



Jenkins

Hello every one, Thank you for your great support to make me another review of Automation Tools, Today we are going to see the Hands on Tour of **Jenkins**.

What is Jenkins?

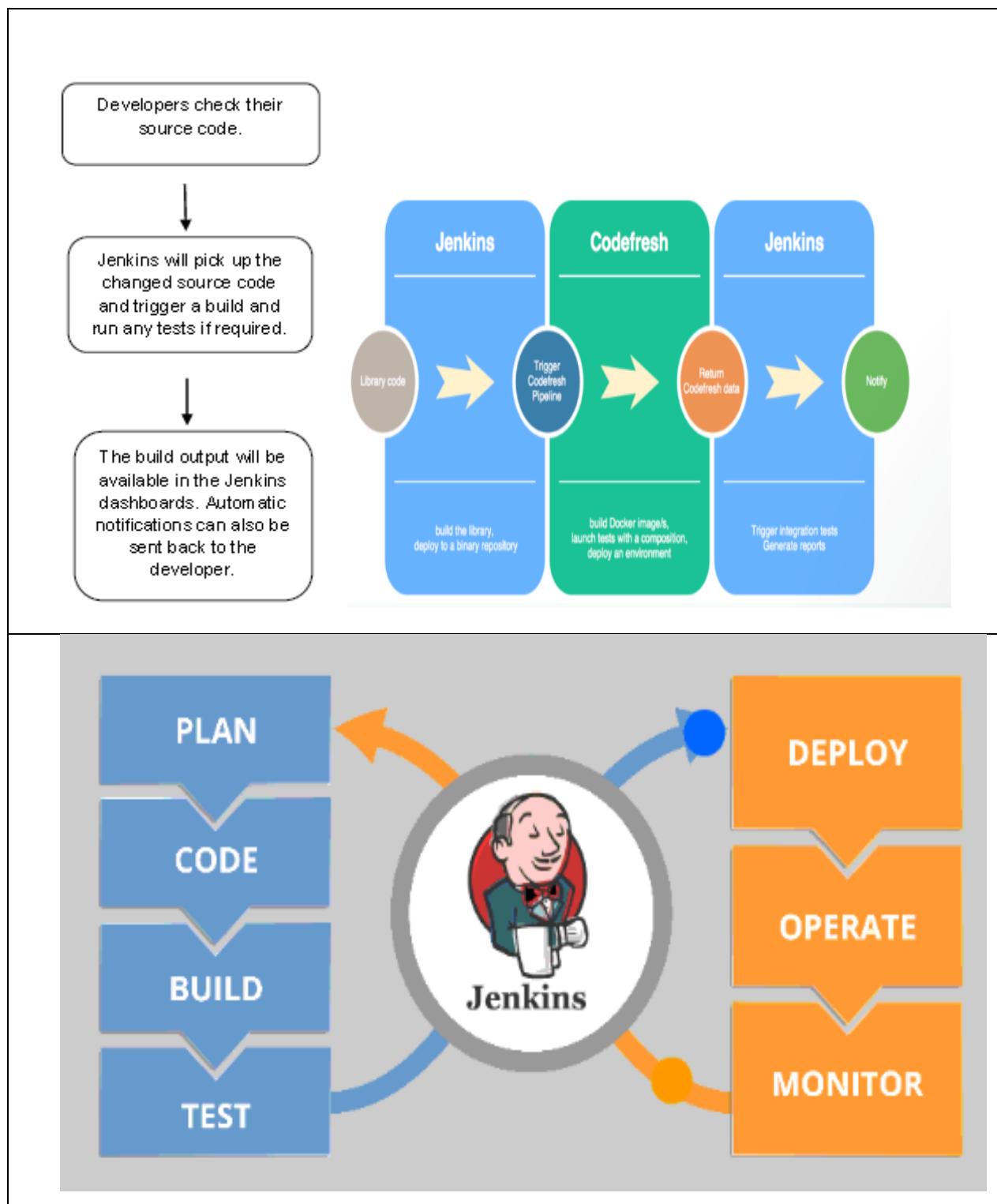
Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies. With Jenkins, organizations can accelerate the software development process through automation. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis and much more.

Jenkins achieves Continuous Integration with the help of plugins. Plugins allows the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example: Git, Maven 2 project, Amazon EC2, HTML publisher etc.

Why Jenkins?

Jenkins is 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.

Architect of Jenkins:



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

fork was created from the Hudson source code, which brought about the introduction of Jenkins.

What is Continuous Integration?

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

System Requirements

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

Jenkins - Installation

Download Jenkins

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

Reference Link : <https://jenkins.io/download/>



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



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

Starting Jenkins

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

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

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

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

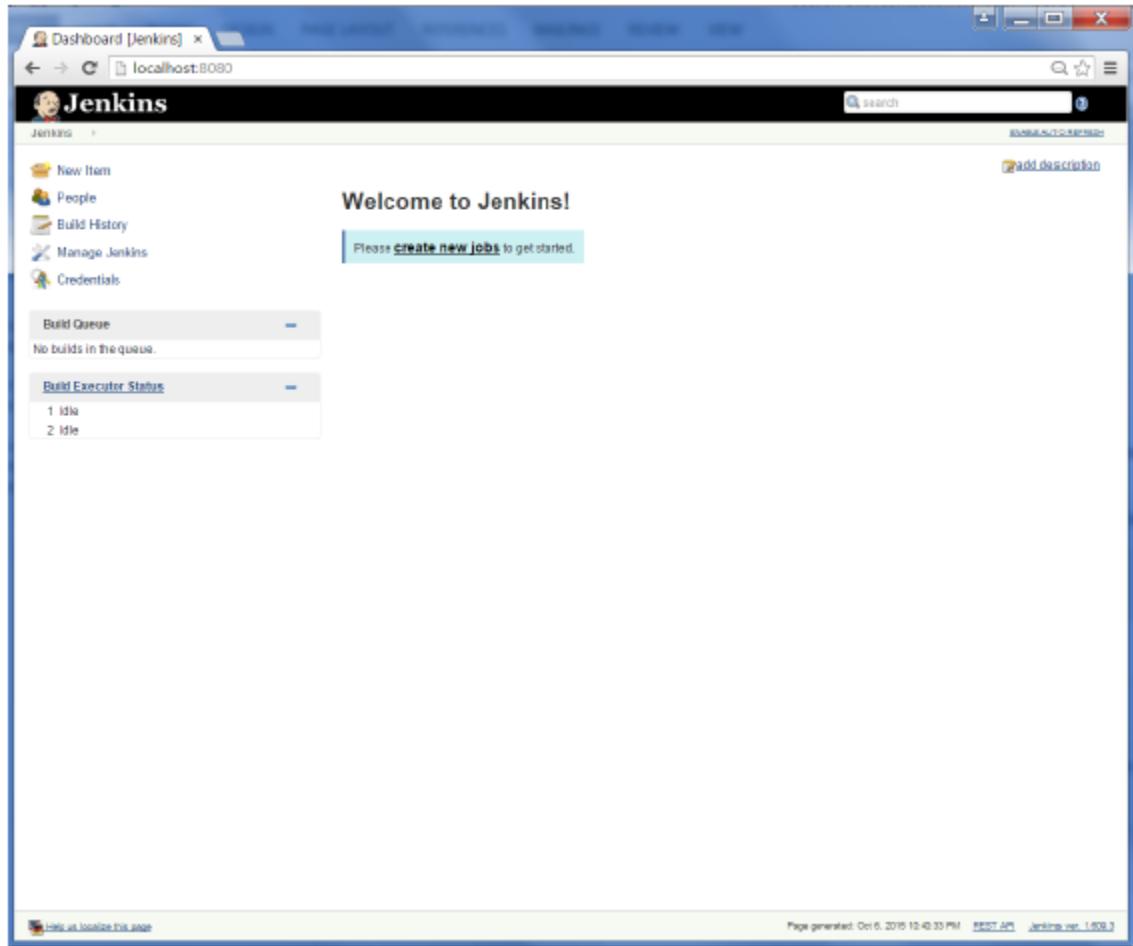
Once the processing is complete without major errors, the following line will come in the output of the command prompt.

```
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.

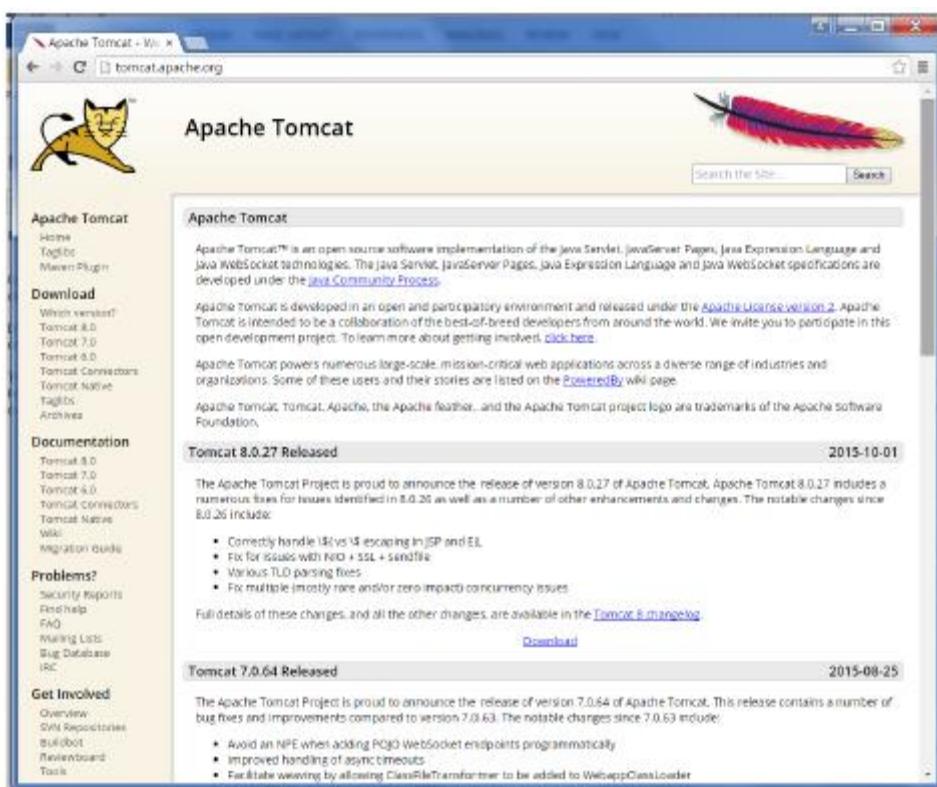
OS	Output
Windows	Append the String; C:\Program Files\Java\jdk1.7.0_60\bin to the end of the

	system variable PATH.
Linux	export PATH=\$PATH:\$JAVA_HOME/bin/

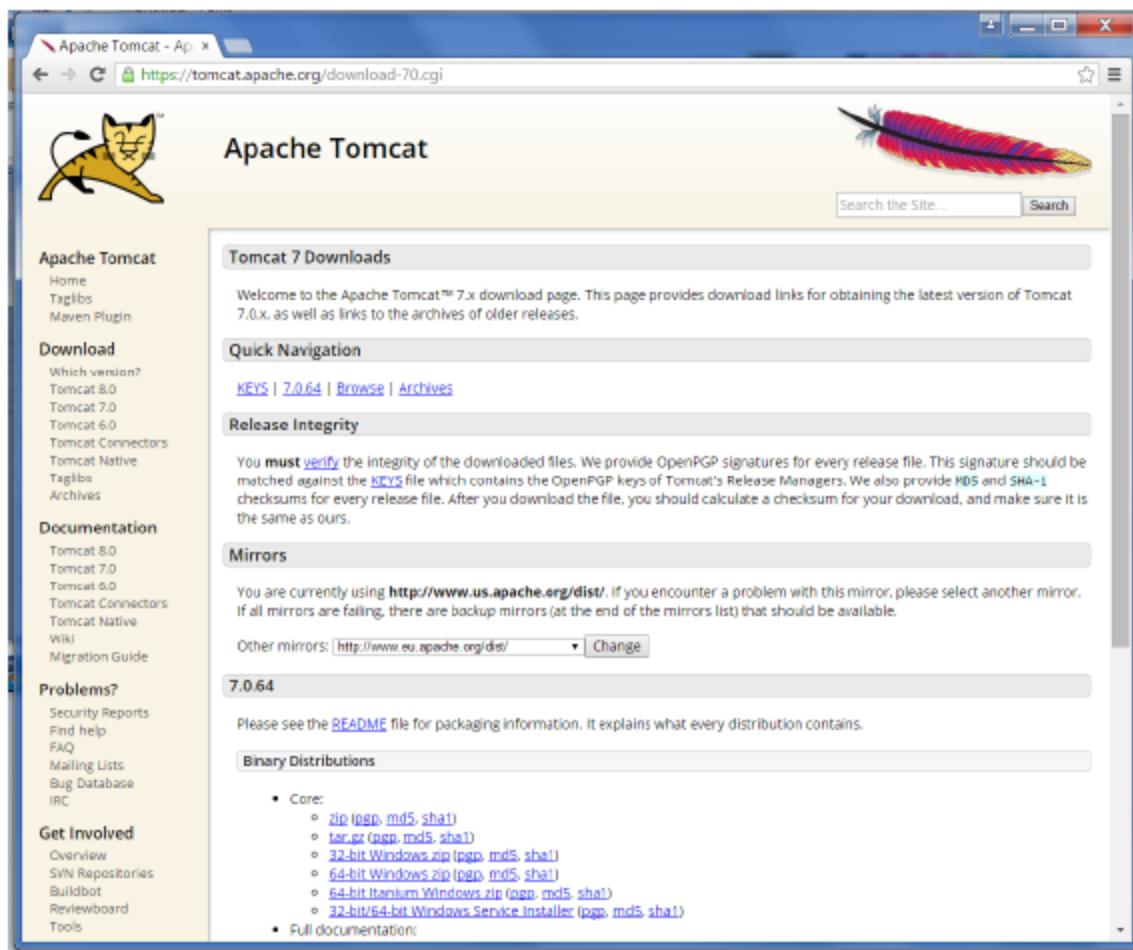
Verify the command java-version from command prompt as explained above.

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.



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



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

Then unzip the contents of the downloaded zip file.

Step 4: Jenkins and Tomcat Setup

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

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

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

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

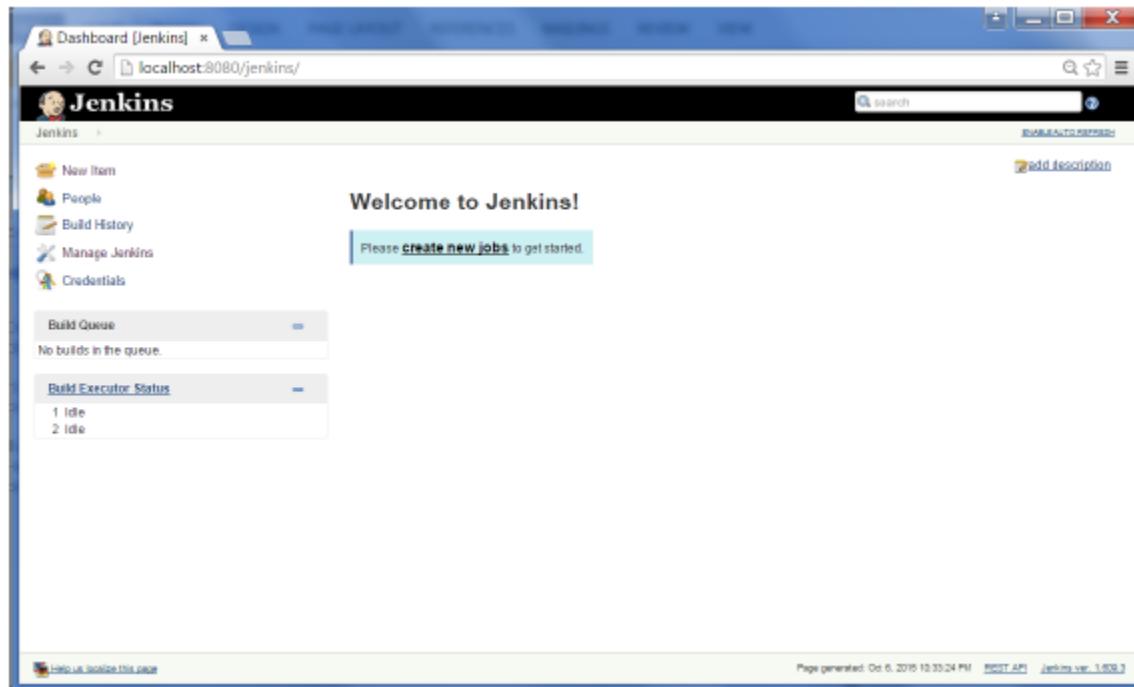
```
INFO: Server startup in 1302 ms
```

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

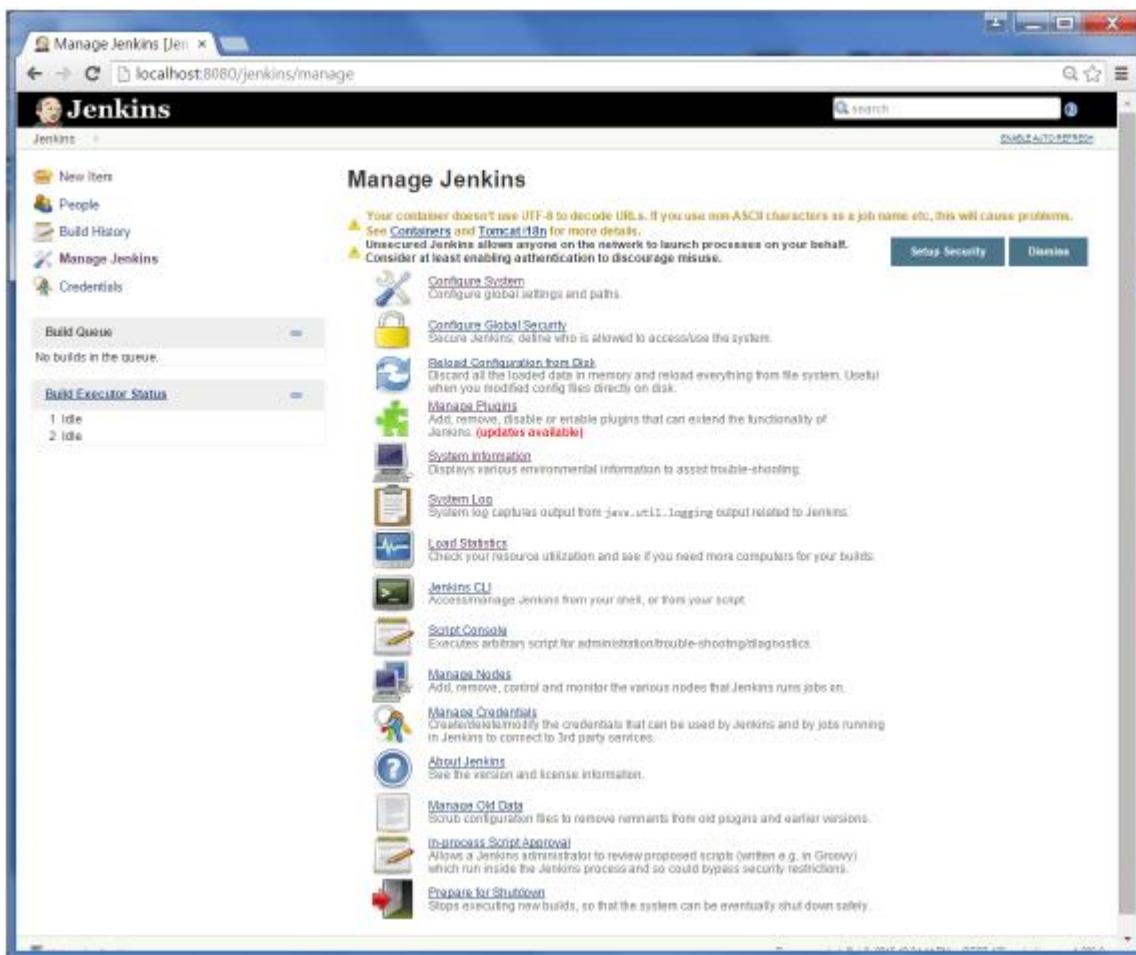


Jenkins - Git Setup

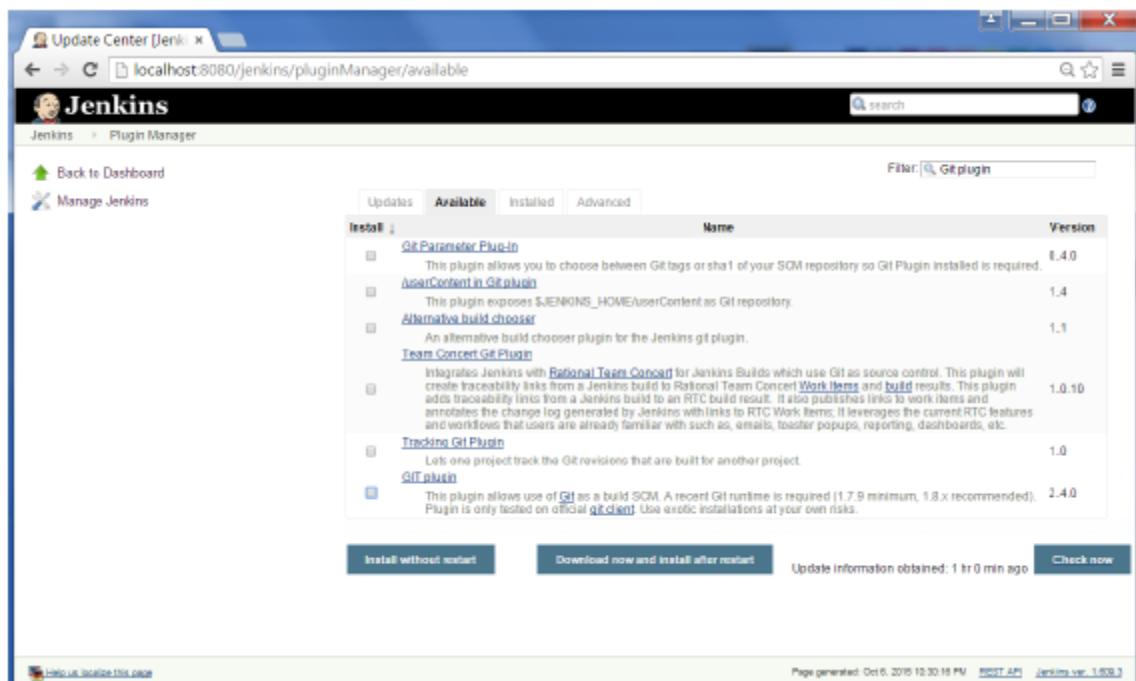
For this exercise, you have to ensure that Internet connectivity is present from the machine on which Jenkins is installed. In your Jenkins Dashboard (Home screen), click the Manage Jenkins option on the left hand side.



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



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



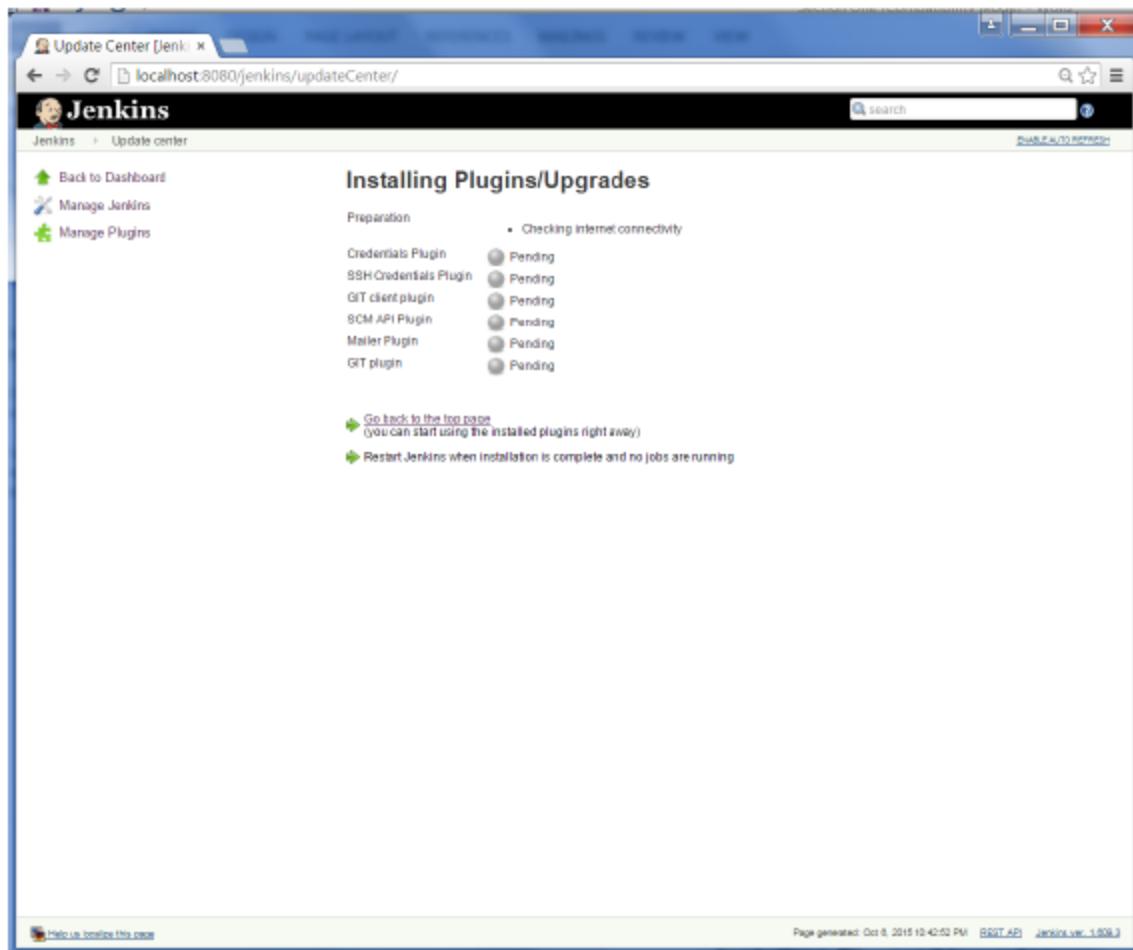
The list will then be filtered. Check the Git Plugin option and click on the button ‘Install without restart’

The screenshot shows the Jenkins Update Center interface. The URL in the address bar is `localhost:8080/jenkins/pluginManager/available`. The page title is "Jenkins". The navigation bar includes "Back to Dashboard" and "Manage Jenkins". The main content area has a search bar with the placeholder "search" and a filter bar with the text "Filter: Git plugin". Below these are tabs for "Updates", "Available" (which is selected), "Installed", and "Advanced". A table lists several plugins under the heading "Install":

Install	Name	Version
<input type="checkbox"/>	Git Parameter Plug-in This plugin allows you to choose between Git tags or sha1 of your SCM repository so Git Plugin installed is required.	0.4.0
<input type="checkbox"/>	UserContent in Git plugin This plugin exposes \$JENKINS_HOME/UserContent as Git repository.	1.4
<input type="checkbox"/>	Alternative build chooser An alternative build chooser plugin for the Jenkins git plugin.	1.1
<input type="checkbox"/>	Team Concert Git Plugin 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.	1.0.10
<input type="checkbox"/>	Tracking Git Plugin Lets one project track the Git revisions that are built for another project.	1.0
<input checked="" type="checkbox"/>	GIT plugin 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.	2.4.0

At the bottom of the table are three buttons: "Install without restart" (highlighted in blue), "Download now and install after restart", and "Check now". Below the table is a status message: "Update information obtained: 1 hr 0 min ago". At the very bottom of the page are links for "Help us improve this page", "Page generated: Oct 8, 2015 10:30:16 PM", "REST API", and "Jenkins ver. 1.808.3".

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

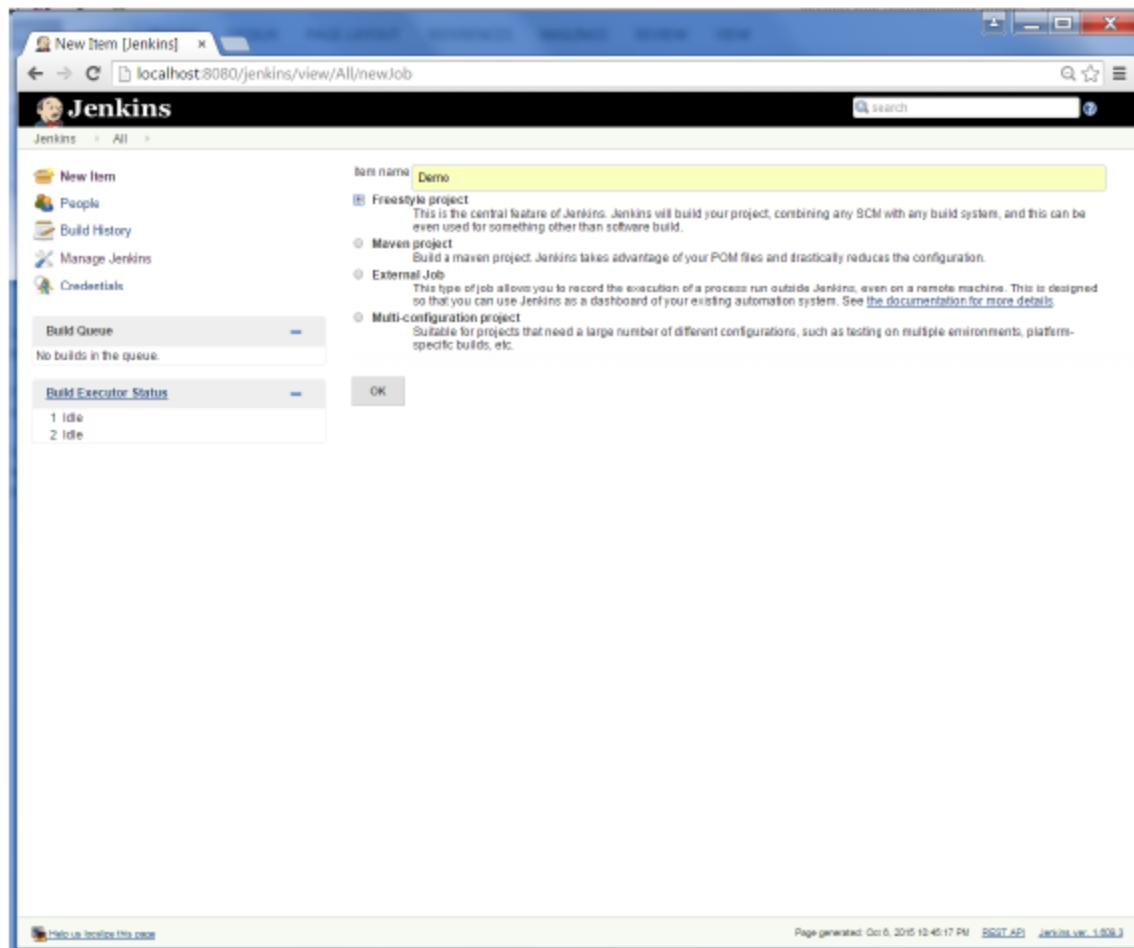


	Preparation	
Credentials Plugin	Pending	• Checking Internet connectivity
SSH Credentials Plugin	Pending	
GIT client plugin	Pending	
SCM API Plugin	Pending	
Mailer Plugin	Pending	
GIT plugin	Pending	

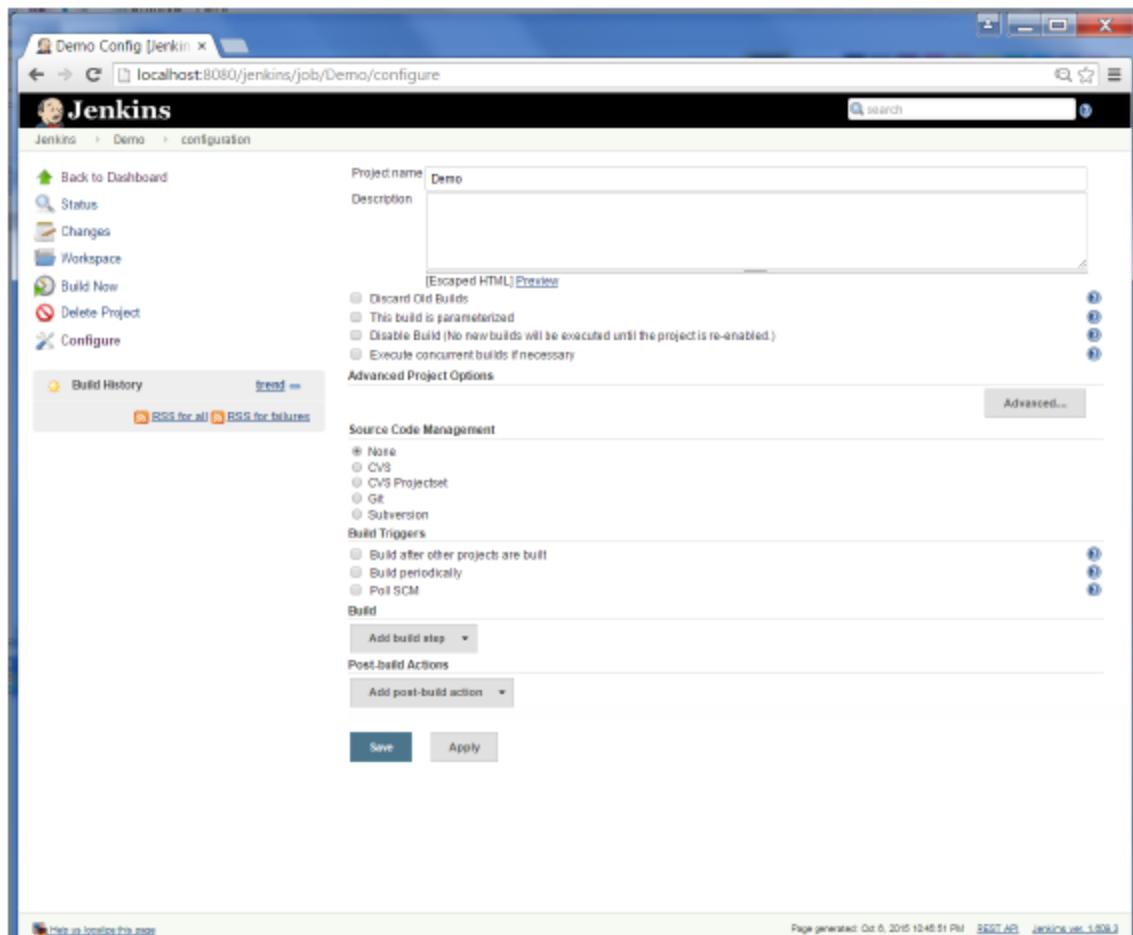
[Go back to the top page](#)
(you can start using the installed plugins right away)
[Restart Jenkins when installation is complete and no jobs are running](#)

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

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



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



Jenkins – Maven Setup

Step 1: Downloading and Setting Up Maven

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

Maven – Download

https://maven.apache.org/download.cgi

Apache Maven Project

Maven™

Last Published: 2015-09-24

Apache / Maven / Download Apache Maven

MAIN

- Welcome
- License
- Download**
- Install
- Configure
- Run
- IDE Integration
- ABOUT MAVEN**
- What is Maven?
- Features
- FAQ
- Support and Training
- DOCUMENTATION**
- Maven Plugins
- Index (category)
- Running Maven
- User Centre >
- Plugin Developer Centre
- Maven Repository Centre
- Maven Developer Centre
- Books and Resources

Downloading Apache Maven 3.3.3

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

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

Other mirrors: <http://www.eu.apache.org/dist/>

System Requirements

Java Development Kit (JDK)	Maven 3.3 requires JDK 1.7 or above to execute - it still allows you to build against 1.3 and other JDK versions by Using Toolchains
Memory	No minimum requirement
Disk	Approximately 10MB is required for the Maven Installation itself. In addition to that, additional disk space will be used for your local Maven repository. The size of your local repository will vary depending on usage but expect at least 500MB.
Operating System	No minimum requirement. Start up scripts are included as shell scripts and Windows batch files.

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the installation instructions. Use a source archive if you intend to build Maven yourself.

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

	Link	Checksum	Signature

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

The screenshot shows the Apache Maven Download page at <https://maven.apache.org/download.cgi>. The left sidebar contains links for Support and Training, Documentation, Maven Plugins, Index (category), Running Maven, User Centre, Plugin Developer Centre, Maven Repository Centre, Maven Developer Centre, Books and Resources, Security, Community Overview, How to Contribute, Maven Repository, Getting Help, Issue Tracking, Source Repository, The Maven Team, Project Documentation, Project Information, Maven Projects, Ant Tasks, Archetype, and Doxia. The main content area has sections for Memory, Disk, and Operating System requirements. Below these is a 'Files' section with a table of download links, checksums, and signatures for Binary tar.gz, Binary zip, Source tar.gz, and Source zip archives. A note about verifying signatures is present. At the bottom is a 'Previous Releases' section with a link to the Maven Releases History.

	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

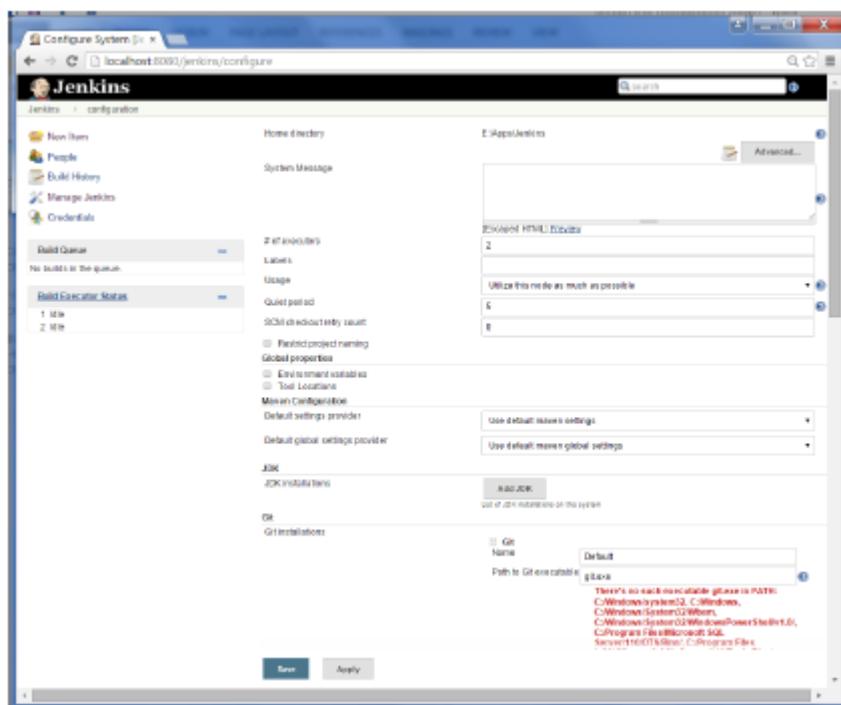
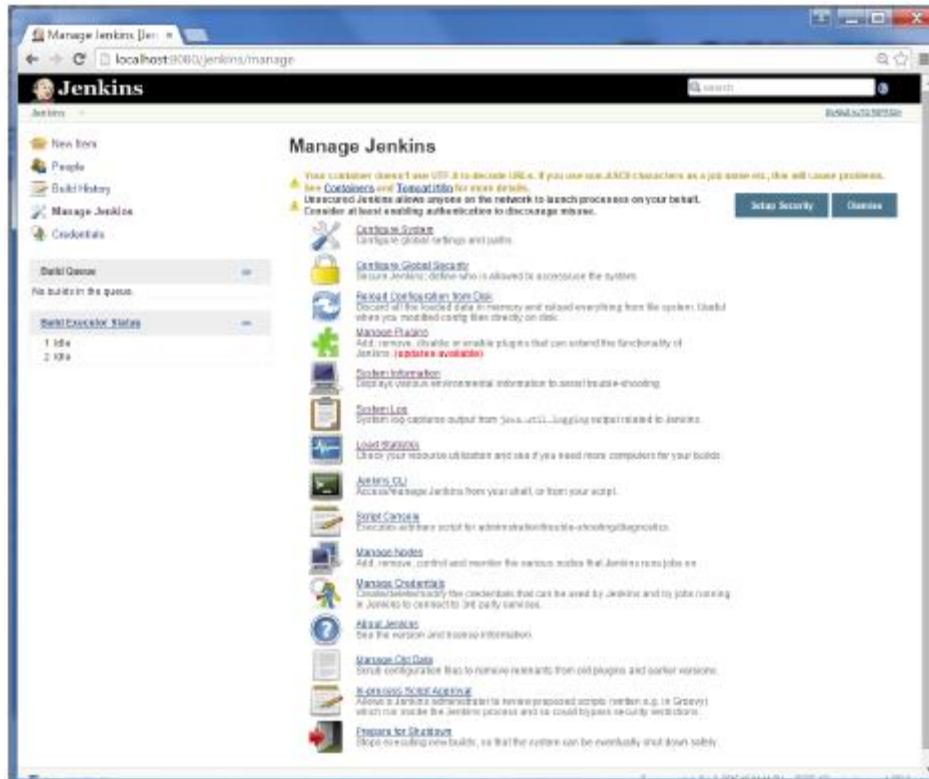
Once the file is downloaded, extract the files to the relevant application folder. For this purpose, the maven files will be placed in E:\Apps\apache-maven-3.3.3.

Step 2: Setting up Jenkins and Maven

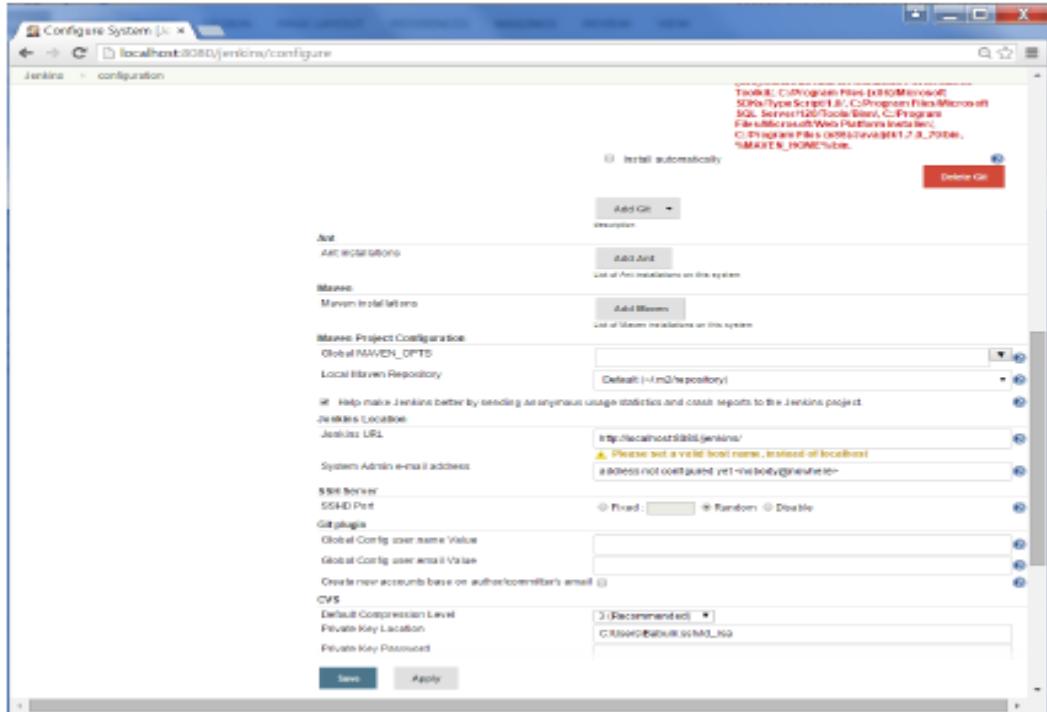
In the Jenkins dashboard (Home screen), click Manage Jenkins from the left-hand side menu.

The screenshot shows the Jenkins dashboard at <http://localhost:8080/jenkins/>. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. The main content area displays a 'Welcome to Jenkins!' message with a button to 'create new jobs'. It also shows sections for Build Queue (empty) and Build Executor Status (two idle executors). The bottom of the page includes a footer with links for Help, Jenkins Home, and Jenkins version information.

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



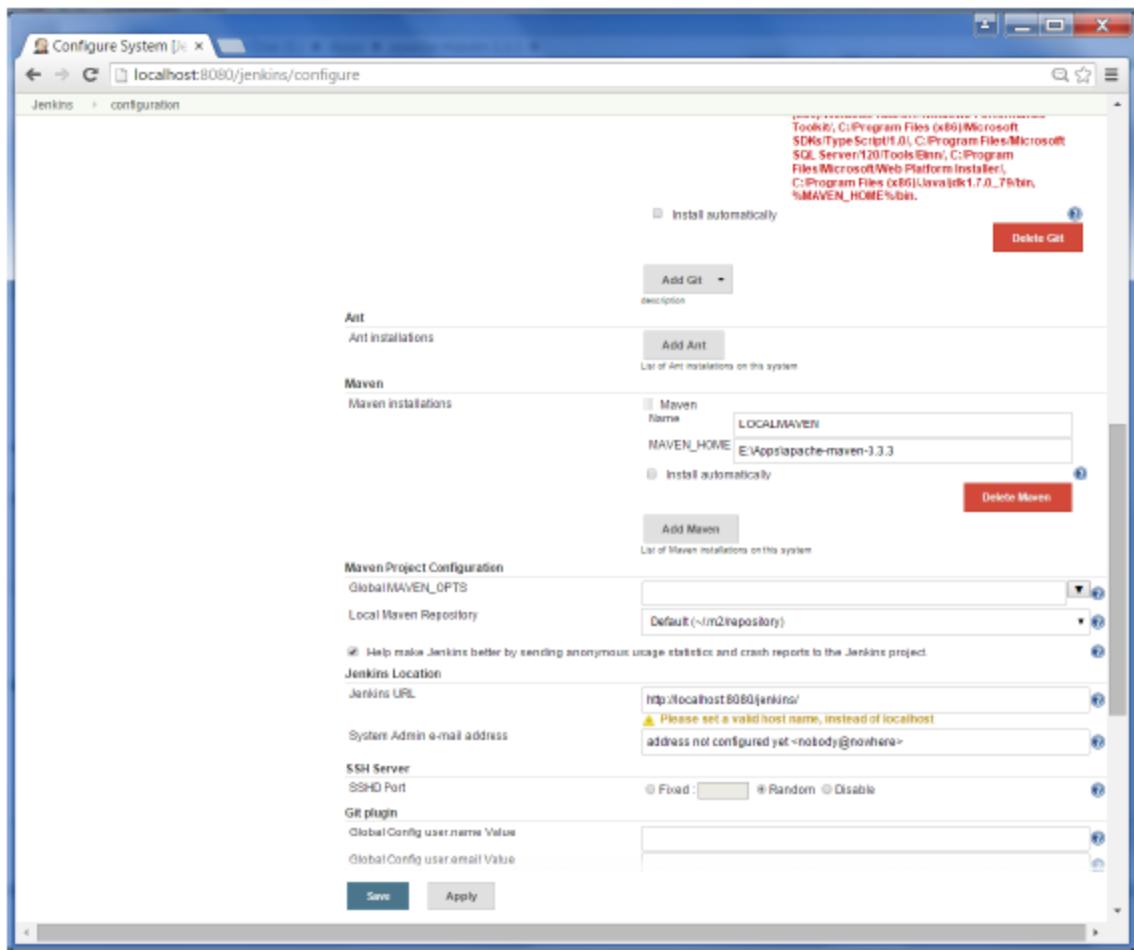
In the Configure system screen, scroll down till you see the Maven section and then click on the '**Add Maven**' button.



Uncheck the ‘Install automatically’ option.

Add any name for the setting and the location of the MAVEN_HOME.

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



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

Dashboard [Jenkins] x

localhost:8080/jenkins/

Jenkins

New item People Build History Manage Jenkins Credentials

Add description

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

Icon: ☀️

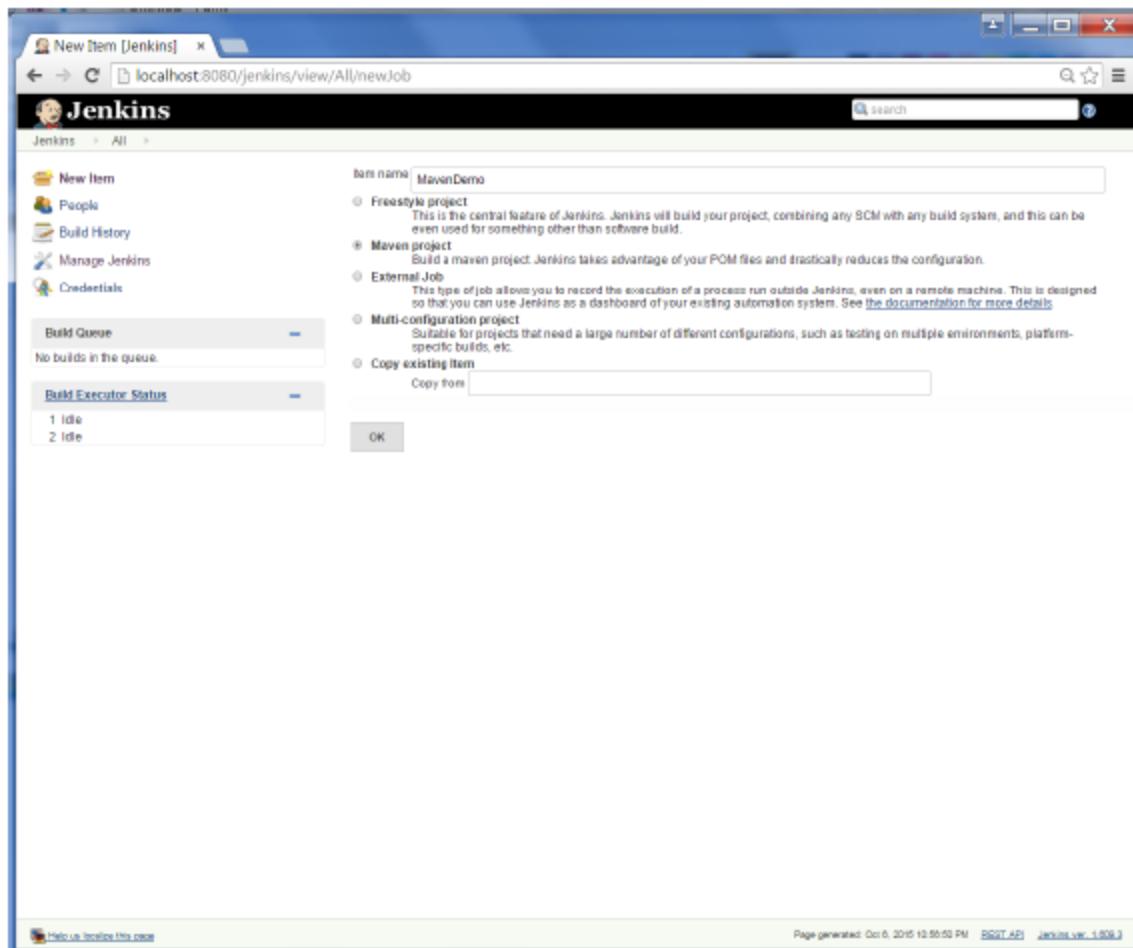
Legend: RSS for all RSS for failures RSS for just latest builds

Build Queue: No builds in the queue.

Build Executor Status: 1 Idle 2 Idle

Help us improve this page Page generated: Oct 6, 2015 12:55:57 PM REST API Jenkins ver. 1.509.3

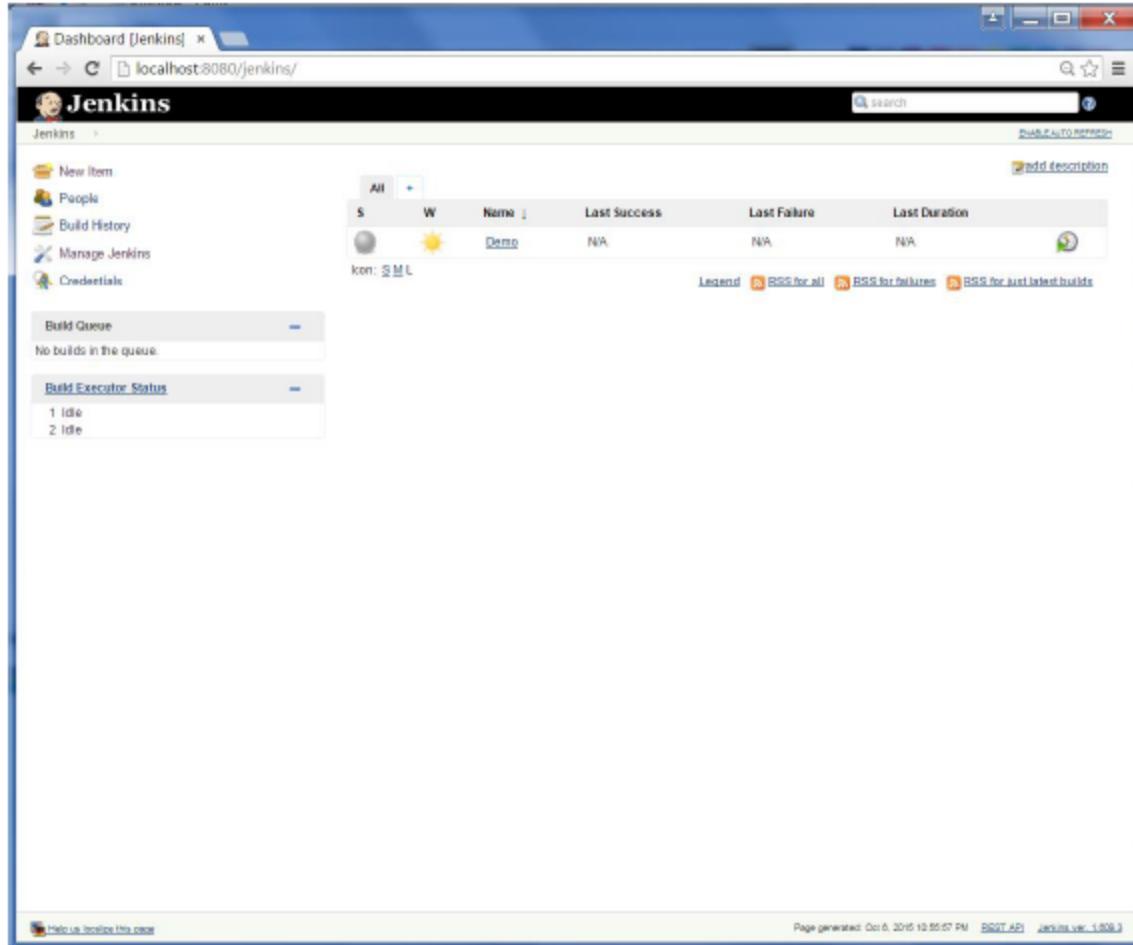
The screenshot shows the Jenkins dashboard at the URL `localhost:8080/jenkins/`. On the left, there's a sidebar with links for 'New item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. The main area has a table with columns for All, S, W, Name, Last Success, Last Failure, and Last Duration. A single row is present for the 'Demo' job, which has an icon of a sun, indicating it's healthy. Below the table is a legend for RSS feeds. On the left, there are two collapsed sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 Idle' and '2 Idle'). At the bottom, there's a footer with a link to help improve the page, the timestamp 'Page generated: Oct 6, 2015 12:55:57 PM', a 'REST API' link, and the Jenkins version 'Jenkins ver. 1.509.3'.



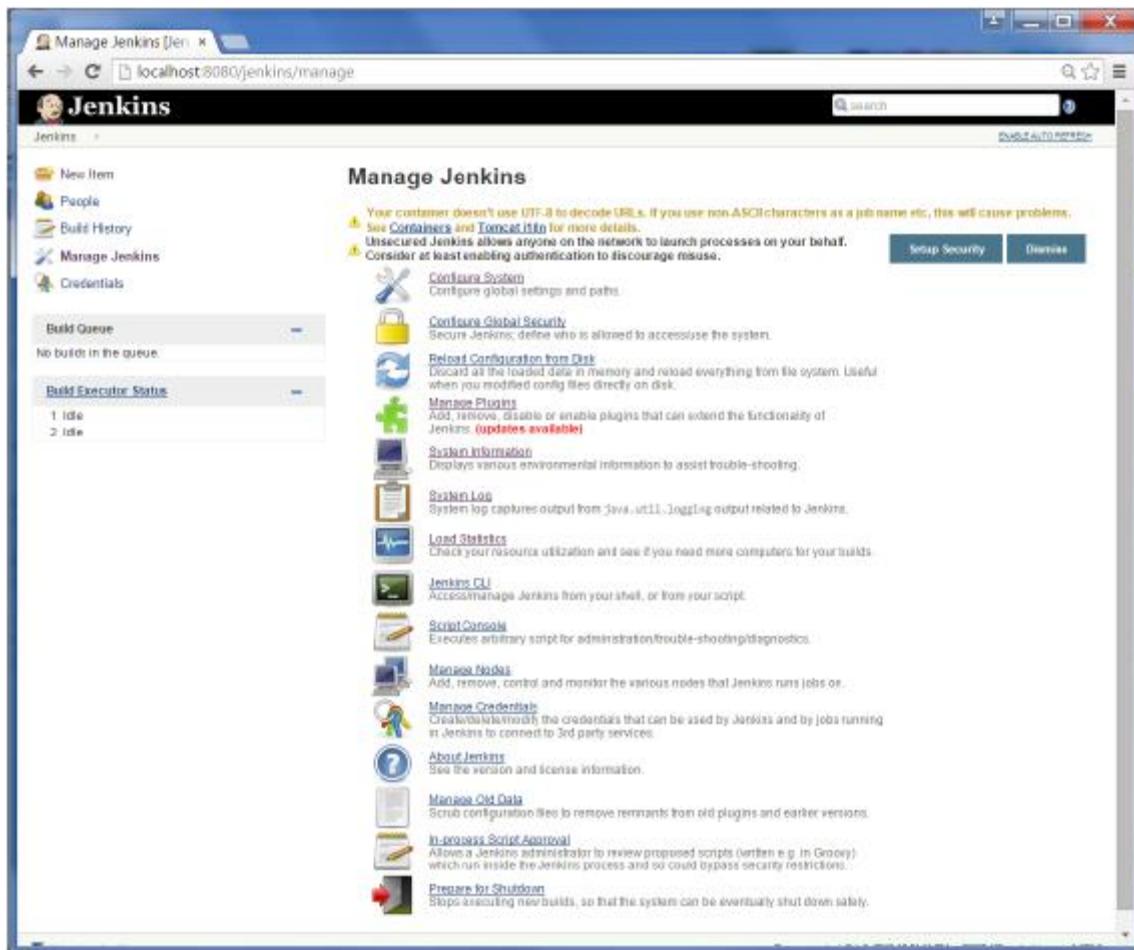
Jenkins - Configuration

You probably would have seen a couple of times in the previous exercises wherein we had to configure options within Jenkins. The following shows the various configuration options in Jenkins.

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



You will then be presented with the following screen –



Click on Configure system. Discussed below are some of the Jenkins configuration settings which can be carried out.

Jenkins Home Directory

Jenkins needs some disk space to perform builds and keep archives. One can check this location from the configuration screen of Jenkins. By default, this is set to **~/jenkins**, and this location will initially be stored **within your user profile location**. In a proper environment, you need to change this location to an adequate location to store all relevant builds and archives. Once can do this in the following ways

- Set "JENKINS_HOME" environment variable to the new home directory before launching the servlet container.
- Set "JENKINS_HOME" system property to the servlet container.
- Set JNDI environment entry "JENKINS_HOME" to the new directory.

The following example will use the first option of setting the "JENKINS_HOME" environment variable.

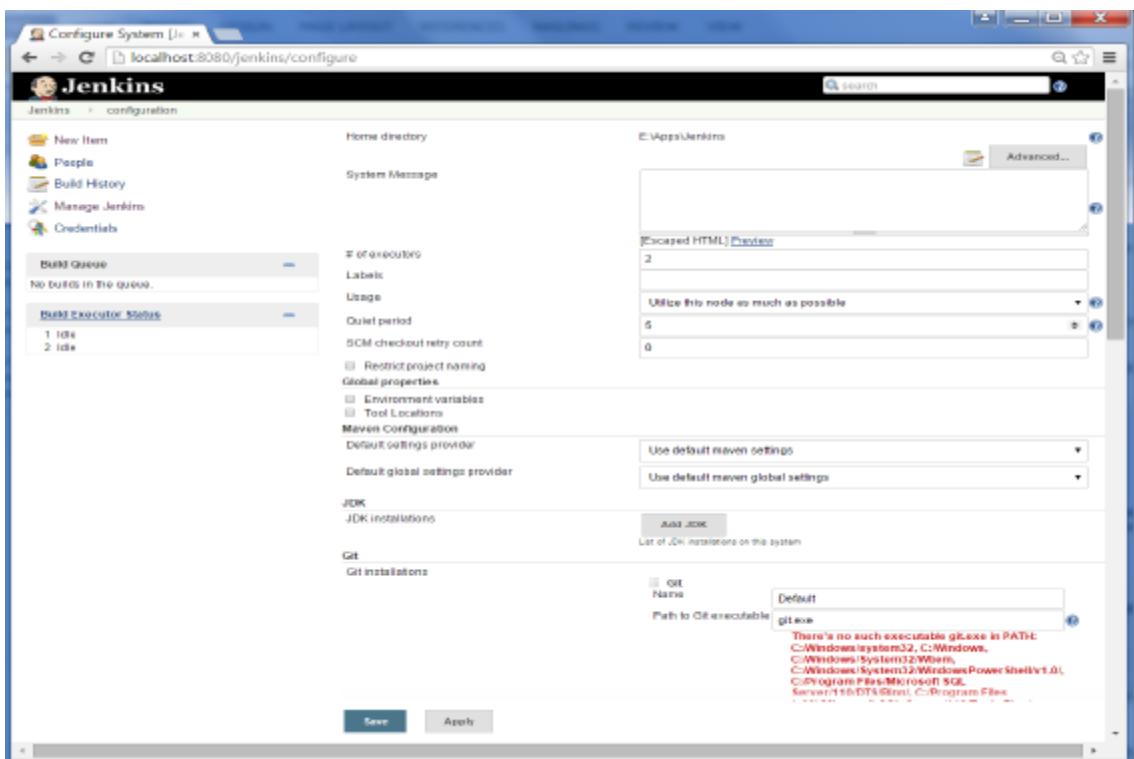
First create a new folder E:\Apps\Jenkins. Copy all the contents from the existing ~/.jenkins to this new directory.

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

OS	Output
Windows	Set Environmental variable JENKINS_HOME to you're the location you desire. As an example you can set it to E:\Apps\Jenkins
Linux	export JENKINS_HOME =/usr/local/Jenkins or the location you desire.

In the Jenkins dashboard, click Manage Jenkins from the left hand side menu. Then click on 'Configure System' from the right hand side.

In the Home directory, you will now see the new directory which has been configured.



of executors

This refers to the total number of concurrent job executions that can take place on the Jenkins machine. This can be changed based on requirements. Sometimes the recommendation is to keep this number the same as the number of CPU on the machines for better performance.

Environment Variables

This is used to add custom environment variables which will apply to all the jobs. These are key-value pairs and can be accessed and used in Builds wherever required.

Jenkins URL

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

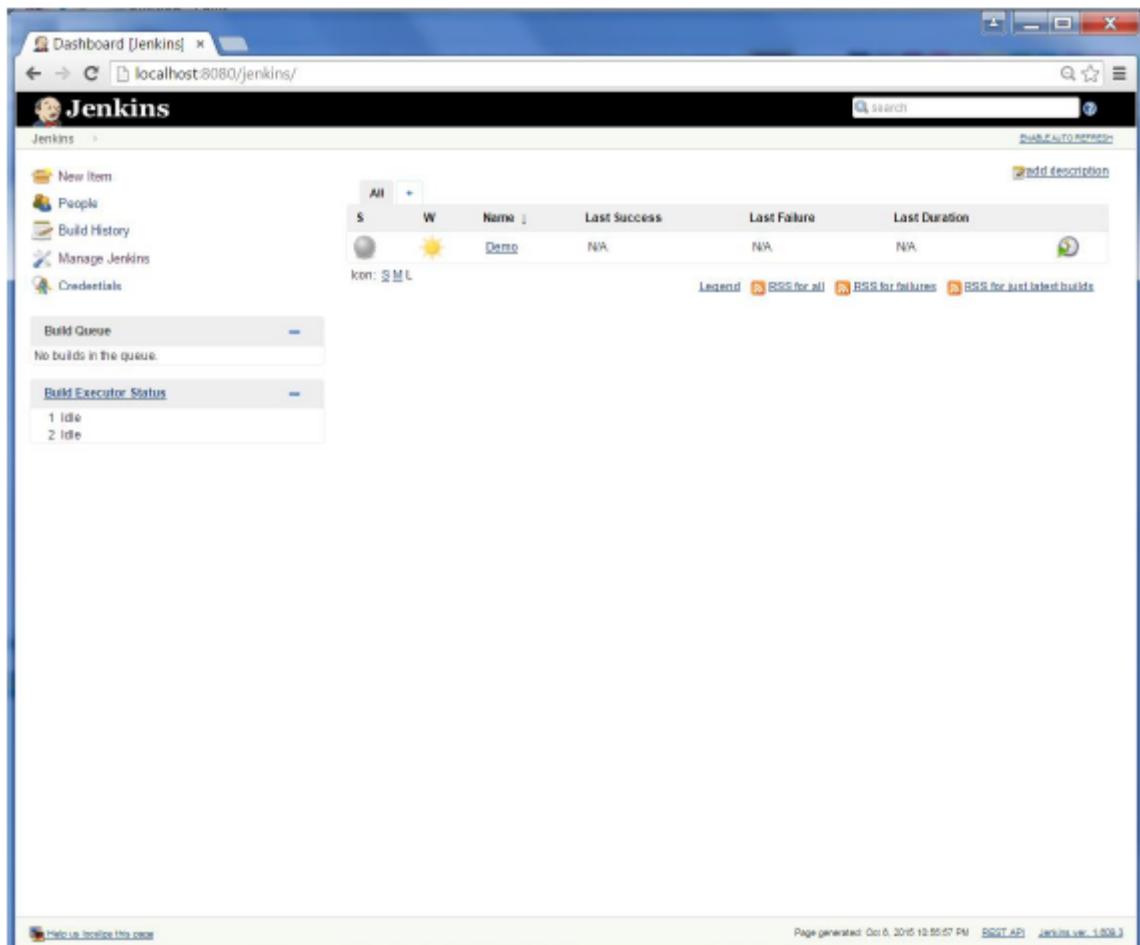
Email Notification

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

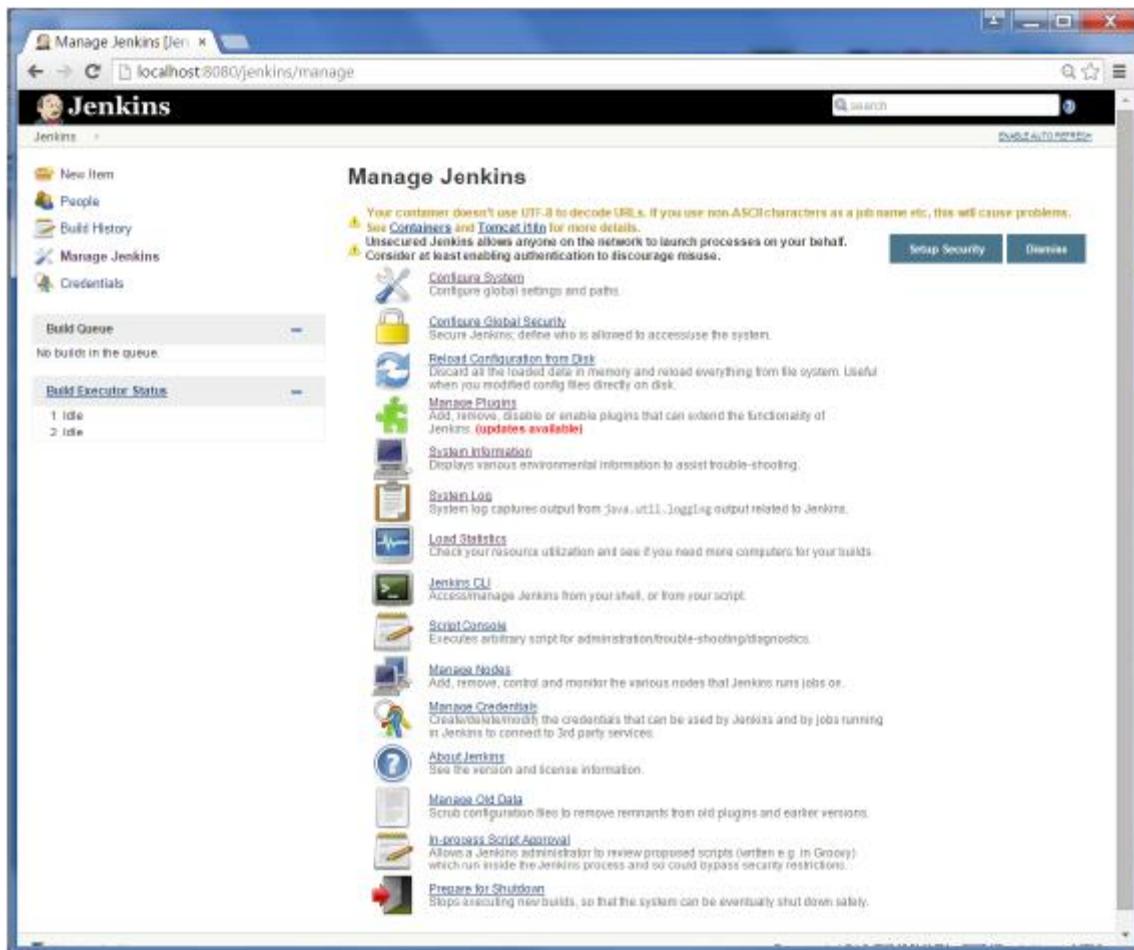
Jenkins - Management

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

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



You will then be presented with the following screen –



Some of the management options are as follows –

Configure System

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

Reload Configuration from Disk

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

Manage Plugin

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

The screenshot shows the Jenkins Update Center interface. At the top, there's a navigation bar with links for 'Back to Dashboard' and 'Manage Jenkins'. Below that is a search bar and a 'Filter' dropdown. The main area is titled 'Plugin Manager' and contains a table of available plugins. The table has columns for 'Name', 'Version', and 'Instaled'. A 'Updates' tab is selected. The table lists several plugins:

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

At the bottom of the screen, there are three buttons: 'Download now and install after restart', 'Update information obtained: 1 hr 36 min ago', and 'Check now'.

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
ant.toolkit	sun.awt.windows.WToolkit
catalina.base	E:\Apps\tomcat7
catalina.home	E:\Apps\tomcat7
catalina.useNaming	true
common.loader	\$catalina.base\$lib\$catalina.base\$lib\$jar;\$catalina.home\$lib;\$catalina.home\$lib\$jar
file.encoding	Cp1252
file.encoding.pkg	sun.io
file.separator	\
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.path	E:\Apps\tomcat7\bin\bootstrap.jar;E:\Apps\tomcat7\bin\tomcat-juli.jar;5.1.0;E:\Apps\tomcat7\endorsed
java.endorsed.dirs	C:\Program Files (x86)\Java\jdk1.7.0_79\jre\lib\ext;C:\Windows\Sun\Java\lib\ext
java.home	C:\Program Files (x86)\Java\jdk1.7.0_79\jre
java.io.tmpdir	E:\Apps\tomcat7\temp
java.library.path	C:\Program Files (x86)\Java\jdk1.7.0_79\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Windows\system32;C:\Windows;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\10.0\Tools\Binn\;C:\Program Files\Microsoft SQL Server\10.0\Tools\Binn\ManagementStudio\;C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\;C:\Program Files\Microsoft\Windows Performance Toolkit\10\;C:\Program Files (x86)\Windows Kits\8.1\Windows Performance Toolkit\;C:\Program Files (x86)\Microsoft SDKs\TypeScript\1.0\;C:\Program Files\Microsoft SQL Server\12.0\Tools\Binn\;C:\Program Files\Microsoft\Web Platform Installer\;C:\Program Files (x86)\Java\jdk1.7.0_79\bin;%MAVEN_HOME%\bin..
java.naming.factory.initial	org.apache.naming.java.javaURLContextFactory
java.naming.factory.url.pkgs	org.apache.naming
java.runtime.name	Java(TM) SE Runtime Environment
java.runtime.version	1.7.0_79-b15
java.specification.name	Java Platform API Specification
java.specification.vendor	Oracle Corporation
java.specification.version	1.7
java.util.logging.config.file	E:\Apps\tomcat7\conf\logging.properties
java.util.logging.manager	org.apache.juli.ClassLoaderLogManager
java.vendor	Oracle Corporation
java.vendor.url	http://java.oracle.com/
java.vendor.url_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.

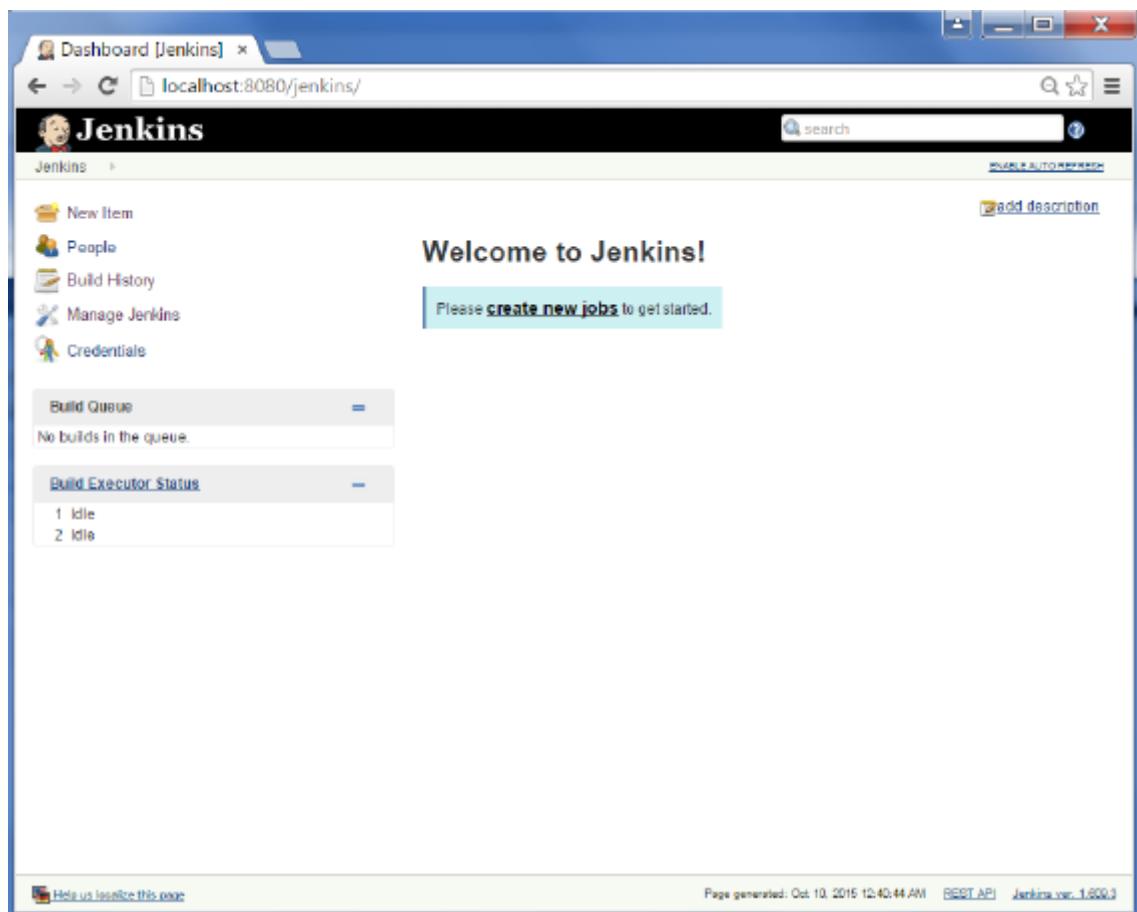
Prepare for Shutdown

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.

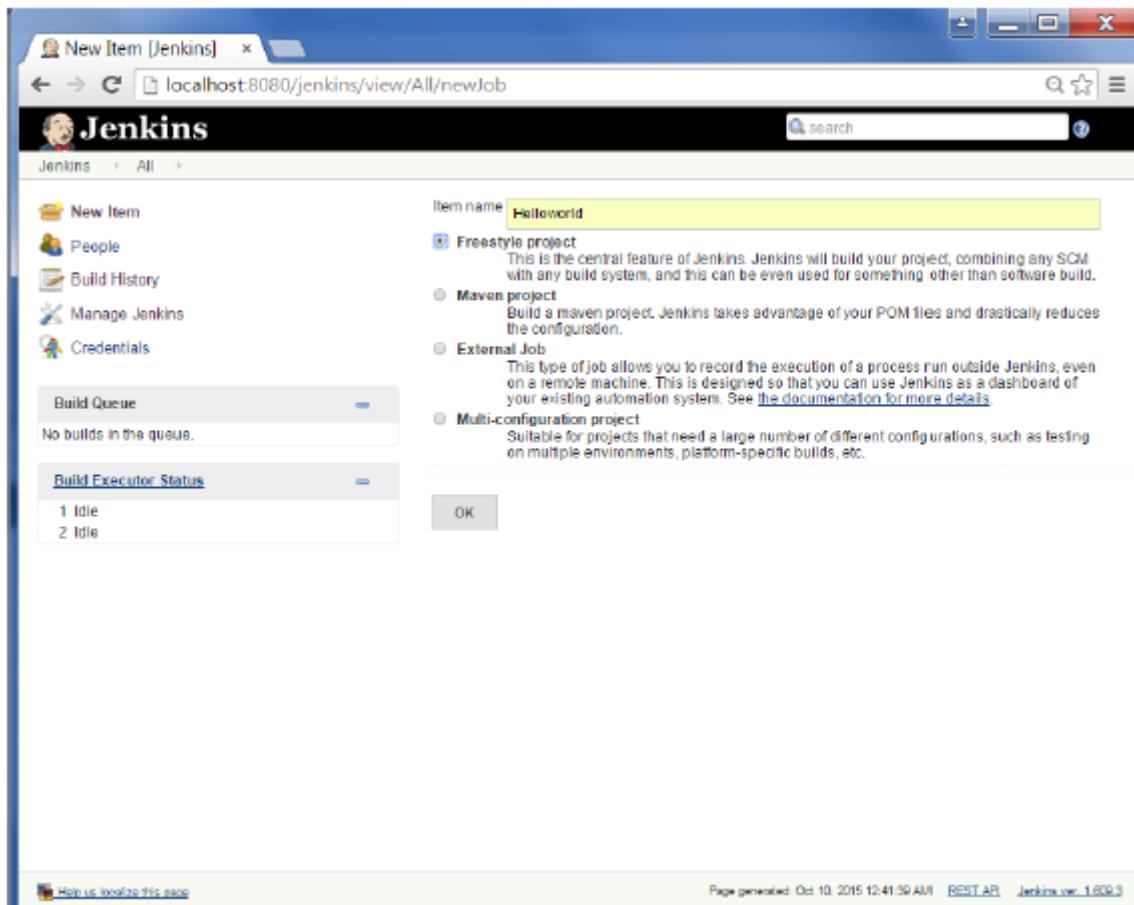
Jenkins - Setup Build Jobs

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

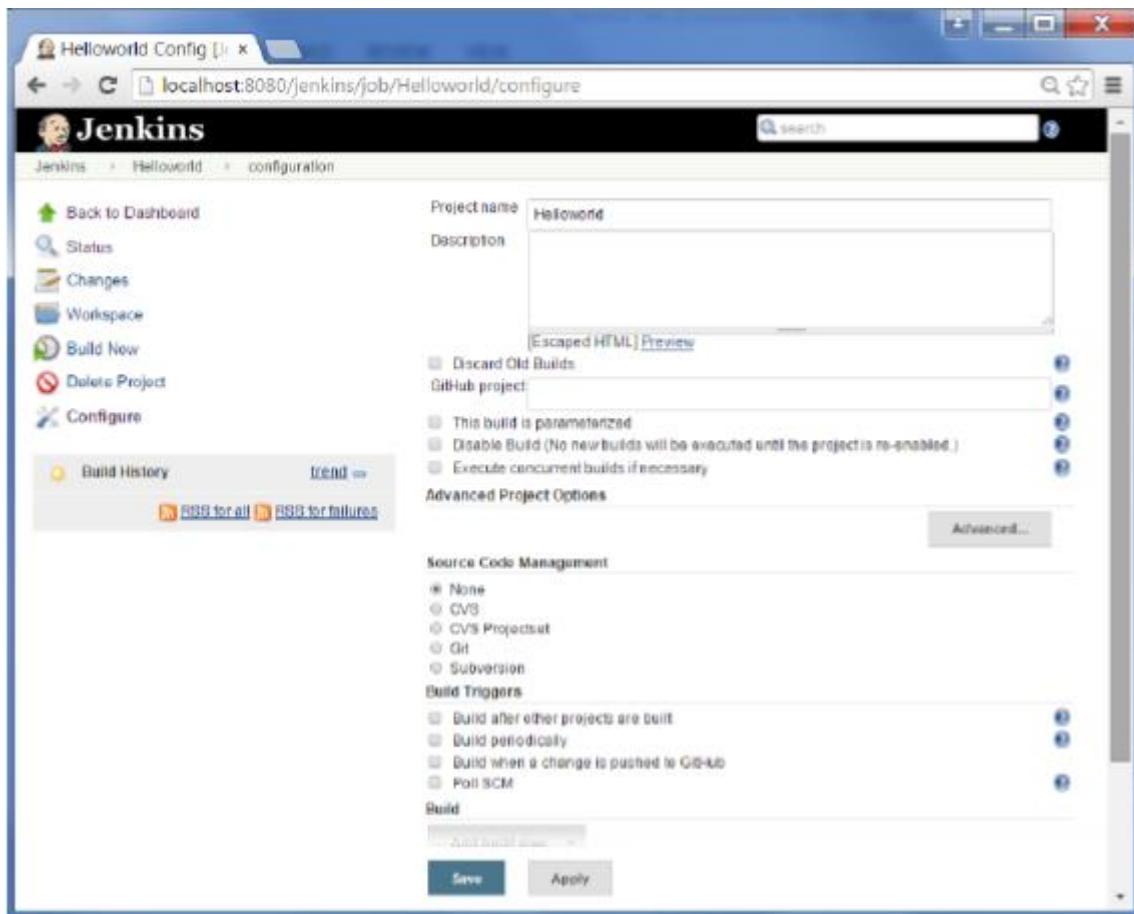
Step 1 – Go to the Jenkins dashboard and Click on New Item



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

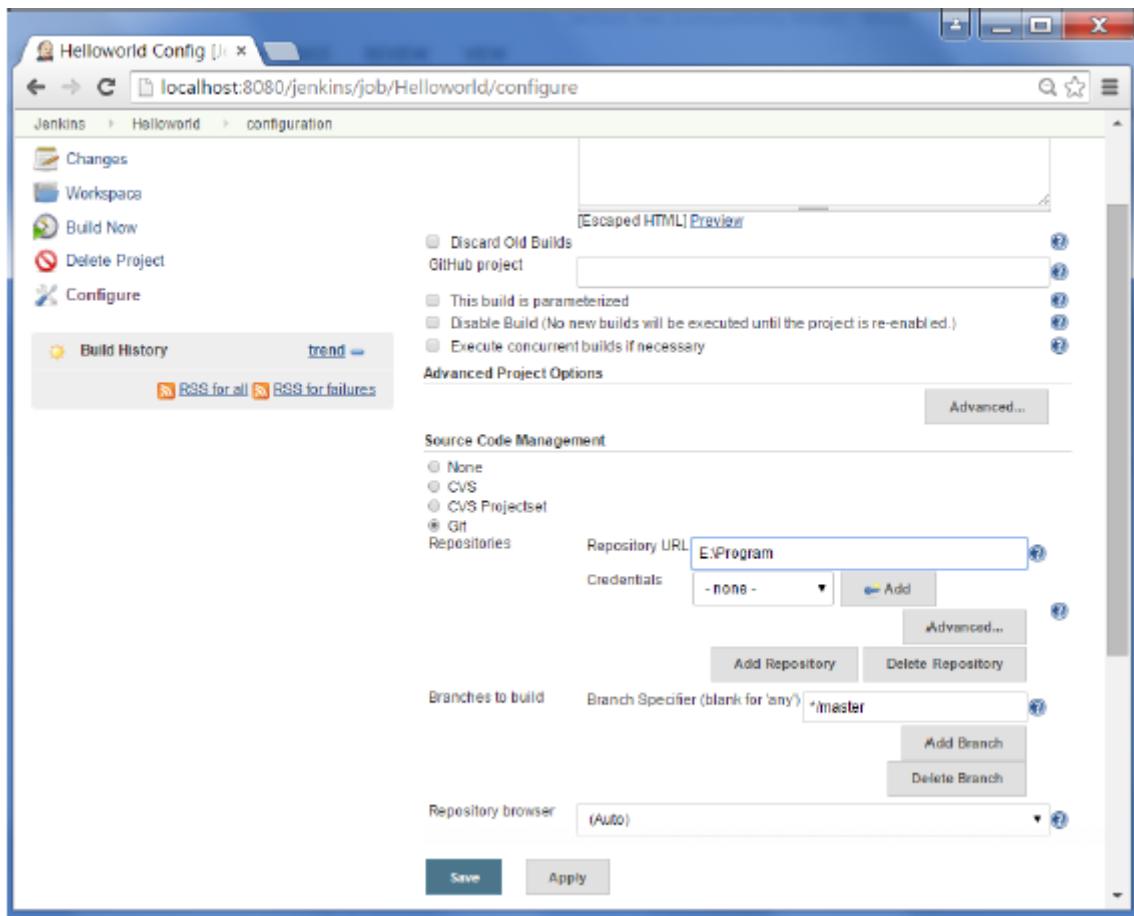


Step 3 – The following screen will come up in which you can specify the details of the job.

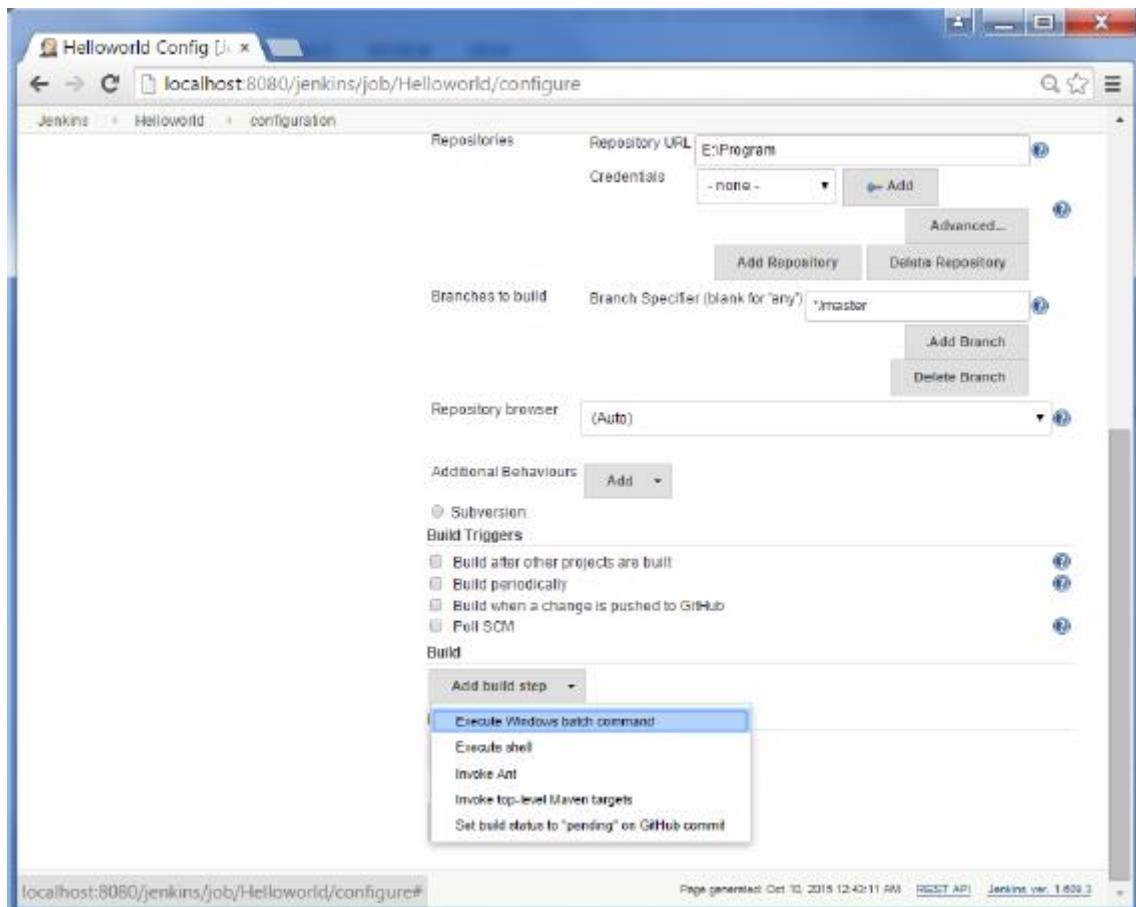


Step 4 – We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a ‘HelloWorld.java’ file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

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

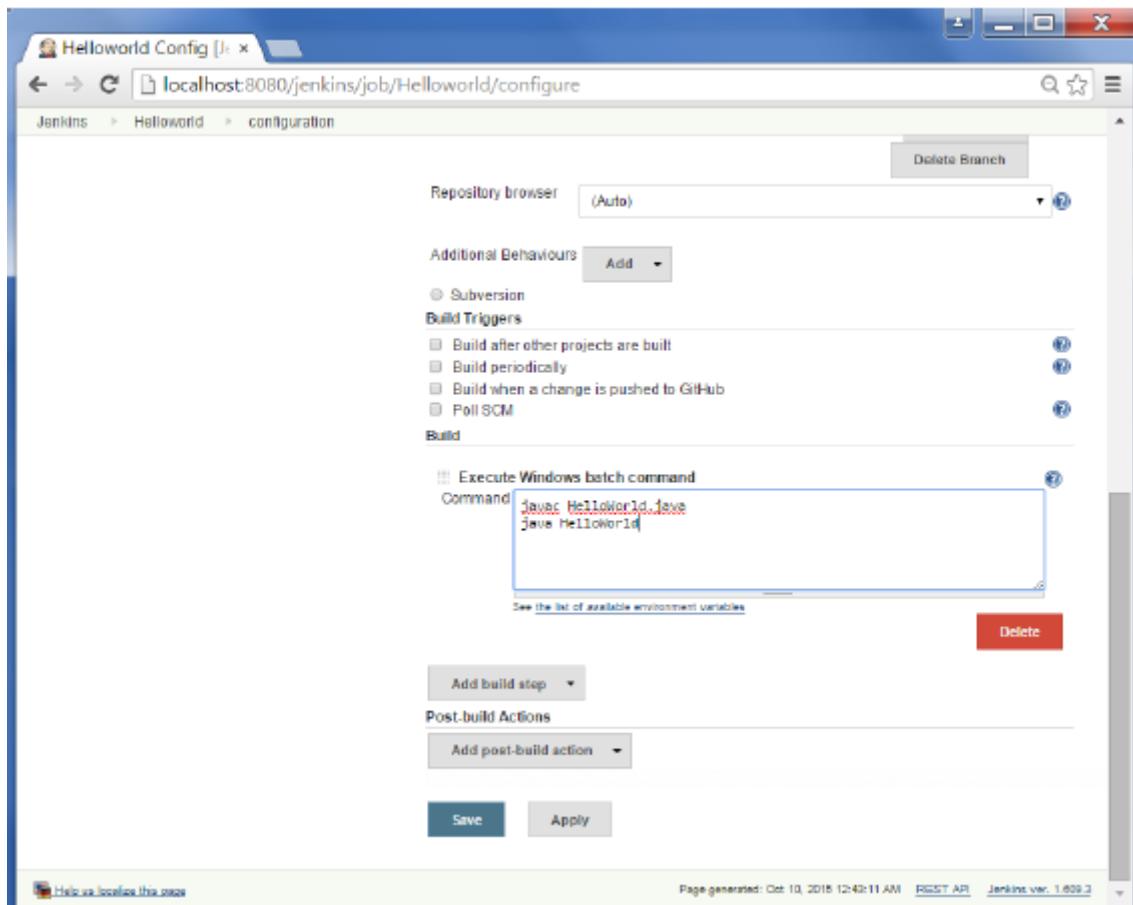


Step 5 – Now go to the Build section and click on Add build step → Execute Windows batch command



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

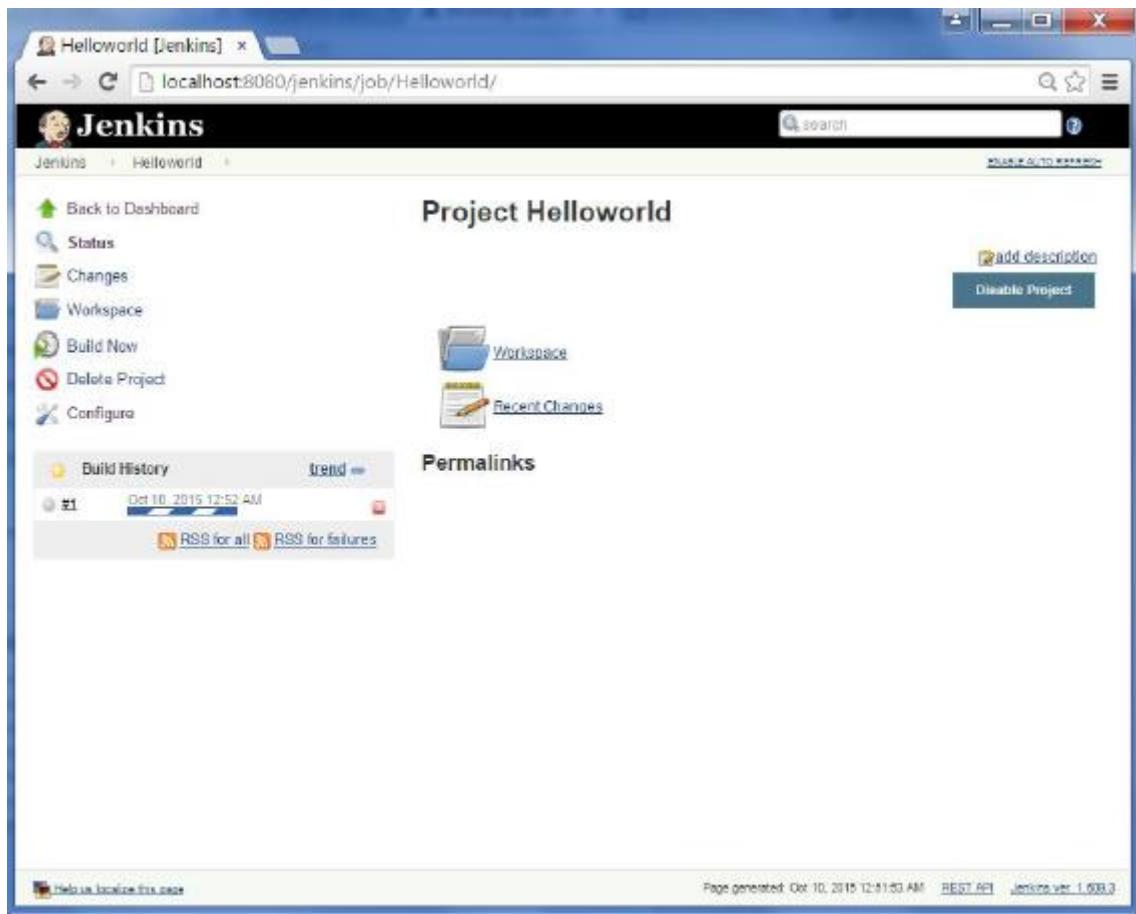
```
Java HelloWorld.java  
Java HelloWorld
```



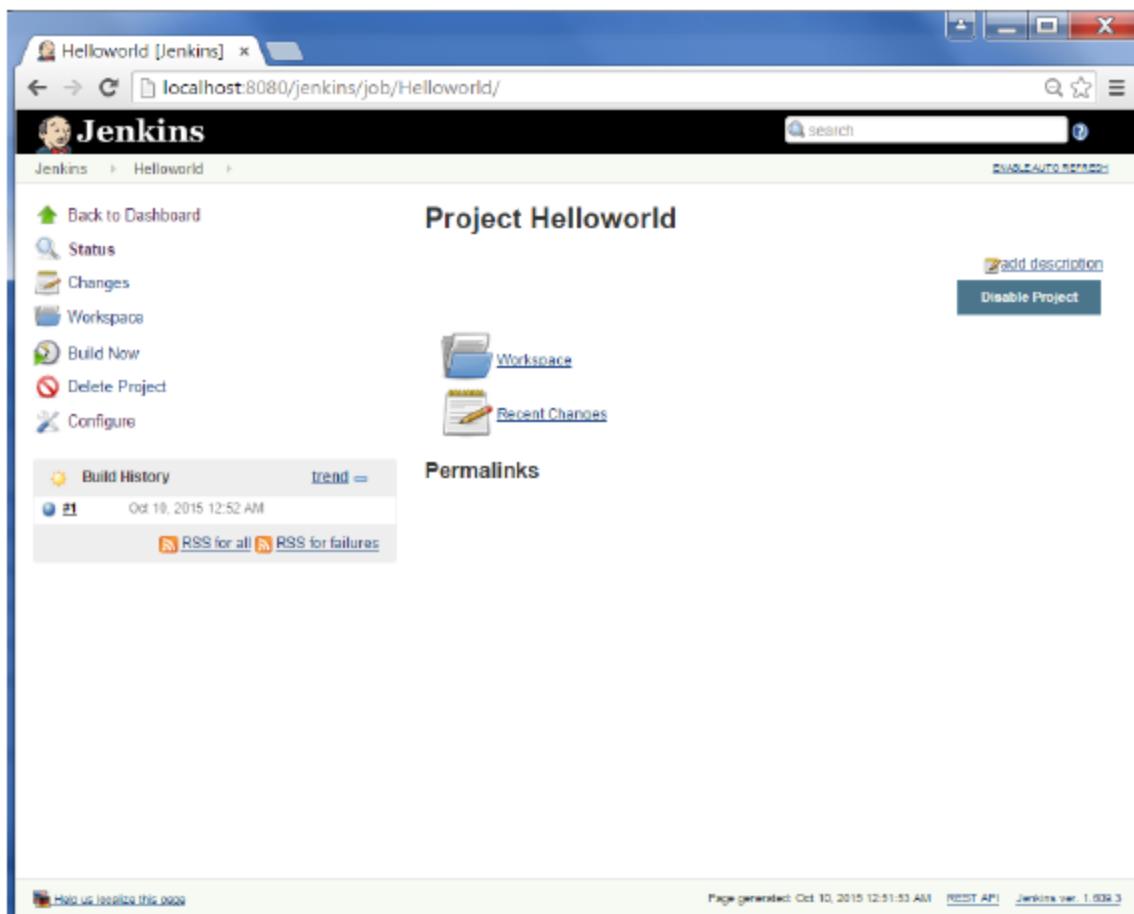
Step 7 – Once saved, you can click on the Build Now option to see if you have successfully defined the job.

The screenshot shows a web browser window with the title "Helloworld [Jenkins]". The address bar displays "localhost:8080/jenkins/job/Helloworld/". The main content area is titled "Project Helloworld". On the left, there is a sidebar with links: "Back to Dashboard", "Status", "Changes", "Workspace", "Build Now", "Delete Project", and "Configure". Below these are buttons for "Build History" (highlighted in yellow), "trend", "RSS for all", and "RSS for failures". The right side of the page features two buttons: "Workspace" (with a folder icon) and "Recent Changes" (with a document icon). At the bottom, there is a "Permalinks" section and footer text including "Help us localize this page", "Page generated: Oct 10, 2016 12:01:03 AM", "REST API", and "Jenkins ver. 1.600.3".

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



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



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

Helloworld #1 [Jenkins] x

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

Jenkins

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

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

Started 4 min 40 sec ago Took 4.7 sec

No changes.

Started by anonymous user

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

Add description

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

Helloworld #12 Cons x

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

Jenkins

Back to Project Status Changes Console Output View as plain text Edit Build Information Delete Build Git Build Data No Tags Previous Build

Console Output

```
Started by user anonymous
Building in workspace E:\Jenkins\jobs\Helloworld\workspace
> C:\Program Files\Git\bin\git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url E:\Program #
timeout=10
Fetching upstream changes from E:\Program
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> C:\Program Files\Git\bin\git.exe -c core.askpass=true fetch --tags --progress
E:\Program\+refs/heads/*:refs/remotes/origin/*
> C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/master^{commit}" # timeout=10
> C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/master^{commit}" # timeout=10
Checking out Revision 42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
(refs/heads/master)
> C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -
42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
> C:\Program Files\Git\bin\git.exe rev-list
42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f # timeout=10
[workspace] $ cmd /c call E:\Apps\tomcat7\temp\nudson1928478766077504601.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
```

Help us localize this page Page generated: Oct 10, 2015 10:14:21 PM REST API Jenkins ver. 1.609.3

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

Jenkins - Unit Testing

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

xUnit Plugin - Jenkins

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

Dashboard > Jenkins > Plugins > xUnit Plugin

Browse Search

 xUnit Plugin

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

Jenkins

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

Documents

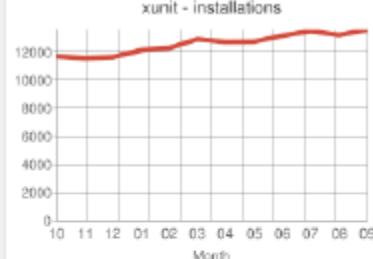
- Meet Jenkins
- Use Jenkins
- Extend Jenkins
- Plugins
- Servlet Container Notes

Plugin Information

Plugin ID	xunit	Changes	In Latest Release Since Latest Release
Latest Release	1.98 (archives)	Source Code	GitHub
Latest Release Date	Oct 09, 2015	Issue Tracking	Open Issues
Required Core Dependencies	1.680.1 junit (version: 1.6)	Pull Requests	Pull Requests
Maintainer(s)	Gregory Boissinot (id: gboissinot)		

Usage

xunit - installations



Month	Installations
Oct 2014	11692
Nov 2014	11557
Dec 2014	11631
Jan 2015	12106
Feb 2015	12262
Mar 2015	12891
Apr 2015	12694
May 2015	12716
Jun 2015	13143
Jul 2015	13470
Aug 2015	13192
Sep 2015	13563

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

CppUnit output



Features

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

Supported tools

Embedded tools

- * [JUnit itself](#)
- * [AUnit](#)
- * [MSTest](#) (imported from [MSTest Plugin](#))
- * [NUnit](#) (imported from [NUnit Plugin](#))
- * [UnitTest++](#)
- * [Boost Test Library](#)
- * [PHPUnit](#)
- * [Free Pascal Unit](#)
- * [CppUnit](#)
- * [MbUnit](#)
- * [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

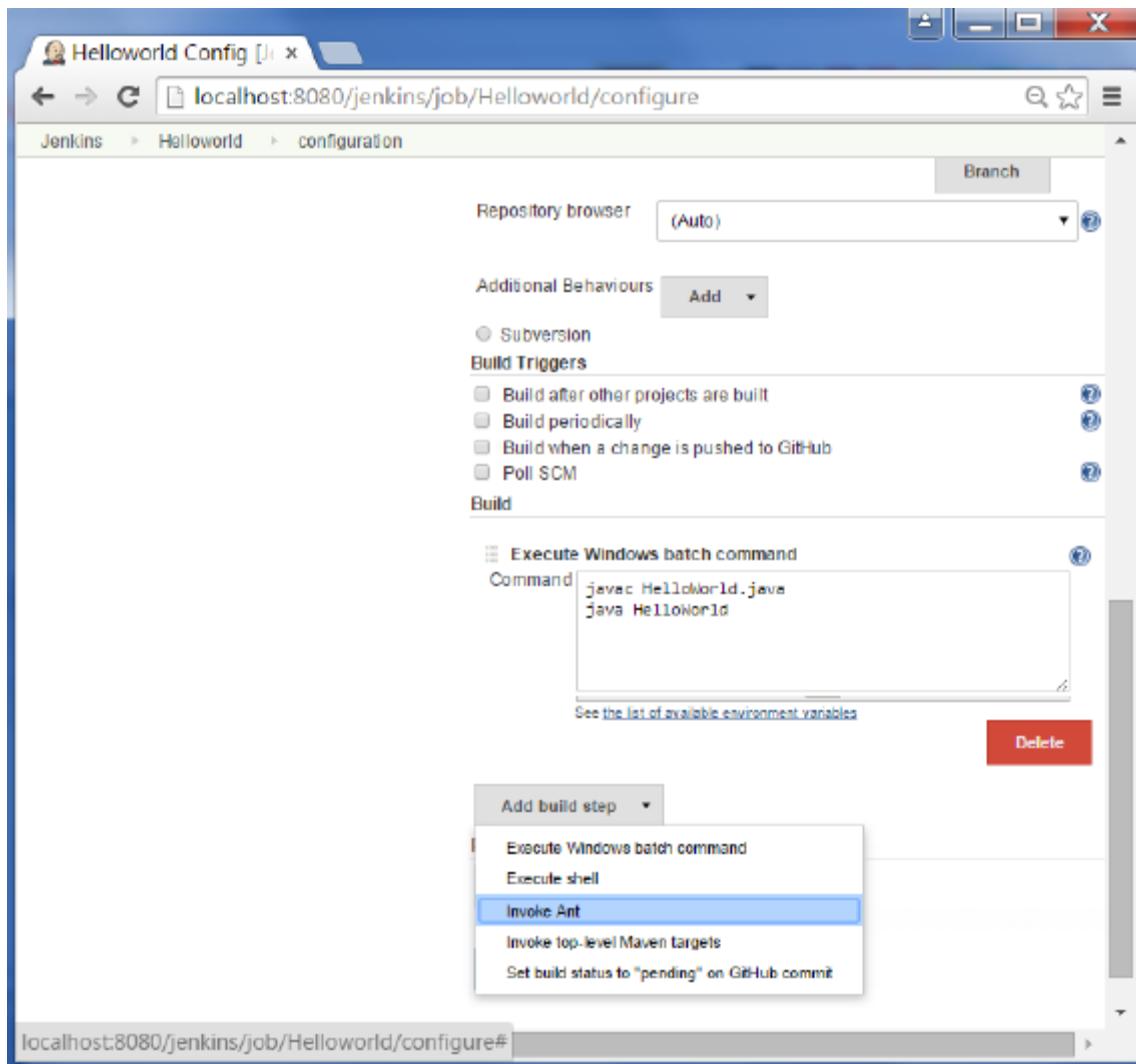
The following example will consider

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

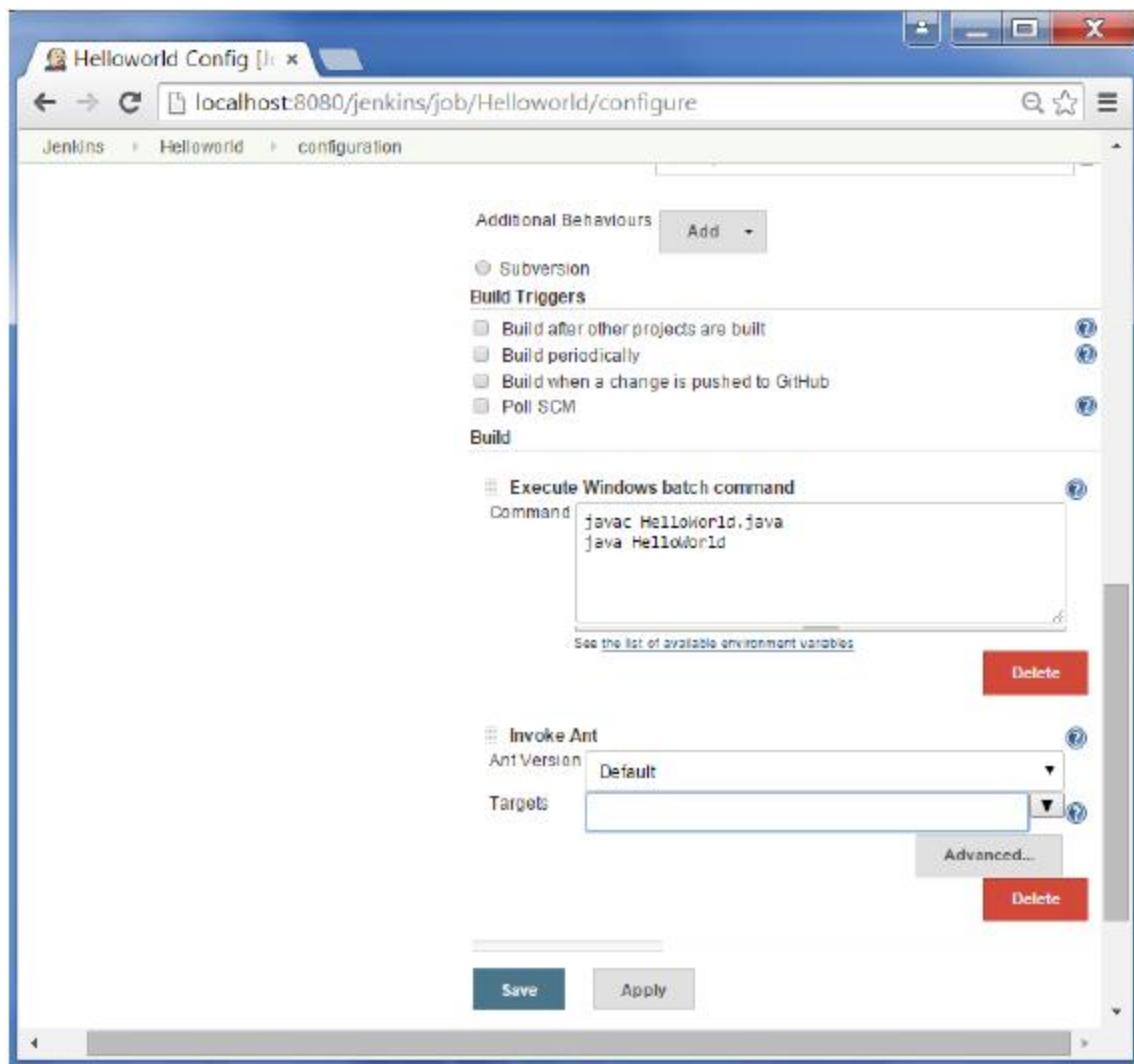
Step 1 – Go to the Jenkins dashboard and Click on the existing HelloWorld project and choose the Configure option

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. 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' (showing 'master' with 1 idle and 2 idle executors, and 'build_slave' which is offline). The main area displays a table of projects. One project, 'HelloWorld', is selected, showing its last build information: '5 sec - #11' (Last Success), '2 days 23 hr - #10' (Last Failure), and '3.2 sec' (Last Duration). A context menu is open over the 'HelloWorld' row, with options: 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure' (which is highlighted with a blue background).

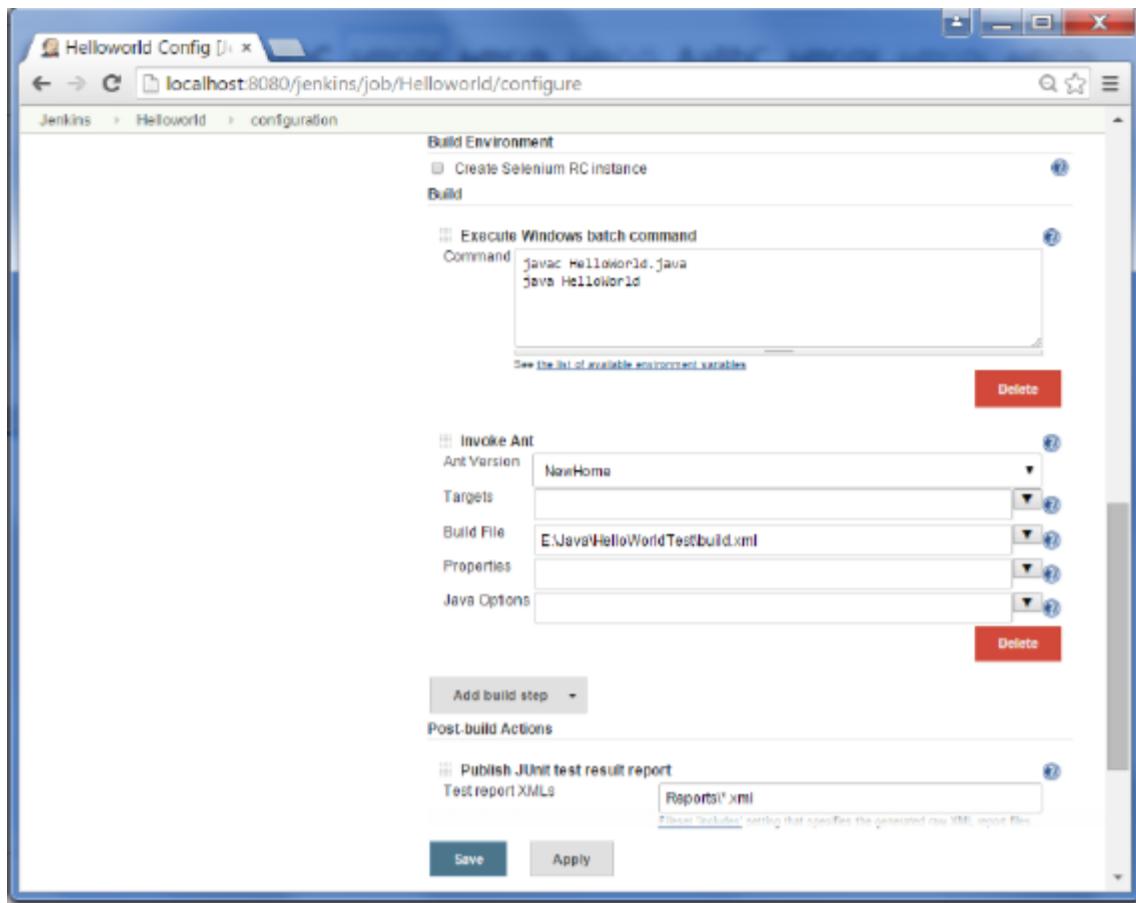
Step 2 – Browse to the section to Add a Build step and choose the option to Invoke Ant.



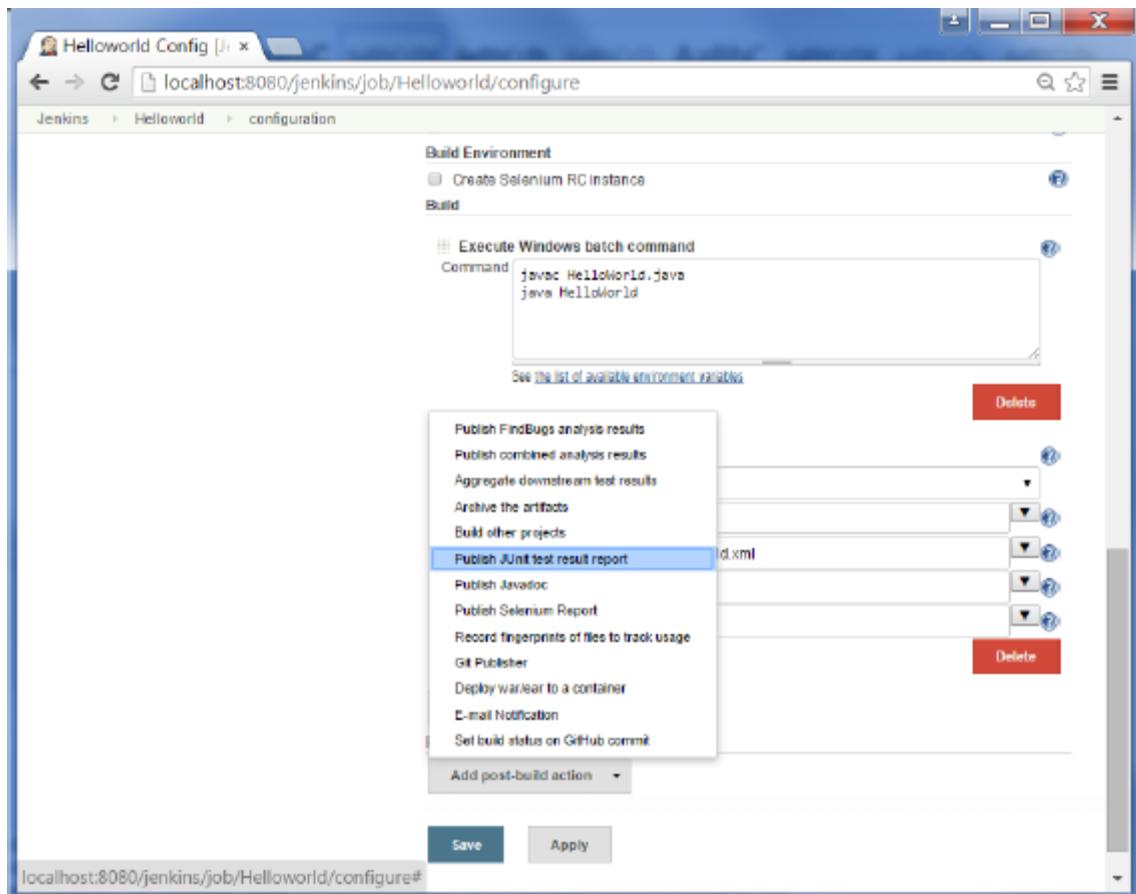
Step 3 – Click on the Advanced button.



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

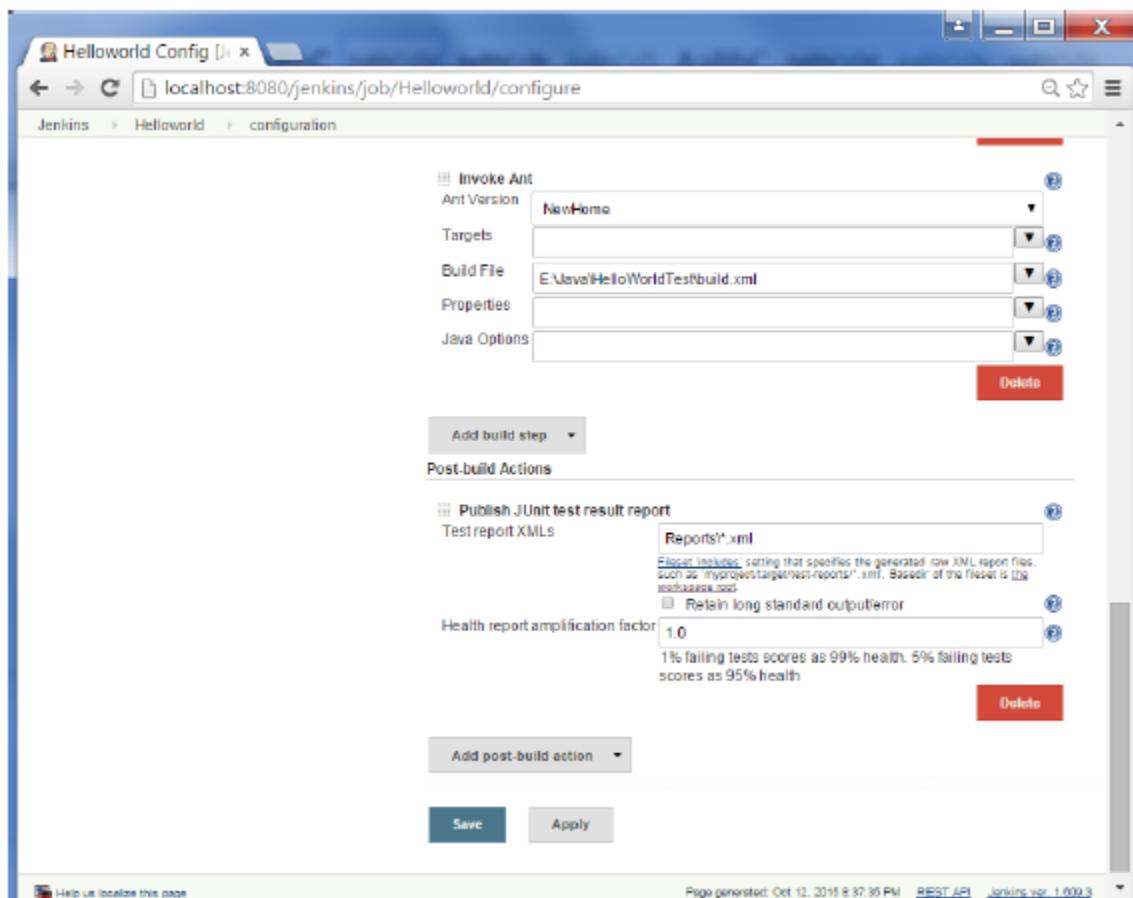


Step 5 – Next click the option to Add post-build option and choose the option of “Publish Junit test result report”



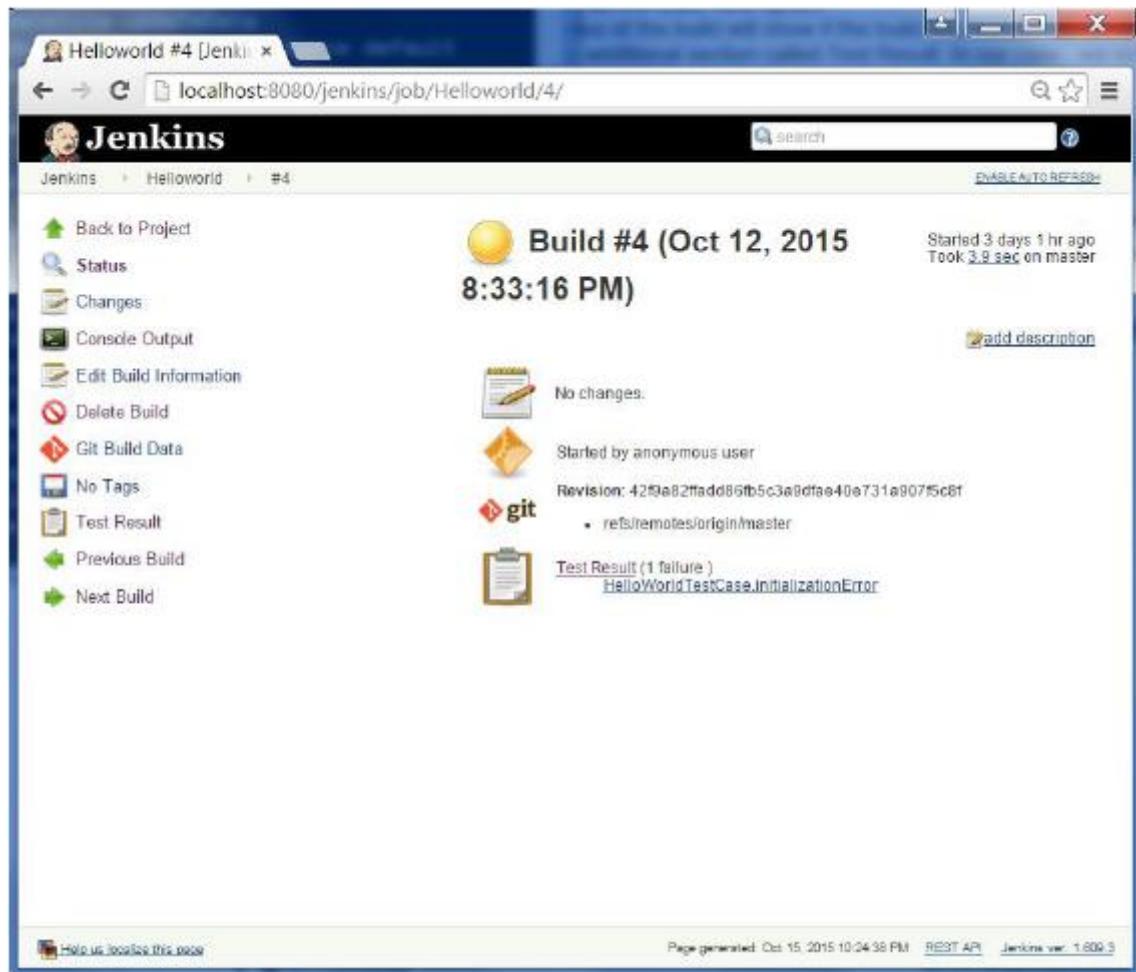
Step 6 – In the Test reports XML’s, enter the location as shown below. Ensure that Reports is a folder which is created in the HelloWorld project workspace. The “*.xml” basically tells Jenkins to pick up the result xml files which are produced by the running of the Junit test cases. These xml files which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

Once done, click the Save option at the end.



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

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



The screenshot shows the Jenkins interface for a build named "Helloworld #4". The main title is "Build #4 (Oct 12, 2015) 8:33:16 PM". Key details include "Started 3 days 1 hr ago" and "Took 3.3 sec on master". A sidebar on the left lists options like Back to Project, Status, Changes, Console Output, Edit Build Information, Delete Build, Git Build Data, No Tags, Test Result, Previous Build, and Next Build. The central area displays build artifacts: "No changes.", "Started by anonymous user", "Revision: 429a82ffadd85fb5c3a9d9a40a731a8075c8f", and a "git" icon. Below these are "Test Result (1 failure)" and a link to "HelloWorldTestCase.InitializationError". The bottom of the page includes links for "Help us localize this page", "Page generated: Oct 15, 2015 10:24:38 PM", "REST API", and "Jenkins ver: 1.609.3".

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

The screenshot shows a Jenkins Test Result page for a build named "Helloworld #4 Test R". The URL is localhost:8080/jenkins/job/Helloworld/4/testReport/. The main title is "Test Result" with a subtitle "1 failures". A summary bar indicates "1 tests Took 10 ms". Below this, there is a section titled "All Failed Tests" with a single entry: "HelloWorldTestCase InitializationError" (Duration: 10 ms, Age: 1). There is also a section titled "All Tests" with a table showing the following data:

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

At the bottom of the page, there are links for "Help us localize this page", "Page generated: Oct 12, 2015 8:45:49 PM", "REST API", and "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 at localhost:8080/jenkins/manage. The left sidebar includes links for New Item, People, Build History, Manage Jenkins (which is selected), and Credentials. Under Manage Jenkins, there are sections for Build Queue (empty) and Build Executor Status (2 idle). The main content area is titled 'Manage Jenkins' and contains several configuration links:

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

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

Jenkins Plugin Manager

Available

Name	Version
Selenium Auto Exec Server(AES) plugin	0.5
Hudson Selenium plugin	0.4
Selenium HTML report	0.94
TestingBot plugin	1.11
TestLink Plugin	3.10
Nemvana Plugin for Jenkins	1.02.06
Sauce OnDemand plugin	1.141
Selenium Builder plugin	1.14
SeleniumRC plugin	2.2

Install without restart Download now and install after restart Update Information obtained

Step 3 – Go to Configure system.

Manage Jenkins

Your container doesn't use UTF-8 to decode URLs. If you use non-ASCII characters as a job name etc., this will cause problems. See [Containers](#) and [Tomcat HTTP](#) for more details.

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

ENABLE AUTO REFRESH

Configure System

Configure Global Security

Reload Configuration from Disk

Manage Plugins

System Information

System Log

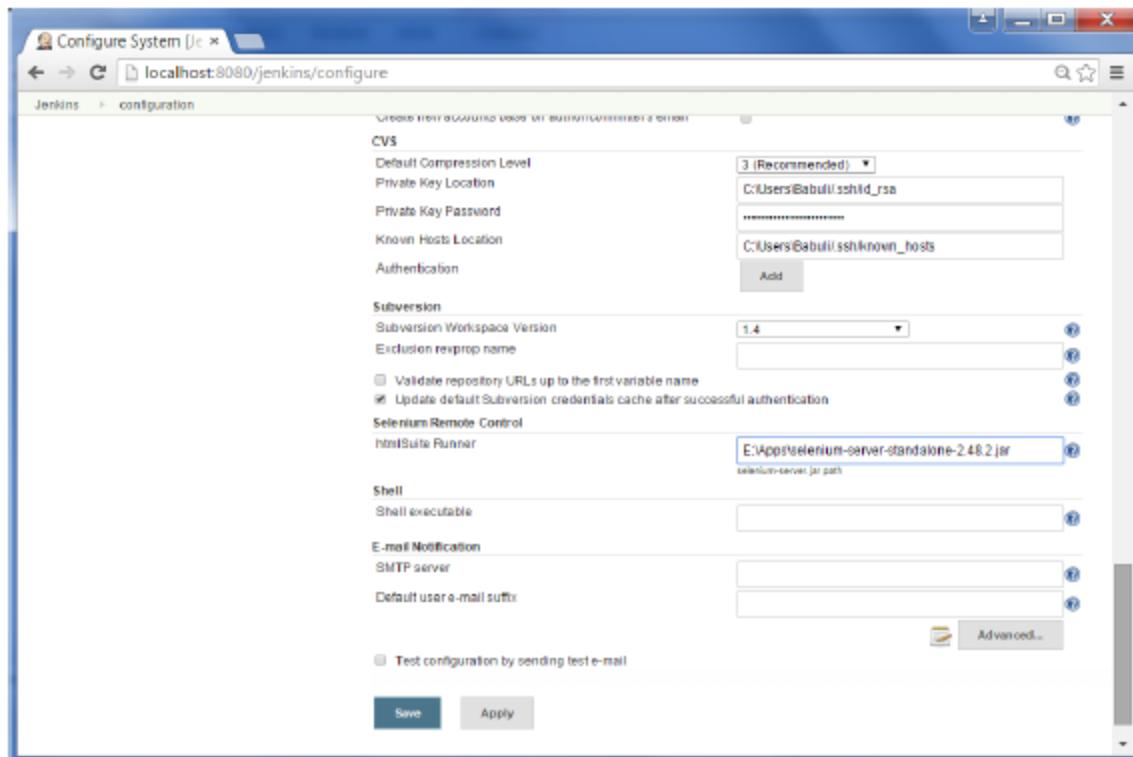
Load Statistics

Jenkins CLI

Script Console

Manage Nodes

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 a web browser window with the URL www.seleniumhq.org/download/. The page is titled "SeleniumHQ Browser Automation". The main content area is titled "Downloads". It contains sections for "Selenium Downloads", "Latest Releases", "Previous Releases", "Source Code", and "Maven Information". Below these is a "Donate to Selenium" section with a "Donate" button and payment method icons (PayPal, VISA, MasterCard). Another section discusses "through sponsorship" and links to the "Selenium project". The "Selenium Sponsors" section lists "See who supports the Selenium project." At the bottom, there is a "BrowserStack" logo and a note about language bindings.

SeleniumHQ Browser Automation

edit this page search selenium: Go

Projects Download Documentation Support About

Selenium Downloads

Latest Releases

Previous Releases

Source Code

Maven Information

Downloads

Below is where you can find the latest releases of all the Selenium components. You can also find a list of [previous releases](#), [source code](#), and additional information for [Maven users](#) (Maven is a popular Java build tool).

Selenium Standalone Server

The Selenium Server is needed in order to run either Selenium RC style scripts or Remote Selenium WebDriver ones. The 2.x server is a drop-in replacement for the old Selenium RC server and is designed to be backwards compatible with your existing infrastructure.

[Download version 2.48.2](#)

To use the Selenium Server in a Grid configuration [see the wiki page](#).

The Internet Explorer Driver Server

This is required if you want to make use of the latest and greatest features of the WebDriver InternetExplorerDriver. Please make sure that this is available on your \$PATH (or %PATH% on Windows) in order for the IE Driver to work as expected.

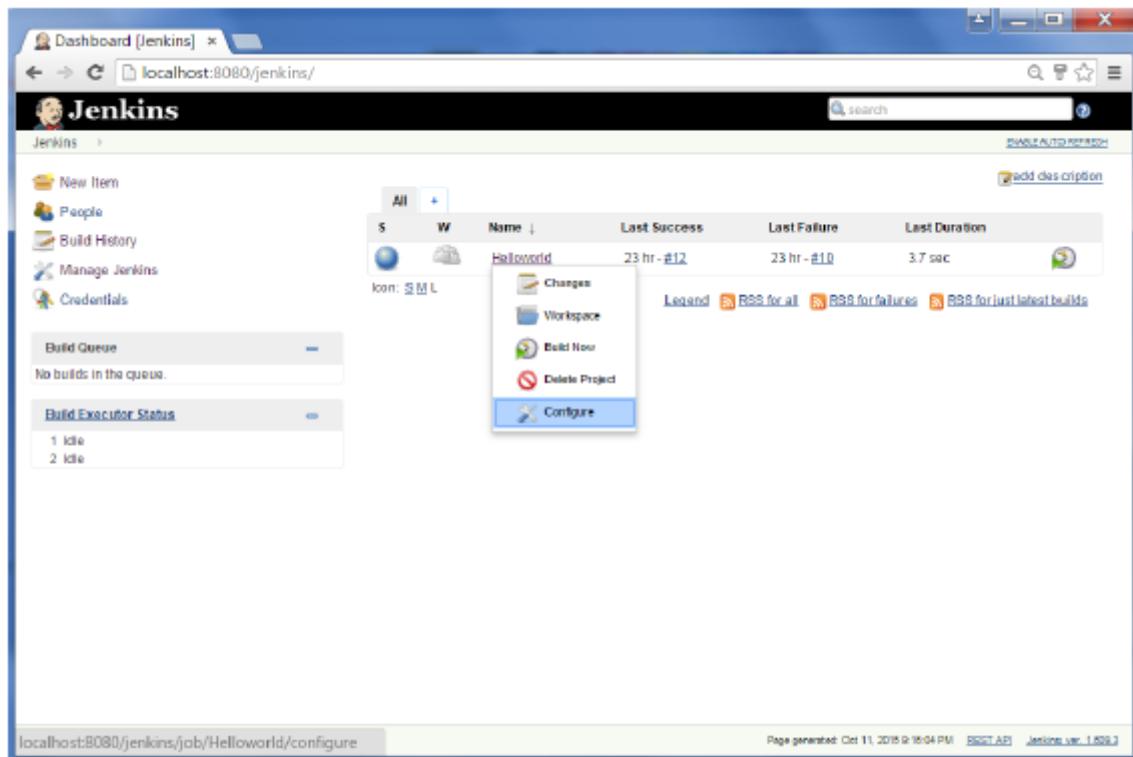
[Download version 2.48.0 for \(recommended\) 32 bit Windows IE or 64 bit Windows IE CHangelog](#)

Selenium Client & WebDriver Language Bindings

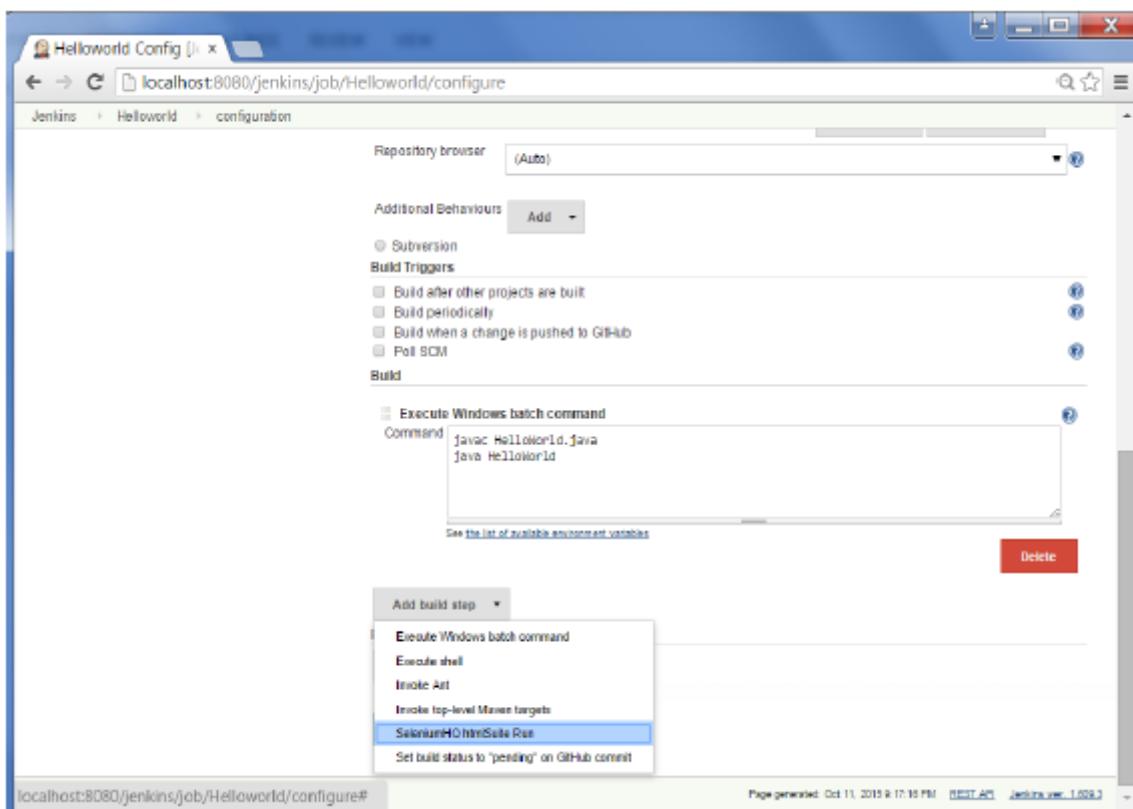
In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

BrowserStack While language bindings for [other languages exist](#), these are the core ones that are supported by the main project hosted on [google code](#).

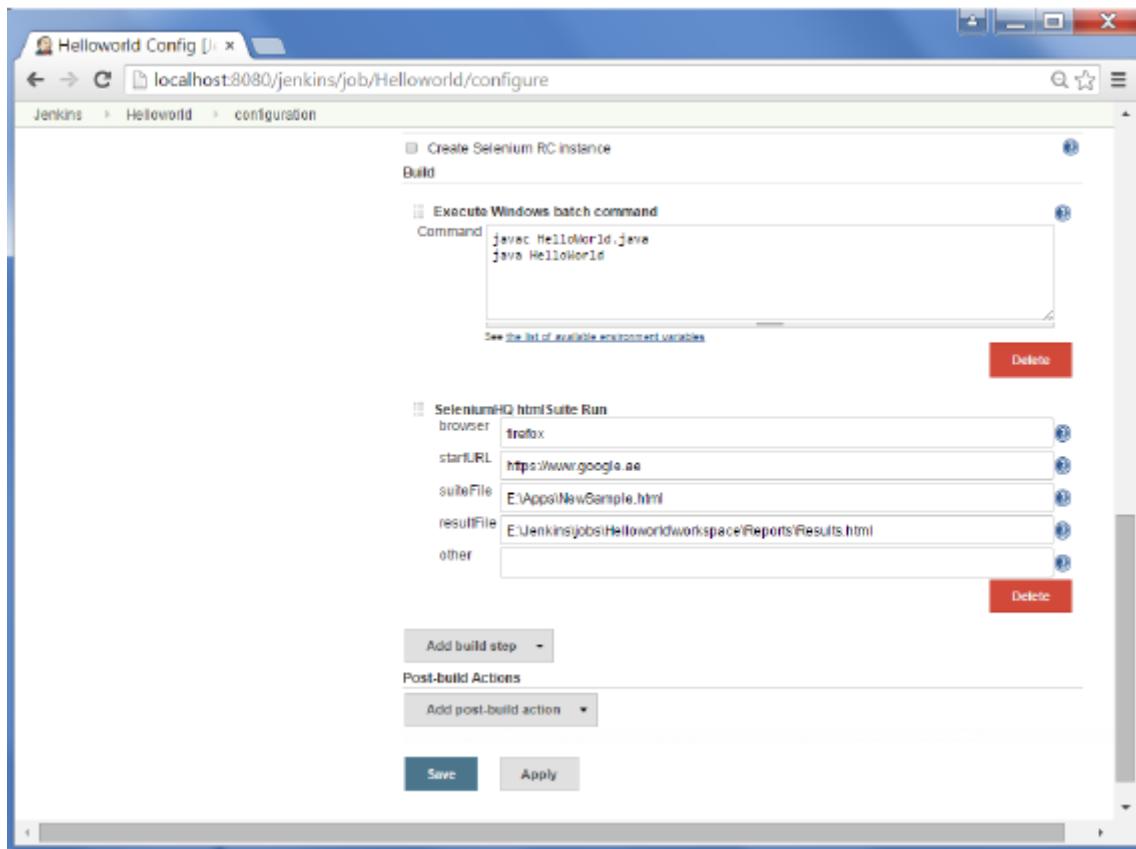
Step 5 – Go back to your dashboard and click on the Configure option for the HelloWorld project.



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



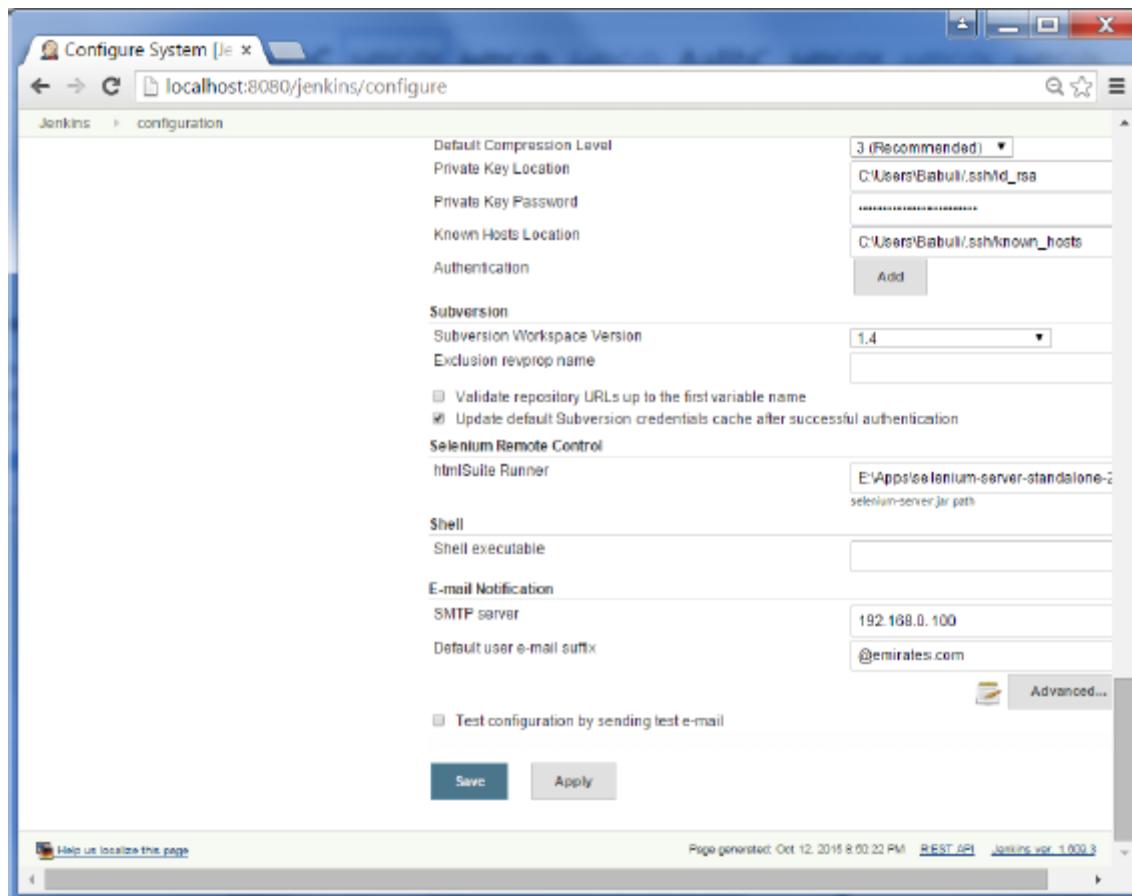
Step 7 – Add the necessary details for the selenium test. Here the suiteFile is the TestSuite generated by using the Selenium IDE. Click on Save and execute a build. Now the post build will launch the selenium driver, and execute the html test.



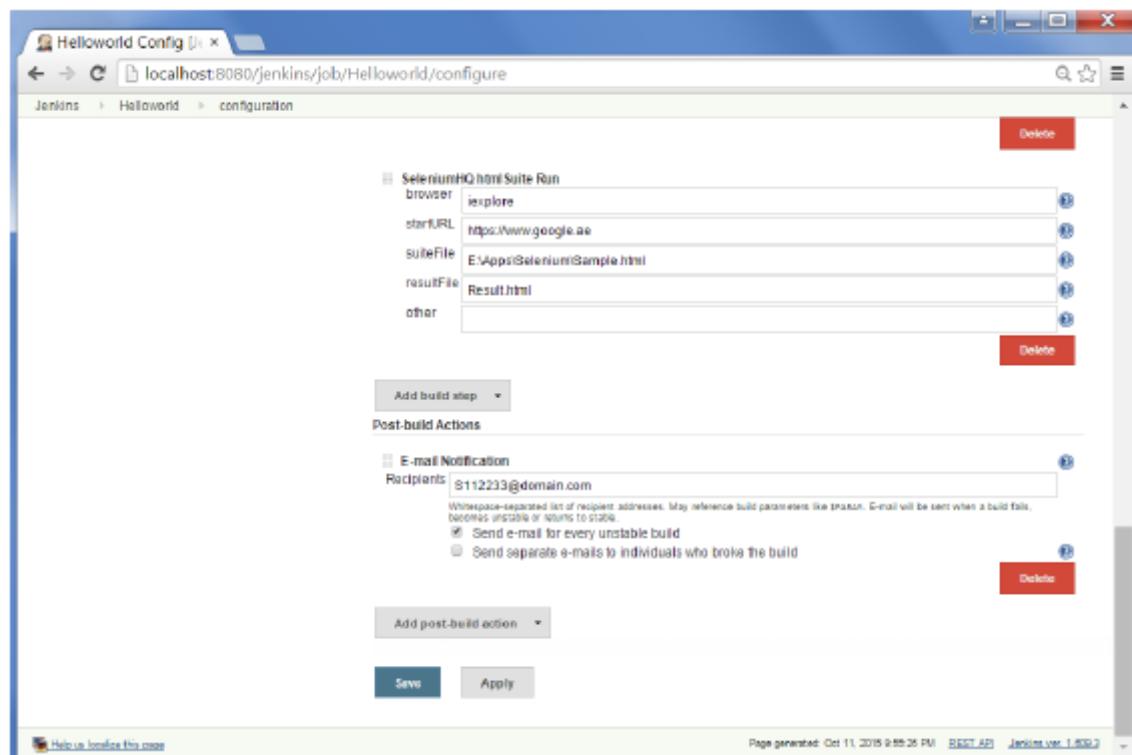
Jenkins - Notification

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

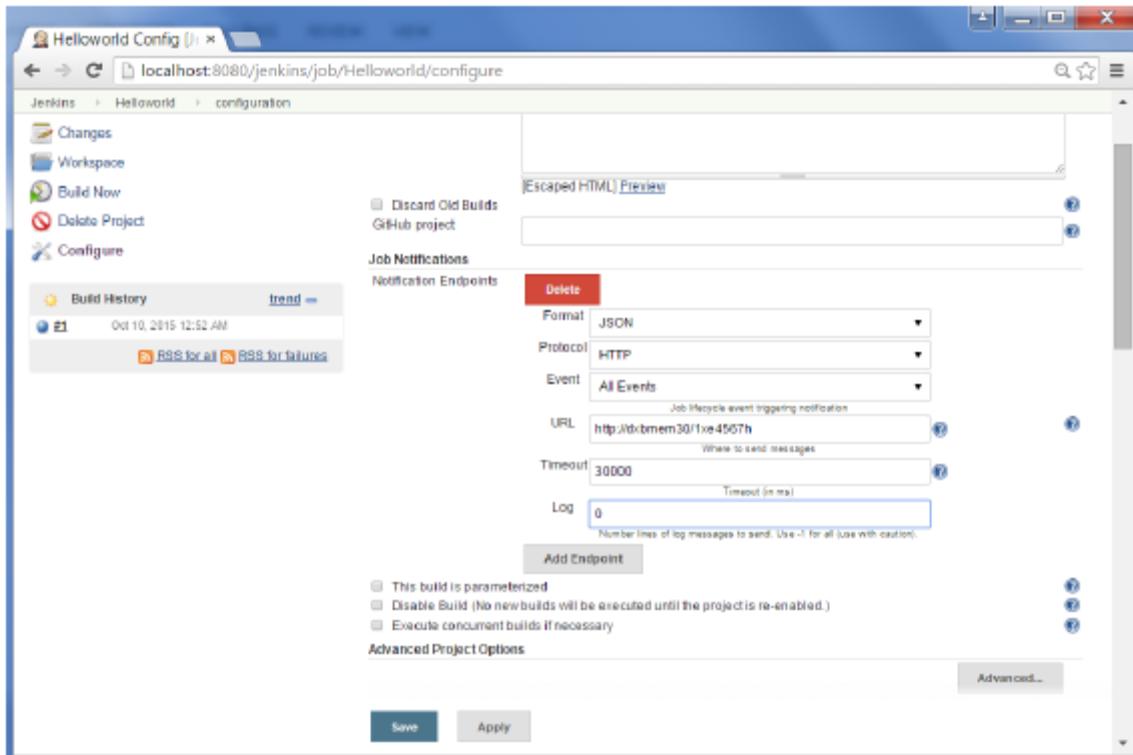
Step 1 – Configuring an SMTP server. Goto Manage Jenkins → Configure System. Go to the E-mail notification section and enter the required SMTP server and user email-suffix details.



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



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



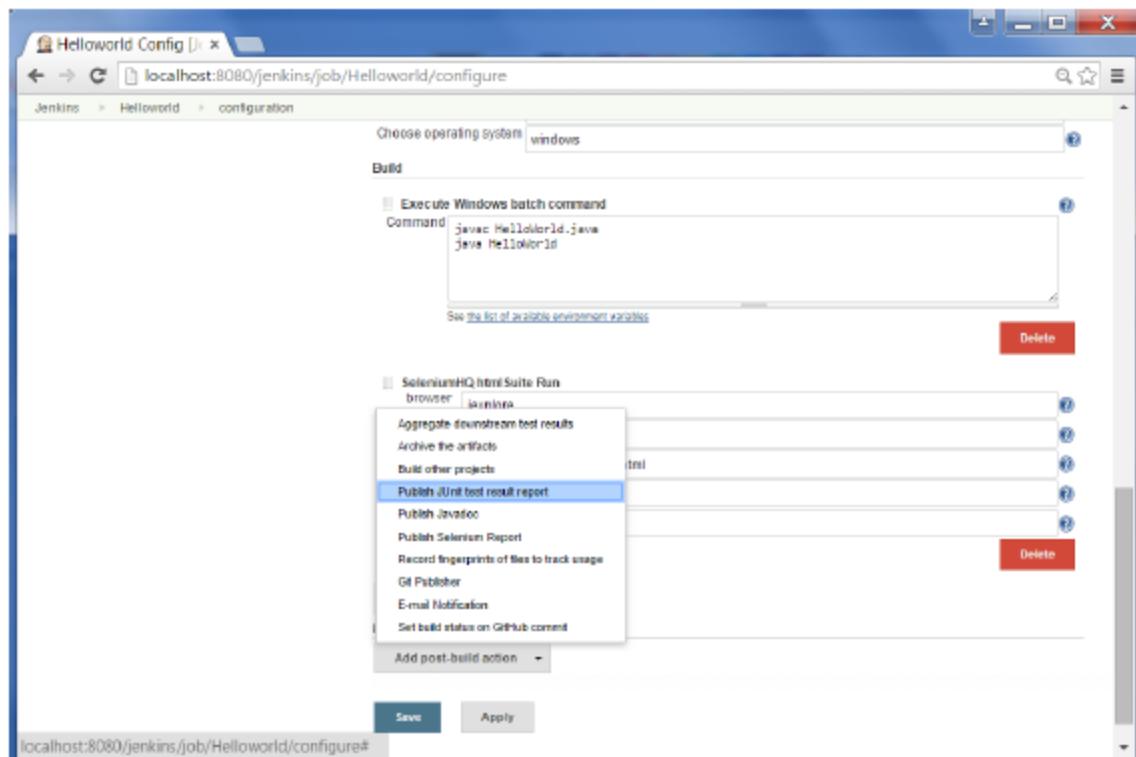
Here are the details of each option –

- "Format" – This is the notification payload format which can either be JSON or XML.
- "Protocol" – protocol to use for sending notification messages, HTTP, TCP or UDP.
- "Event" – The job events that trigger notifications: Job Started, Job Completed, Job Finalized or All Events (the default option).
- "URL" – URL to send notifications to. It takes the form of "http://host" for HTTP protocol, and "host:port" for TCP and UDP protocols.
- "Timeout" – Timeout in milliseconds for sending notification request, 30 seconds by default.

Jenkins - Reporting

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

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



Jenkins - Code Analysis

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

The screenshot shows the Jenkins Static Code Analysis Plug-ins page. On the left, there's a sidebar with links like Home, Mailing lists, Source code, Bugtracker, Security, Events, Donation, Commercial Support, and Wiki Site Map. Under Documents, there are links to Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, Servlet Container, and Notes. The main content area has a title 'Static Code Analysis Plug-ins' with a sub-section for 'analysis-core'. It shows the Plugin ID as 'analysis-core', Latest Release as '1.74 (archives)' from 'Sep 07, 2015', and Required Core as '1.596.1'. Dependencies listed include 'art', 'token-magic', 'maven-plugin', 'matrix-project', and 'dashboard-view'. The 'Changes' section shows 'In Latest Release Since Latest Release' with a GitHub link. The 'Usage' section contains a line graph titled 'analysis-core - installations' showing a steady increase from approximately 25,000 in October 2014 to over 30,000 in September 2015. The 'Installations' section lists dates from '2014-Oct 26415' to '2015-Sep 29858'. A message at the bottom says 'Waiting for wiki.jenkins-ci.org...'.

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

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

The plugins can provide information such as

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

Jenkins - Distributed Builds

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

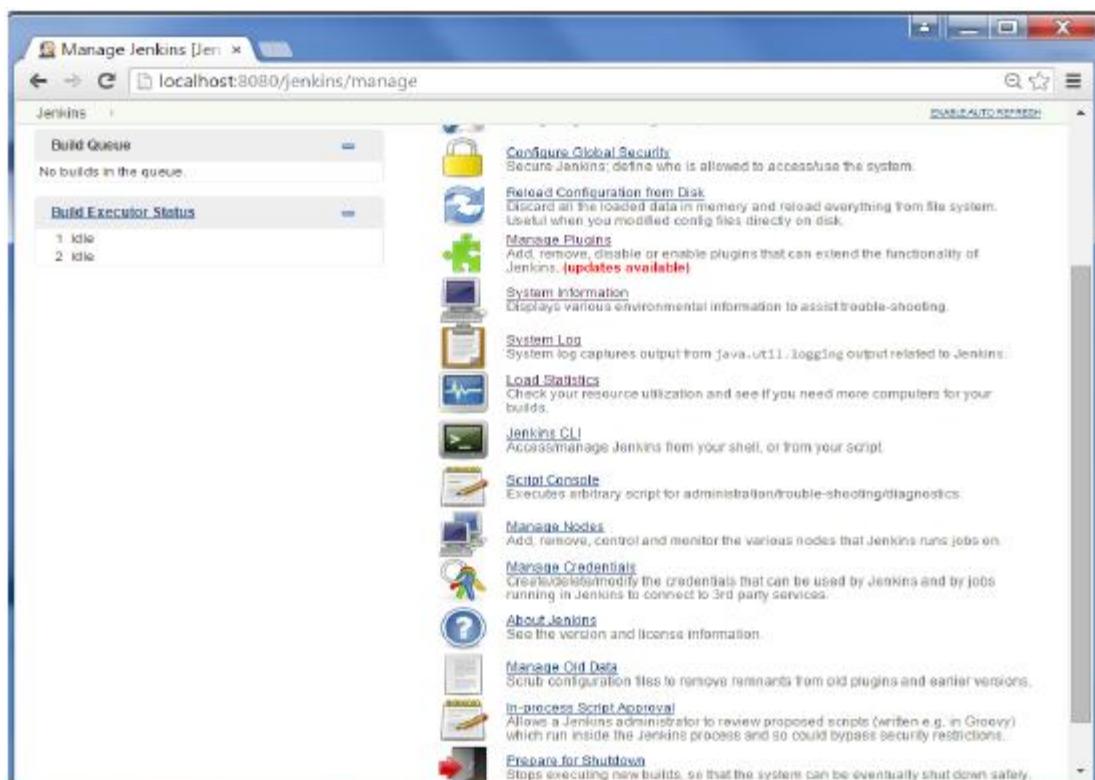
Sometimes you might also need several different environments to test your builds. In this case using a slave to represent each of your required environments is almost a must.

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

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

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

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

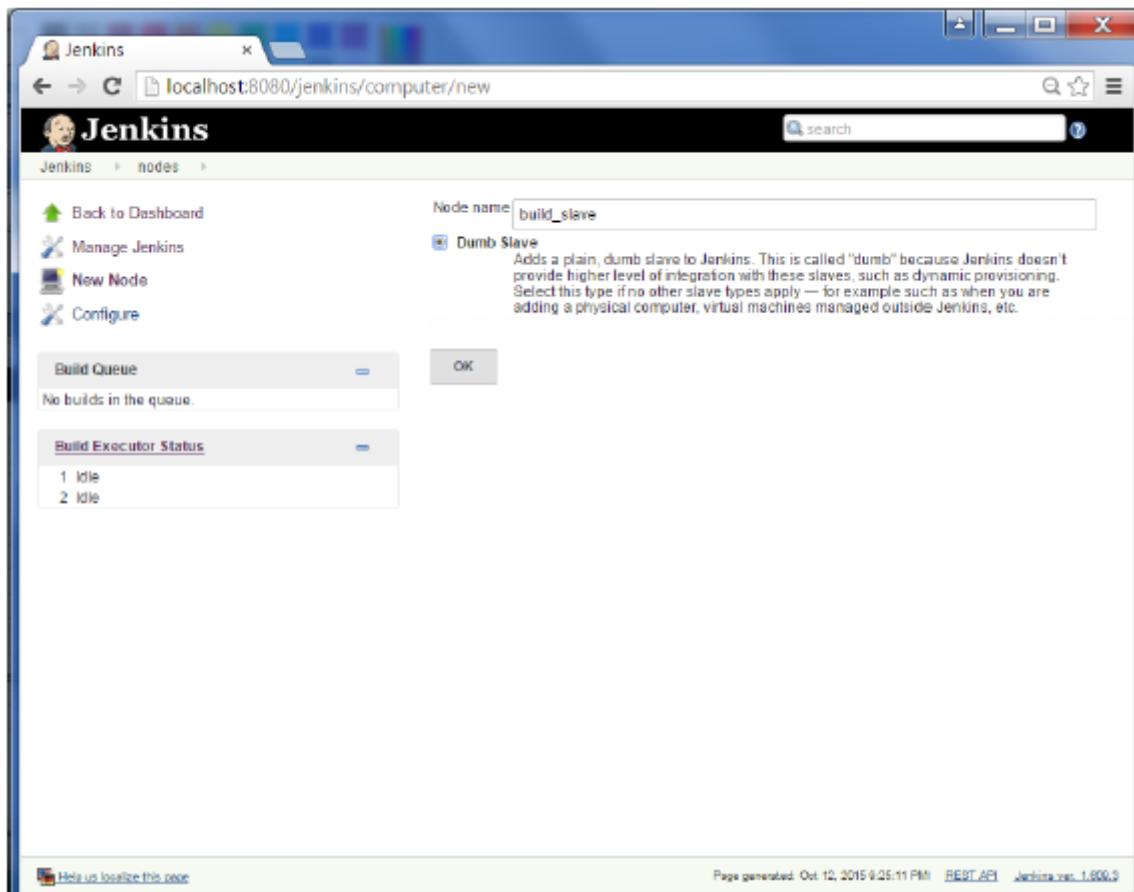


Step 2 – Click on New Node

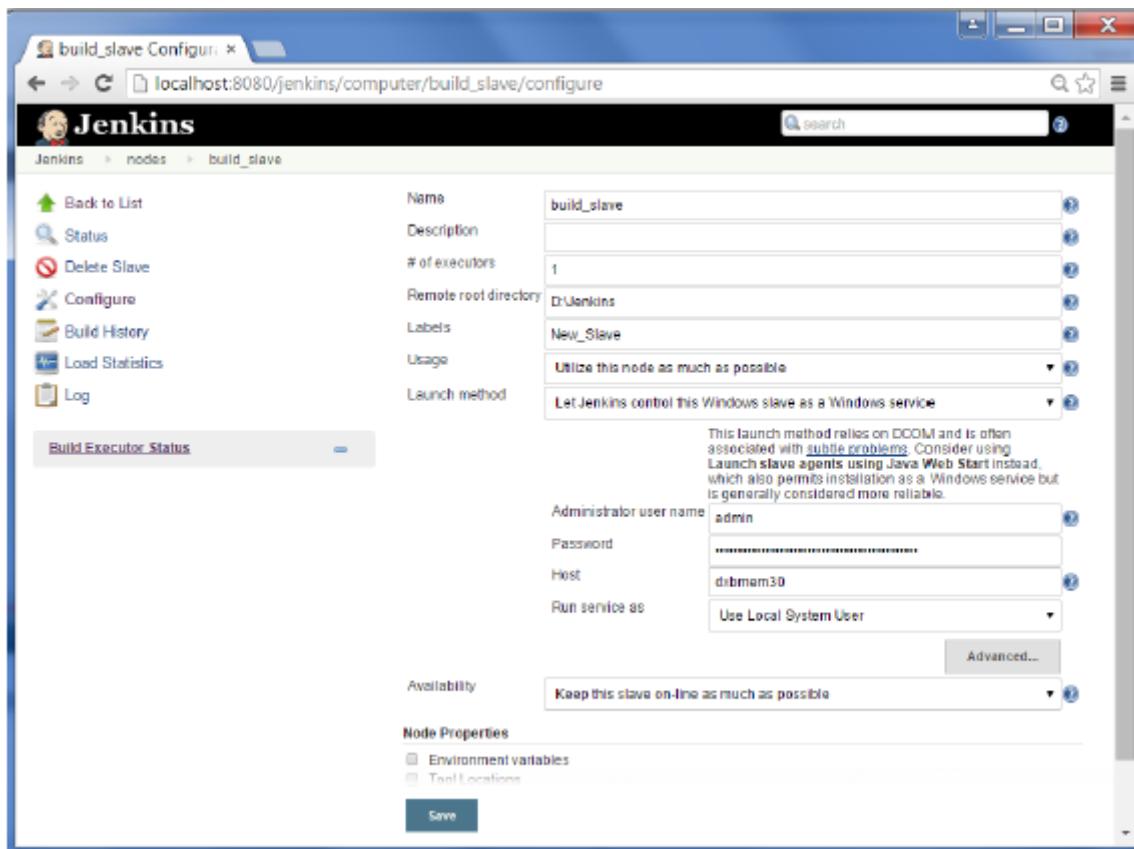
The screenshot shows the Jenkins 'Nodes' page. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. Below this is a search bar and a 'DYNAMIC AUTO REFRESH' button. The main content area has a table titled 'Nodes' with columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, and Free. One row is present, showing 'master' as the name, 'Windows 7 (x86)' as the architecture, 'In sync' as the clock difference, and 220.89 GB and 12.13 GB respectively for disk and swap space. Below the table is a 'Refresh status' button. On the left, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (2 Idle). At the bottom, there are links for 'Help us localize this page', 'Page generated: Oct 12, 2015 9:22:44 PM', 'REST API', and 'Jenkins ver. 1.806.2'.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free
	master	Windows 7 (x86)	In sync	220.89 GB	12.13 GB	

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.

The screenshot shows the Jenkins 'Nodes' page at localhost:8080/jenkins/computer/. The left sidebar includes links for Back to Dashboard, Manage Jenkins, New Node, and Configure. The main content displays a table of nodes:

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

Below the table are sections for Build Queue (No builds in the queue) and Build Executor Status (master: 1 idle, 2 idle; build_slave: offline). A 'Refresh status' button is located at the bottom right of the table area. The footer includes links for Help us localize this page, Page generated: Oct 12, 2015 9:31:49 PM, REST API, Jenkins ver. 1.600.3, and navigation arrows.

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 Update Center interface. The URL in the address bar is `localhost:8080/jenkins/pluginManager/available`. The page title is "Update Center [Jenkins]". The left sidebar shows "Jenkins" and "Plugin Manager". The main content area displays a table of available plugins:

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

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

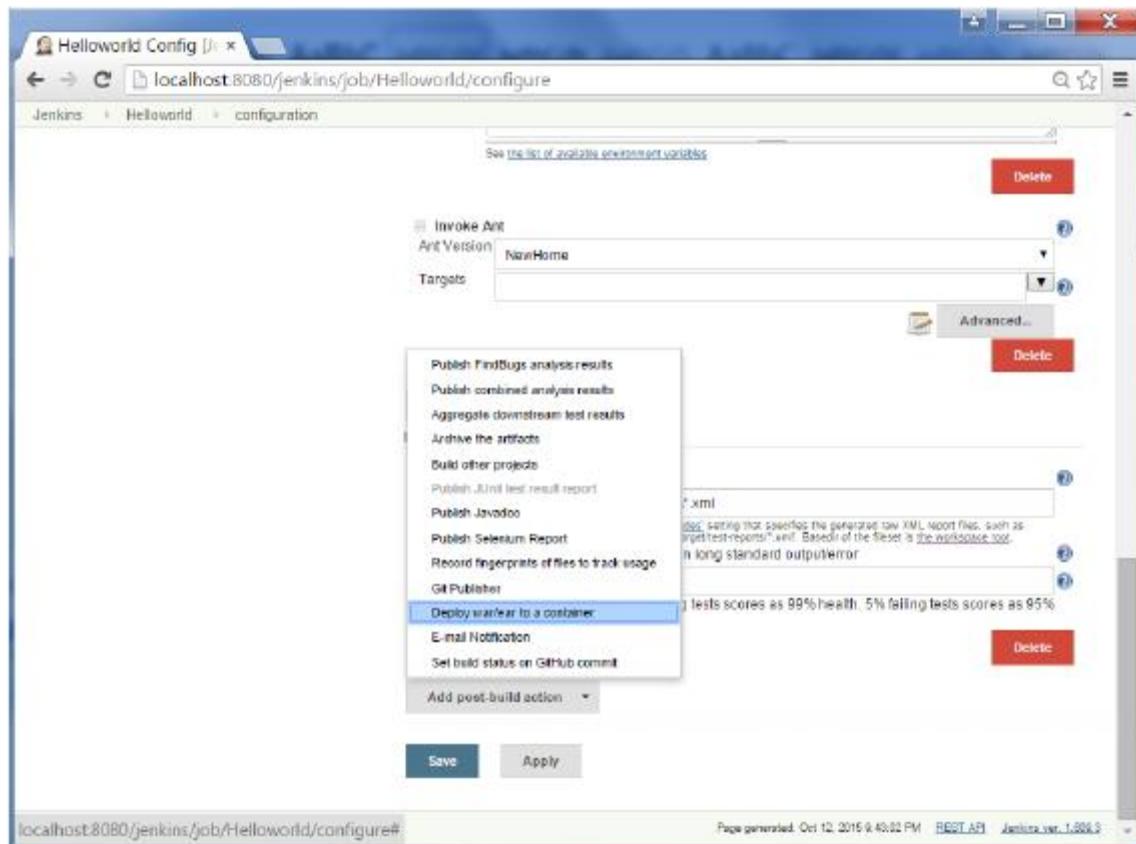
This plugin takes a war/ear file and deploys that to a running remote application server at the end of a build.

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

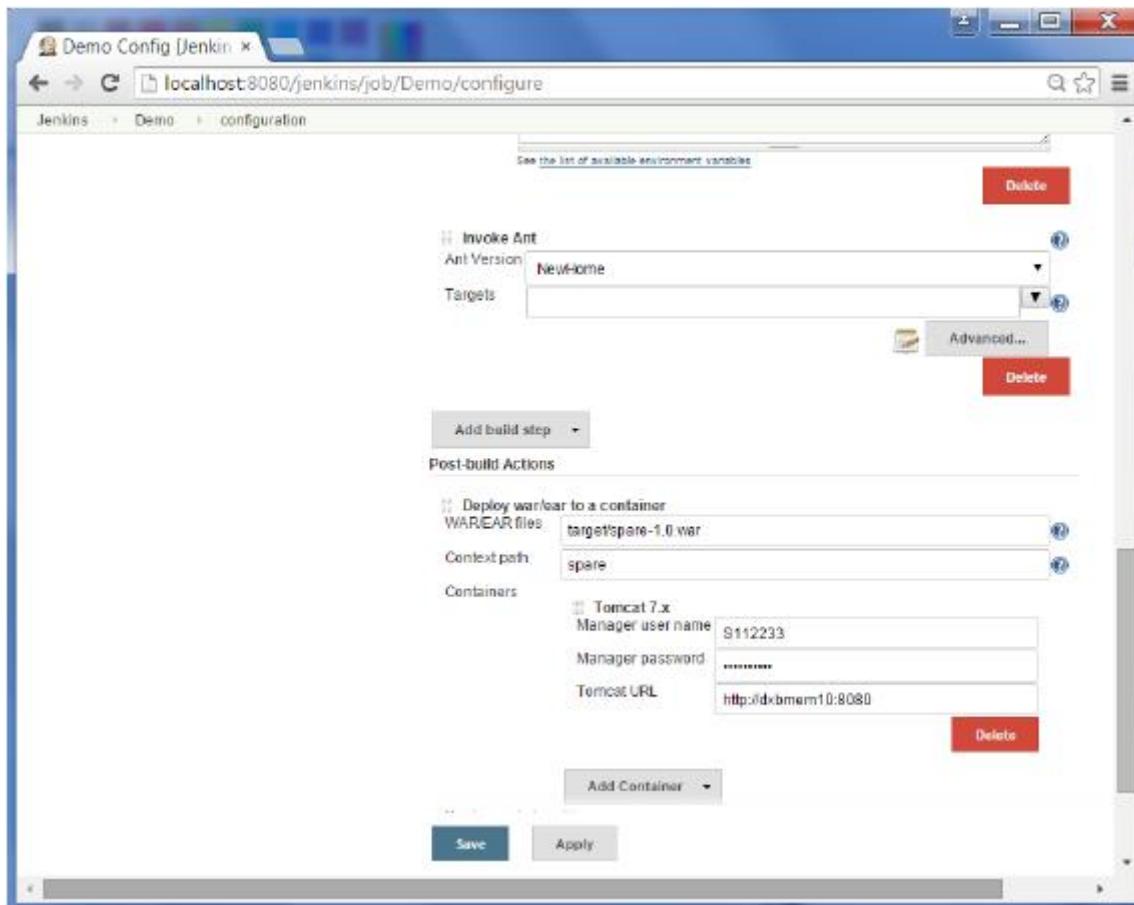
JBoss 3.x/4.x

Glassfish 2.x/3.x

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



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



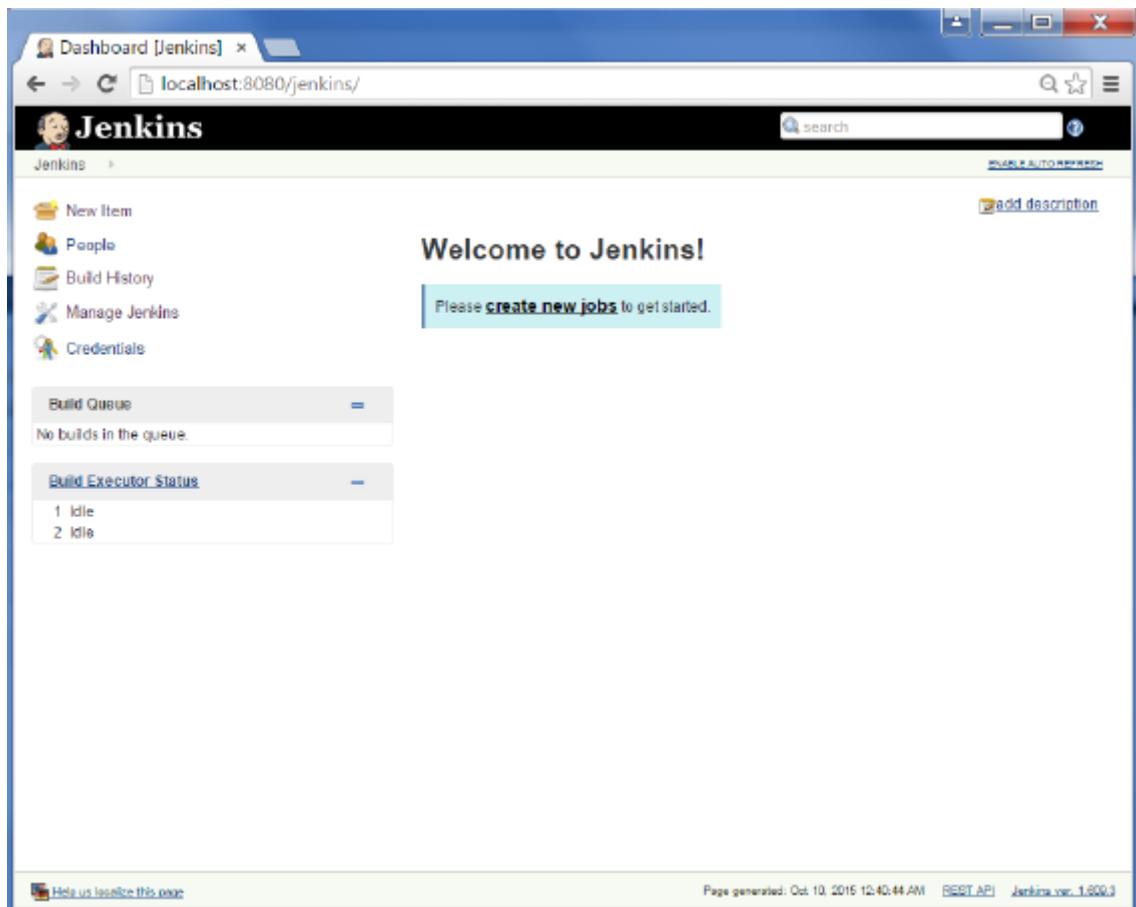
Jenkins - Metrics & Trends

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

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

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

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



Step 2 – Go to the Manage Plugins option.

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected), and 'Credentials'. Below this are sections for 'Build Queue' (empty) and 'Build Executor Status' (2 idle). The main content area is titled 'Manage Jenkins' and contains several management tools:

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

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

The screenshot shows the Jenkins Plugin Manager interface. The URL in the browser is `localhost:8080/jenkins/pluginManager/available`. The 'Available' tab is selected. A search bar at the top right contains the filter text `build-history-metrics-plugin`. A single plugin entry is listed:

Install	Name	Version
<input checked="" type="checkbox"/>	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** and **Download now and install after restart**. To the right, a message says "Update information obtained: 2 mi". At the bottom, there are links for "Help us localize this page", "Page generated: Oct 24, 2015 9:53:24 PM", "REST API", and "Jenkins ver. 1.600.3".

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

The screenshot shows the Jenkins 'Installing Plugins/Upgrades' confirmation page. The URL in the browser is `localhost:8080/jenkins/updateCenter/`. The title of the page is **Installing Plugins/Upgrades**.

The 'Preparation' section lists the following steps:

- + Checking Internet connectivity
- + Checking update center connectivity
- + Success

The 'Build History Metrics plugin' status is shown as **Success**.

Two green checkmark icons provide instructions:

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

At the bottom, there are links for "Help us localize this page", "Page generated: Oct 24, 2015 9:53:27 PM", "REST API", and "Jenkins ver. 1.600.3".

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

The screenshot shows the Jenkins interface for the 'Helloworld' project. On the left, there's a sidebar with links like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. Below that is the 'Build History' section, which lists builds #12 through #1, with the most recent at the top. At the bottom of this section are links for 'RSS for all' and 'RSS for failures'. To the right of the sidebar, the main content area has a title 'Project Helloworld'. It features two large buttons: 'Workspace' (with a folder icon) and 'Recent Changes' (with a document icon). Above these buttons are 'add description' and 'Disable Project' buttons. The central part of the page contains three tables showing performance metrics:

	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

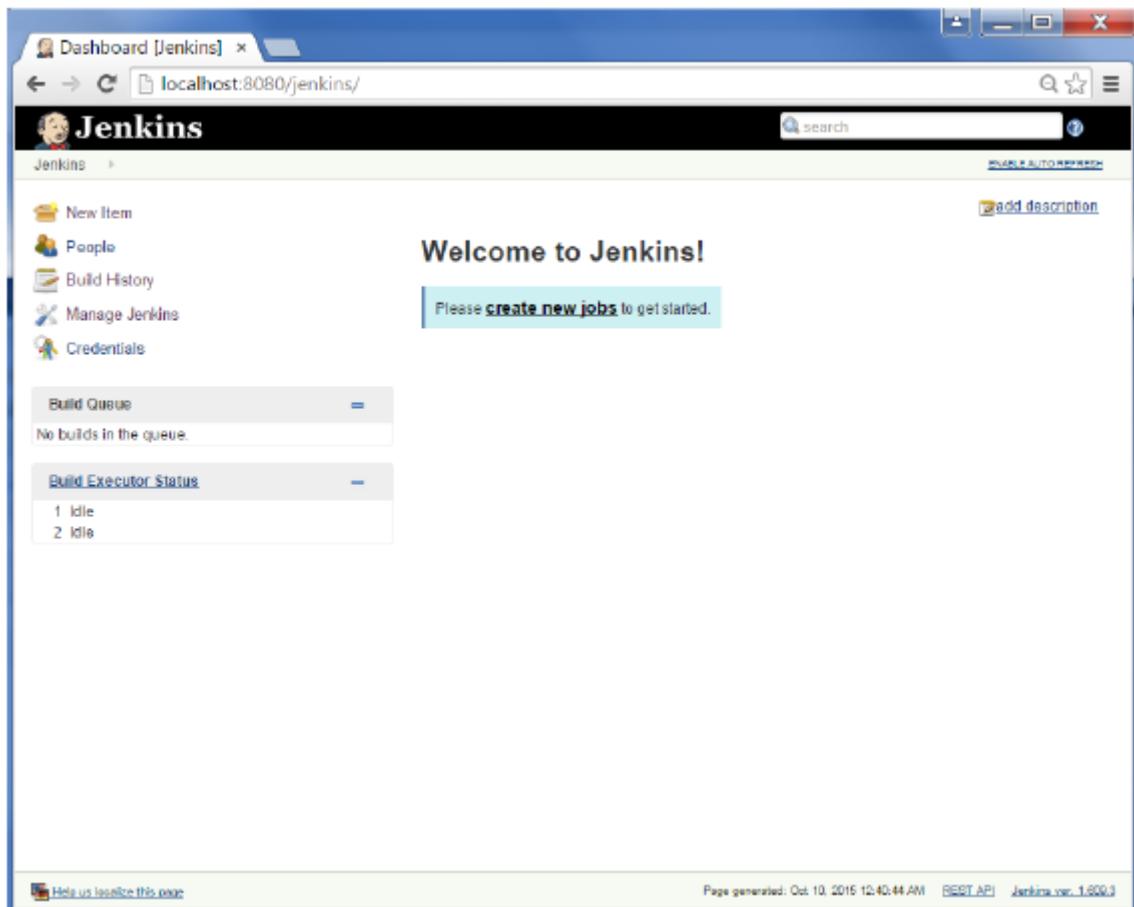
Below these tables is a section titled 'Permalinks' containing a bulleted list of links to specific build logs:

- 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, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 9:57:10 PM', 'REST API', and 'Jenkins ver. 1.600.3'.

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

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

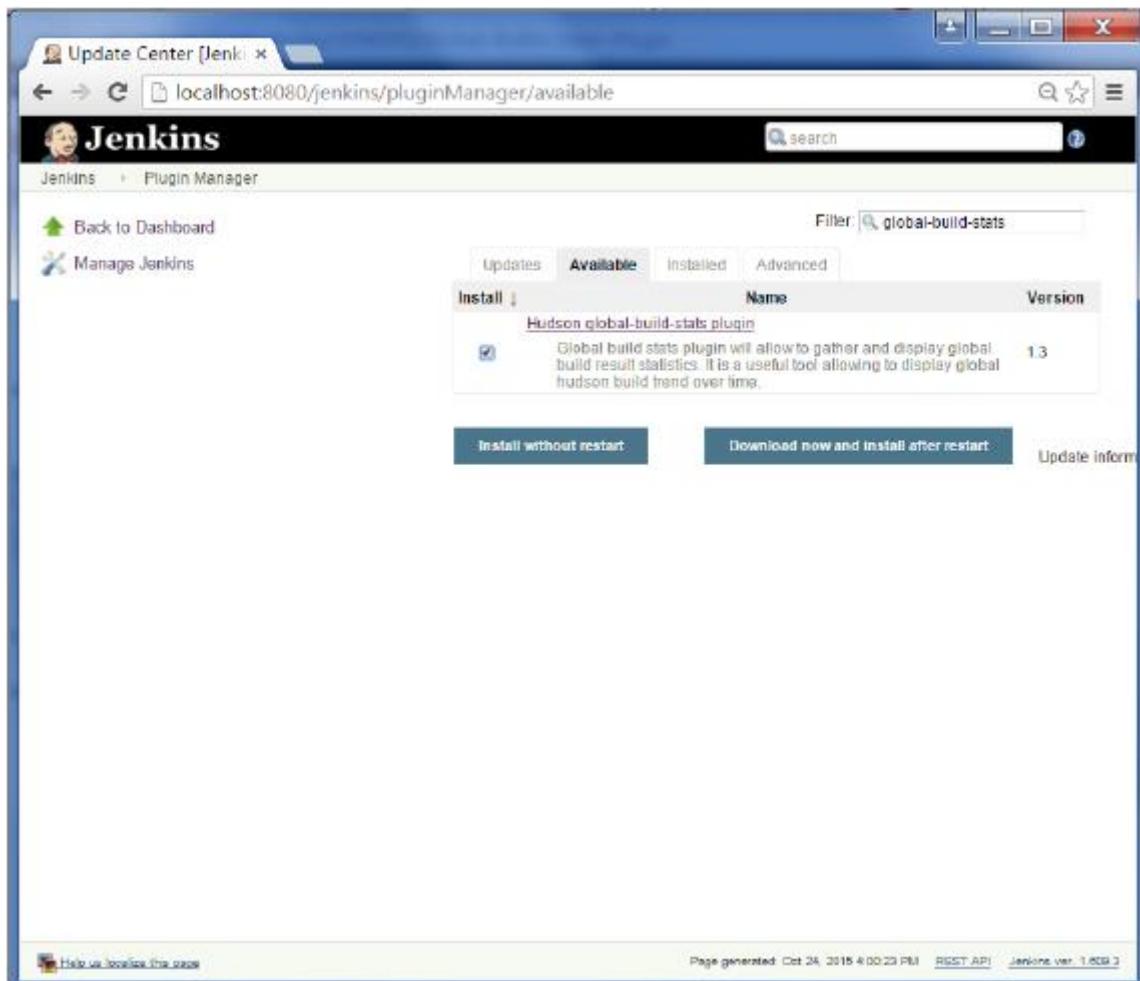


Step 2 – Go to the Manage Plugins option

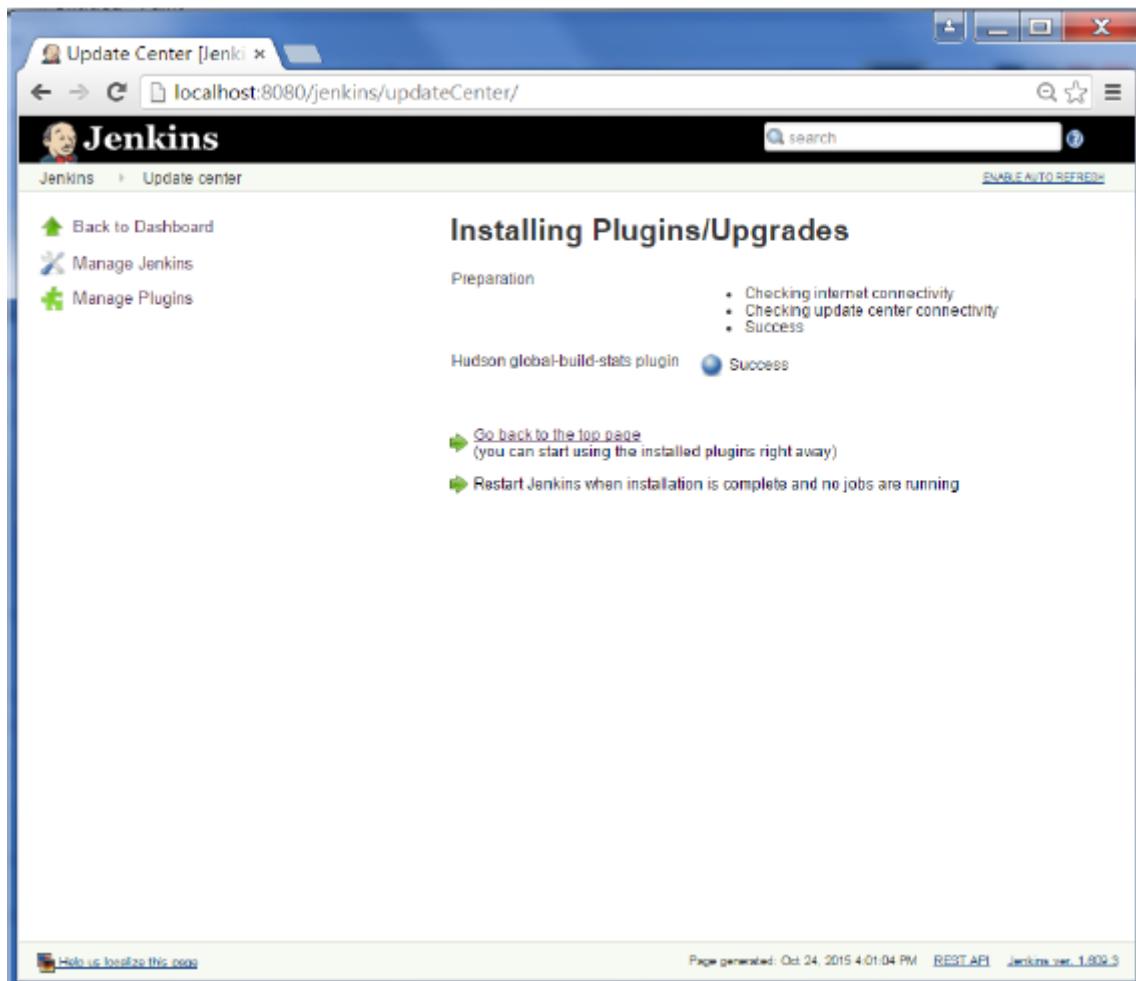
The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected), and 'Credentials'. Below these are two expandable sections: 'Build Queue' (empty) and 'Build Executor Status' (showing 1 idle and 2 idle executors). 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 the system.
- Reload Configuration from Disk**: Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins. [\(updates available\)](#)
- System Information**: Displays various environmental information to assist trouble-shooting.
- System Log**: System log captures output from java.util.logging output related to Jenkins.
- Load Statistics**: Check your resource utilization and see if you need more computers for your builds.
- Jenkins CLI**: Access/manage Jenkins from your shell, or from your script.
- Script Console**: Executes arbitrary script for administration/trouble-shooting/diagnostics.
- Manage Nodes**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Manage Credentials**: Create/delete/modify the credentials that can be used by Jenkins and by jobs running in Jenkins to connect to 3rd party services.
- About Jenkins**: See this screen for general information.

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



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

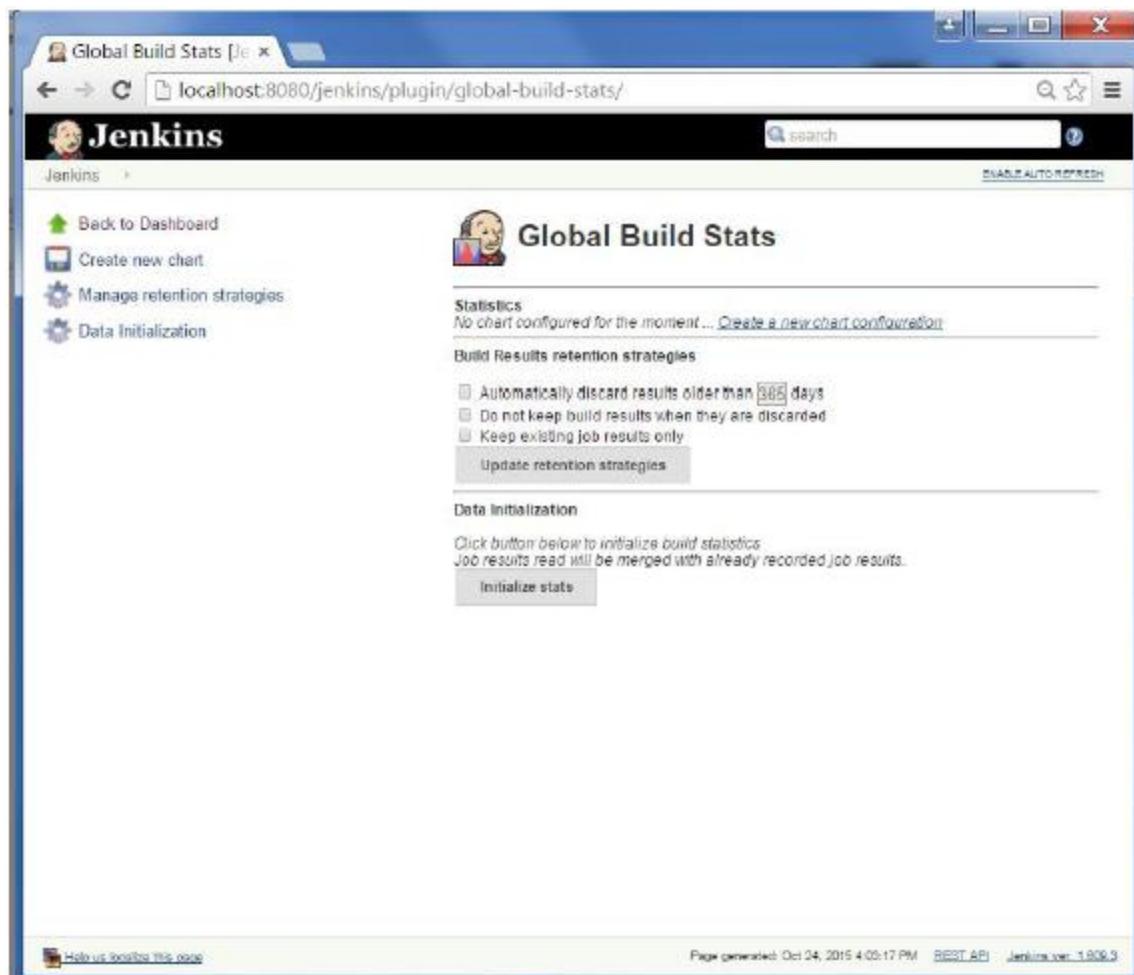


To see the Global statistics, please follow the Step 5 through 8.

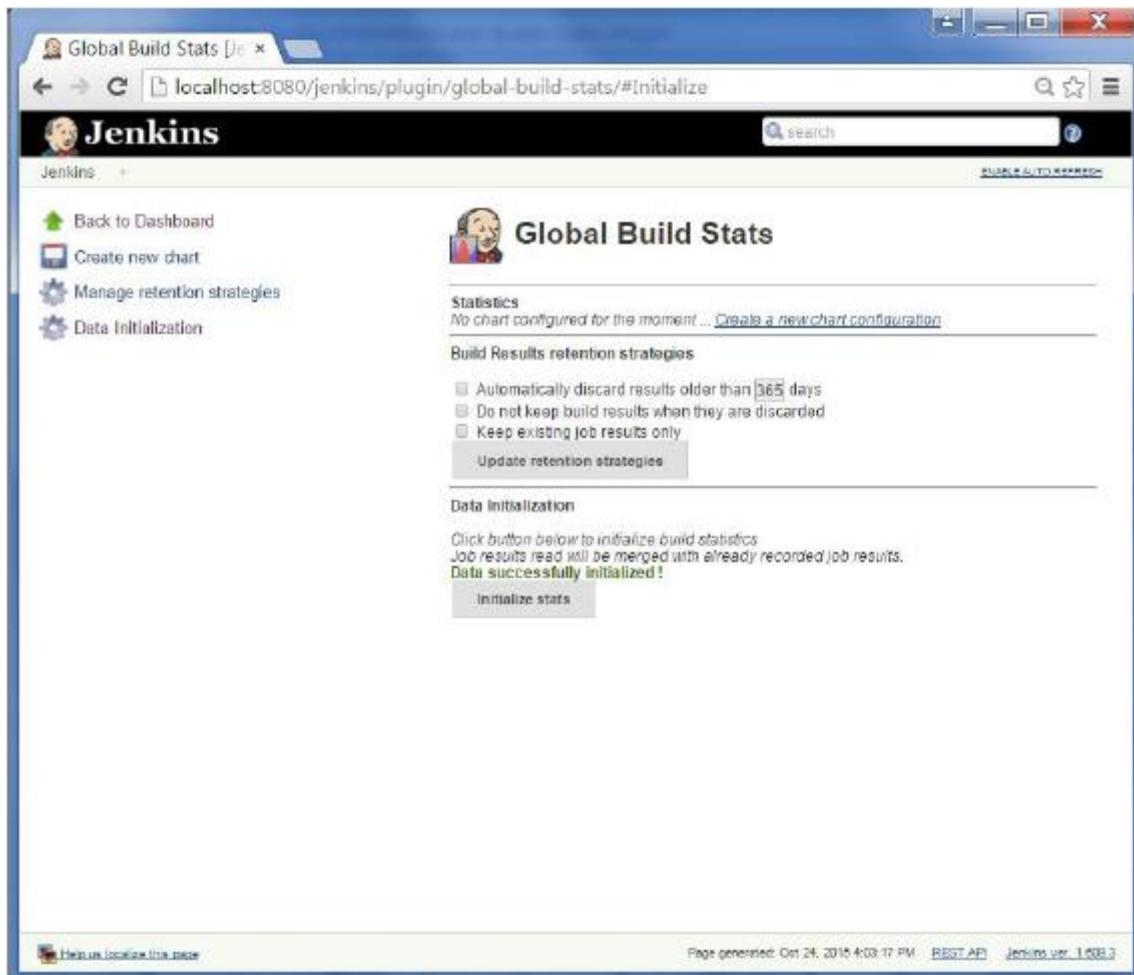
Step 5 – Go to the Jenkins dashboard and click on Manage Jenkins. In the Manage Jenkins screen, scroll down and now you will now see an option called ‘Global Build Stats’. Click on this link.



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



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



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

- Title – Any title information, for this example is given as ‘Demo’
- Chart Width – 800
- Chart Height – 600
- Chart time scale – Daily
- Chart time length – 30 days

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

Global Build Stats | Jenkins

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

Jenkins

Back to Dashboard Create new chart Manage retention strategies Data Initialization

Global Build Stats

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

Build Results retention strategies

Adding new chart

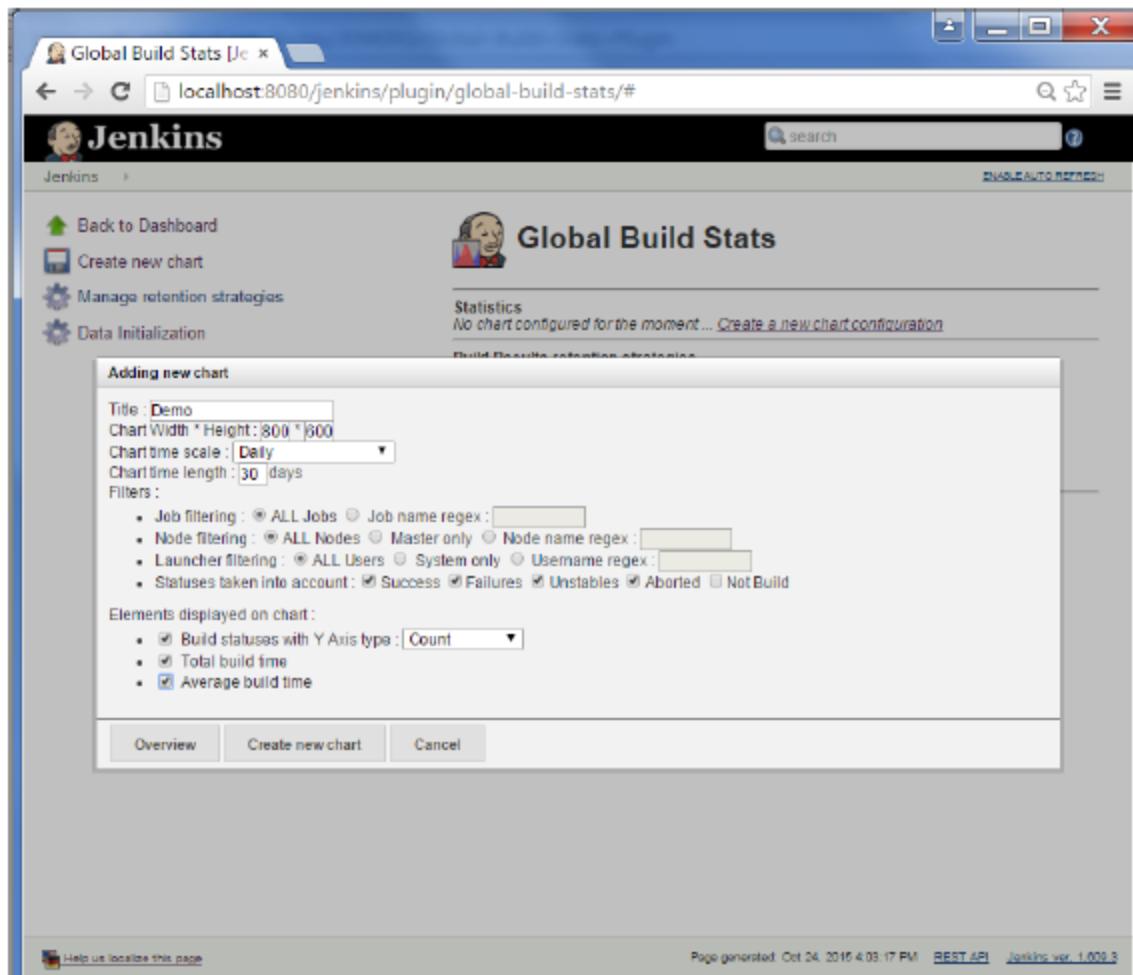
Title: Demo
Chart Width * Height: 800 * 300
Chart time scale: Daily
Chart time length: 30 days
Filters:
• Job filtering: ALL Jobs Job name regex:
• Node filtering: ALL Nodes Master only Node name regex:
• Launcher filtering: ALL Users System only Username regex:
• Statuses taken into account: Success Failures Unstables Aborted Not Build

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

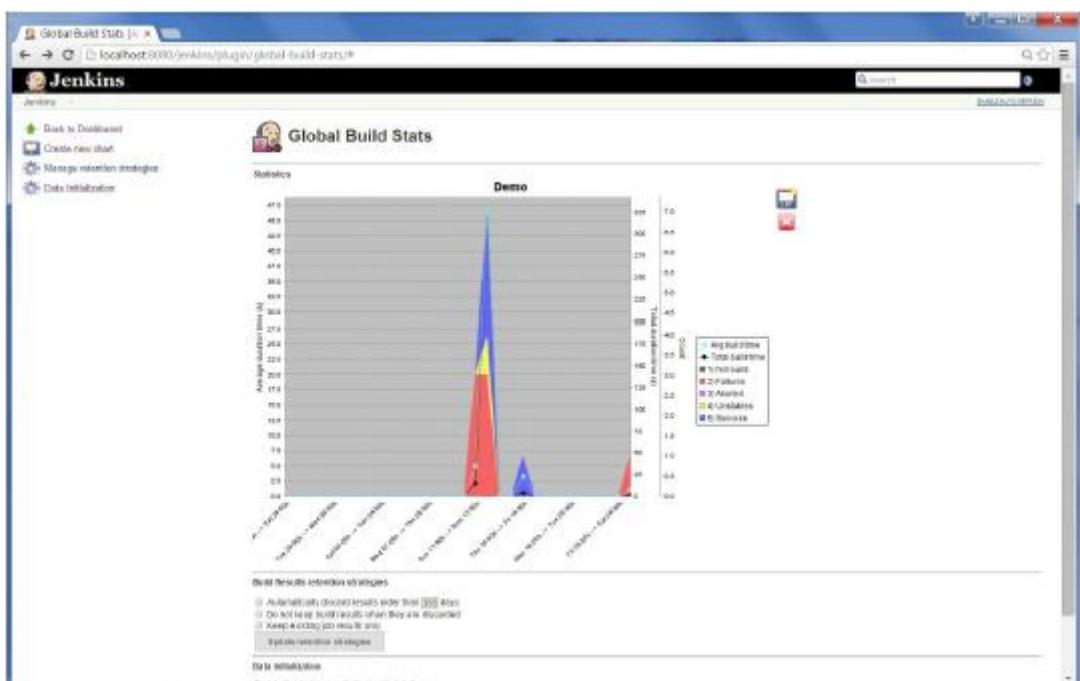
Overview Create new chart Cancel

Help us localize this page

Page generated: Oct 24, 2015 4:03:17 PM | REST API | Jenkins ver. 1.609.3



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



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

The screenshot shows the Jenkins Global Build Search page. At the top, there are filters for 'Job filtering' (radio buttons for 'All Jobs' and 'Job name regex'), 'Node filtering' (radio buttons for 'All Nodes', 'Master only', and 'Node name regex'), 'Launcher filtering' (radio buttons for 'All Users', 'System only', and 'Username regex'), and a 'Statuses taken into account' dropdown with options for Success, Failure, Unstable, Aborted, and Not Built. Below the filters is a 'Search' button. The main area is titled 'Search results' and contains a table with three rows of build data:

Status	Job Name	#	Date	Duration	Node name	Launched by
Failure	Hellosword	12	Oct 11, 2015 11:04:57 PM	5.6 sec	master	SYSTEM
Failure	Hellosword	12	Oct 11, 2015 19:51:07 PM	4.6 sec	master	SYSTEM
Failure	Hellosword	12	Oct 11, 2015 19:48:11 PM	4.2 sec	master	SYSTEM

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.

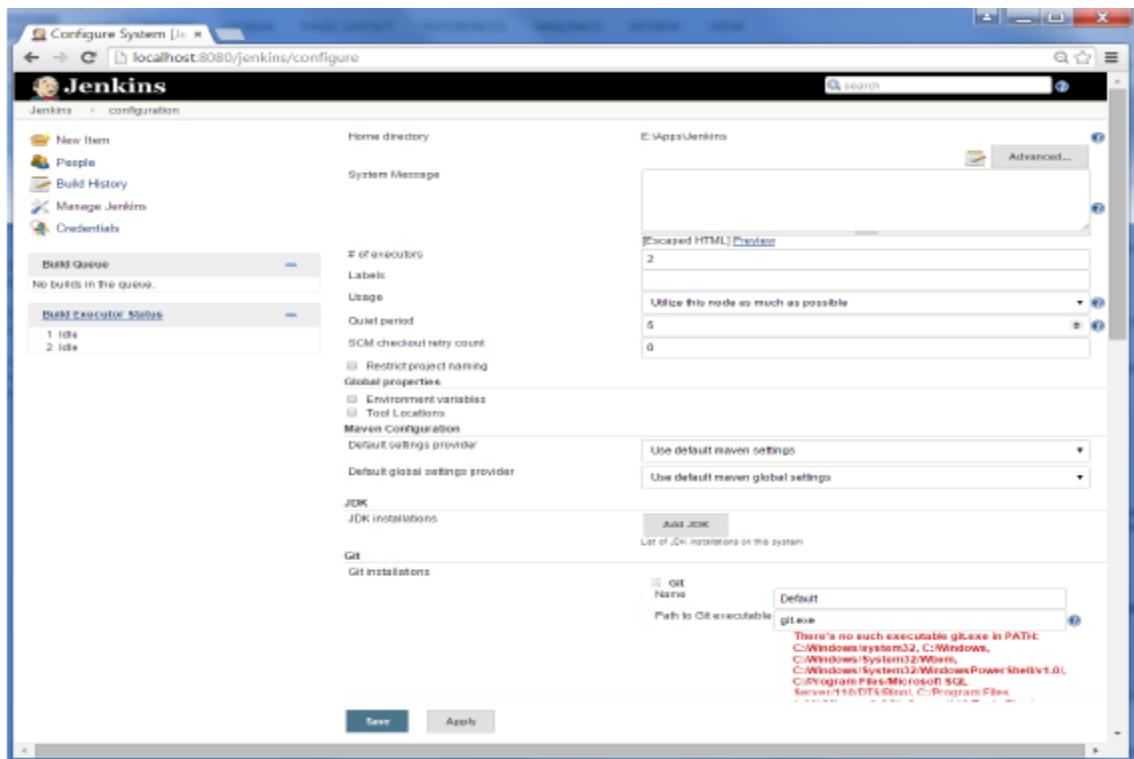
`http://localhost:8080/jenkins/exit` – shutdown jenkins

`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.

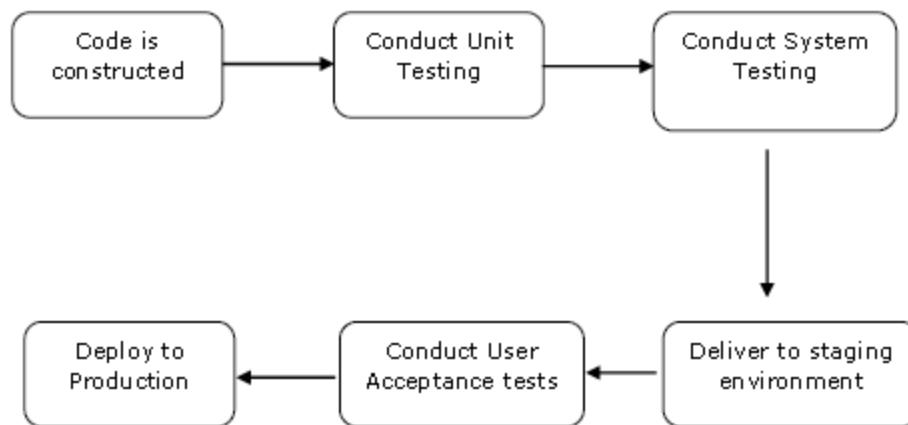


Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined and doing continuous builds, always ensure that Jenkins is setup on a drive that has enough hard disk space. If you hard disk runs out of space, then all builds on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

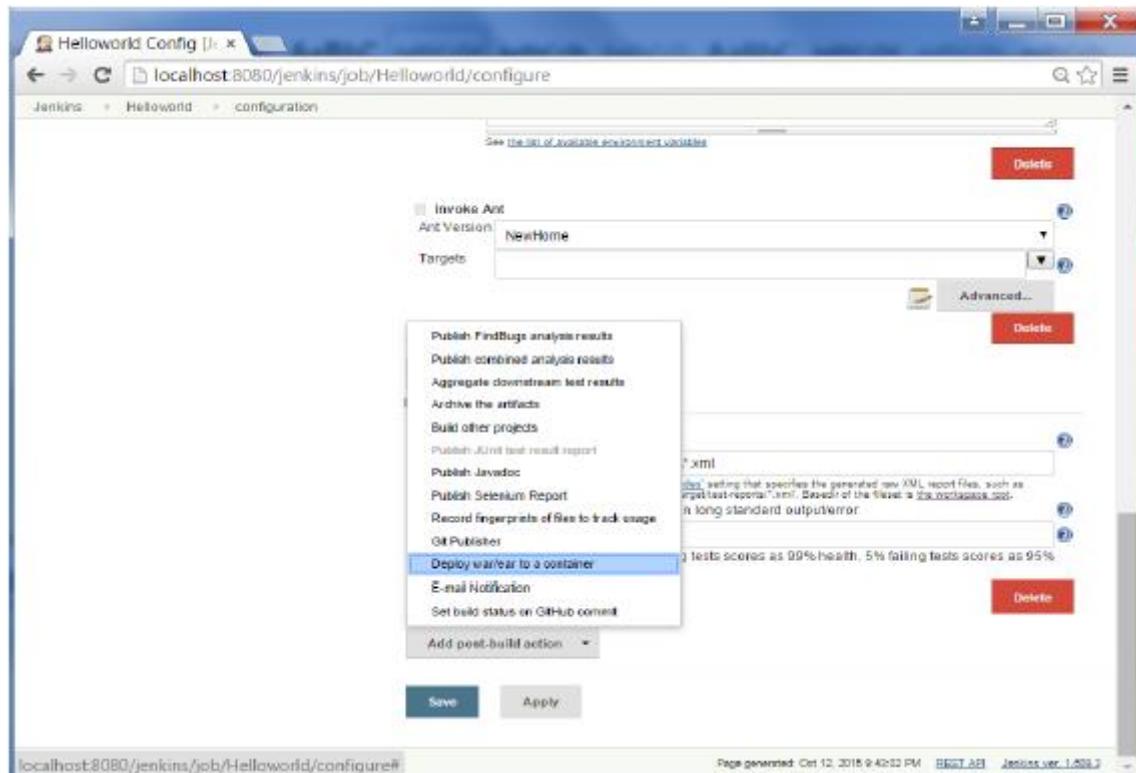
Another best practice is to write cron jobs or maintenance tasks that can carry out clean-up operations to avoid the disk where Jenkins is setup from becoming full.

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.



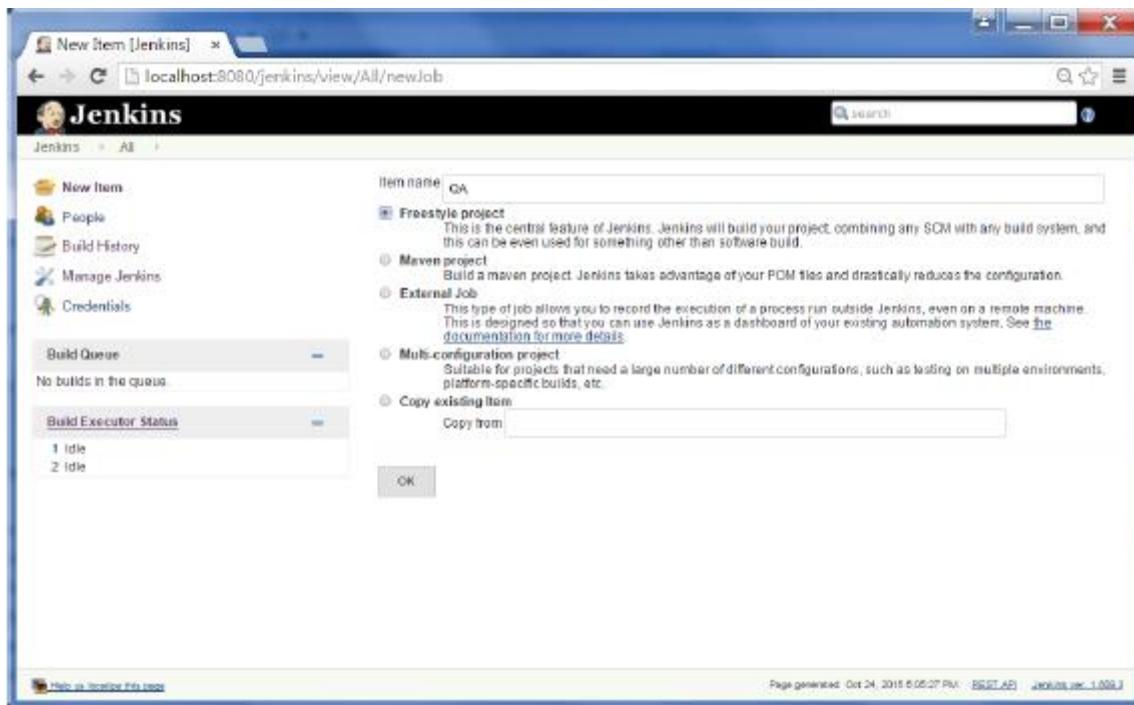
The main part of Continuous deployment is to ensure that the entire process which is shown above is automated. Jenkins achieves all of this via various plugins, one of them being the “Deploy to container Plugin” which was seen in the earlier lessons.



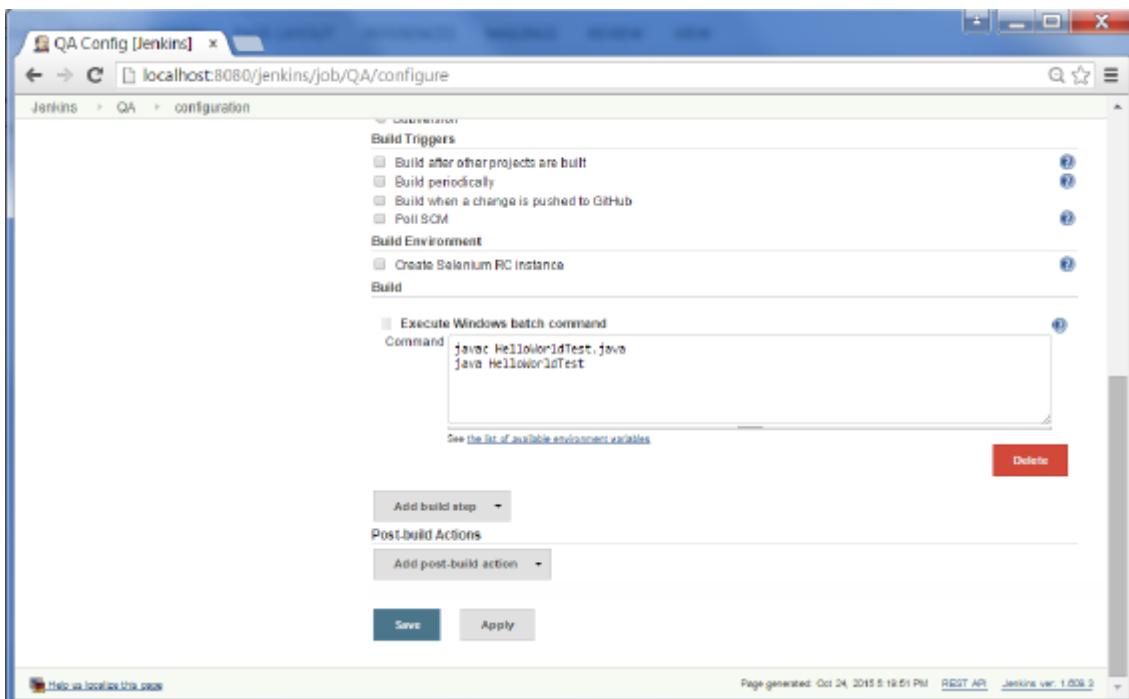
There are plugins available which can actually give you a graphical representation of the Continuous deployment process. But first lets create another project in Jenkins, so that we can see best how this works.

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.



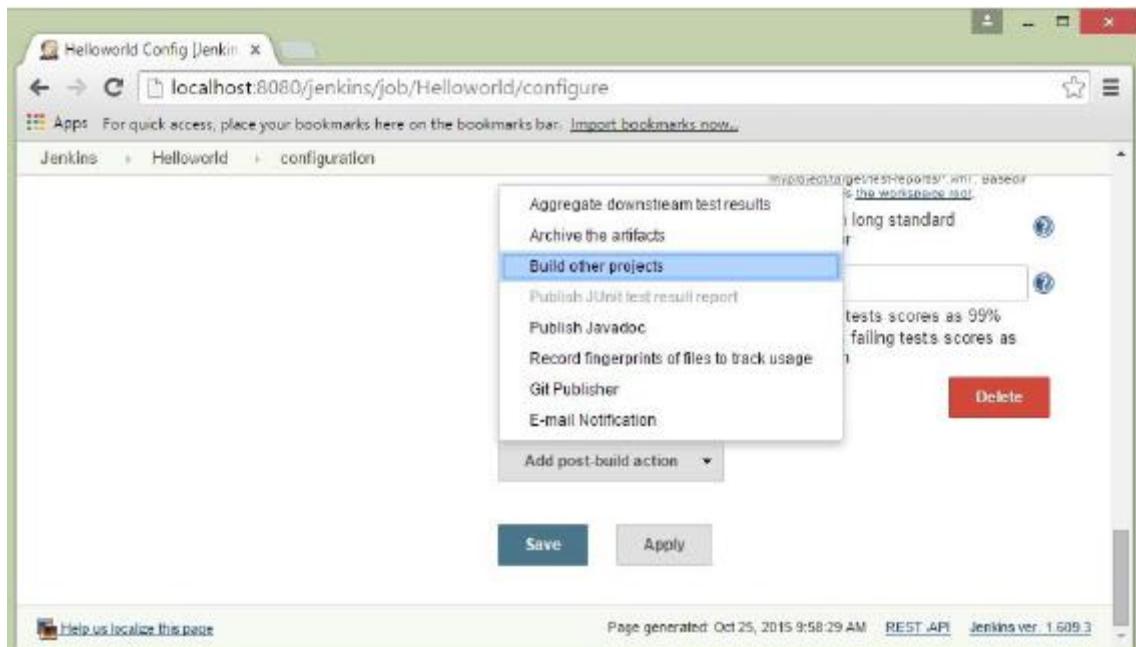
So our project QA is now setup. You can do a build to see if it builds properly.

The screenshot shows the Jenkins Project QA page for a job named 'QA'. The left sidebar contains links for Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main content area is titled 'Project QA' and displays a 'Build History' table with four entries from October 24, 2015, at various times. Below the build history are three tables: MTTR, MTTF, and Standard Deviation, all showing values for Last 7 Days, Last 30 Days, and All Time. A 'Recent Changes' section is also present. On the right side, there are buttons for 'Add description', 'Disable Project', and 'Configure'. At the bottom, there is a 'Permalinks' section with two links: 'Last build (#4), 6.4 sec ago' and 'Last stable build (#4), 6.4 sec ago'.

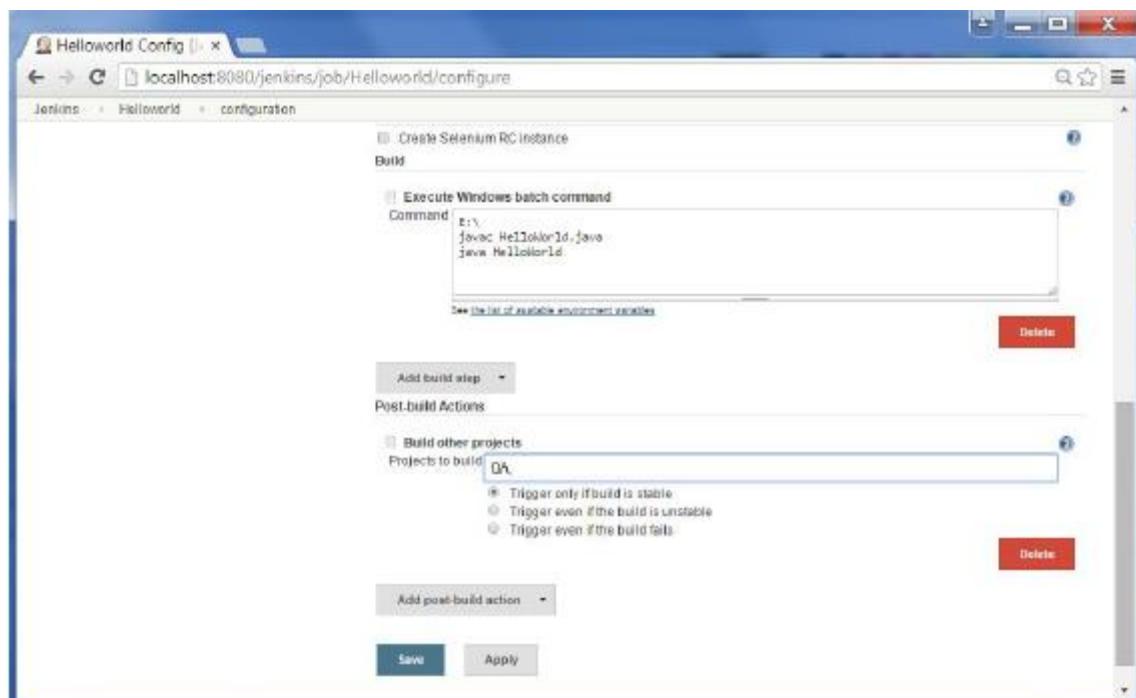
Step 3 – Now go to your Helloworld project and click on the Configure option

The screenshot shows the Jenkins Dashboard. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. The main area displays a list of projects: 'Helloworld' (selected), 'HelloWorld', 'Icon', and 'Just latest builds'. For the 'Helloworld' project, details are shown: Name (Helloworld), Last Success (12 days - #15), Last Failure (12 days - #14), and Last Duration (6.6 sec). Below the project list are links for Changes, Workspace, Build Now, Delete Project, and Configure. The 'Configure' link is highlighted with a blue border. The URL in the browser bar is 'localhost:8080/jenkins/job/Helloworld/configure'.

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



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.

```

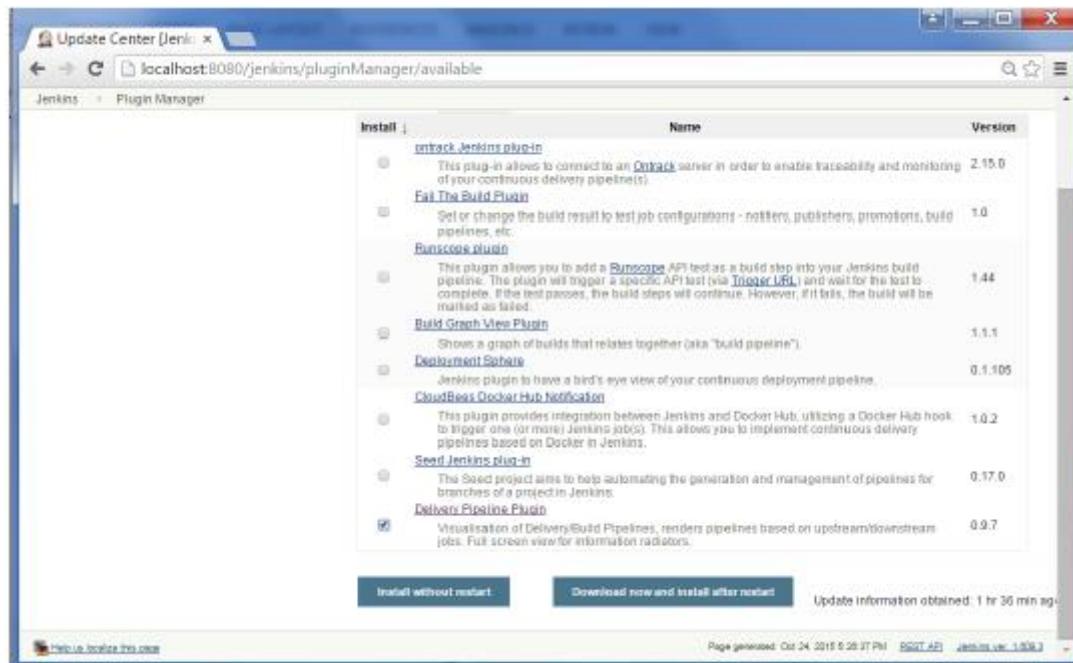
Started by user anonymous
Building in workspace E:\Jenkins\jobs\HelloWorld\workspace
[workspace] $ cmd /c call E:\Apache\tomcat7\temp\hudson3970974123996969633.bat
E:\Jenkins\jobs\HelloWorld\workspace>cmd
'El!' is not recognized as an internal or external command,
operable program or batch file.

E:\Jenkins\jobs\HelloWorld\workspace>java -jar HelloWorld.jar
Hello World

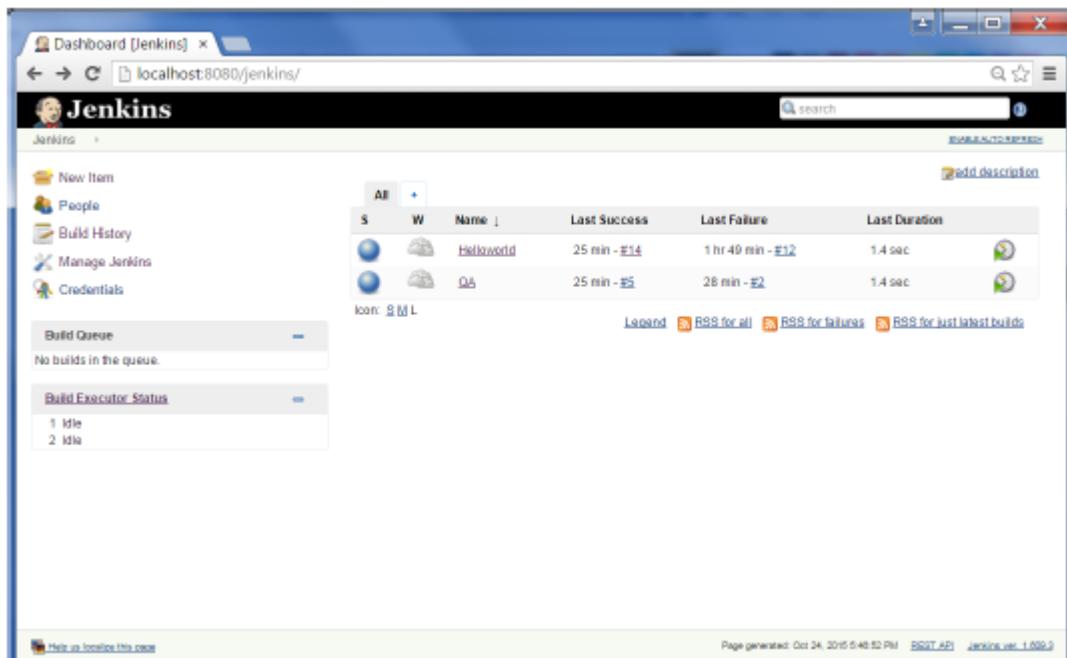
E:\Jenkins\jobs\HelloWorld\workspace>exit 0
Warning: you have no plugins providing access control for builds, so falling back to legacy behavior of permitting any downstream build to be triggered
Triggering a new build of Job
Finished: SUCCESS

```

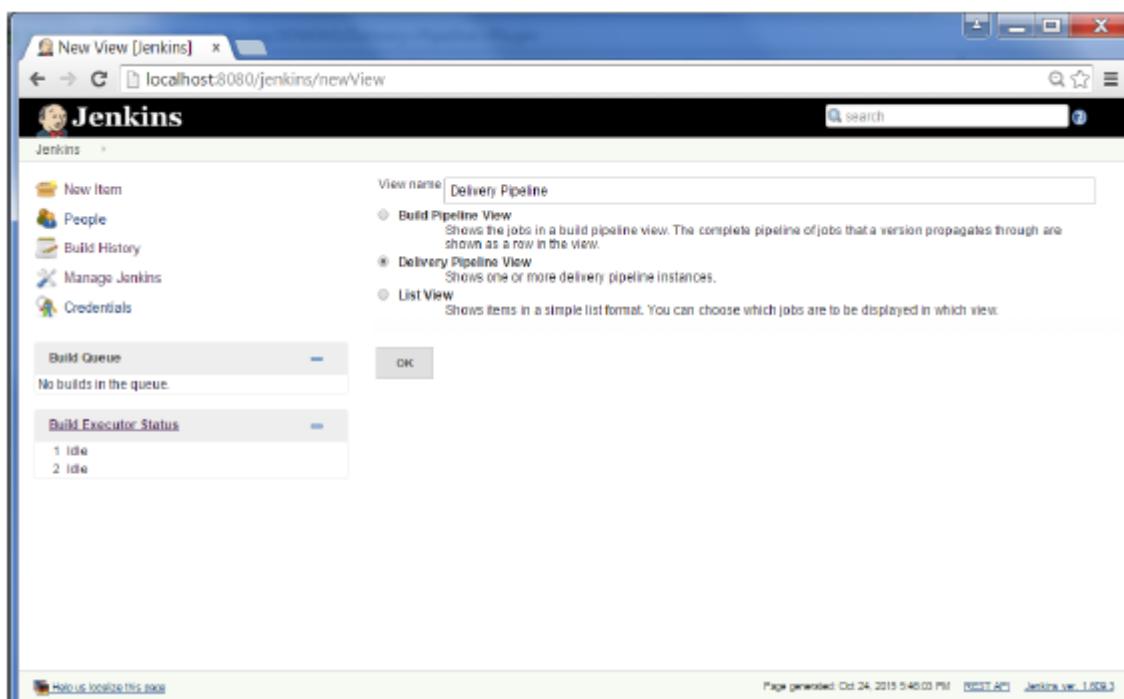
Step 7 – Let now install the Delivery pipeline plugin. Go to Manage Jenkins → Manage Plugin's. In the available tab, search for ‘Delivery Pipeline Plugin’. Click On Install without Restart. Once done, restart the Jenkins instance.



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.



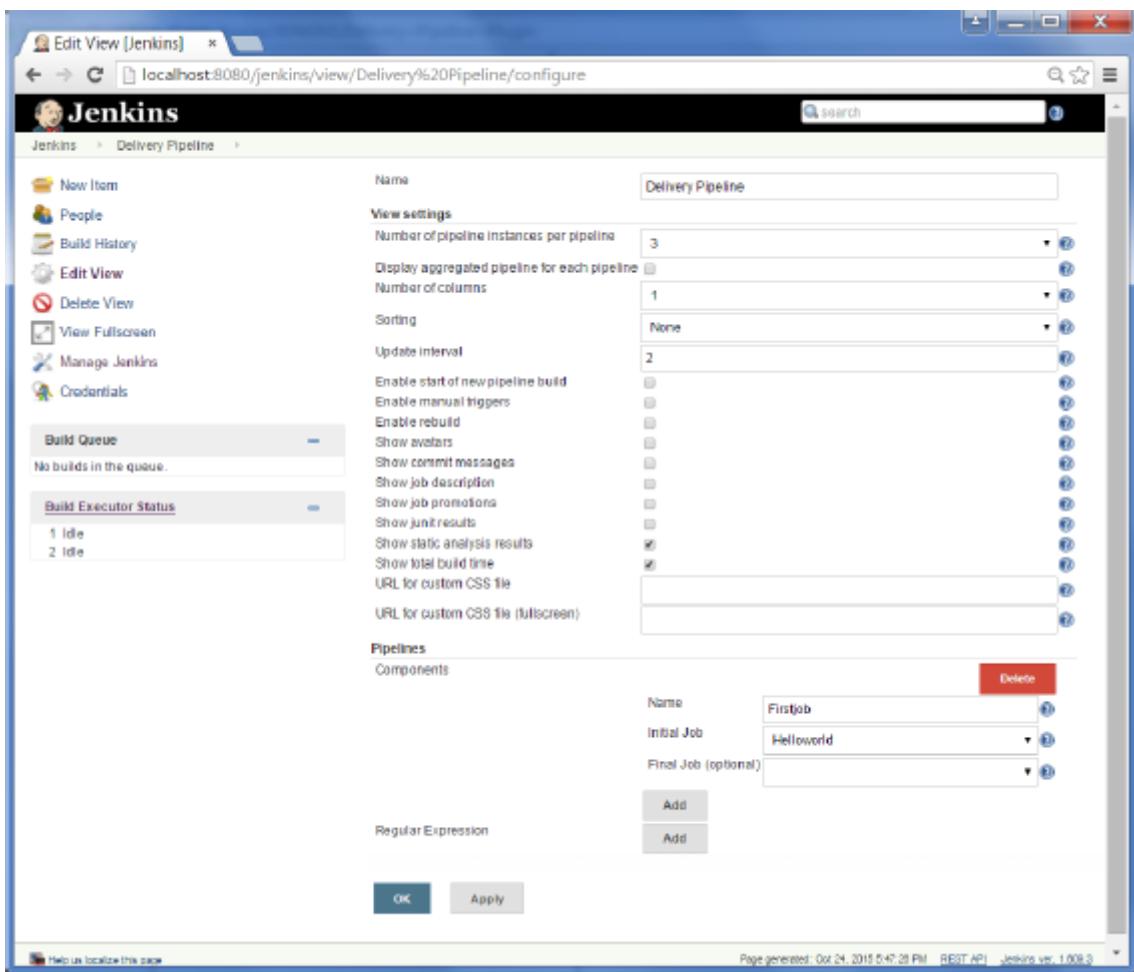
Step 9 – Enter any name for the View name and choose the option ‘Delivery Pipeline View’.



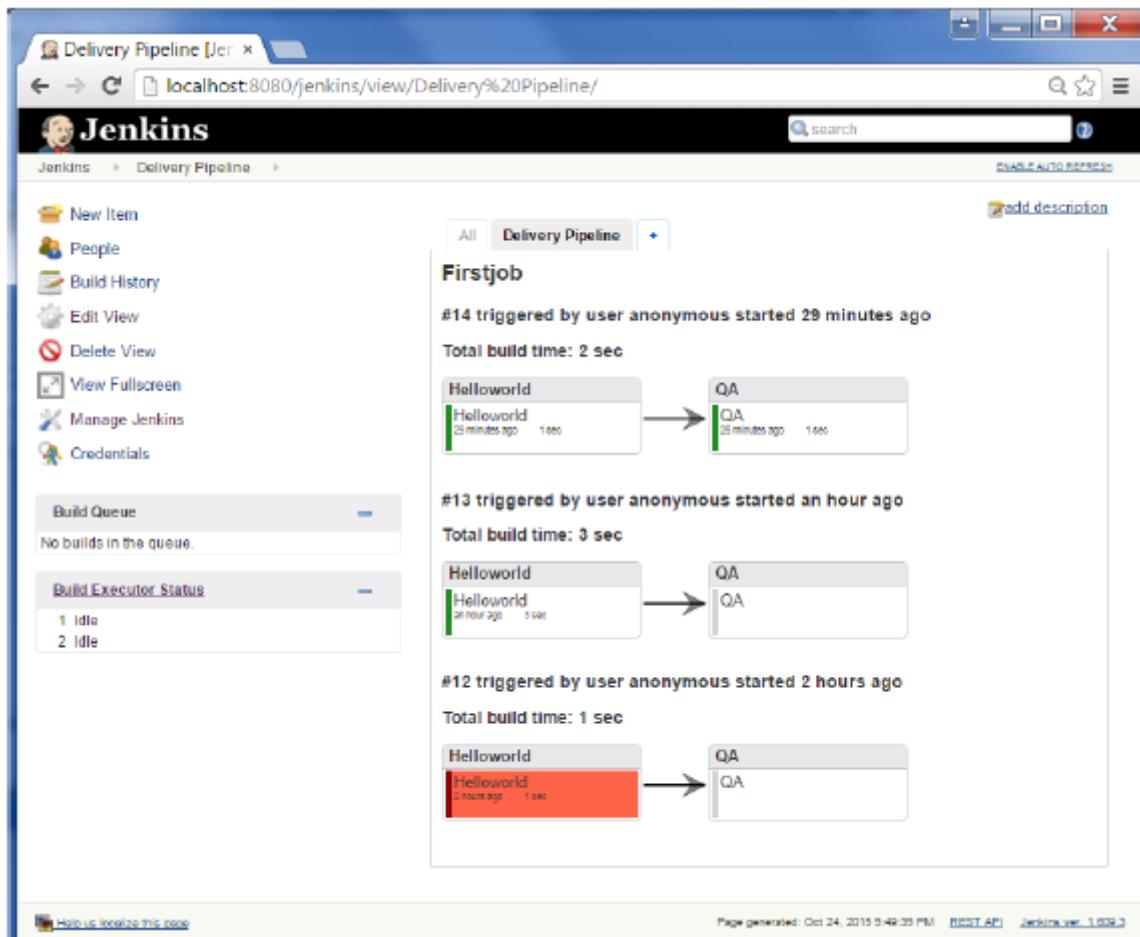
Step 10 – In the next screen, you can leave the default options. One can change the following settings –

- 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.

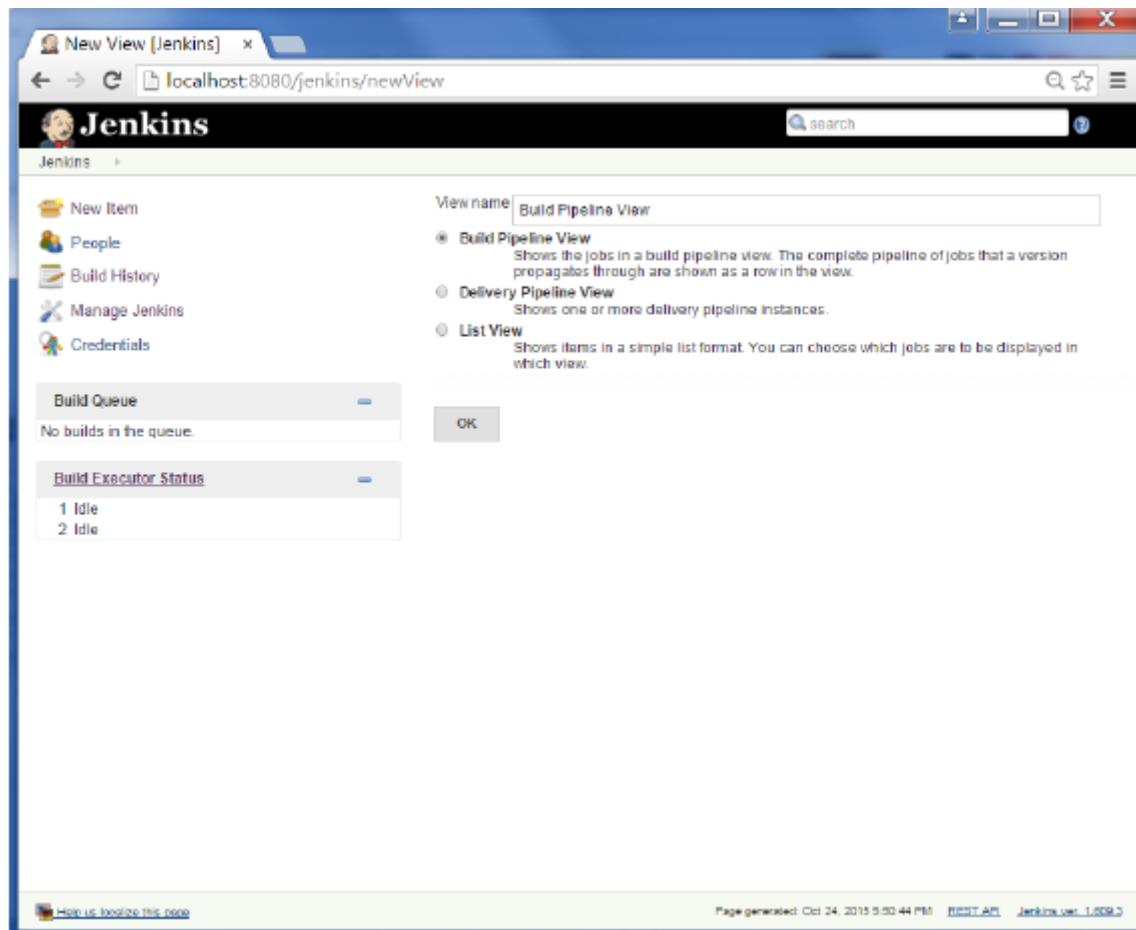
Step 1 – Go to Manage Jenkins → Manage Plugins. In the available tab, search for ‘Build Pipeline Plugin’. Click On Install without Restart. Once done, restart the Jenkins instance.

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Jenkins]". The address bar shows "localhost:8080/jenkins/pluginManager/available". The main area has a "Available" tab selected. A search bar at the top right contains the text "Build pipeline". Below it is a table with columns "Name" and "Version". The first item in the list is the "Build Pipeline Plugin" with version 1.4.8, which is currently selected for installation. Other listed plugins include "Fail The Build Plugin" (version 1.0), "Runscope plugin" (version 1.44), "Build Graph View Plugin" (version 1.1.1), and "Delivery Pipeline Plugin" (version 0.9.7). At the bottom are two buttons: "Install without restart" and "Download now and install after restart".

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

The screenshot shows the Jenkins Dashboard. The title bar says "Dashboard [Jenkins]". The address bar shows "localhost:8080/jenkins/". The main area has a sidebar with links like "New item", "People", "Build History", "Manage Jenkins", and "Credentials". The main content area has a table titled "All" with columns "Name", "Last Success", "Last Failure", and "Last Duration". It lists two builds: "Helloworld" (last success 25 min - 8:14, last failure 1 hr 49 min - 8:12, duration 1.4 sec) and "QA" (last success 25 min - 8:5, last failure 28 min - 8:2, duration 1.4 sec). Below the table are sections for "Build Queue" (empty) and "Build Executor Status" (1 idle, 2 idle). At the bottom is a footer with links "Help us localize this page", "Page generated: Oct 24, 2015 4:49:09 PM", "REST API", and "Jenkins ver. 1.609.3".

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



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

Edit View [Jenkins] > localhost:8080/jenkins/view/Build%20Pipeline%20View/configure

Jenkins

Build Pipeline View

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

Build Queue No builds in the queue.

Build Executor Status 1 Idle 2 Idle

Name: Build Pipeline View Description:

Filter build queue: Build Pipeline View Title: Layout: Based on upstream/downstream relations

This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs.

Select Initial Job: HelloWorld

No Of Displayed Builds: 1 3

Restrict triggers to most recent successful builds: Yes No

Always allow manual trigger on pipeline steps: Yes No

Show pipeline project headers: Yes No

Show pipeline parameters in project headers: Yes No

Show pipeline parameters in revision box: Yes No

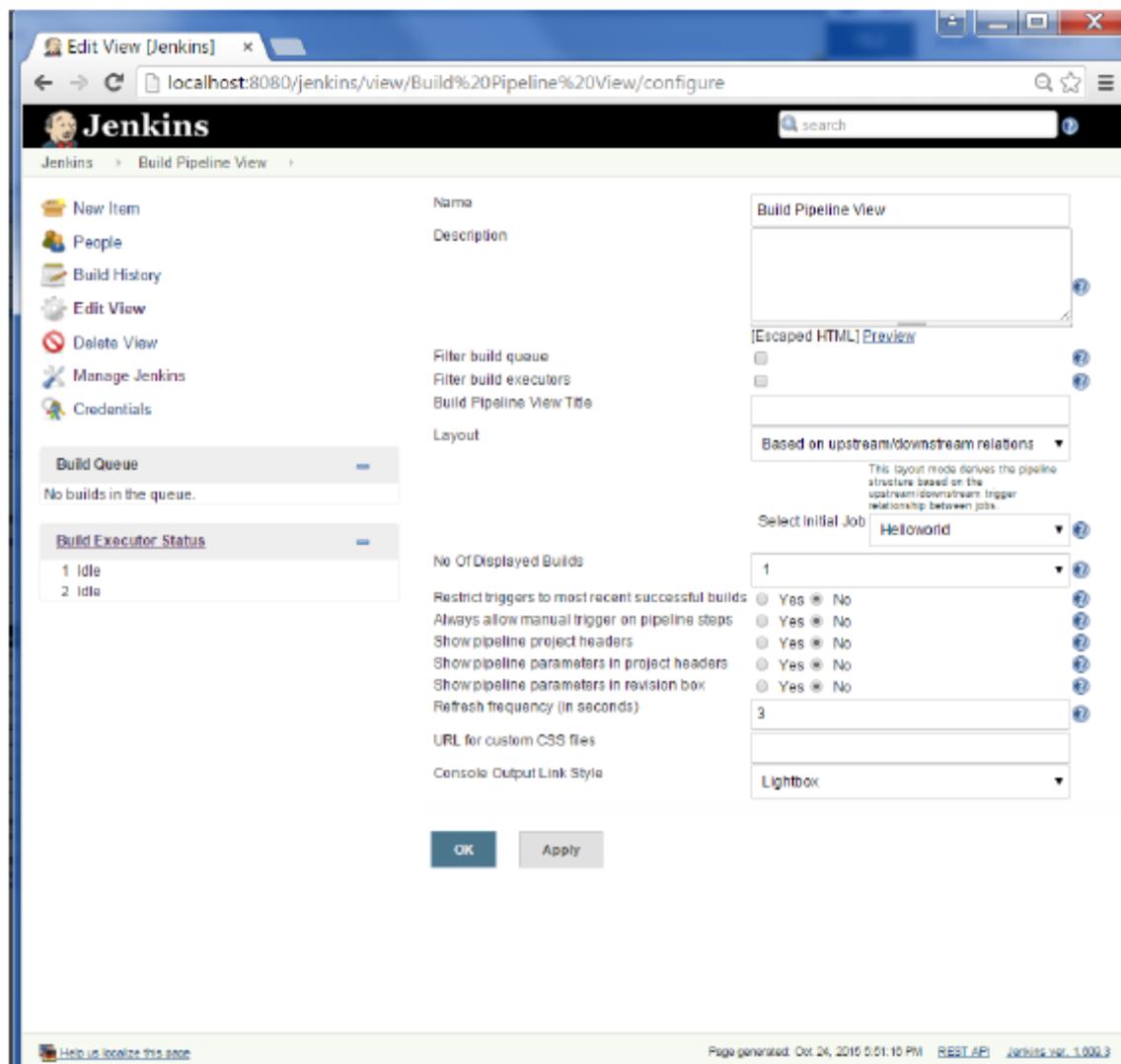
Refresh frequency (in seconds):

URL for custom CSS files:

Console Output Link Style: Lightbox

OK Apply

Help us localize this page Page generated: Oct 24, 2015 5:51:10 PM REST API Jenkins ver. 1.609.3



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

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

Jenkins

Build Pipeline

Pipeline #14 #14 HelloWorld #15

Run History Configure Add Step Cancel Stop



Jenkins - Managing Plugins

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

The screenshot shows a web browser window with the title "Plugins - Jenkins - Jenkins". The URL in the address bar is <https://wiki.jenkins-ci.org/display/JENKINS/Plugins>. The page content is titled "Plugins". A sidebar on the left has sections for "Jenkins" (Home, Mailing lists, Source code, Bugtracker, Security Advisories, Events, Donation, Commercial Support, Wiki Site Map) and "Documents" (Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, Servlet Container Notes). The main content area starts with "1 How to install plugins" and lists several sub-sections: 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 notifications, 4.6 Slave launchers and controllers, 4.7 Build reports, 4.8 Artifact upenders, 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 Game development).

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

Uninstalling Plugins

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

The screenshot shows the Jenkins Plugin Manager interface. The 'Installed' tab is selected. A table lists several Jenkins plugins:

Enabled	Name	Version	Previously Installed Version	Pinned	Uninstall
✓	Ant Plugin	1.2			Uninstall
✓	Build History Metrics Plugin	1.6			Uninstall
✓	Build Pipeline Plugin	1.4.8			Uninstall
✓	Credentials Plugin	1.23	Downgrade to 1.18	Unpin	Uninstall
✓	CVS Plugin	2.11			Uninstall
✓	Custom Pipeline Plugin	0.5.7			Uninstall
✓	Deploy to container Plugin	1.10			Uninstall
✓	External Monitor Job Type Plugin	1.4			Uninstall
✓	Failure Notification Plugin	1.1			Uninstall
✓	FindBugs Plugin	4.6.5			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.

The screenshot shows the Jenkins Plugin Manager interface with the 'Advanced' tab selected. It includes three main sections:

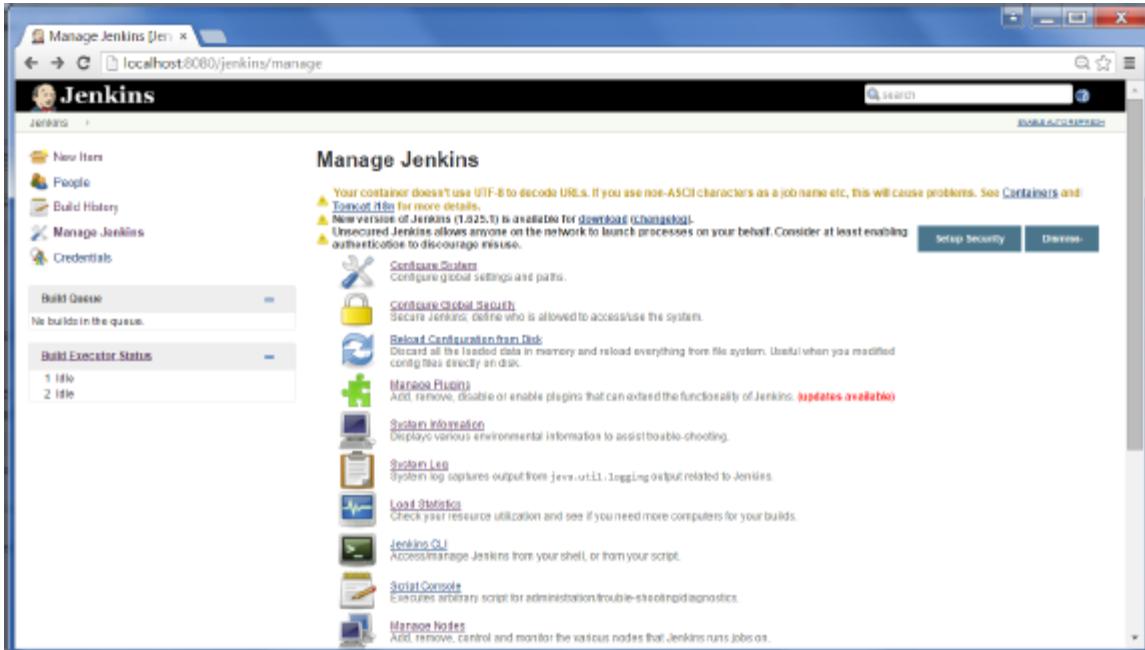
- HTTP Proxy Configuration:** Fields for Server, Port, Username, Password, and No Proxy Host, with a 'Submit' button and an 'Advanced...' link.
- Upload Plugin:** A section for uploading a JAR file from outside the central plugin repository, with a 'File' input field showing 'No file chosen', a 'Choose File' button, and an 'Upload' button.
- Update Site:** A section for specifying the URL of an update site, with a 'URL' input field containing 'http://updates.jenkins-ci.org/update-center/sof' and a 'Submit' button. Below it, a message says 'Update information obtained: 2 hr 5 min ago' and a 'Check now' button.

Jenkins - Security

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

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

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



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

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



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

The screenshot shows a web browser window with the Jenkins 'Sign up' page. The URL in the address bar is `localhost:8080/jenkins/securityRealm/firstUser`. The page title is 'Sign up [Jenkins]'. On the left sidebar, there are links: 'Back to Dashboard', 'Manage Jenkins', and 'Create User'. The main content area has a heading 'Sign up'. It contains five input fields:

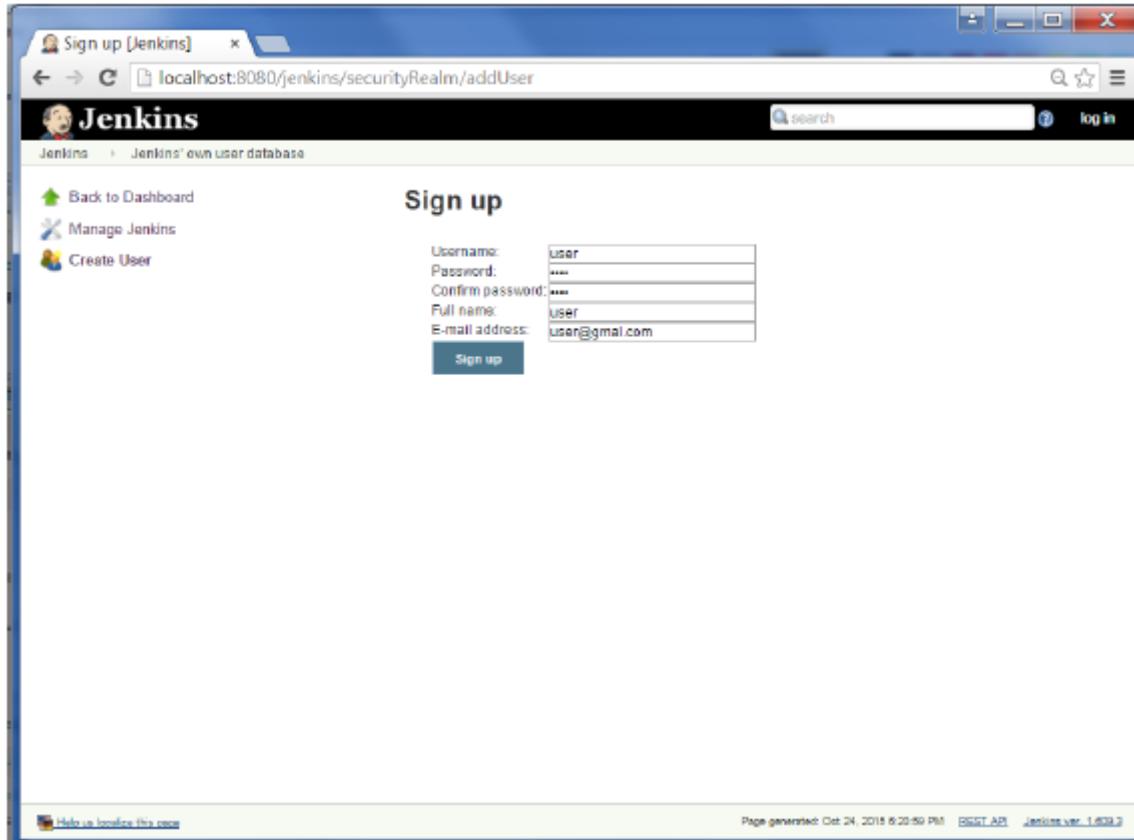
Username:	admin
Password:
Confirm password:
Full name:	Administrator
E-mail address:	al@gmail.com

Below the form is a blue 'Sign up' button. At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 6:19:01 PM', 'REST API', and 'Jenkins ver. 1.606.3'.

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

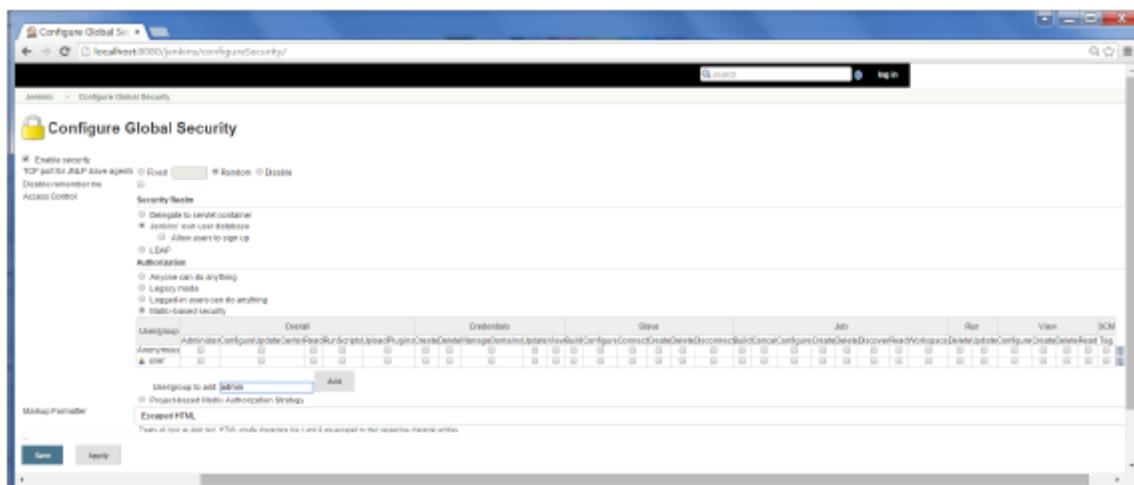


Step 5 – Just like you defined your admin user, start creating other users for the system. As an example, we are just creating another user called ‘user’.



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

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



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

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

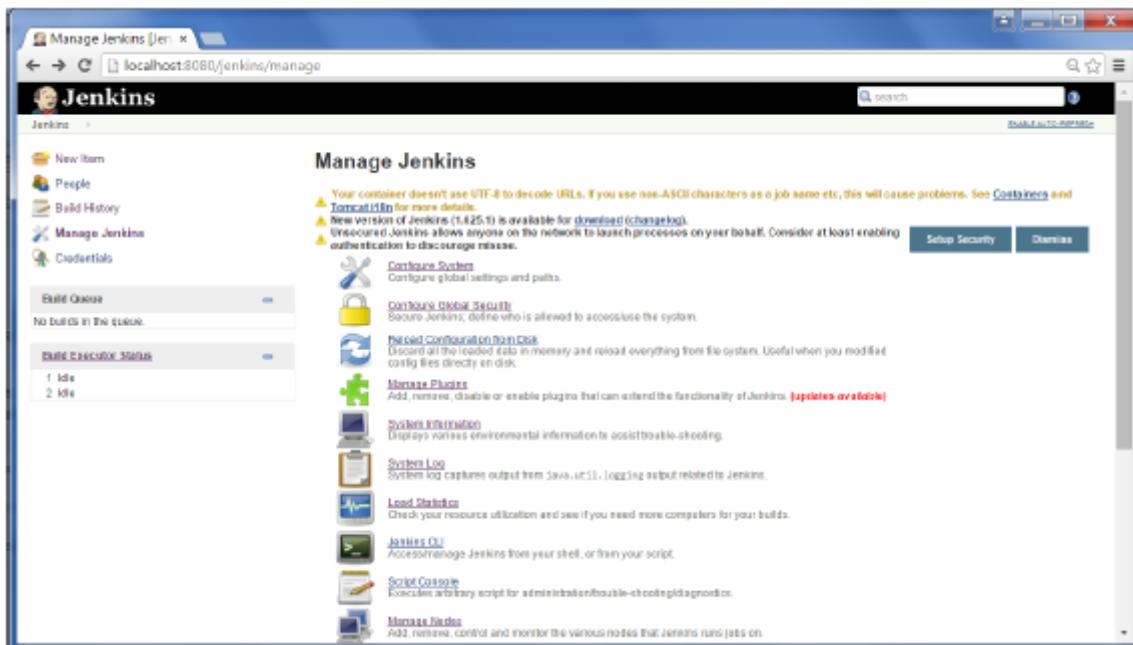
Your Jenkins security is now setup.

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

Jenkins - Backup Plugin

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

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



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

Available Tab Screenshot:

Plugin	Name	Version
Backup plugin	Backup plugin allows archiving and restoring your Jenkins (and Hudson) home directory.	1.8.1
Backup and restore job plugin	Backup up and restore running jobs.	1.0
Install CloudBees Jenkins Enterprise	This plugin converts an OSB installation to a CloudBees Jenkins Enterprise (CBJE) installation. CJE has 20+ plugins that address issues in Jenkins such as: how to migrate from Jenkins to CJE, how to use templates, roles-based access control, backup plugins and others.	15.05.1
CloudBees Free Enterprise Plugins	This plugin installs free enterprise plugins from CloudBees. The following plugins are currently installed: "Folders" easily organize your jobs, "Backup to Cloud" backup your Jenkins into CloudBees cloud, "Wanted Minutes" find out if you are short of slaves and need to add capacity to speed up builds, "CloudBees Status" find out how much of the free CloudBees Jenkins capacity in the cloud is available for your use. *Note: You will be asked to register for a free CloudBees account to use these plugins (This plugin was formerly known as the CloudBees Plugin Gateway plugin).	5.0
Perforce Backup		1.3
ThinBackup	This plugin simply backs up the global and job specific configurations (not the archive or the workspace).	1.7.4

Installation Success Screenshot:

Preparation:

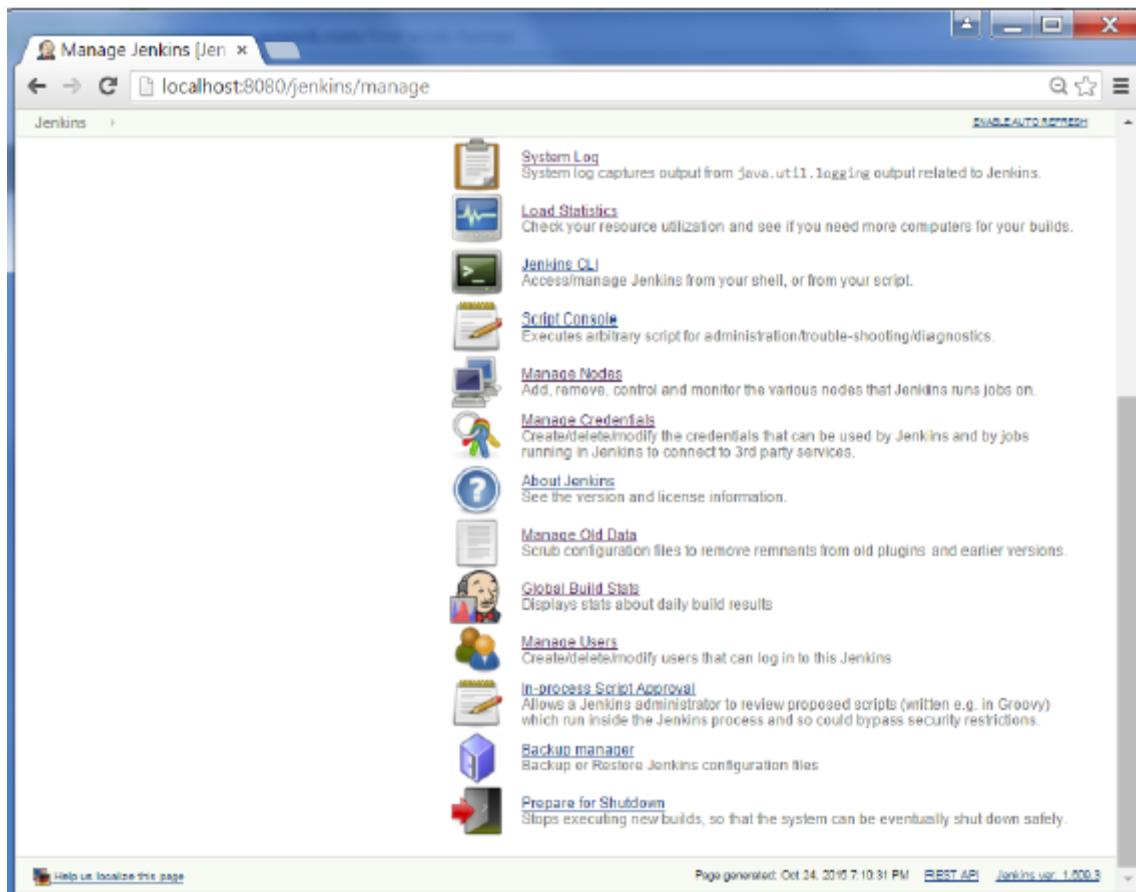
- Checking internet connectivity
- Checking update center connectivity
- Success

Backup plugin: Success

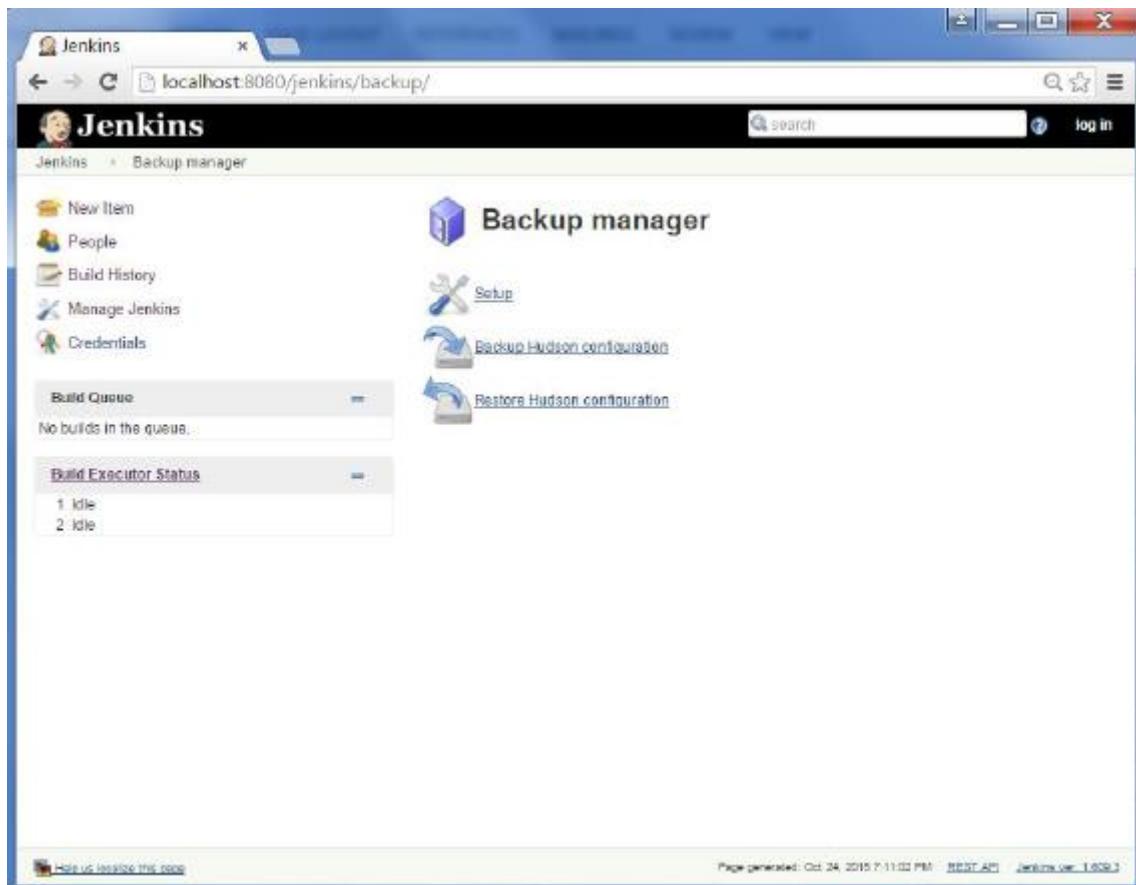
Success Messages:

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

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



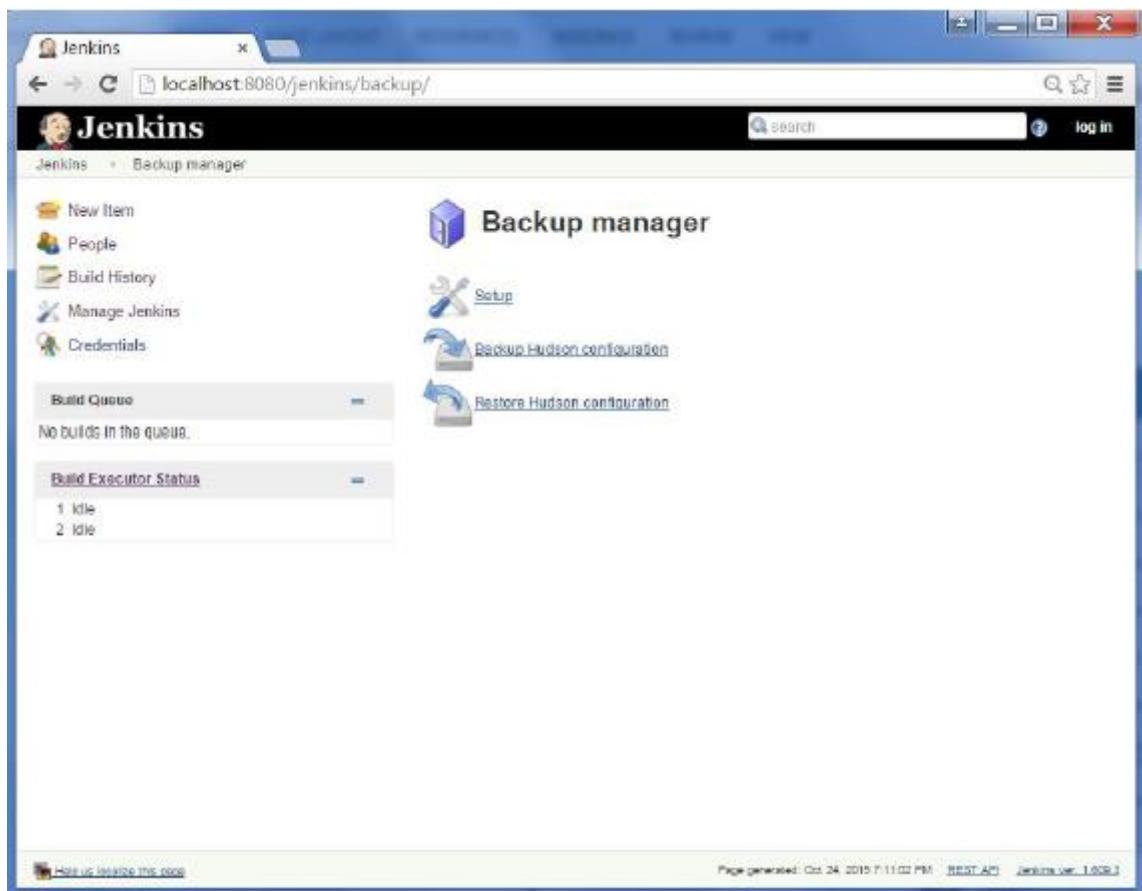
Step 4 – Click on Setup.



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.

The screenshot shows the Jenkins interface at localhost:8080/jenkins/backup/backupsettings. The left sidebar includes links for New Item, People, Build History, Manage Jenkins, and Credentials. Under Build Queue and Build Executor Status, there are no builds listed. The main content area is titled "Backup config files" and contains "Backup configuration" settings. These include "Hudson root directory" set to "E:\Jenkins", "Backup directory" set to "D:\Backup", "Format" set to "zip", "File name template" set to "backup_@date@.@(extension@)", and "Custom exclusions". Below these are "Backup content" options: "Backup job workspace" (selected), "Backup builds history", "Backup maven artifacts archives", and "Backup fingerprints". A "Save" button is at the bottom right. The footer includes links for Help us localize this page, Page generated: Oct 24, 2015 7:17:45 PM, REST API, and Jenkins ver. 1.600.3.

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



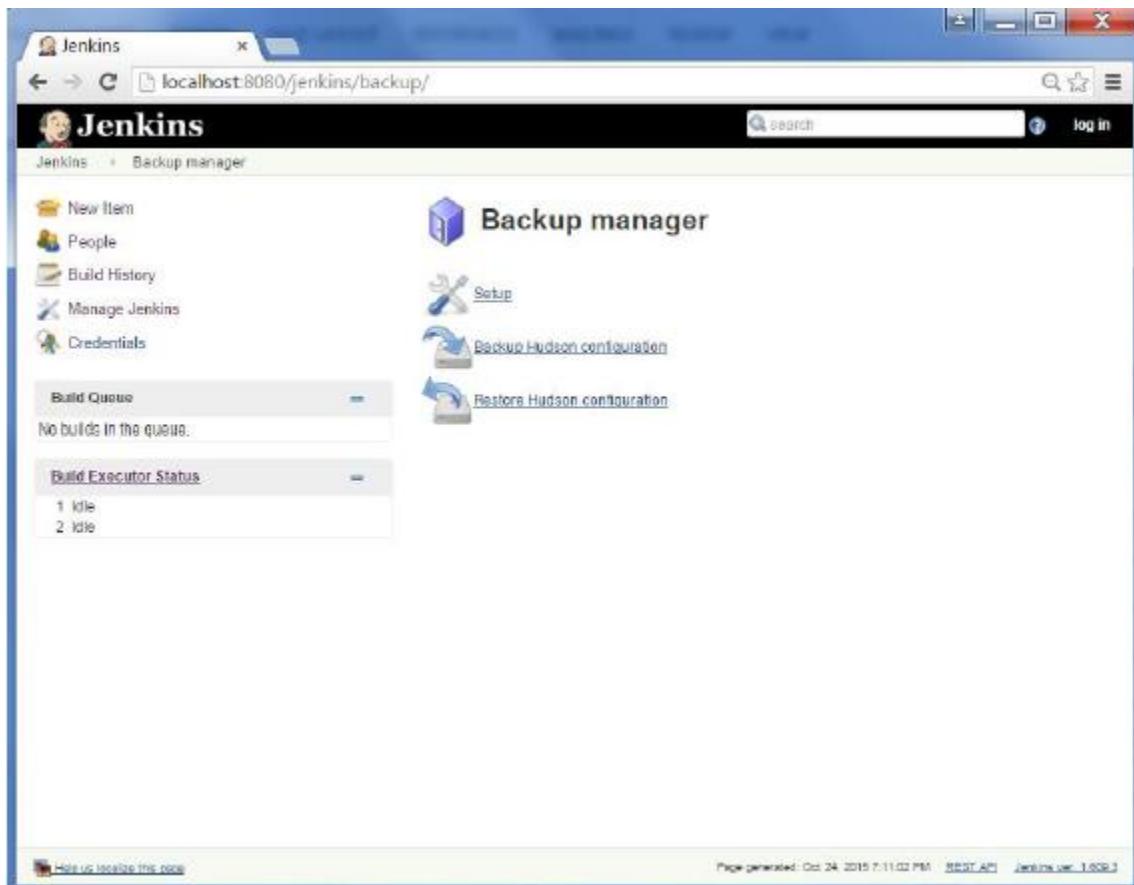
The next screen will show the status of the backup

The screenshot shows a Jenkins web interface titled "Backup manager log". The URL in the address bar is "localhost:8080/jenkins/backup/backup". The main content area displays a log message: "Jenkins is going to shut down" in a red box, followed by a detailed log of the backup process:

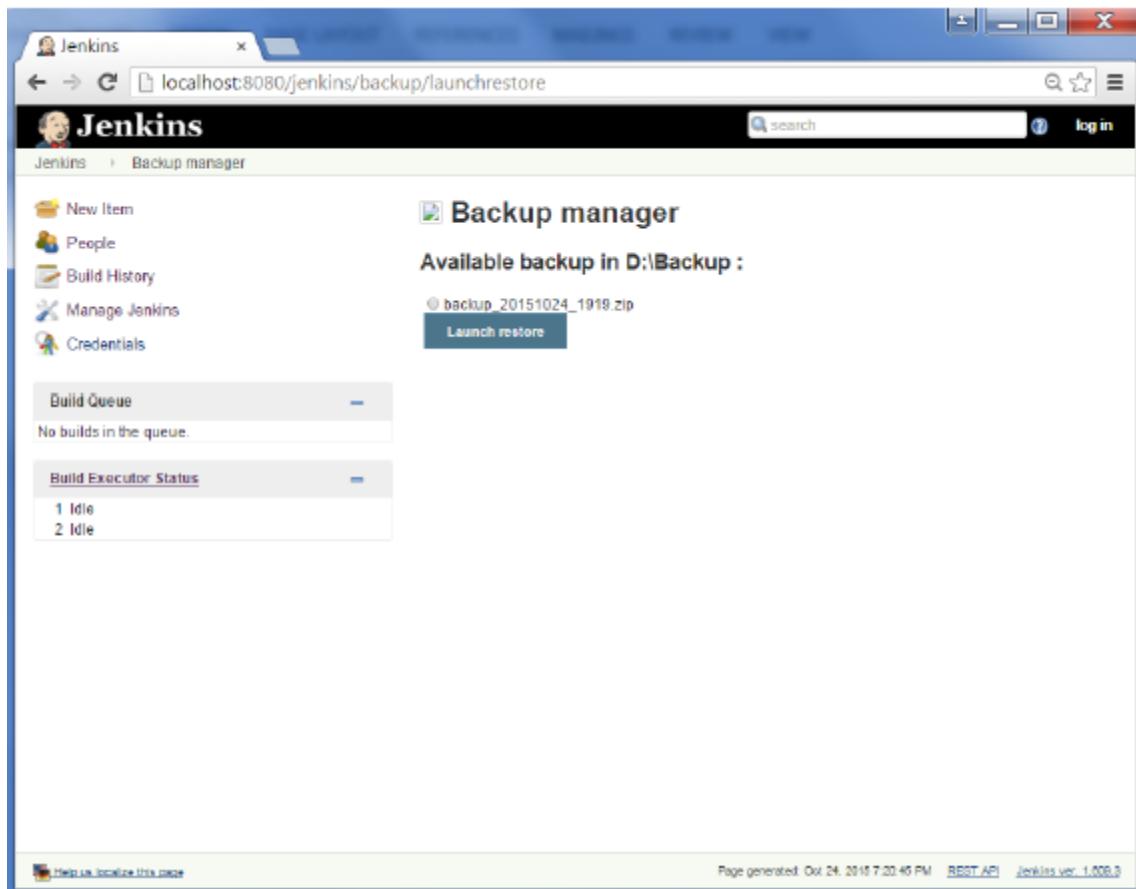
```
[ INFO] Backup started at [18/24/15 19:19:31]
[ INFO] Setting hudson in shutdown mode to avoid files corruptions.
[ INFO] Waiting all jobs end...
[ INFO] Number of running jobs detected : 0
[ INFO] All jobs finished.
[ INFO] Full backup file name : D:\Backup\backup_20151024_1919.zip
[ INFO] Saved files : 011
[ INFO] Number of errors : 0
[ INFO] Cancel hudson shutdown mode
[ INFO] Backup end at [18/24/15 19:19:30]
[ INFO] [19.524s]
```

The left sidebar contains 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).

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



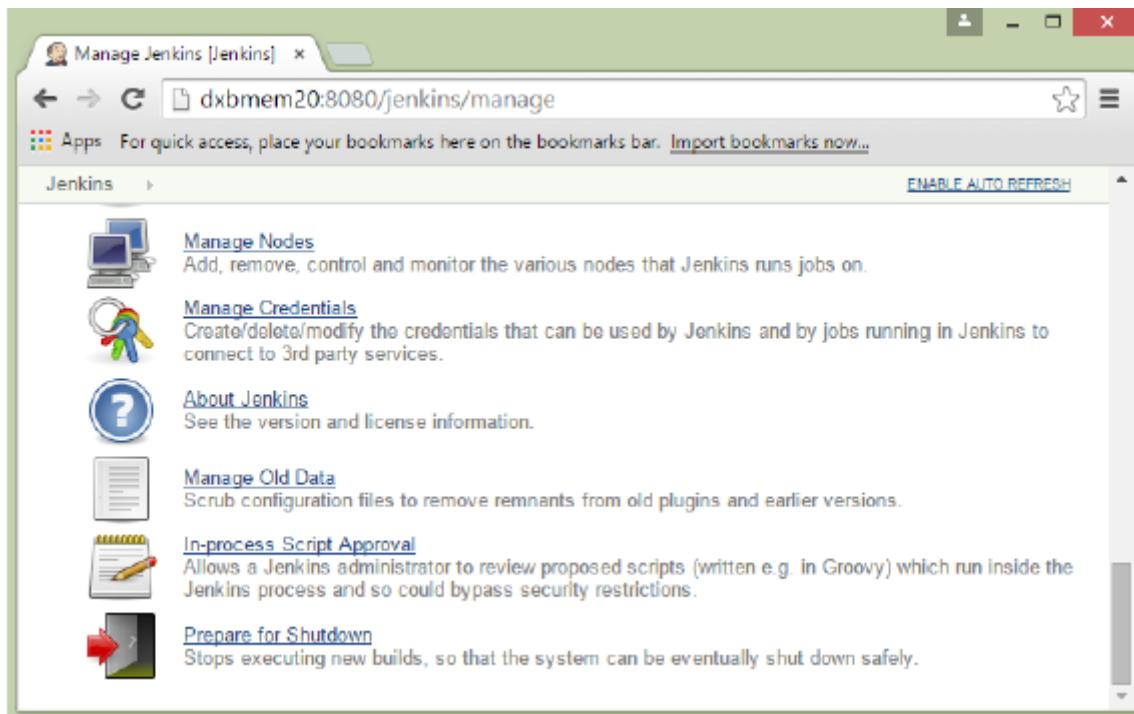
The list of backup's will be shown, click on the appropriate one to click on Launch Restore to begin the restoration of the backup.



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.

A screenshot of a web browser window titled "Nodes [Jenkins]". The URL in the address bar is "dxbmemp20:8080/jenkins/computer/". The page shows a table of nodes. The table has columns: S, Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, and Free Temp Sp. There are two entries: "DXBMEM30" and "master". Both entries show "Windows Server 2012 (x86)" under "Architecture". Under "Clock Difference", both are listed as "In sync". Under "Free Disk Space", DXBMEM30 shows 112.79 GB and master shows 94.20 GB. Under "Free Swap Space", DXBMEM30 shows 4.54 GB and master shows 3.85 GB. Under "Free Temp Sp", DXBMEM30 shows 112.79 and master shows 94.20. A "Refresh status" button is at the bottom right. The table also includes a "Data obtained" row with "13 min" under all columns except the first.

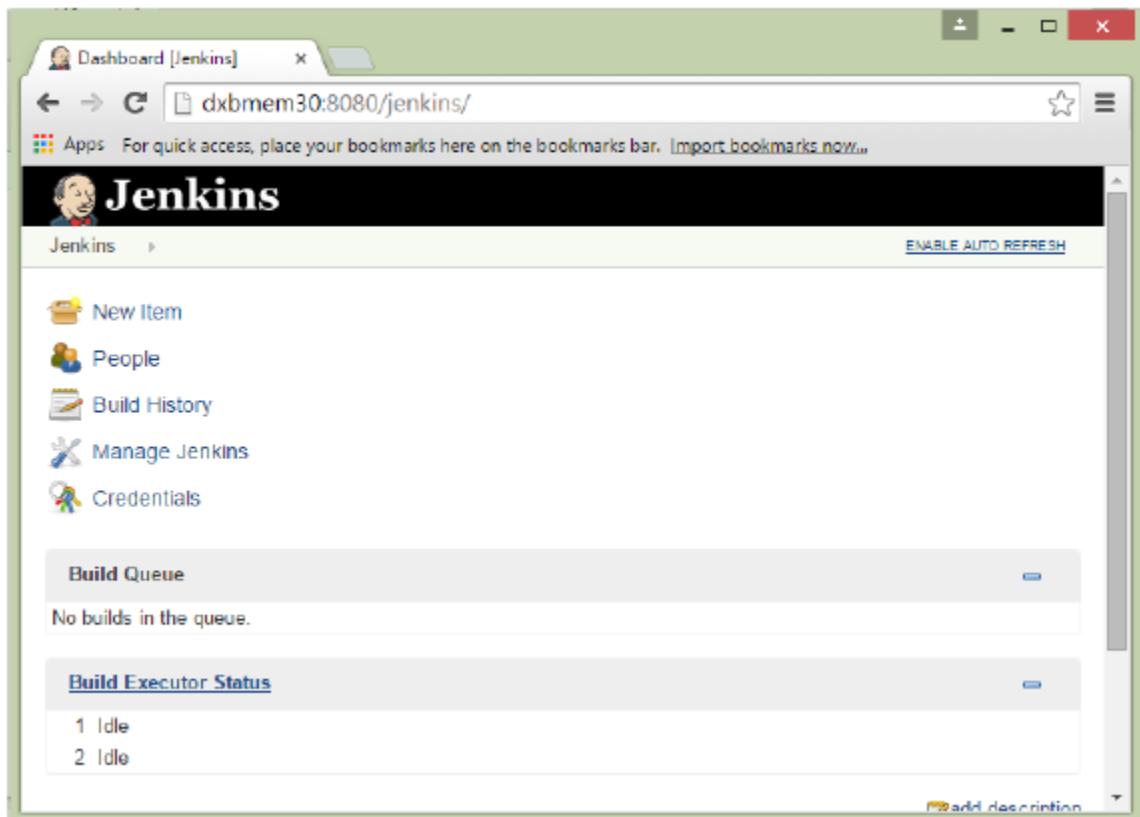
Step 2 – Click on configure for the DXBMEM30 slave machine.

The screenshot shows the Jenkins 'Nodes' page. There are two nodes listed: 'DXBMEM30' (Windows Server 2012) and another unnamed node. The 'DXBMEM30' node has a context menu open over it, with the 'Configure' option highlighted. Other options in the menu include 'Delete Slave' and 'Build History'. A 'Refresh status' button is visible at the bottom right of the table.

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

The screenshot shows the 'DXBMEM30 Configuration' page. The 'Launch method' dropdown is set to 'Launch slave agents via Java Web Start'. Other configuration options shown include Name (DXBMEM30), Description, # of executors (1), Remote root directory (C:\users\administrator EMIRATES\jenkins), Labels, Usage (Utilize this node as much as possible), and an 'Advanced...' button. A 'Save' button is at the bottom left.

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 Jenkins interface for managing nodes. At the top, there's a blue header bar with the title 'Nodes [Jenkins]'. Below it is a browser-style address bar with the URL 'dxbmemo20:8080/jenkins/computer/'. The main content area has a navigation bar with 'Jenkins' and 'nodes' selected, and a 'ENABLE AUTO REFRESH' link. A message says 'No builds in the queue.' Below this is a section titled 'Build Executor Status'.

master

- 1 Idle
- 2 Idle

DXBMEM30 (offline)

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

Refresh status

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

The screenshot shows a web browser window with the URL `dxbmemp20:8080/jenkins/computer/DXBMEM30/`. The page title is "DXBMEM30 [Jenkins]". The main content area displays the following information:

- Connection instructions:
 - Launch agent from browser on slave
 - Run from slave command line:

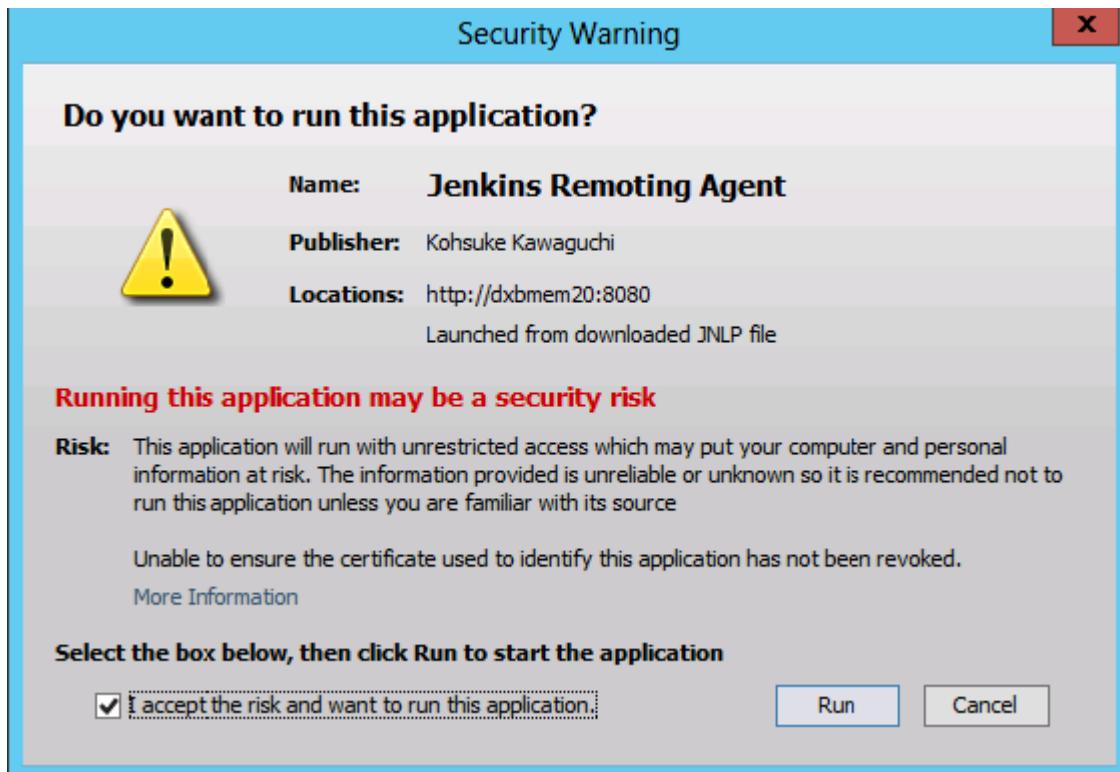
```
javaws http://localhost:8080/jenkins/computer/DXBMEM30/slave-agent.jnlp
```
 - Or if the slave is headless:

```
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.



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



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

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

The screenshot shows the Jenkins configuration page for the 'HelloWorld' job. The URL is `dxbmem20:8080/jenkins/job/HelloWorld/configure`. The configuration includes:

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

Label Expression: `DXBMEM30`
Slaves in [label]: 1

Advanced Project Options

Source Code Management

Save Apply

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

The screenshot shows the Jenkins configuration page for the 'HelloWorld' job, focusing on the build steps. A 'SeleniumHQ htmlSuite Run' step is selected, with the following configuration:

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

Buttons: Delete, Add build step, Save, Apply

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

Thank you for Viewing, Hope this Jenkins hands on Tour will make you learn some basic Information about the work flow of Jenkins and real time experience.

Many thanks for viewing

Senthilkumar.Sivakumar