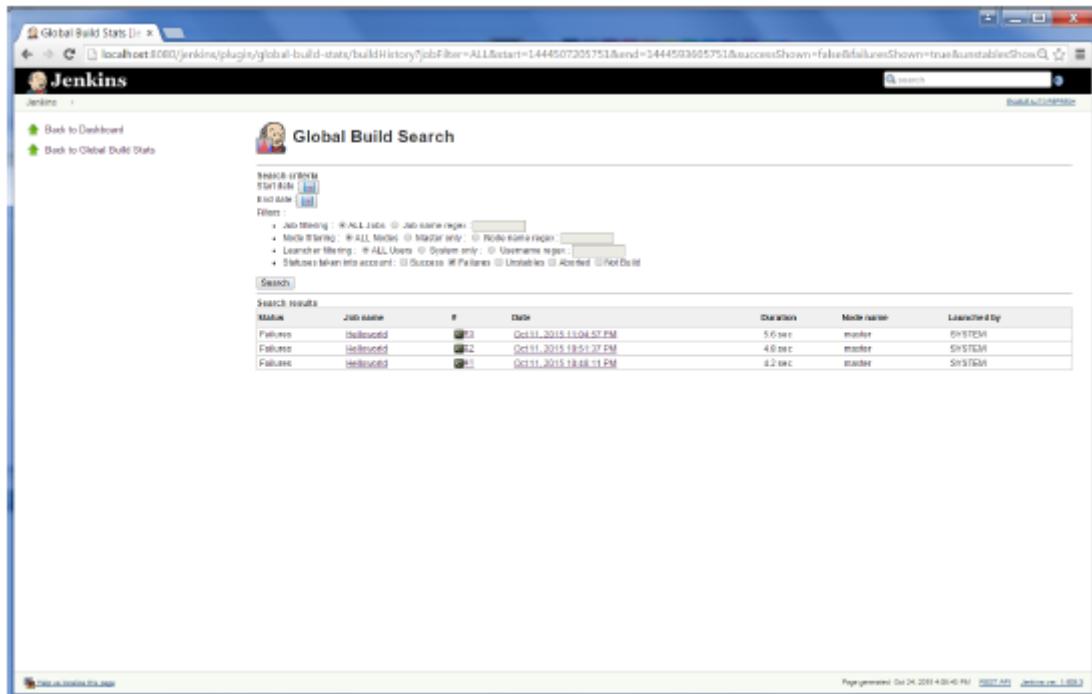
- Build statuses with Y Axis type : Count
- Total build time
- Average build time

Buttons: Overview, Create new chart, Cancel

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.



Jenkins - Server Maintenance

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

URL Options

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

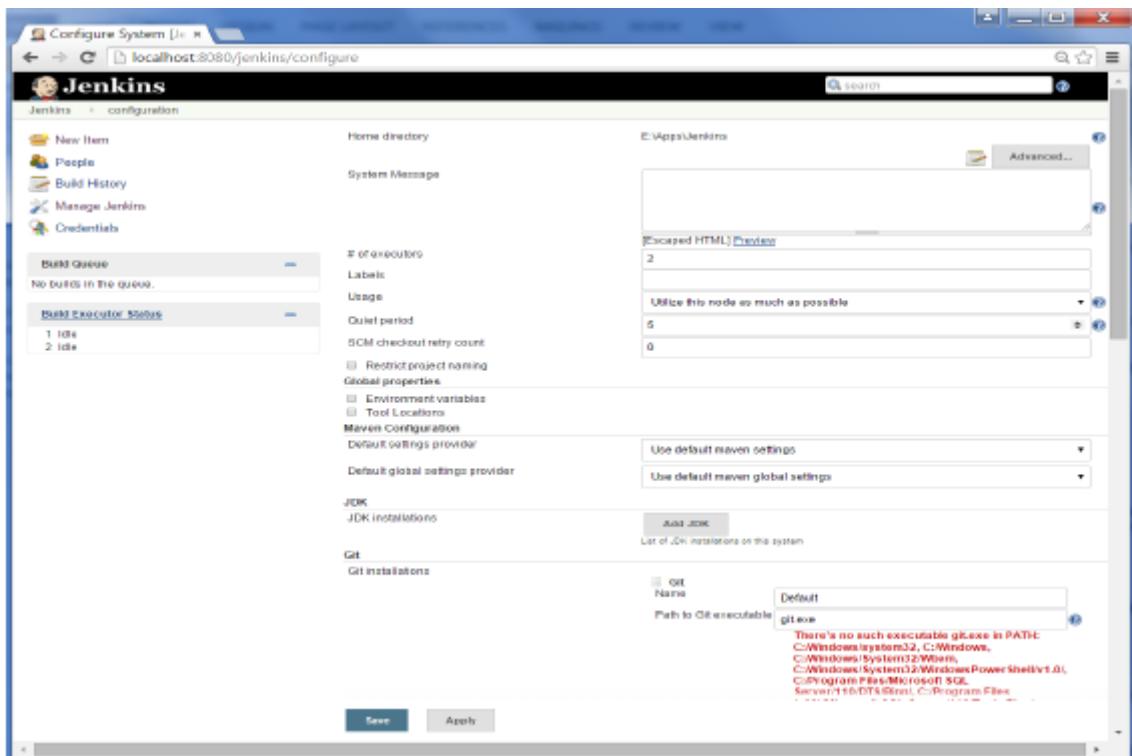
http://localhost:8080/jenkins/exit – shutdown jenkins

http://localhost:8080/jenkins/restart – restart jenkins

http://localhost:8080/jenkins/reload – to reload the configuration

Backup Jenkins Home

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

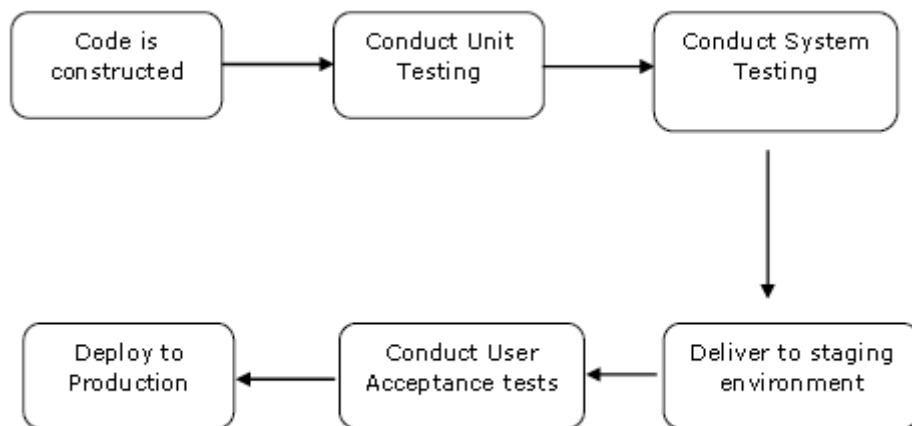


Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined and doing continuous builds, always ensure that Jenkins is setup on a drive that has enough hard disk space. If you hard disk runs out of space, then all builds on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

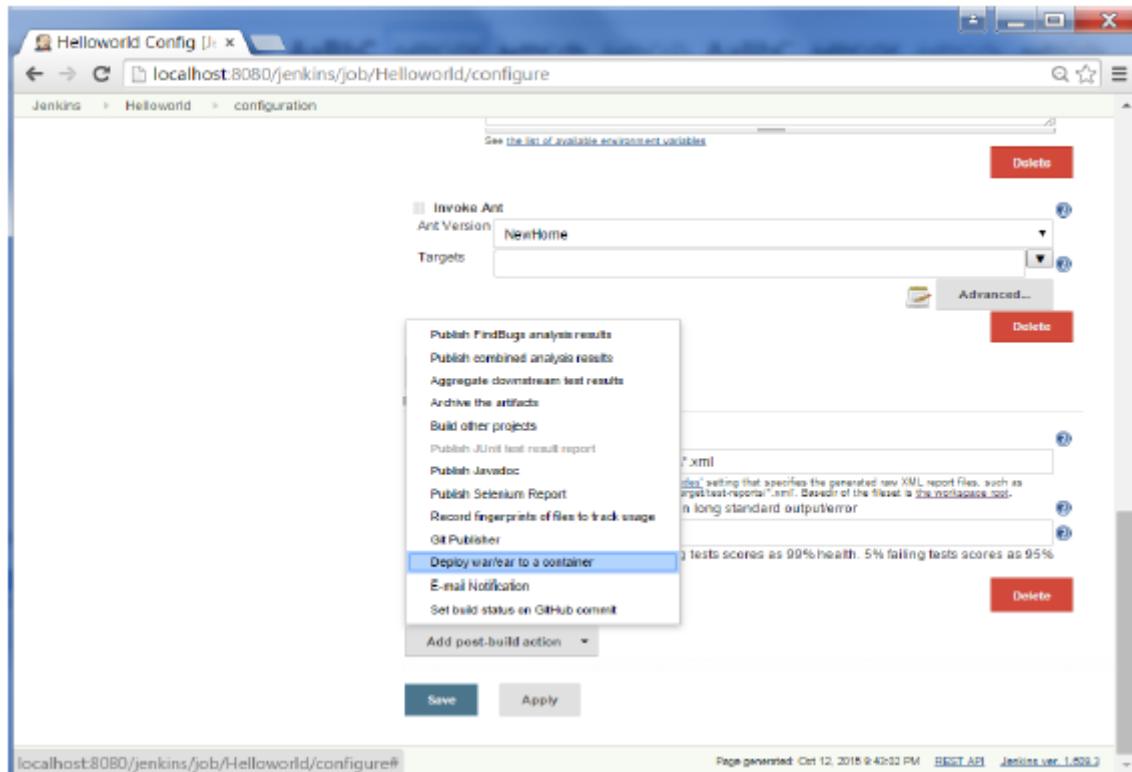
Another best practice is to write cron jobs or maintenance tasks that can carry out clean-up operations to avoid the disk where Jenkins is setup from becoming full.

Jenkins - Continuous Deployment

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



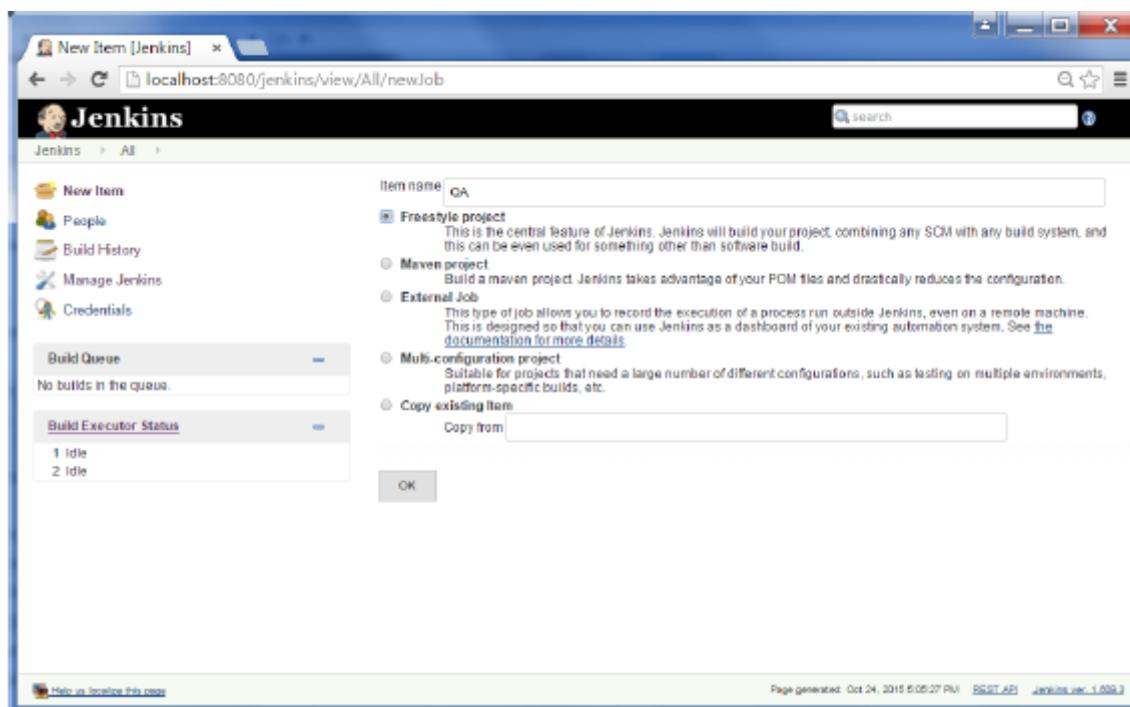
The main part of Continuous deployment is to ensure that the entire process which is shown above is automated. Jenkins achieves all of this via various plugins, one of them being the "Deploy to container Plugin" which was seen in the earlier lessons.



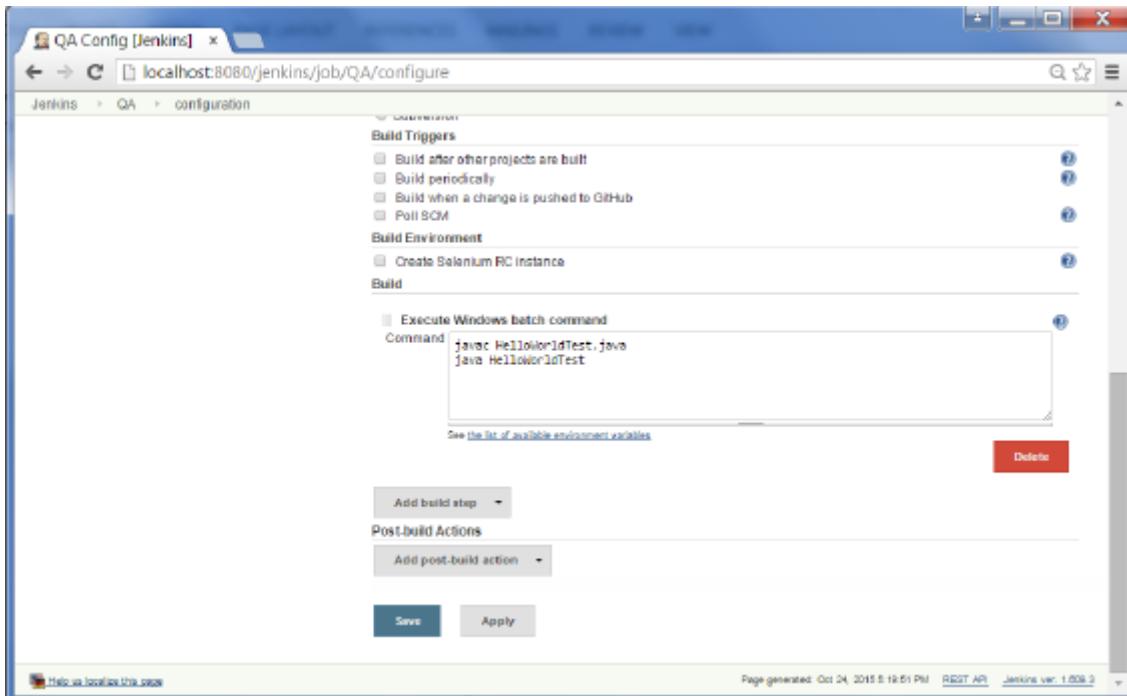
There are plugins available which can actually give you a graphical representation of the Continuous deployment process. But first lets create another project in Jenkins, so that we can see best how this works.

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

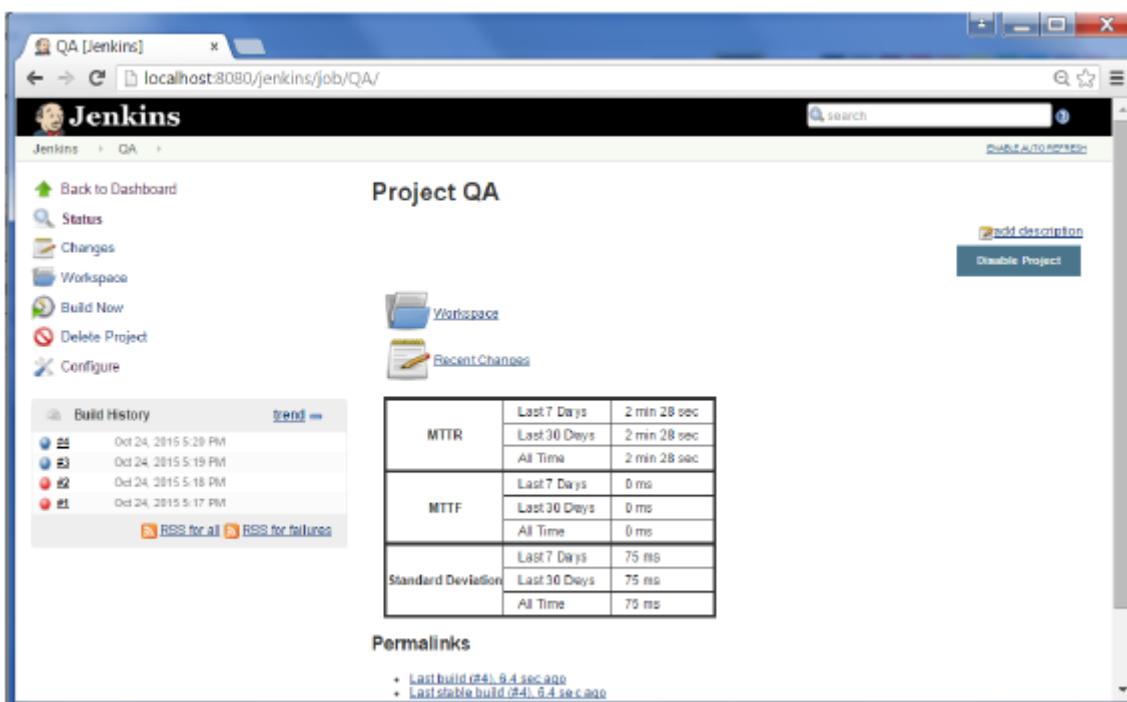
Step 1 – Go to the Jenkins dashboard and click on New Item. Choose a ‘Freestyle project’ and enter the project name as ‘QA’. Click on the Ok button to create the project.



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



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



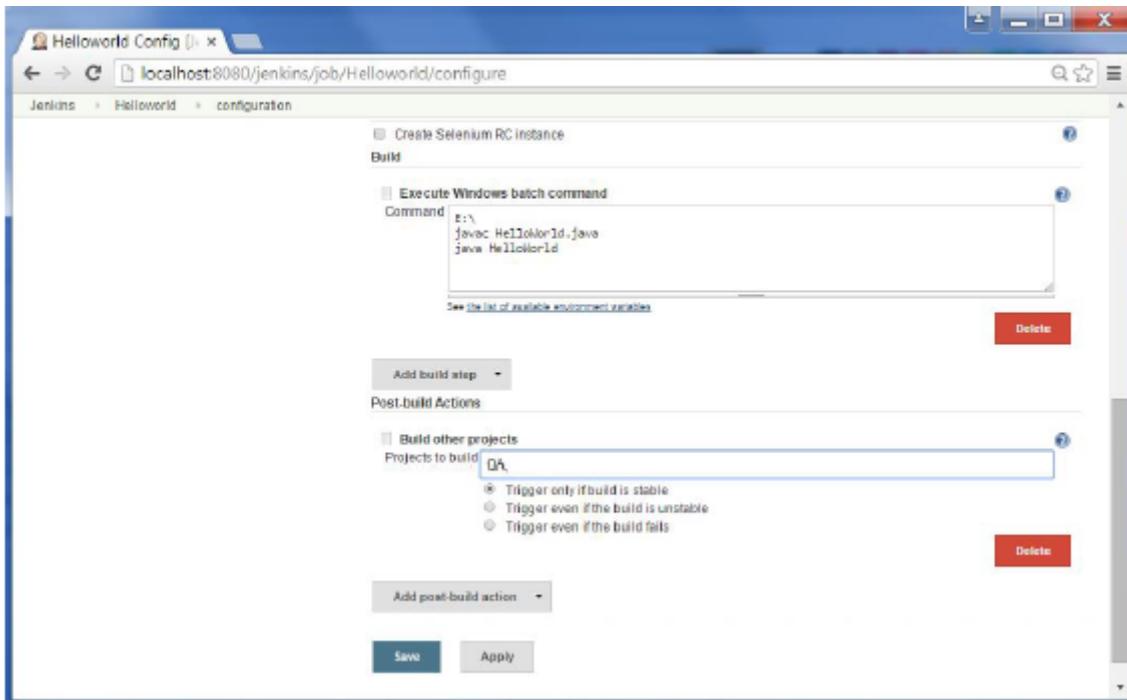
Step 3 – Now go to your Helloworld project and click on the Configure option

The screenshot shows the Jenkins dashboard with the 'Helloworld' job listed. A context menu is open over the job entry, with the 'Configure' option highlighted.

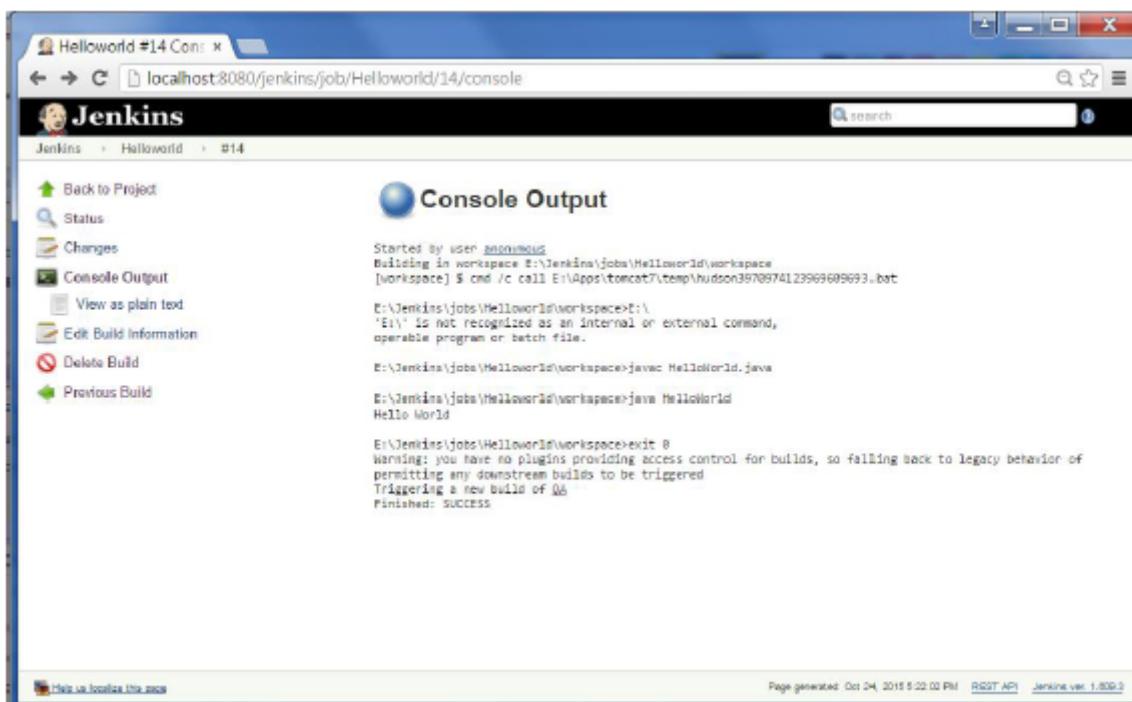
Step 4 – In the project configuration, choose the ‘Add post-build action’ and choose ‘Build other projects’

The screenshot shows the 'Helloworld' configuration page. The 'Build other projects' option is selected in the 'Post-build Actions' dropdown.

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



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



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

The screenshot shows the Jenkins Update Center interface. The 'available' tab is selected. A list of available plugins is displayed, with the 'Delivery Pipeline Plugin' checked for installation. The plugin details show it's version 0.9.7 and provides a full-screen view for information radiators.

Name	Version
ontrack Jenkins plug-in	2.15.0
Fail The Build Plugin	1.0
Runscope plugin	1.44
Build Graph View Plugin	1.1.1
Deployment Sphere	0.1.105
Jenkins plugin to have a bird's eye view of your continuous deployment pipeline.	
CloudBees Docker Hub Notification	1.0.2
Seed Jenkins plug-in	0.17.0
Delivery Pipeline Plugin	0.9.7

Buttons at the bottom include 'Install without restart', 'Download now and install after restart', and 'Update information obtained: 1 hr 36 min ago'.

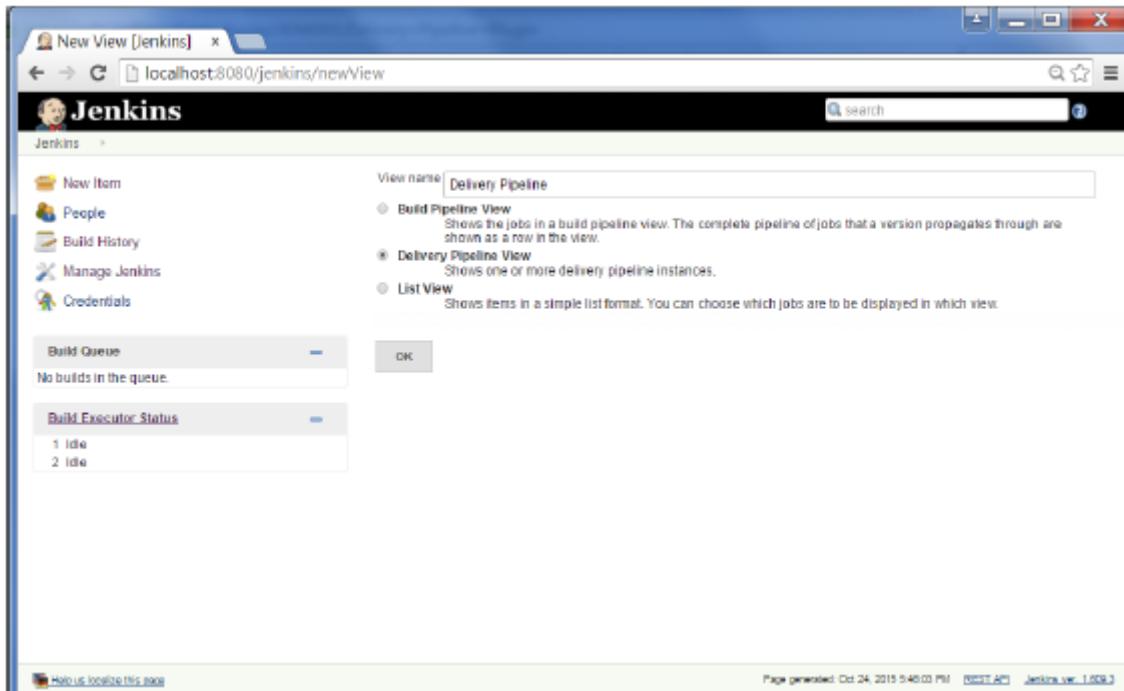
Step 8 – To see the Delivery pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the ‘All’ Tab.

The screenshot shows the Jenkins Dashboard. The 'All' tab is selected. Two build jobs are listed: 'HelloWorld' and 'QA'. Each job has a status icon, name, last success time, last failure time, and last duration. Below the table, there are links for 'Legend', 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'.

S	W	Name	Last Success	Last Failure	Last Duration
		HelloWorld	25 min - #14	1 hr 40 min - #12	1.4 sec
		QA	25 min - #5	28 min - #2	1.4 sec

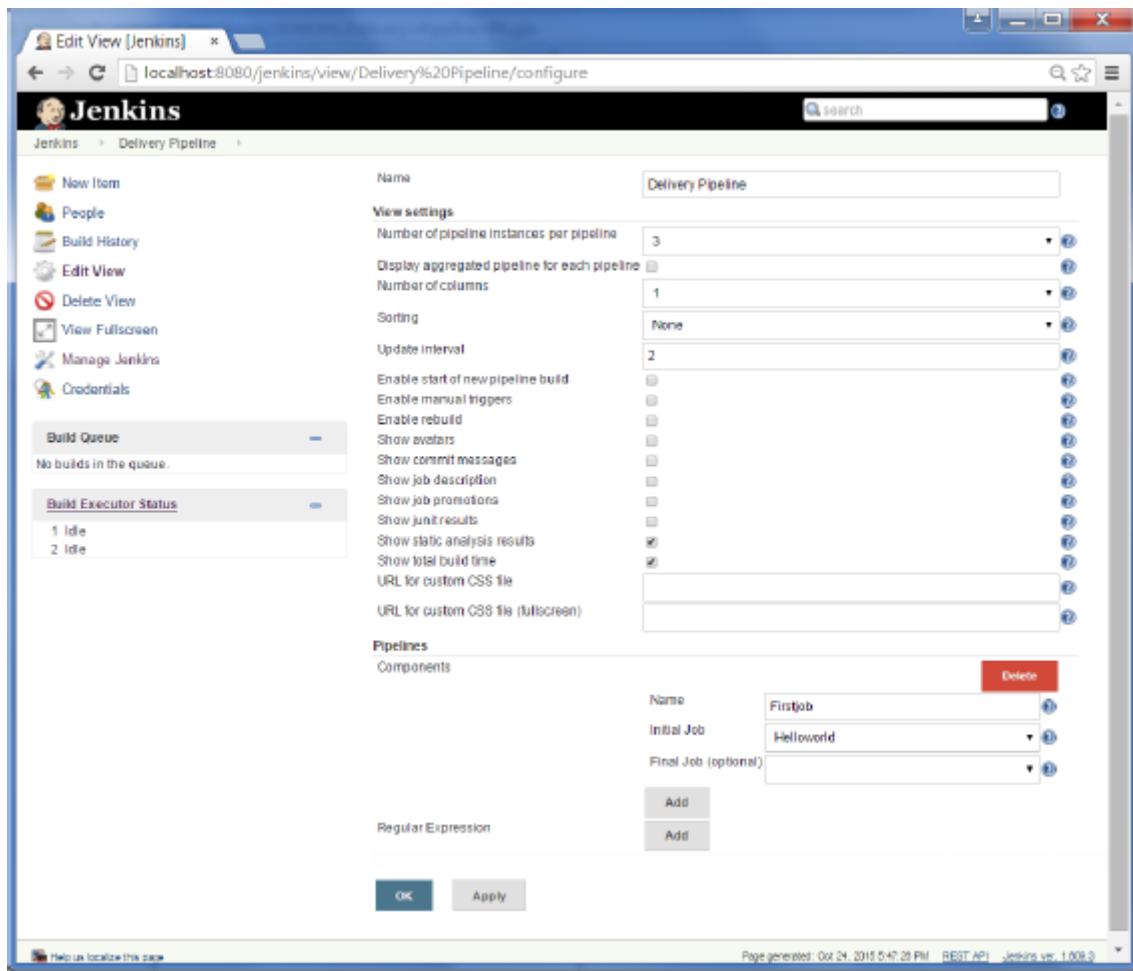
Icons for 'Build Queue' and 'Build Executor Status' are shown below the table.

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

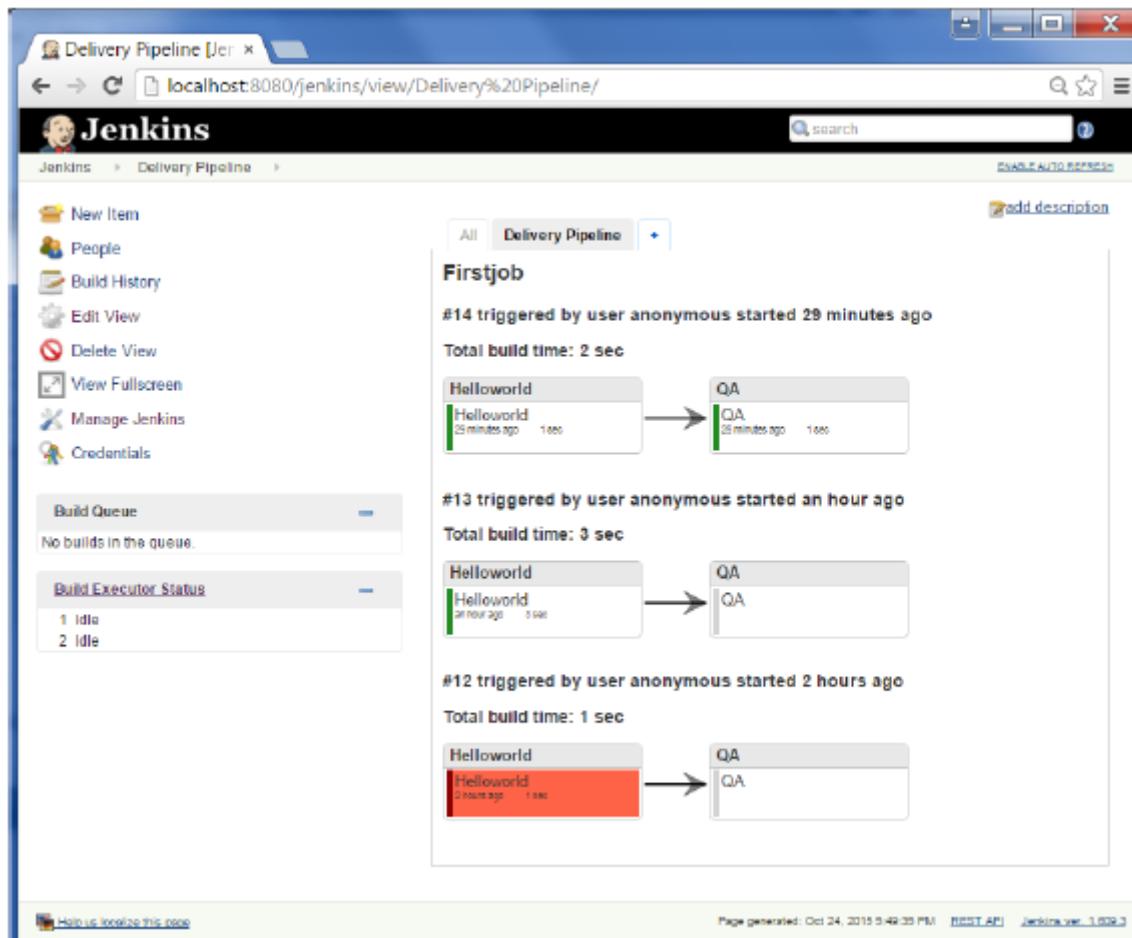


Step 10 – In the next screen, you can leave the default options. One can change the following settings –

- 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 Plugin Manager interface. The URL is `localhost:8080/jenkins/pluginManager/available`. A search bar at the top right contains the text 'Build pipeline'. Below it, there are tabs: 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. A filter bar also contains the text 'Build pipeline'. The main area displays a list of available plugins:

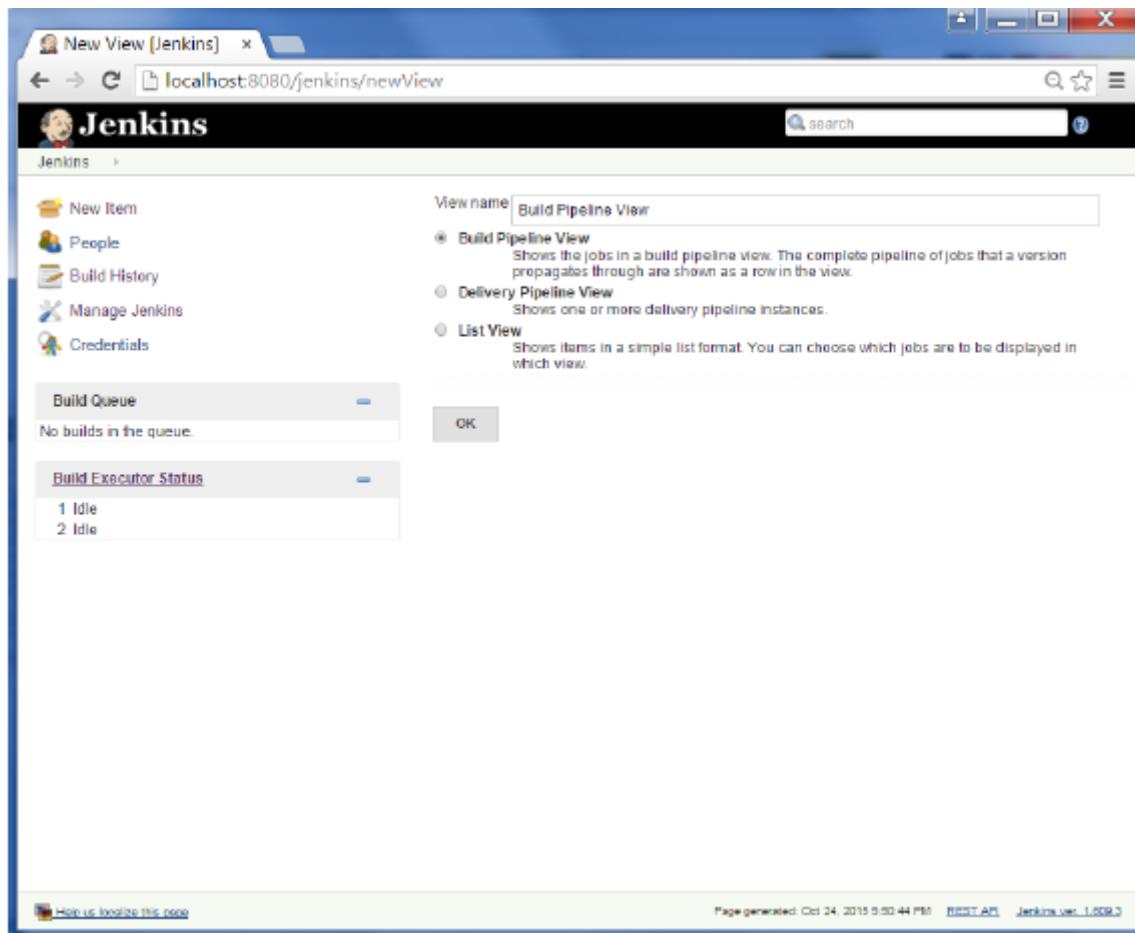
Install	Name	Version
<input checked="" type="checkbox"/>	Build Pipeline Plugin	1.4.8
<input type="checkbox"/>	Fail The Build Plugin	1.0
<input type="checkbox"/>	Runscope plugin	1.44
<input type="checkbox"/>	Build Graph View Plugin	1.1.1
<input type="checkbox"/>	Delivery Pipeline Plugin	0.9.7

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

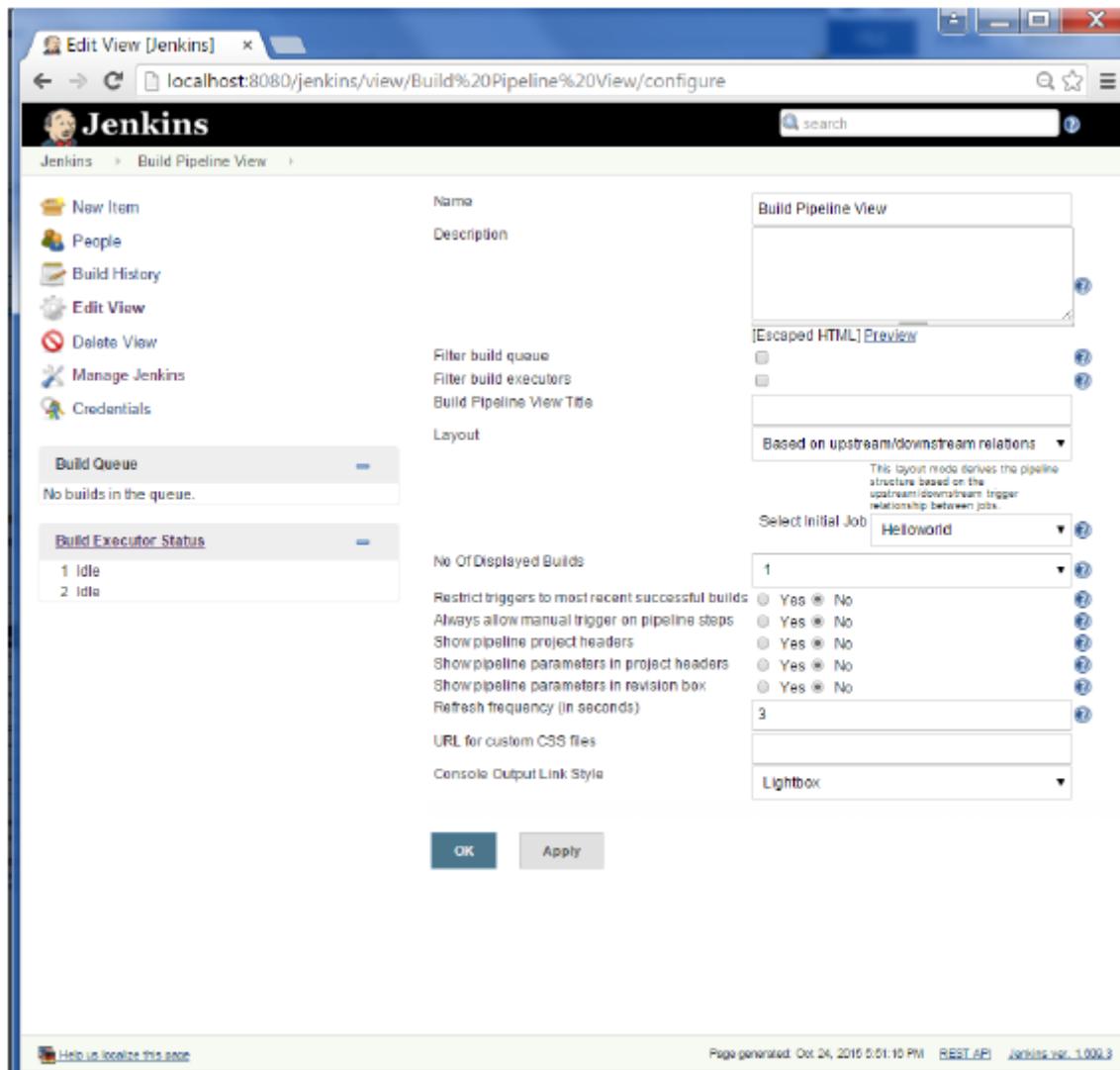
Step 2 – To see the Build pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the ‘All’ Tab.

The screenshot shows the Jenkins Dashboard. The URL is `localhost:8080/jenkins/`. On the left, there's a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. The main area has sections for 'Build Queue' (empty) and 'Build Executor Status' (1 idle, 2 idle). In the center, there's a table titled 'All' with columns: S, W, Name, Last Success, Last Failure, and Last Duration. It lists two entries: 'HelloWorld' and 'QA'. At the bottom of the table, there's a legend for RSS feeds: 'RSS for all', 'RSS for failures', and 'RSS for just failed builds'. The status bar at the bottom indicates the page was generated on Oct 24, 2015, at 4:48:09 PM.

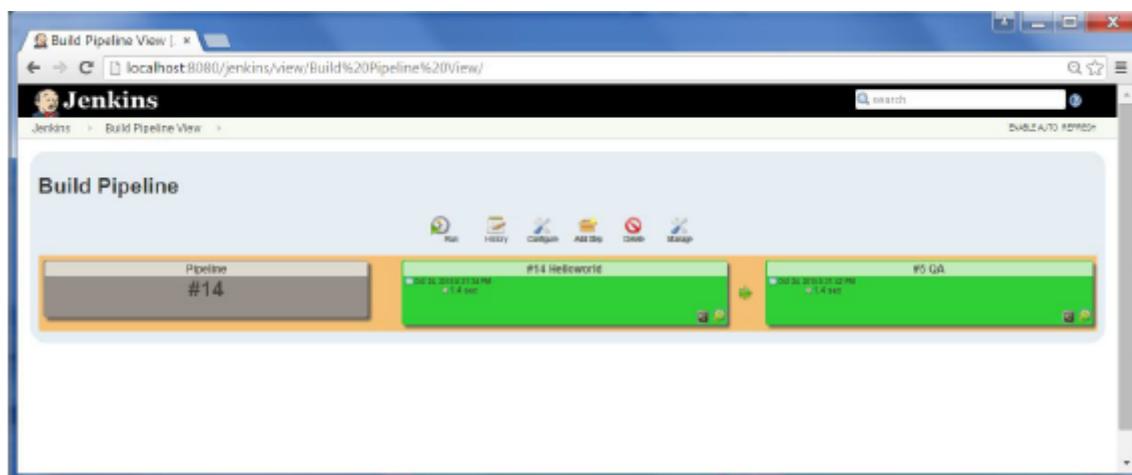
Step 3 – Enter any name for the View name and choose the option ‘Build Pipeline View’.



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 from the official documentation 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. The 'Documents' section contains links for Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, and Servlet Container Notes. The main content area displays a hierarchical tree of plugin categories:

- 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 Slaves 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

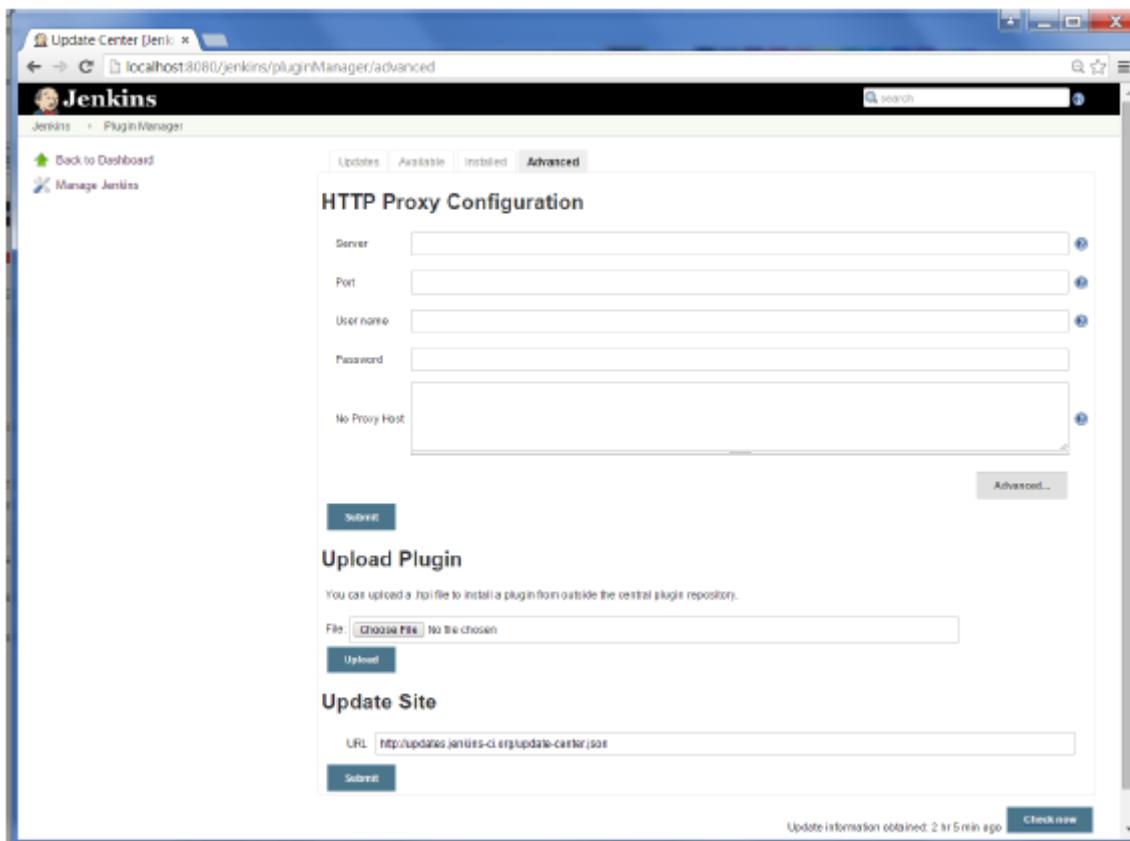
Uninstalling Plugins

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.

Enabled	Name	Version	Previously installed version	Pinned	Uninstall
<input checked="" type="checkbox"/>	Ant Plugin	1.2			Uninstall
<input checked="" type="checkbox"/>	Build History Metrics Plugin	1.2			Uninstall
<input checked="" type="checkbox"/>	Build Pipeline Plugin	1.48			Uninstall
<input checked="" type="checkbox"/>	Credentials Plugin	1.23	Downgrade to 1.18	Open	Uninstall
<input checked="" type="checkbox"/>	CVSPlugin	2.15			Uninstall
<input checked="" type="checkbox"/>	Delivery Pipeline Plugin	0.9.1			Uninstall
<input checked="" type="checkbox"/>	Deploy to container Plugin	1.10			Uninstall
<input checked="" type="checkbox"/>	External Monitor Job Type Plugin	3.4			Uninstall
<input checked="" type="checkbox"/>	Extreme Notification Plugin	3.1			Uninstall
<input checked="" type="checkbox"/>	FindBugs Plugin	4.62			Uninstall

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.

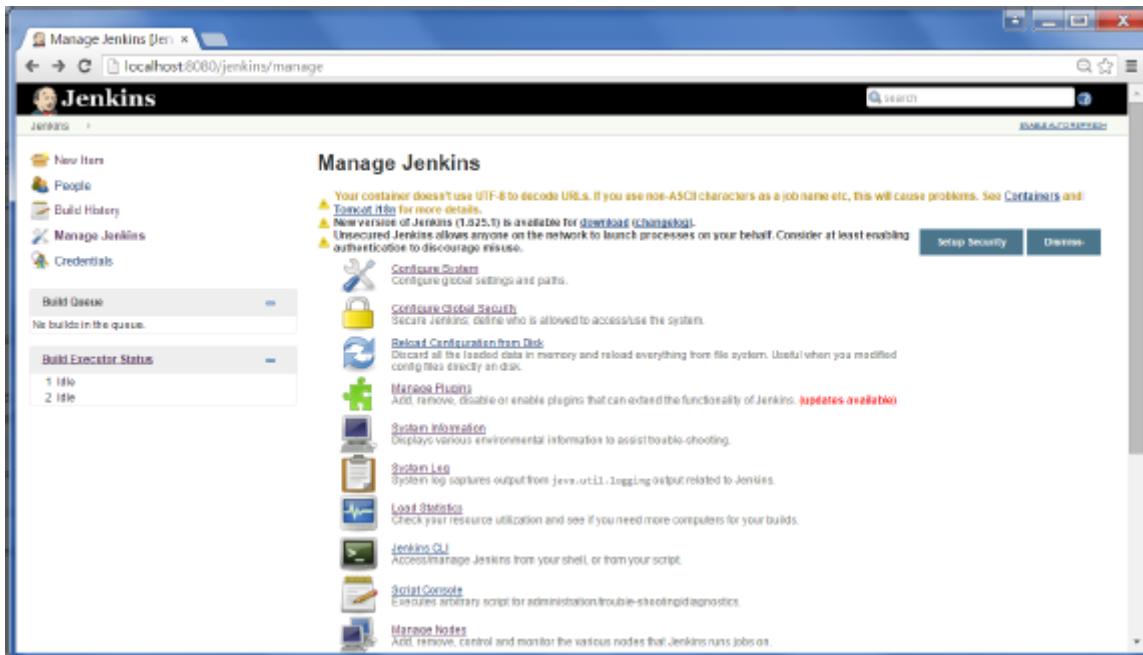


Jenkins - Security

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.

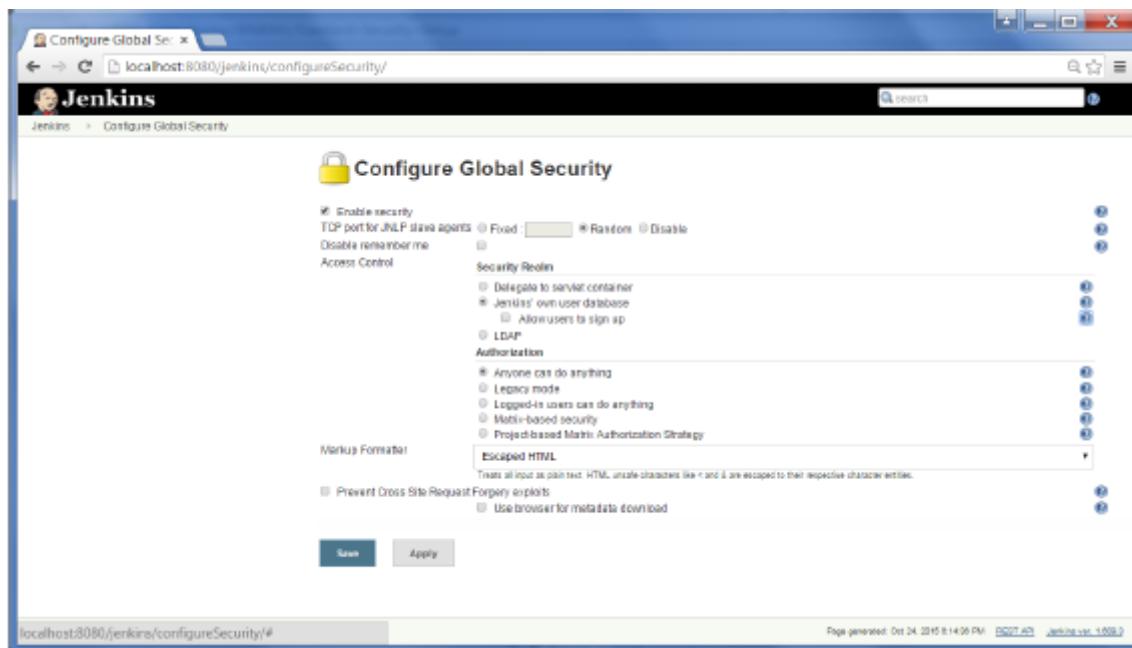
To configure Security in Jenkins, follow the steps given below.

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’.

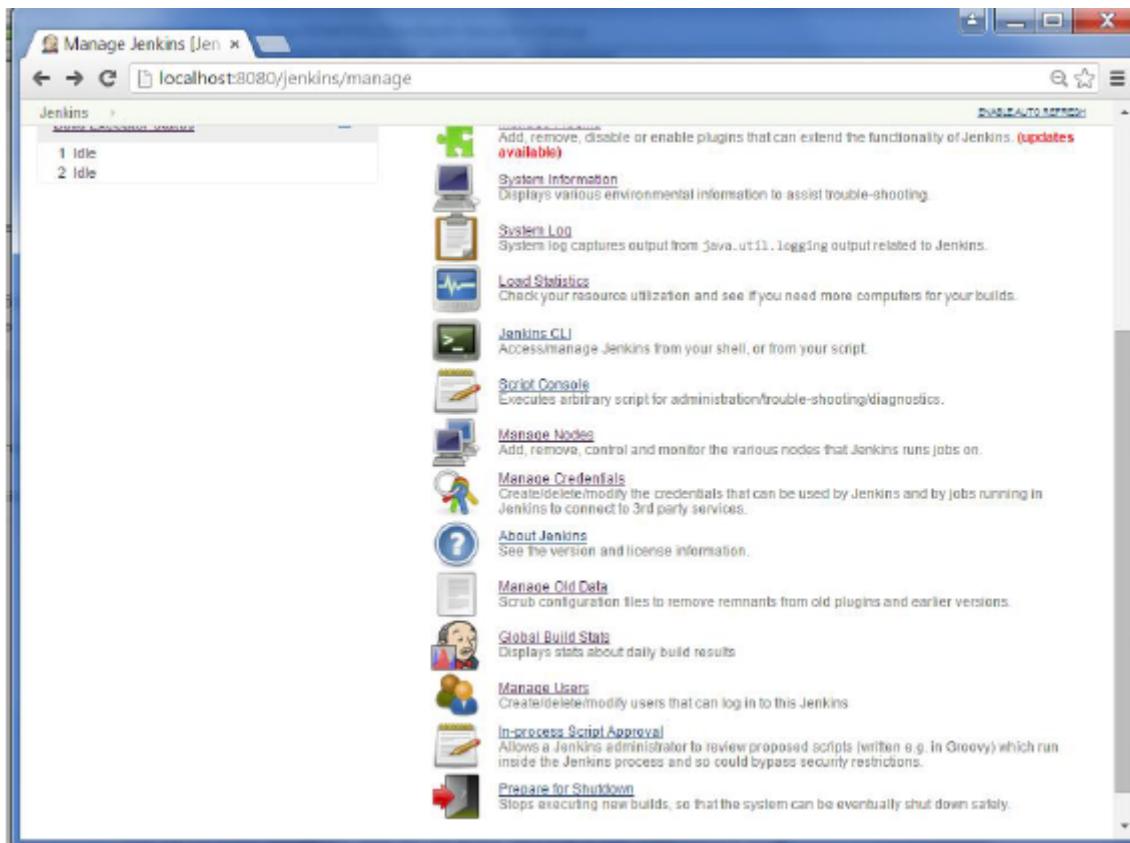
By default you would want a central administrator to define users in the system, hence ensure the ‘Allow users to sign up’ option is unselected. You can leave the rest as it is for now and click the Save button.



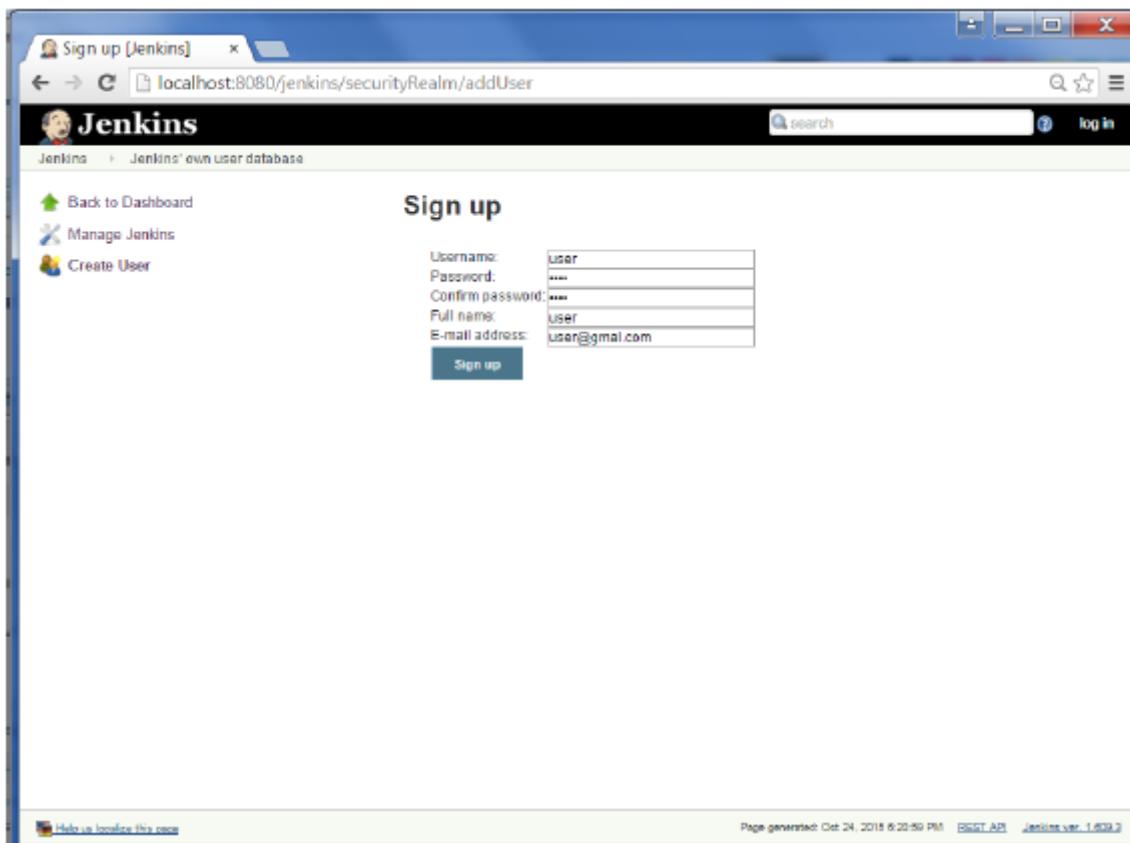
Step 3 – You will be prompted to add your first user. As an example, we are setting up an admin users for the system.

The screenshot shows the Jenkins 'Sign up' page. At the top left, there's a 'Back to Dashboard' link, a 'Manage Jenkins' link, and a 'Create User' link. On the right, there's a search bar and a 'log in' link. The main title is 'Sign up'. Below it, there are five input fields: 'Username' (admin), 'Password' (****), 'Confirm password' (****), 'Full name' (Administrator), and 'E-mail address' (al@gmail.com). A blue 'Sign up' button is at the bottom. The URL in the browser is 'localhost:8080/jenkins/securityRealm/firstUser'.

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

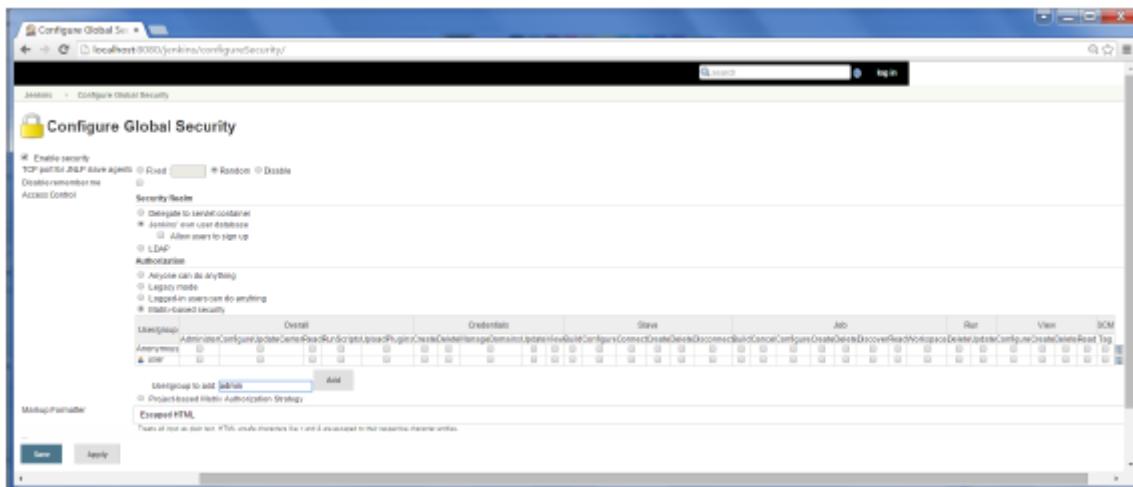


Step 5 – Just like you defined your admin user, start creating other users for the system. As an example, we are just creating another user called ‘user’.



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

Now in the Authorization section, click on ‘Matrix based security’



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

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

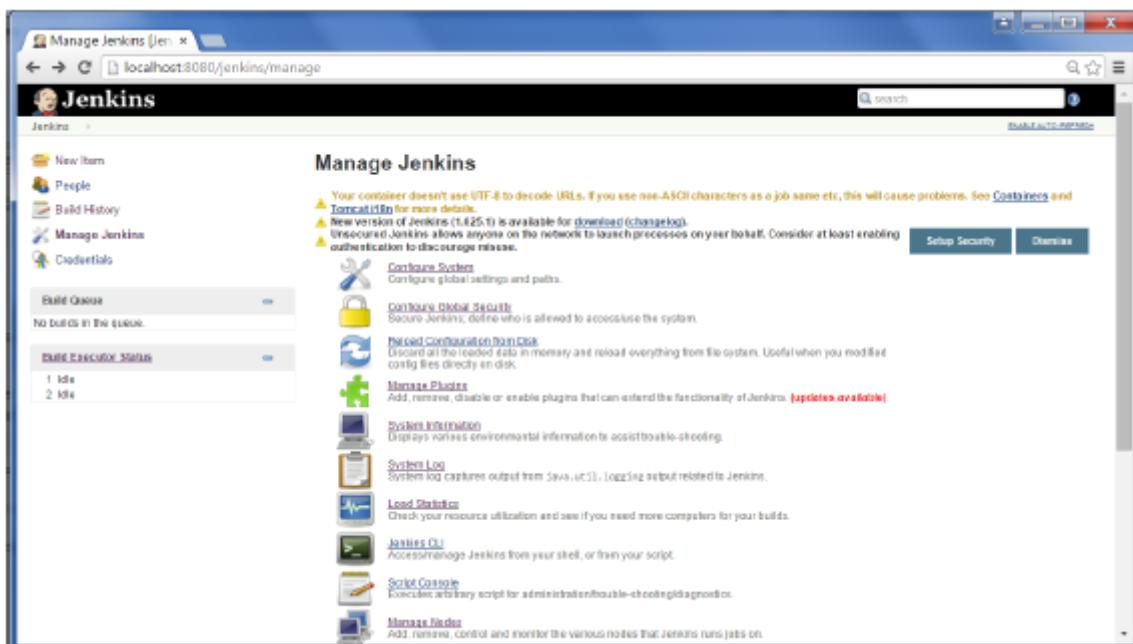
Your Jenkins security is now setup.

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

Jenkins - Backup Plugin

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

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

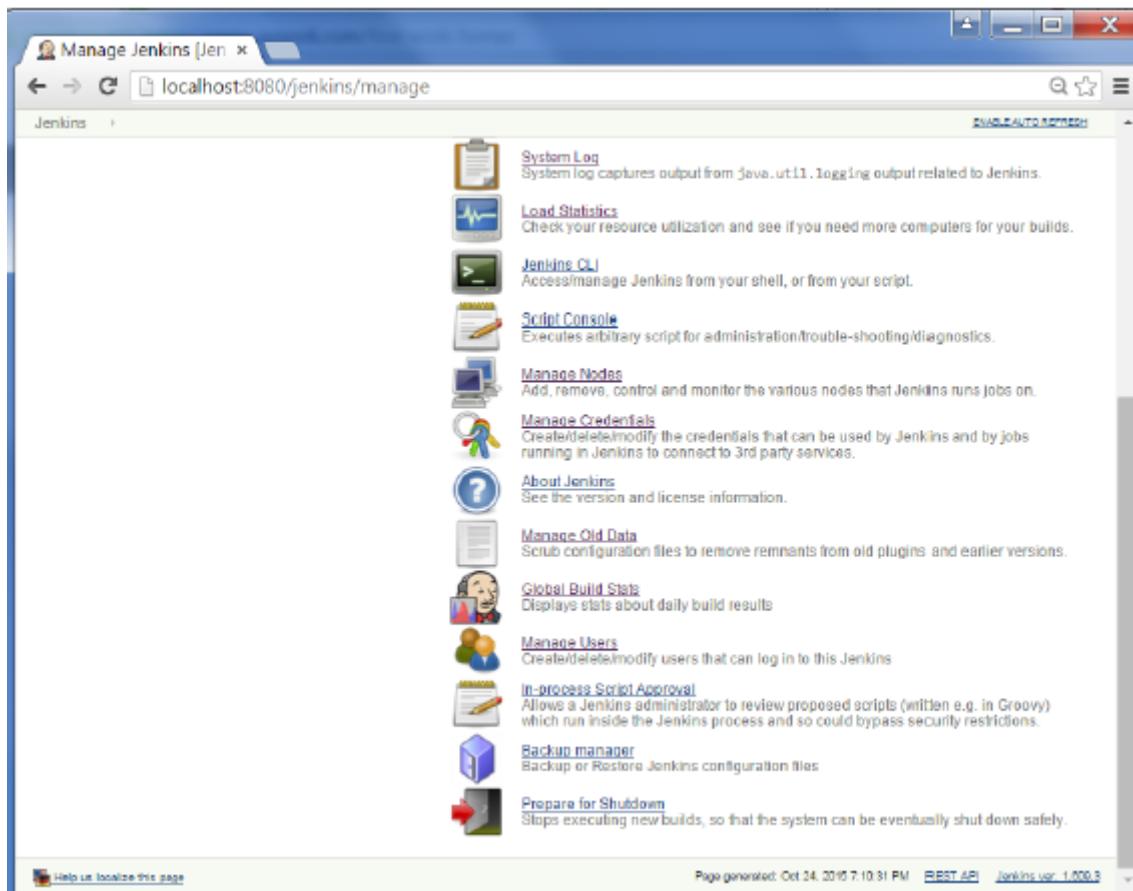


Step 2 – In the available tab, search for 'Backup Plugin'. Click On Install without Restart. Once done, restart the Jenkins instance

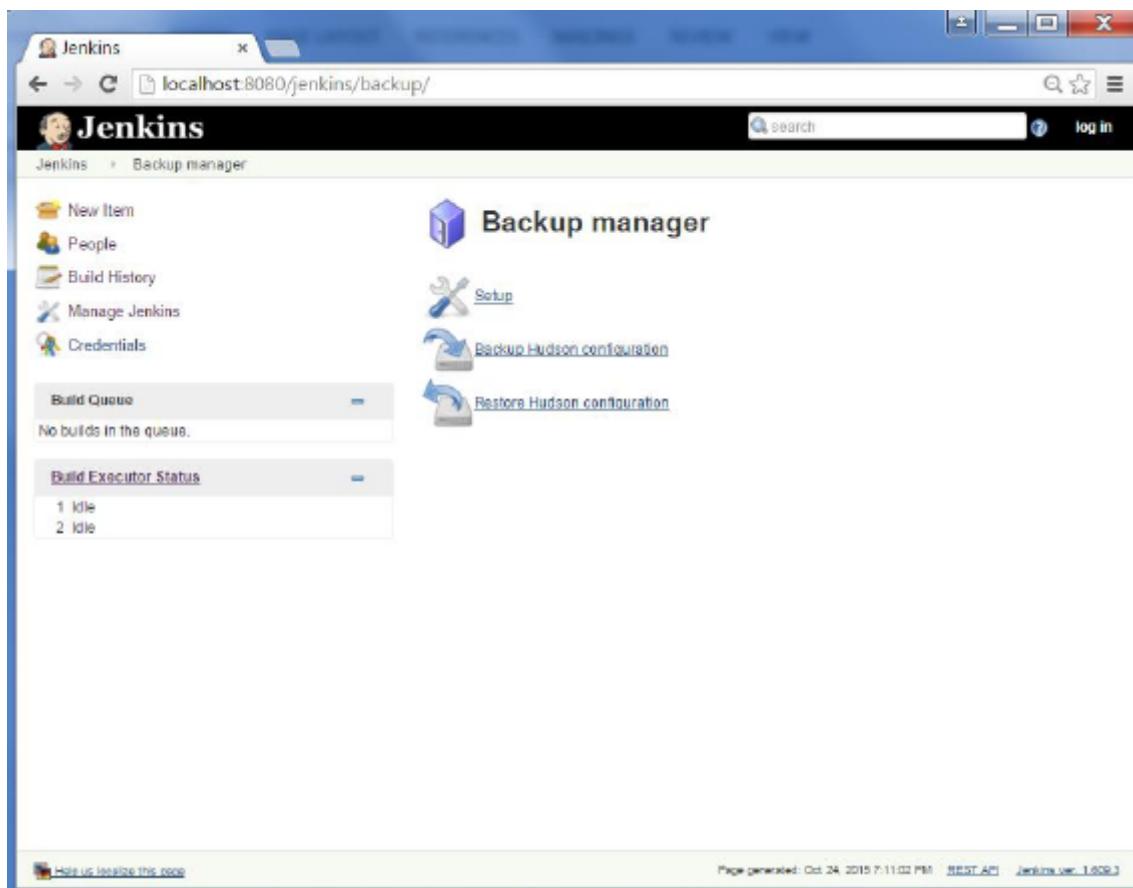
The screenshot shows the Jenkins Plugin Manager interface. The 'Available' tab is selected. A search bar at the top right contains the text 'backup'. A list of available plugins is shown, with 'Backup plugin' selected. The 'Backup plugin' entry includes a brief description: 'Backup plugin allows archiving and restoring your Jenkins (and Hudson) home directory.' It is version 1.8.1. Below this, there are other entries like 'Backup and interrupt job plugin', 'CloudBees Jenkins Enterprise', and 'CloudBees Free Enterprise Plugins'. At the bottom of the list, there are three buttons: 'Install without restart', 'Download now and install after restart', and 'Update information'.

The screenshot shows the Jenkins Update Center interface. The main title is 'Installing Plugins/Upgrades'. Under 'Preparation', it lists: 'Checking Internet connectivity', 'Checking update center connectivity', and 'Success'. Below this, under 'Backup plugin', it shows a blue circular icon with the text 'Success'. At the bottom, 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'. The footer of the page shows the date 'Page generated: Oct 24, 2015 6:26:30 PM' and the Jenkins version 'jenkins ver. 1.600.3'.

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



Step 4 – Click on Setup.



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

Backup config files

Backup configuration

Hudson root directory E:\Jenkins
Backup directory D:\Backup
Format Zip
File name template backup_@date@.extension@
Custom exclusions
 Verbose mode
 Configuration files (.xml) only
 No shutdown
Backup content
 Backup job workspace
 Backup builds history
 Backup maven artifacts archives
 Backup fingerprints

Save

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

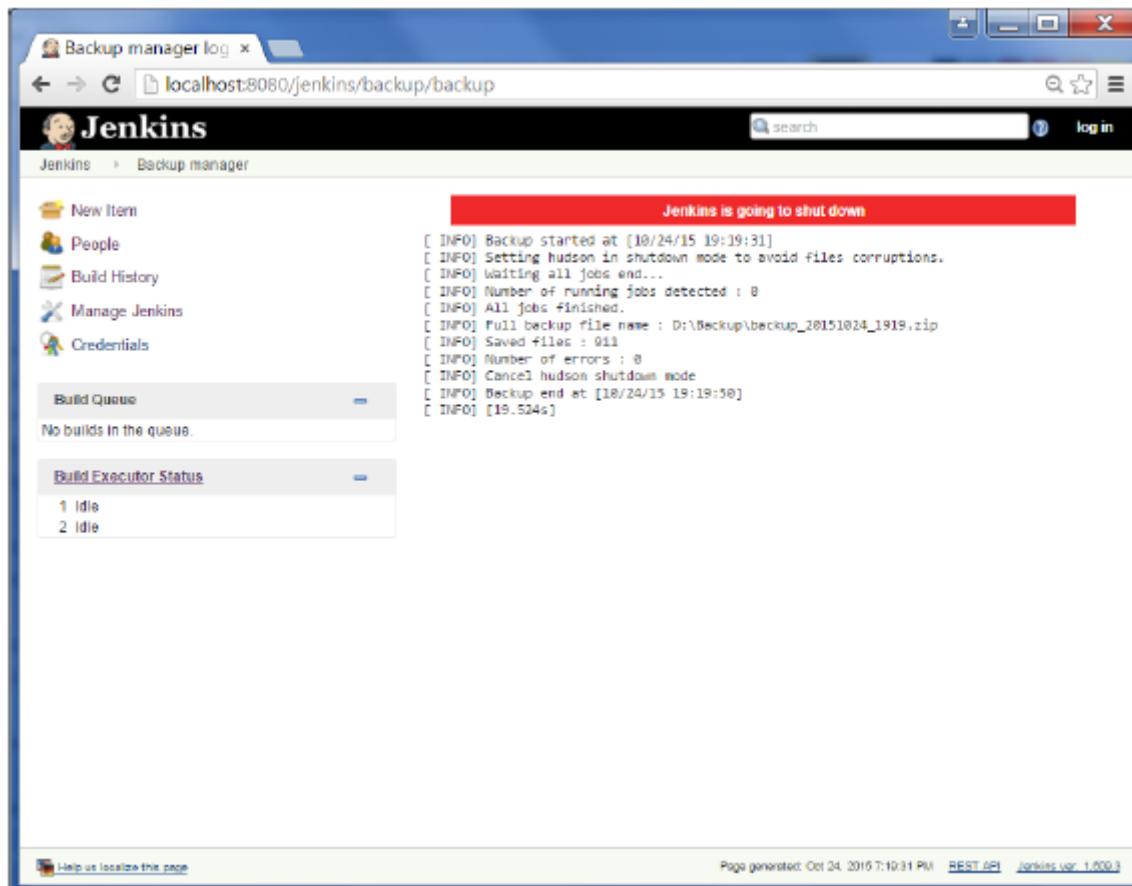
Backup manager

Setup

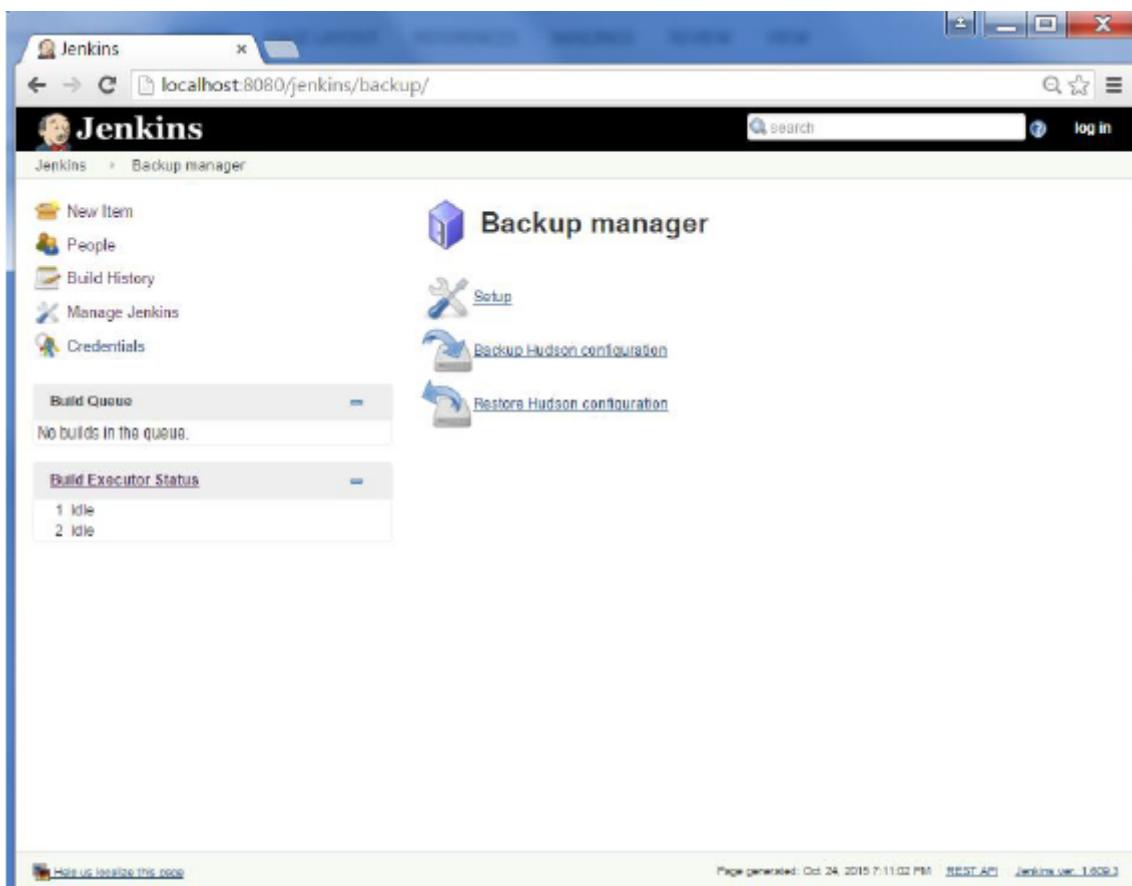
Backup Hudson configuration

Restore Hudson configuration

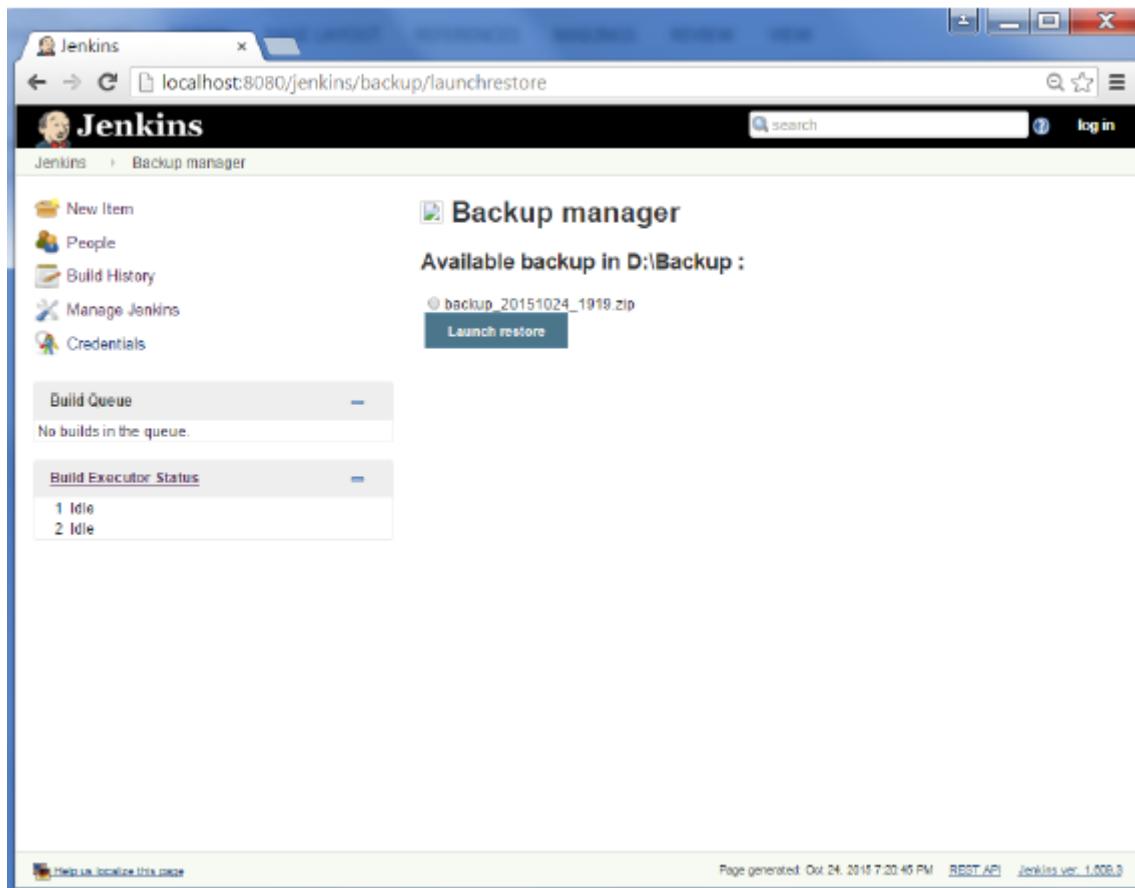
The next screen will show the status of the backup



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



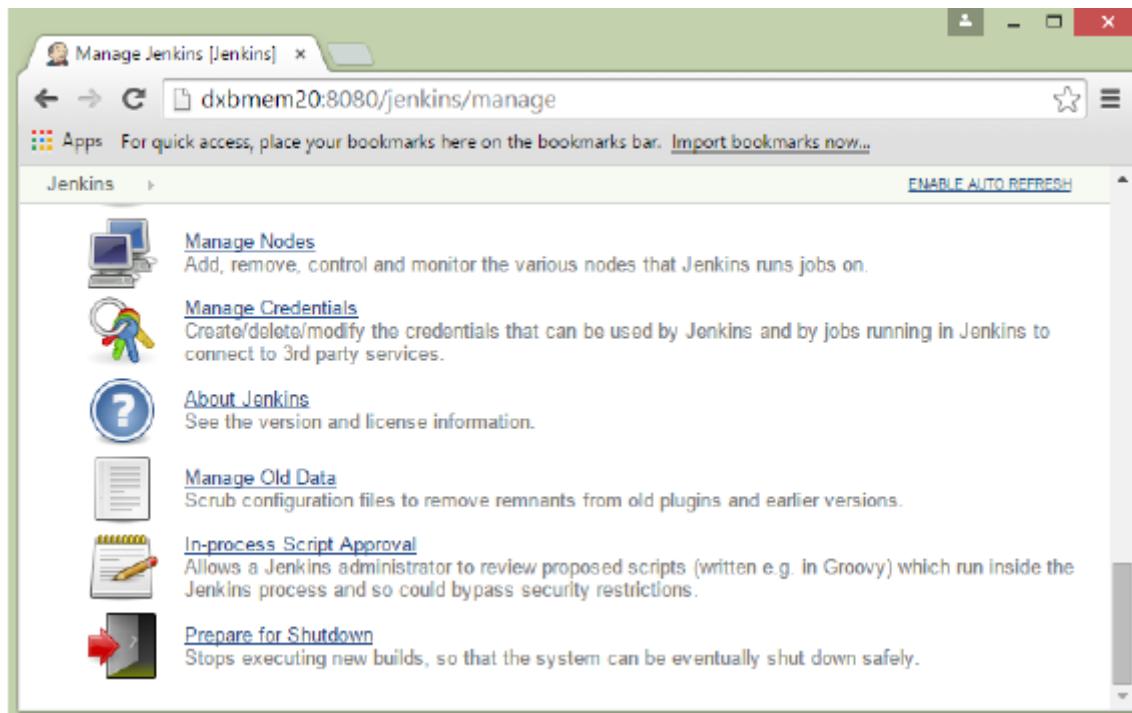
The list of backup's will be shown, click on the appropriate one to click on Launch Restore to begin the restoration of the backup.



Jenkins - Remote Testing

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

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



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. The title bar says 'Nodes [Jenkins]'. The URL in the address bar is 'dxbmemp20:8080/jenkins/computer/'. The main content area shows a table of nodes:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Sp
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 blue button labeled 'Refresh status'. The footer of the browser window shows the URL 'dxbmemp20:8080/jenkins/computer/#'.

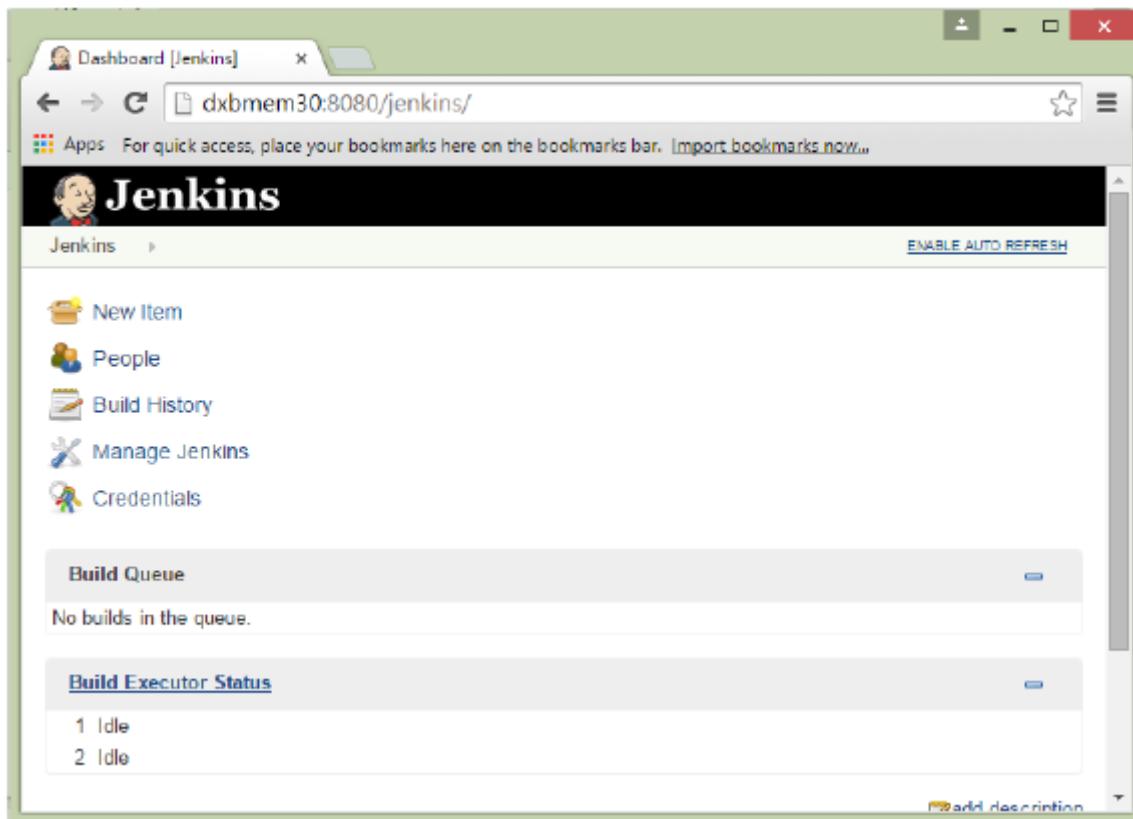
Step 2 – Click on configure for the DXBMEM30 slave machine.

The screenshot shows the Jenkins 'Nodes' page. At the top, there are two nodes listed: '2 Idle' and 'DXBMEM30'. Below this is a table with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, and Free Temp Sp. The 'DXBMEM30' row is selected, and its details are shown in the table. A context menu is open over the 'DXBMEM30' row, 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. It contains fields for Name (DXBMEM30), Description, # of executors (1), Remote root directory (C:\users\administrator.EMIRATES\jenkins), Labels, Usage (Utilize this node as much as possible), and Launch method (set to 'Launch slave agents via Java Web Start'). A 'Save' button is at the bottom left, and an 'Advanced...' button is at the bottom right.

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



Step 5 – Click on the DXBMEM30 instance.

The screenshot shows the 'Nodes' page under the Jenkins navigation. It displays two nodes: 'master' (1 Idle) and 'DXBMEM30' (offline). Below this, a table provides detailed information for each node:

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space
	DXBMEM30	Windows Server 2012 (x86)	In sync	112.79 GB	4.54 GB
	master	Windows Server 2012 (x86)	In sync	94.20 GB	3.85 GB
	Data obtained	45 min	45 min	45 min	45 min

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’

Connect slave to Jenkins one of these ways:

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

Created by anonymous user

Projects tied to DXBMEM30

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

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

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

Do you want to run this application?

Name: Jenkins Remoting Agent

Publisher: Kohsuke Kawaguchi

Locations: <http://dxbmem20:8080>
Launched from downloaded JNLP file

Running this application may be a security risk

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

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

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

I accept the risk and want to run this application.

Run **Cancel**

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

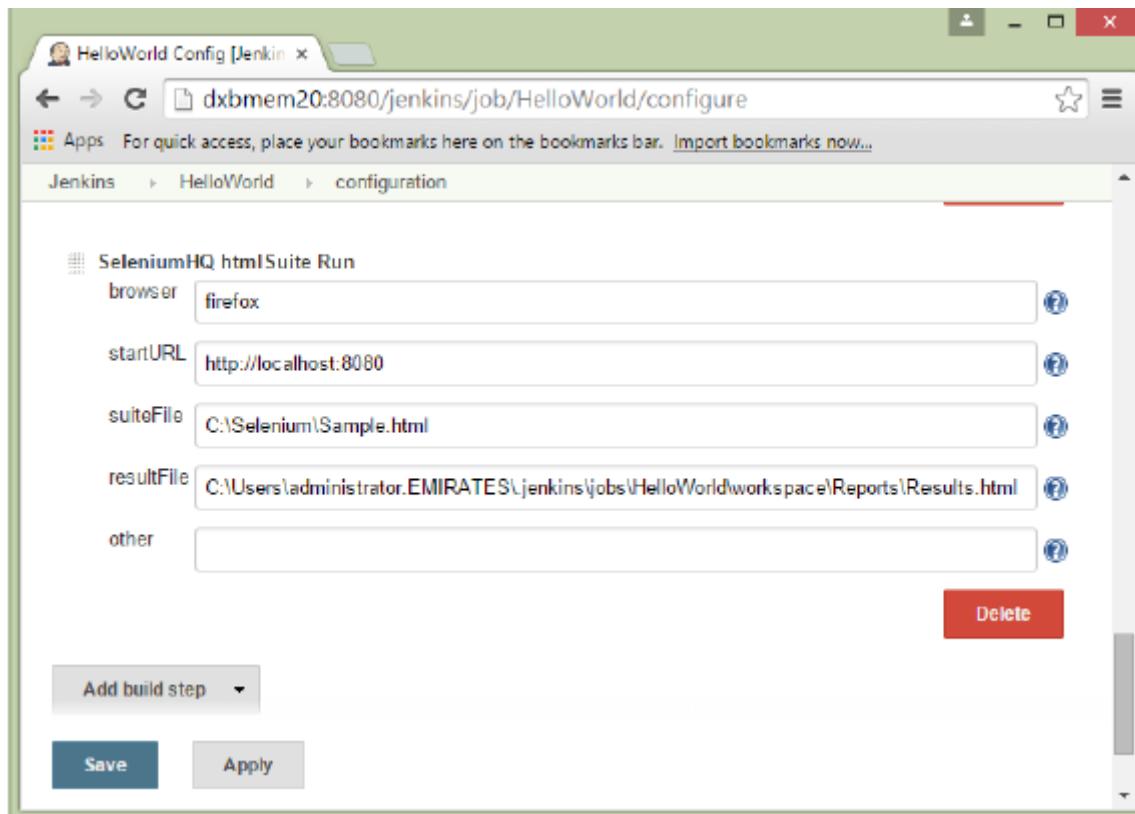


Step 8 – Configuring your tests to run on the slave. Here, you have to ensure that the job being created is meant specifically to only run the selenium tests.

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

A screenshot of a web browser displaying the Jenkins job configuration page for "HelloWorld". The URL in the address bar is "dxbmem20:8080/jenkins/job/HelloWorld/configure". The page shows various configuration options under "Advanced Project Options" and "Source Code Management". The "Label Expression" field is set to "DXBMEM30". The "Restrict where this project can be run" checkbox is checked. There are "Save" and "Apply" buttons at the bottom.

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



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