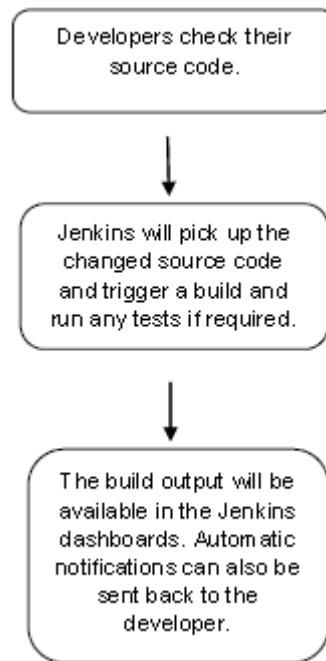


Jenkins - Quick Guide

Jenkins - Overview

Why Jenkins?

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



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

What is Continuous Integration?

Continuous Integration is a development practice that requires developers to integrate code into a shared repository at regular intervals. This concept was meant to remove the problem of finding later occurrence of issues in the build lifecycle. Continuous integration requires the developers to

have frequent builds. The common practice is that whenever a code commit occurs, a build should be triggered.

System Requirements

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

Jenkins - Installation

Download Jenkins

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

The screenshot shows the Jenkins website at <https://jenkins-ci.org>. The page features a large logo of the Jenkins mascot, a man with a red bow tie holding a coffee cup. Below the logo, the word "Jenkins" is written in a large, bold, black font. A banner at the top right says "Find me on GitHub". The main content area has four sections: "Meet Jenkins" (with an icon of a person), "Use Jenkins" (with an icon of two people), "Customize Jenkins" (with an icon of a green puzzle piece), and "Extend Jenkins" (with an icon of a wrench and screwdriver). To the right, there's a sidebar titled "Download Jenkins" with tabs for "Release" and "Long-Term Support Release". The "Release" tab is selected, showing links for "Java Web Archive (.war)" (with "Latest and greatest (1.832)" and "changelog | past releases"), "Native packages" (listing Windows, Ubuntu/Debian, Red Hat/Fedora/CentOS, Mac OS X, openSUSE, FreeBSD, OpenBSD, Gentoo), and "Docker" (with a Docker command: "docker pull jenkinsci/jenkins"). Below this is a "FOLLOW US ON" button with icons for GitHub, LinkedIn, and YouTube. At the bottom, there's a "Latest Blog Items" section with links to "Upcoming in office hours: Jenkins 2.8", "Bay Area JAM", "GUI improvements on the horizon", and "Office hour on form handling in Jenkins".

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



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

Starting Jenkins

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

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

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

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

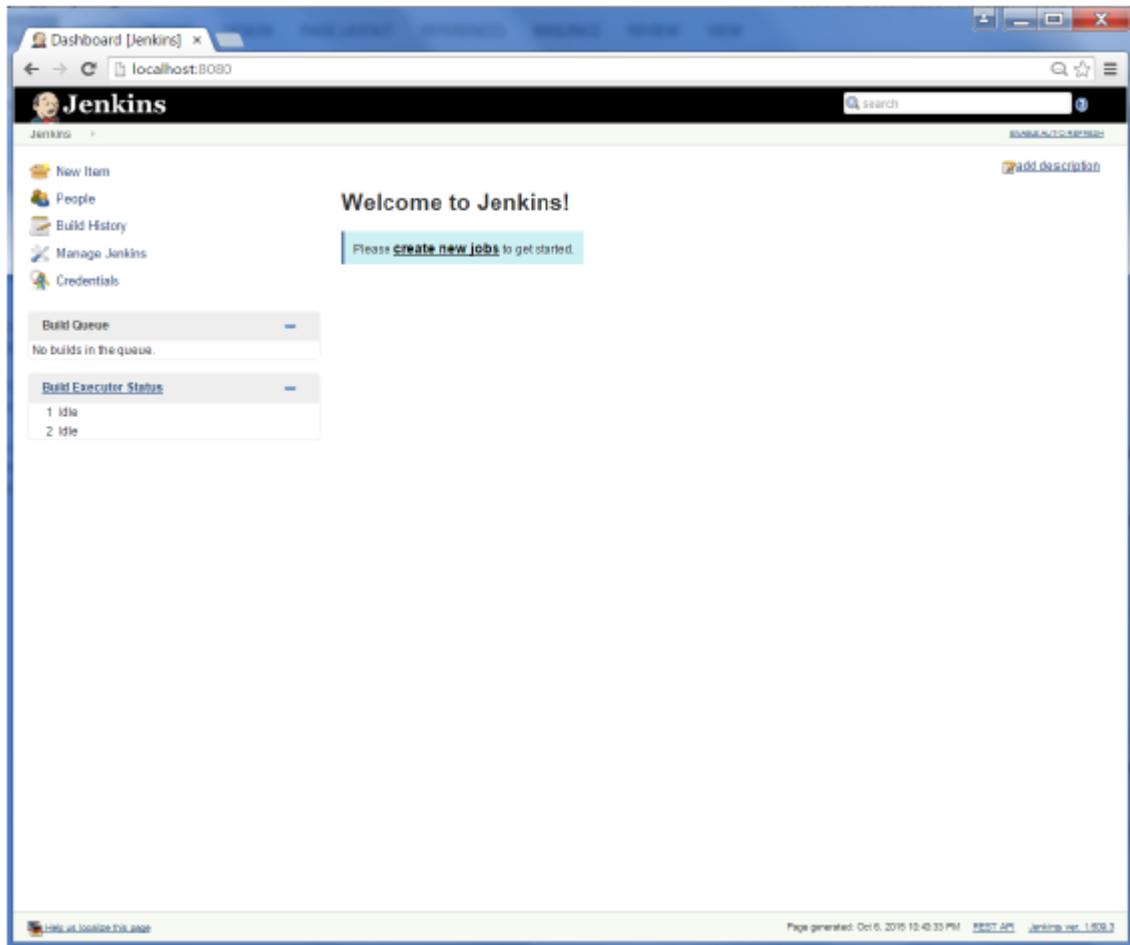
Once the processing is complete without major errors, the following line will come in the output of the command prompt.

INFO: Jenkins is fully up and running

Accessing Jenkins

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

This link will bring up the Jenkins dashboard.



Jenkins – Tomcat Setup

The following prerequisites must be met for Jenkins Tomcat setup.

Step 1: Verifying Java Installation

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

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

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

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

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

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

Step 2: Verifying Java Installation

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

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

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

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

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

Step 3: Download Tomcat

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

The screenshot shows the Apache Tomcat homepage. At the top, there's a navigation bar with links for Home, Taglibs, Maven Plugin, Download, Documentation, Problems?, Get Involved, and a search bar. The main content area features the Apache Tomcat logo (a yellow cat) and the Apache feather logo. A section titled "Apache Tomcat" provides an overview of the project. Below it, two release announcements are listed: "Tomcat 8.0.27 Released" (2015-10-01) and "Tomcat 7.0.64 Released" (2015-08-25). Each announcement includes a list of changes and a "Download" link.

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

The screenshot shows the "Tomcat 7 Downloads" page. The left sidebar has the same navigation as the homepage. The main content area is titled "Tomcat 7 Downloads" and says "Welcome to the Apache Tomcat™ 7.x download page. This page provides download links for obtaining the latest version of Tomcat 7.0.x, as well as links to the archives of older releases." It includes sections for "Quick Navigation" (KEYS | 7.0.64 | Browse | Archives), "Release Integrity" (instructions for verifying file integrity using OpenPGP signatures), "Mirrors" (information about mirrors and how to change them), and "7.0.64" (links to various binary distribution files like zip, tar.gz, and Windows zip files, along with a "Full documentation" link).

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

Then unzip the contents of the downloaded zip file.

Step 4: Jenkins and Tomcat Setup

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

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

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

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

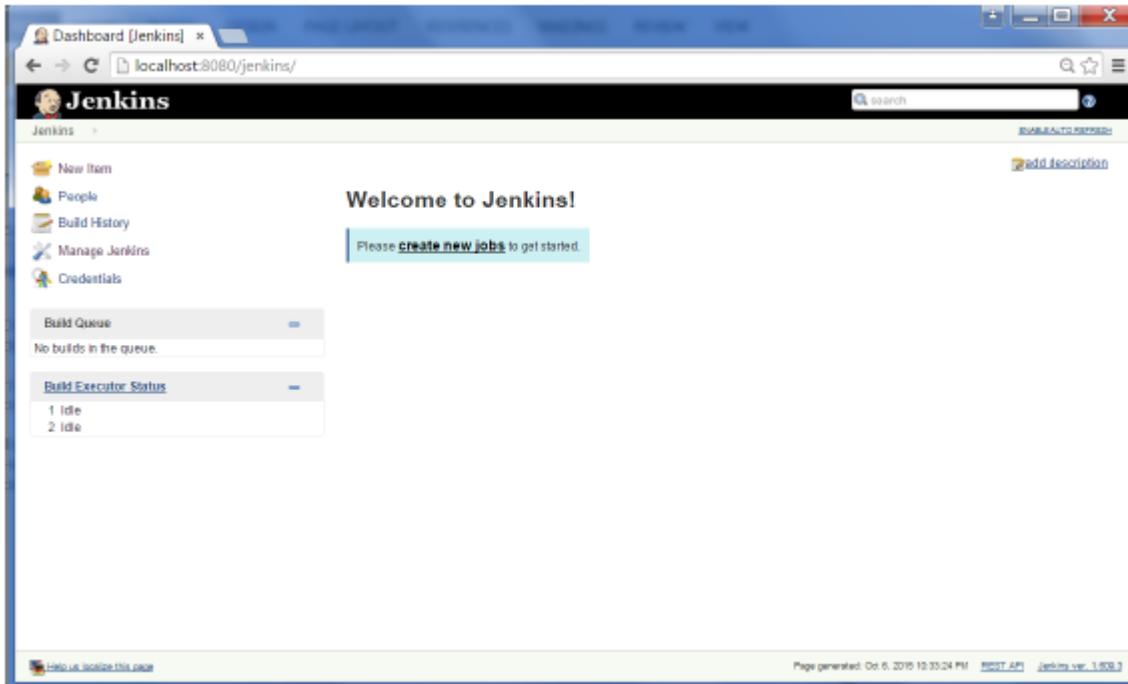
```
INFO: Server startup in 1302 ms
```

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

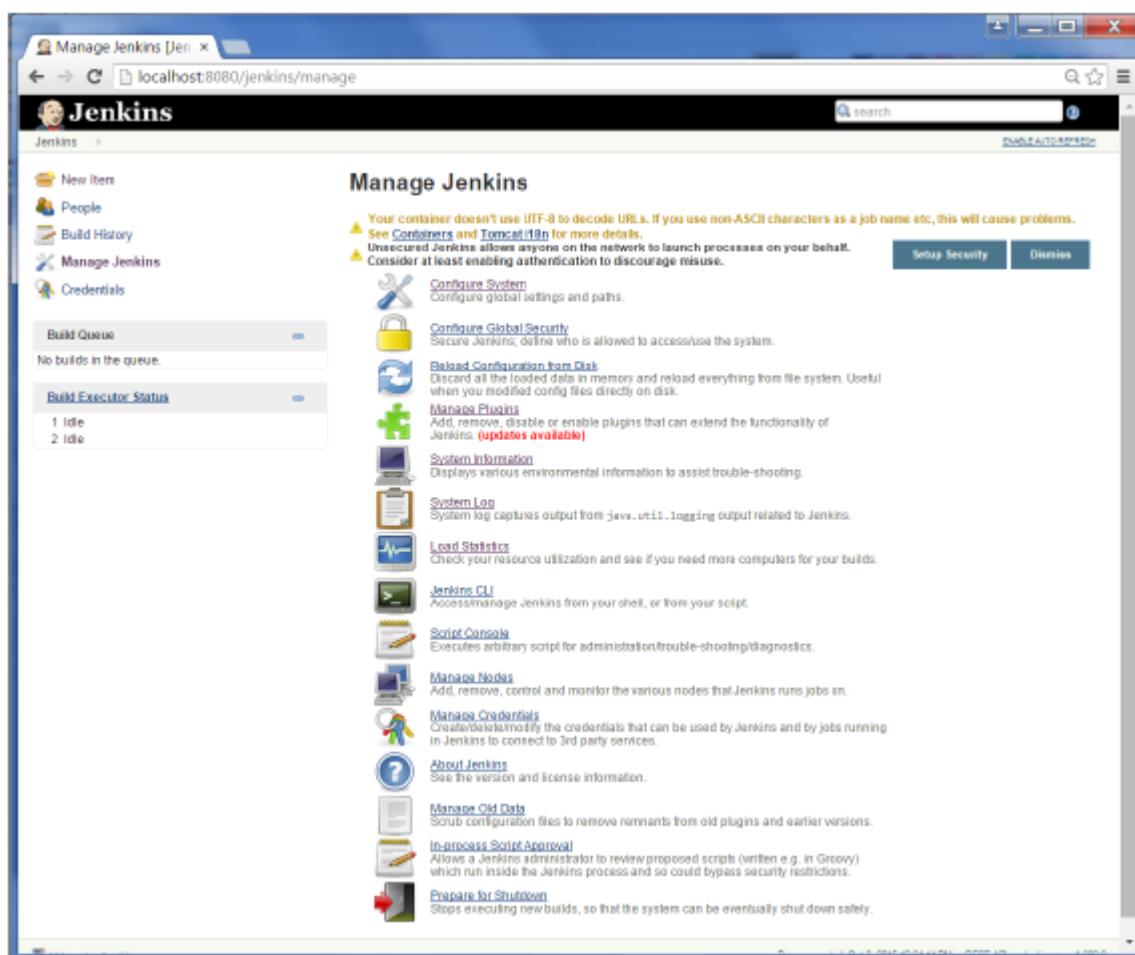


Jenkins - Git Setup

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



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



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

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Jenkins]". The URL in the address bar is "localhost:8080/jenkins/pluginManager/available". The left sidebar has "Back to Dashboard" and "Manage Jenkins" options. The top navigation bar has tabs for "Updates", "Available" (which is selected), "Installed", and "Advanced". A search bar at the top right contains the text "Filter: Git plugin". Below the tabs is a table with columns "Install", "Name", and "Version". The table lists several Git-related plugins:

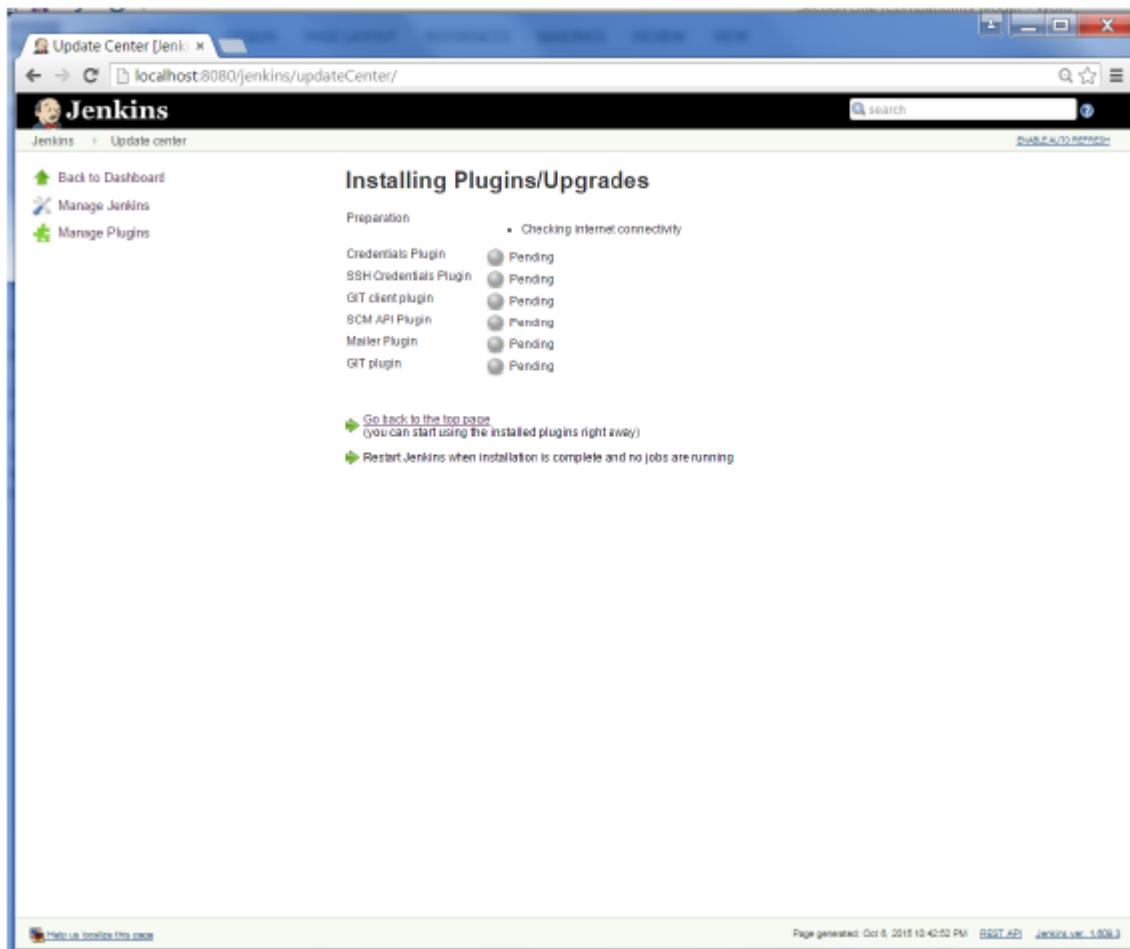
Install	Name	Version
<input type="checkbox"/>	Git Parameter Plug-in	0.4.0
<input type="checkbox"/>	UserContentInGitplugin	1.4
<input type="checkbox"/>	Alternative build chooser	1.1
<input type="checkbox"/>	Team Concert Git Plugin	1.0.10
<input type="checkbox"/>	Tracking Git Plugin	1.0
<input checked=""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

At the bottom of the page are three buttons: "Install without restart", "Download now and install after restart", and "Check now". A status message says "Update information obtained: 1 hr 0 min ago". The footer includes links to "Help us localize this page", "Page generated: Oct 6, 2015 10:30:18 PM", "REST API", and "Jenkins ver. 1.603".

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

This screenshot is identical to the one above, showing the Jenkins Plugin Manager with the "Available" tab selected and the "Git plugin" filter applied. The "GIT plugin" checkbox is checked. The "Install without restart" button is highlighted with a blue background and white text.

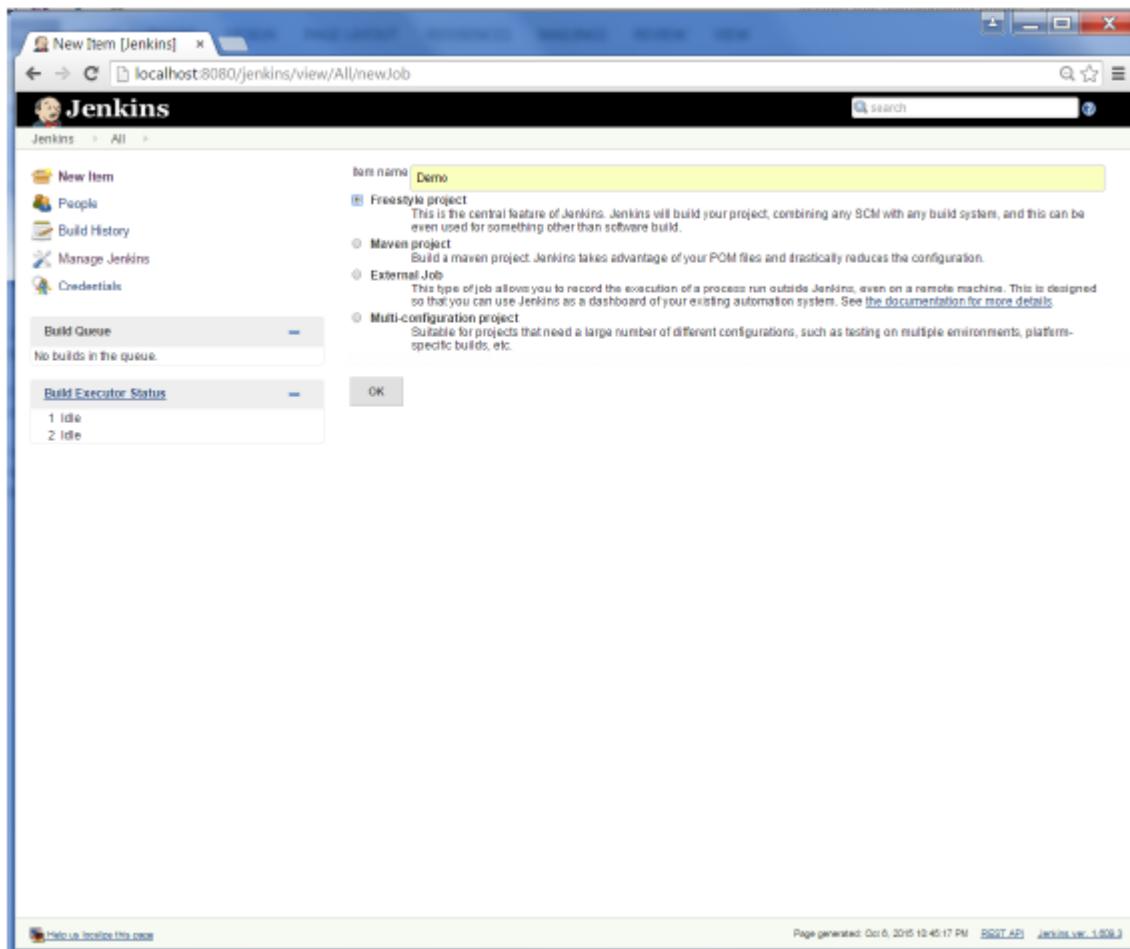
The installation will then begin and the screen will be refreshed to show the status of the download.



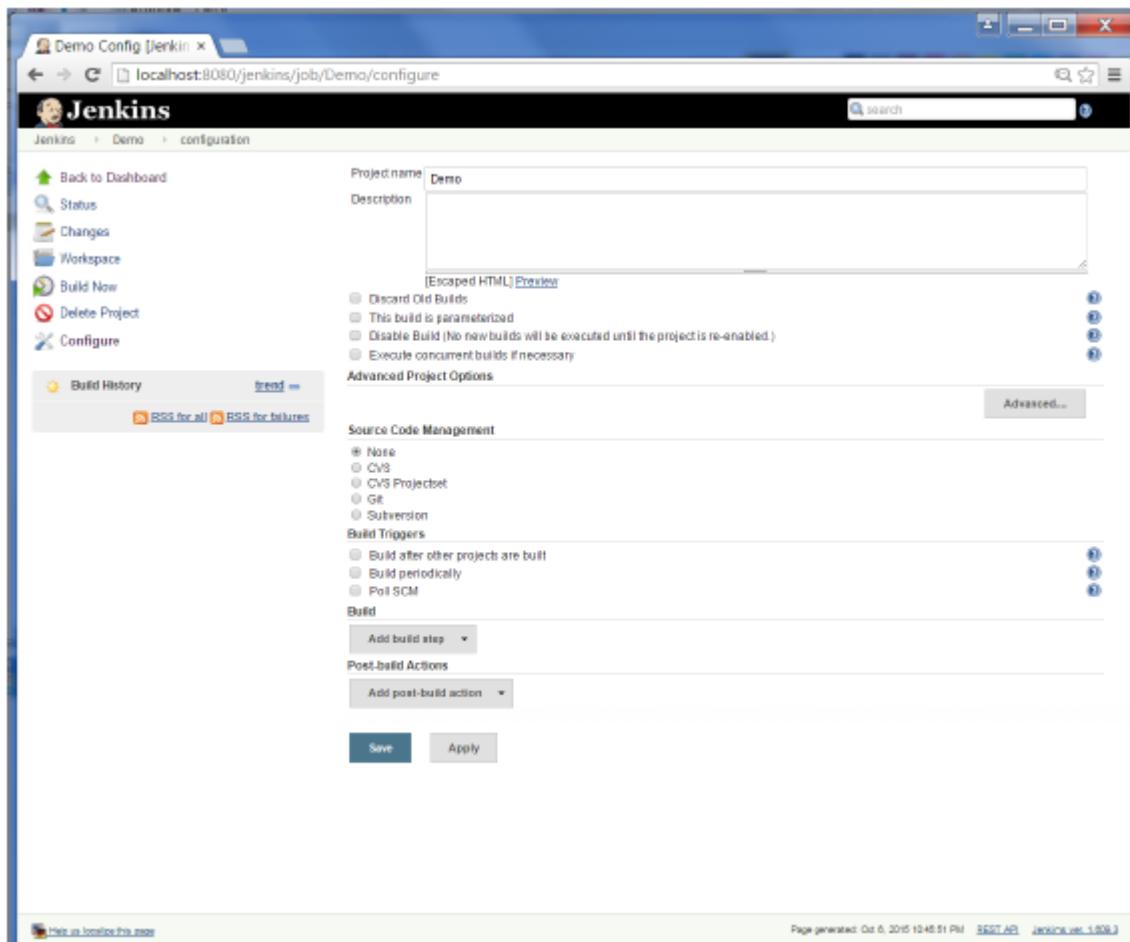
Once all installations are complete, restart Jenkins by issue the following command in the browser.

<http://localhost:8080/jenkins/restart>

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



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



Jenkins – Maven Setup

Step 1: Downloading and Setting Up Maven

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

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

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

Files

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

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

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

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

Previous Releases

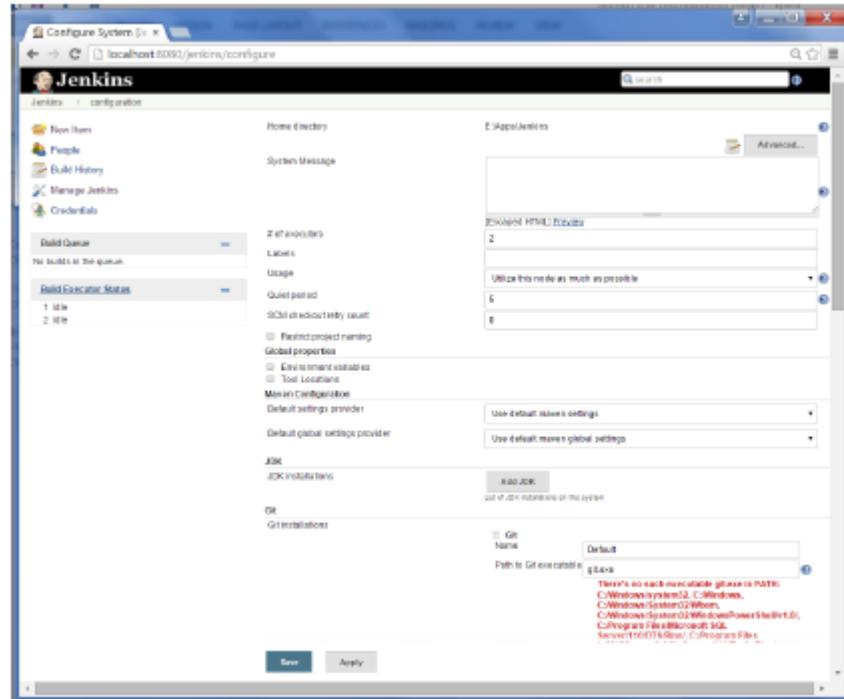
It is strongly recommended to use the latest release version of Apache Maven to take advantage of newest features and bug fixes. If you still want to use an old version you can find more information in the [Maven Releases History](#) and can download files from the [archives](#) for versions 3.0.4+ and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

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

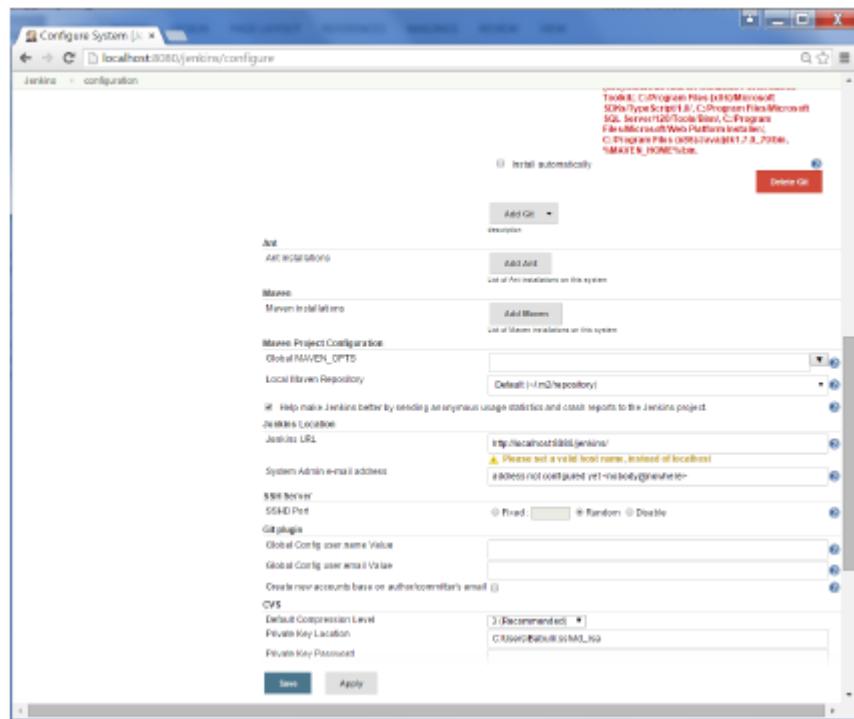
Step 2: Setting up Jenkins and Maven

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

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



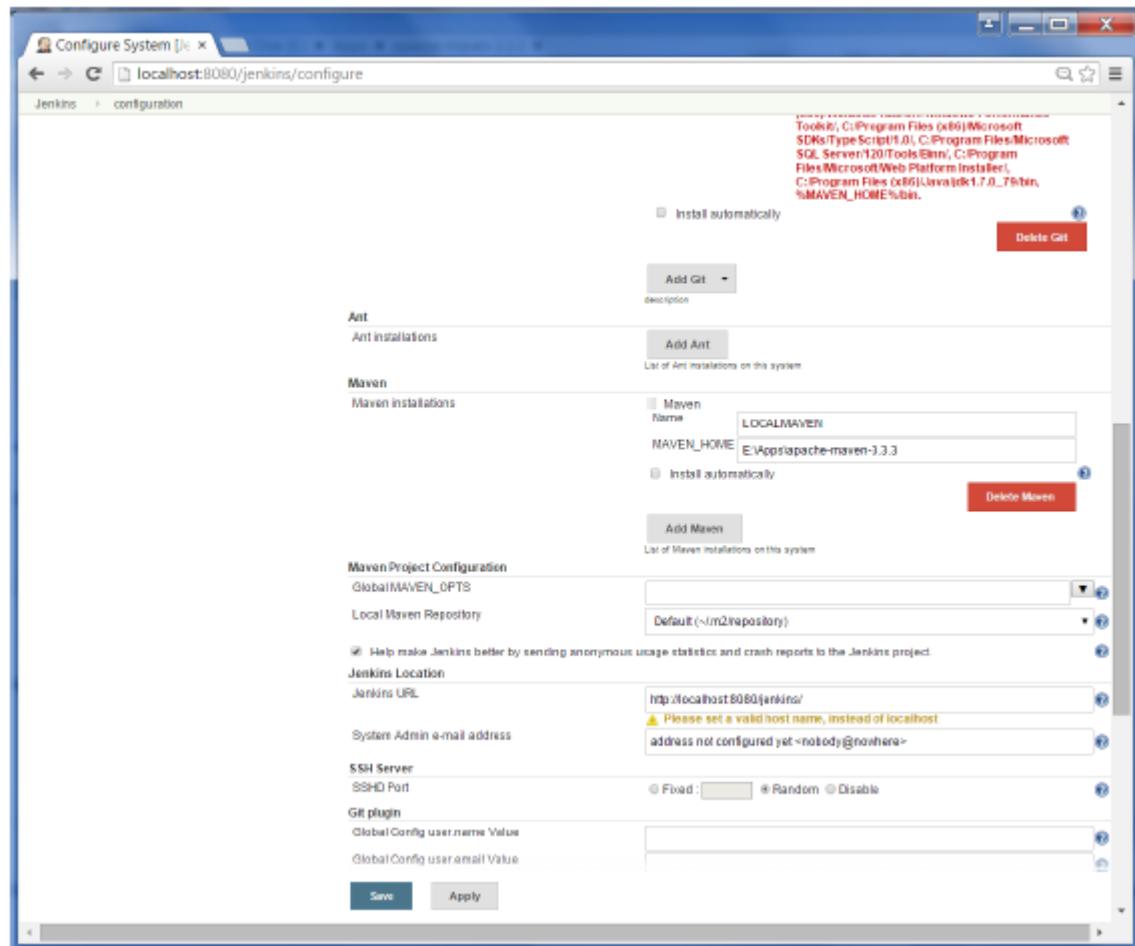
In the Configure system screen, scroll down till you see the Maven section and then click on the 'Add Maven' button.



Uncheck the 'Install automatically' option.

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

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



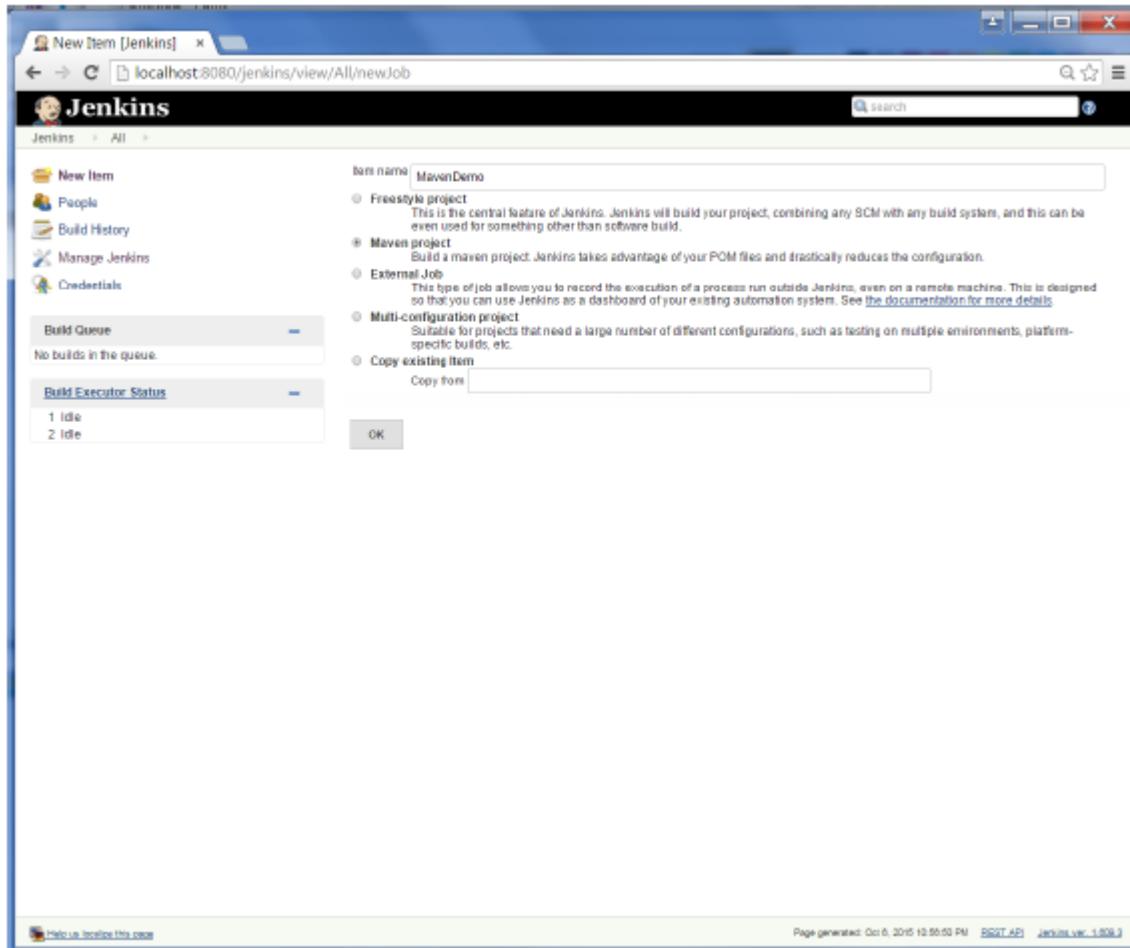
You can now create a job with the ‘Maven project’ option. In the Jenkins dashboard, click the New Item option.

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The main content area displays a table of jobs. There is one job listed:

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

Below the table, there is a legend: "RSS for all" (RSS icon), "RSS for failures" (RSS icon with a red dot), and "RSS for just latest builds" (RSS icon with a green dot).

On the left sidebar, under the "Jenkins" section, the following items are listed: New Item, People, Build History, Manage Jenkins, and Credentials. Under "Build Queue" and "Build Executor Status", both sections show "No builds in the queue." and "1 Idle" respectively.



Jenkins - Configuration

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

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

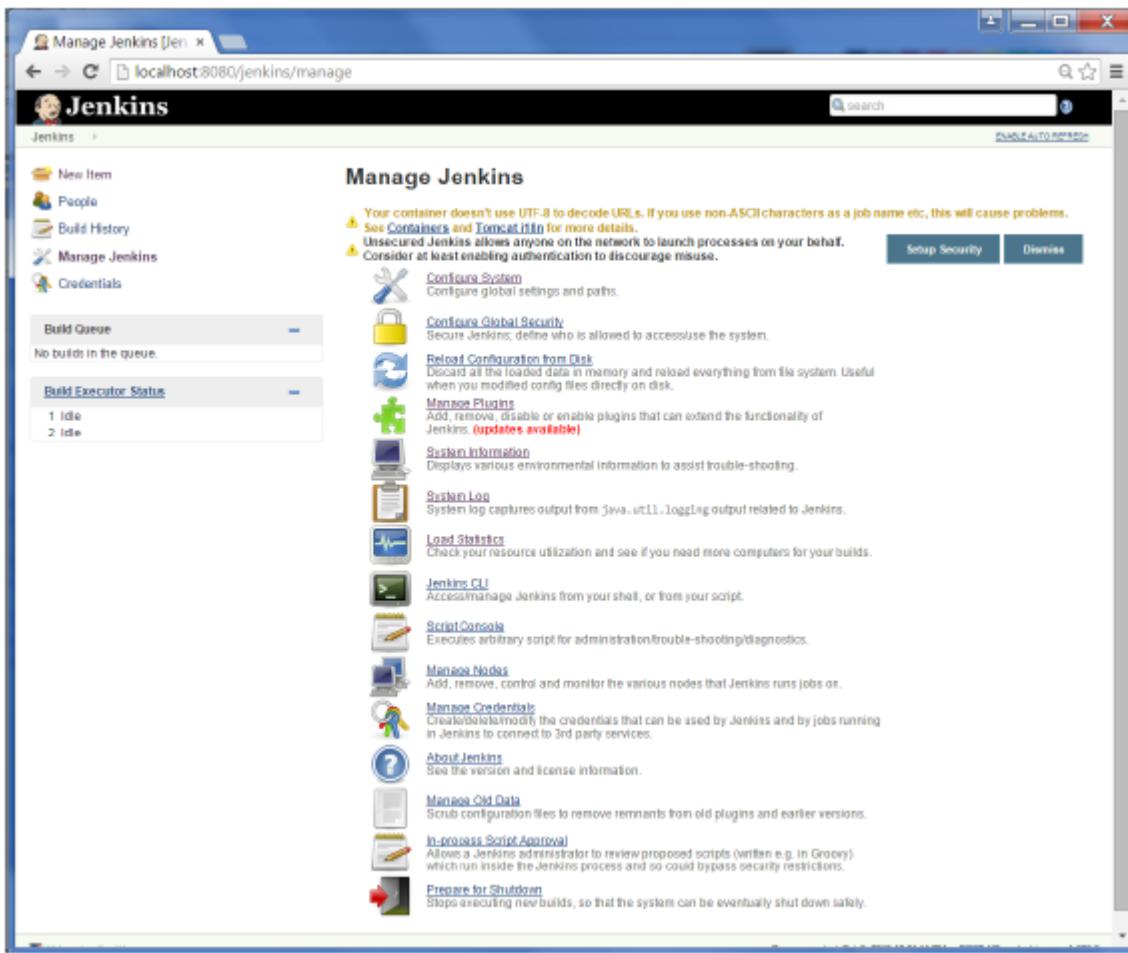
The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The left sidebar includes links for New item, People, Build History, Manage Jenkins, and Credentials. The main area displays a table for the 'Demo' job, which is currently Idle. A legend at the bottom right indicates RSS feeds for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

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

Build Queue: No builds in the queue.

Build Executor Status: 1 Idle, 2 Idle.

You will then be presented with the following screen –



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

Jenkins Home Directory

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

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

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

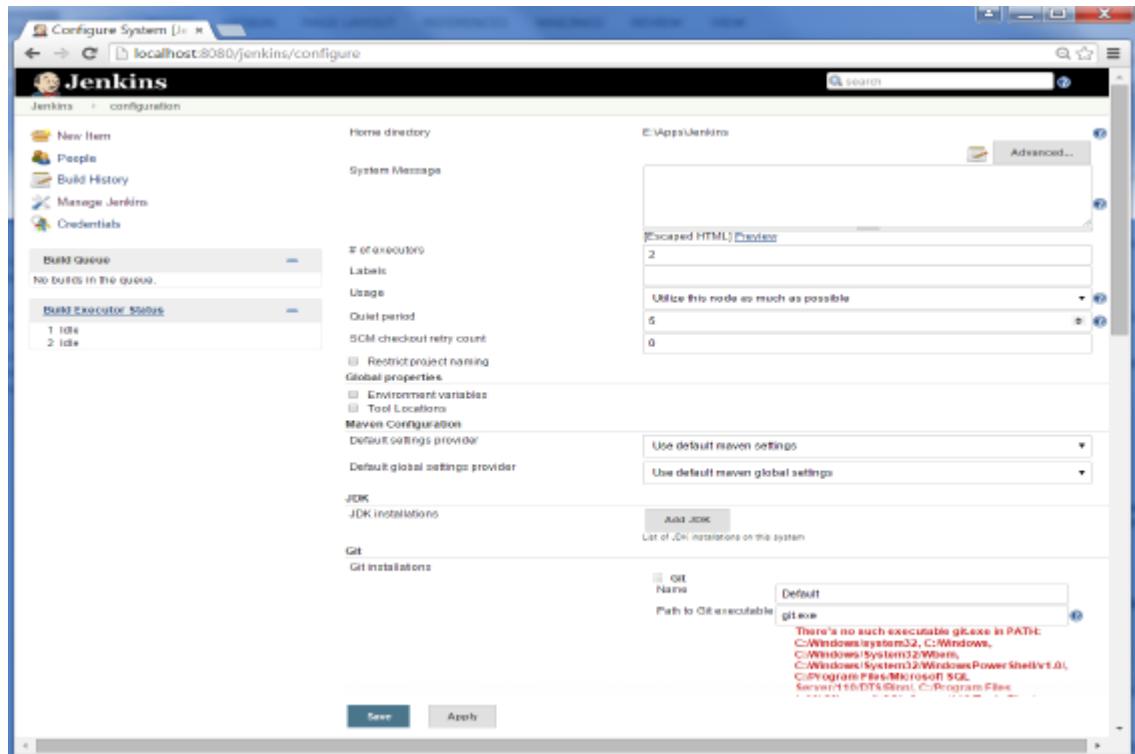
First create a new folder E:\Apps\Jenkins. Copy all the contents from the existing ~/.jenkins to this new directory.

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

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

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

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



of executors

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

Environment Variables

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

Jenkins URL

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

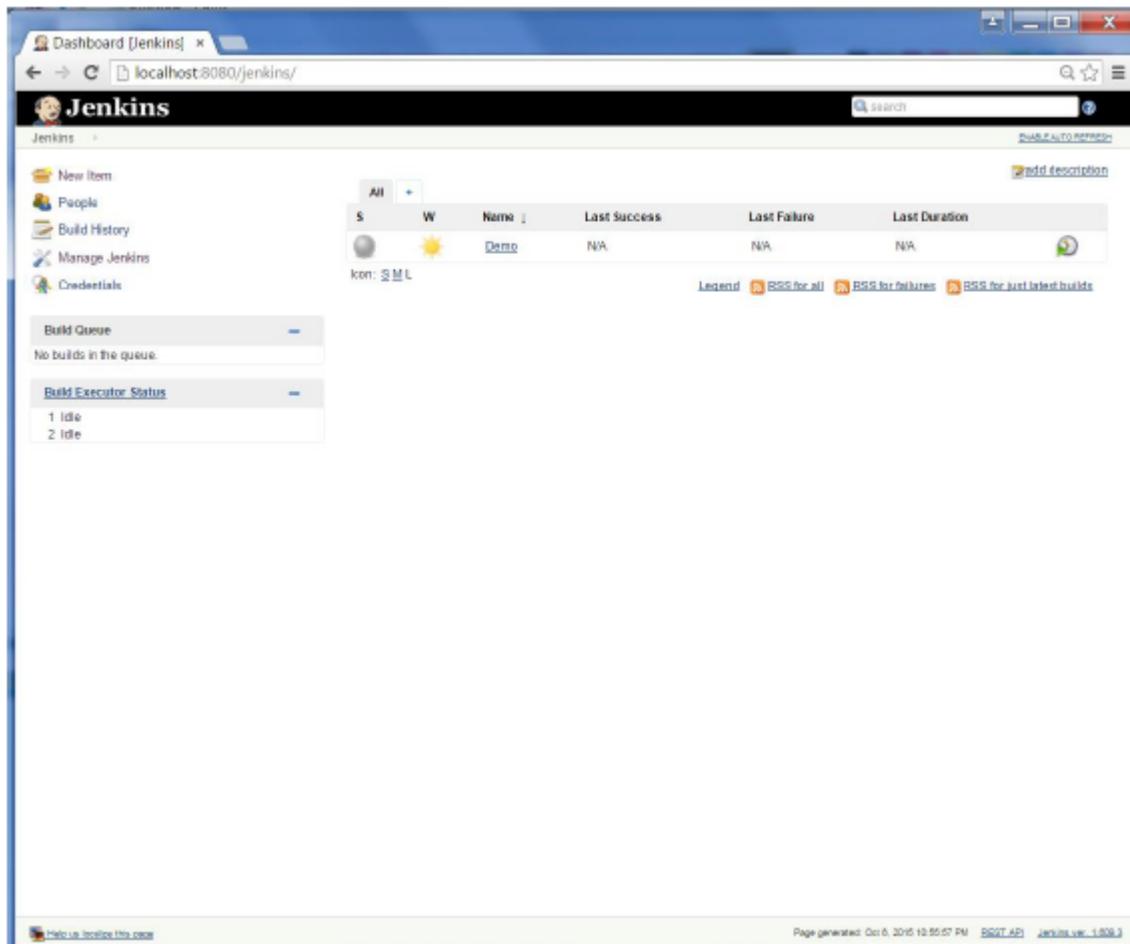
Email Notification

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

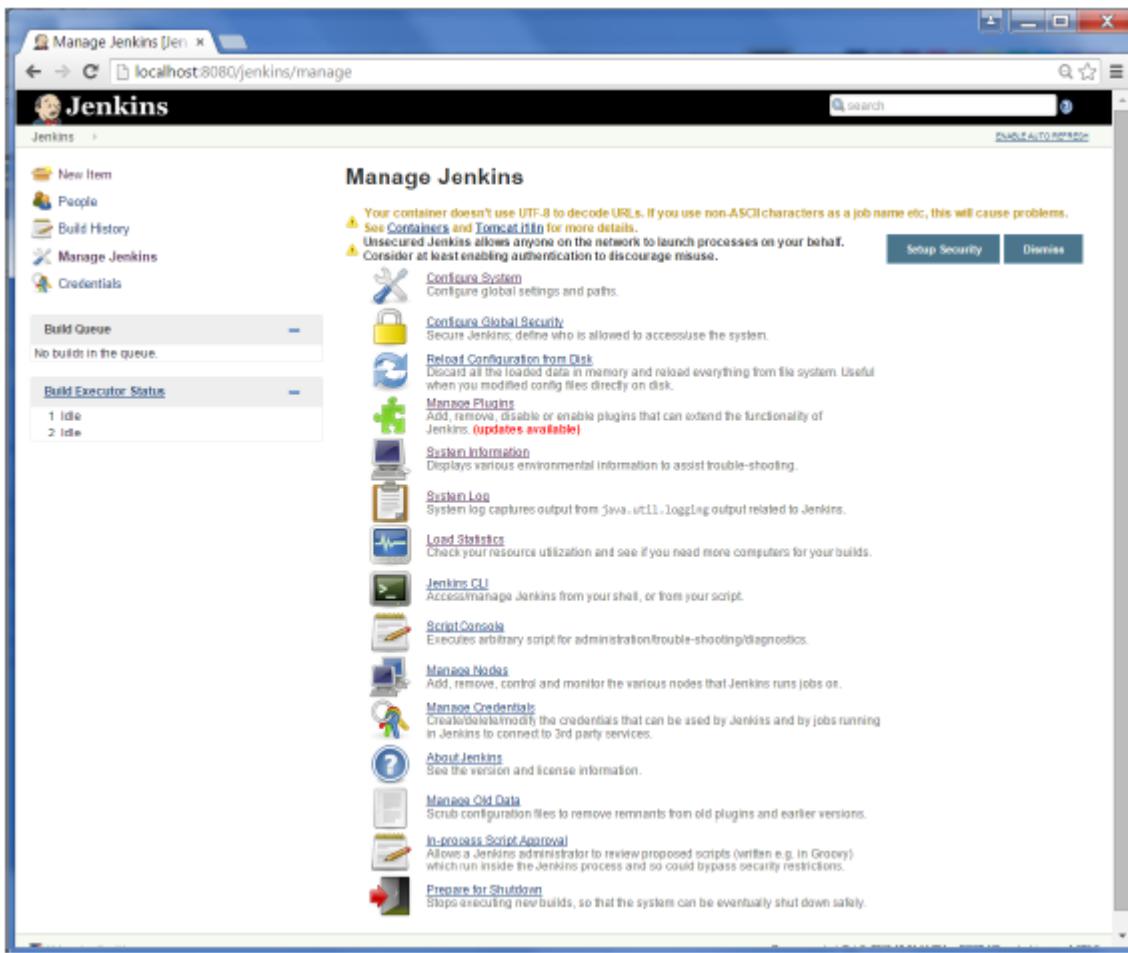
Jenkins - Management

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

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



You will then be presented with the following screen –



Some of the management options are as follows –

Configure System

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

Reload Configuration from Disk

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

Manage Plugin

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

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Jenk] < localhost:8080/jenkins/pluginManager/ Jenkins". The main content area has a header "Updates Available Installed Advanced". Below it is a table titled "Install" with columns "Name", "Version", and "Installed". The table lists various Jenkins plugins:

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

At the bottom, there are buttons "Download now and install after restart" and "Check now". A note says "Select All. None. This page lists updates to the plugins you currently use." The footer includes links "Help us decide this issue", "Page generated: Oct 6, 2015 11:08:25 PM", "Build API", and "Jenkins ver. 1.608.3".

System Information

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

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

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

Name	Value
awt.toolkit	sun.awt.windows.WToolkit
catalina.base	E:\Appstromcat7
catalina.home	E:\Appstromcat7
catalina.useNaming	true
common.loader	\$[catalina.base]\lib,\$[catalina.base]\lib*jar,\$[catalina.home]\lib,\$[catalina.home]\lib*jar
file.encoding	Cp1252
file.encoding.pkg	sun.ja
file.separator	\
java.awt.graphicsenv	sun.awt.Win32GraphicsEnvironment
java.awt.printerjob	sun.awt.windows.WPrinterJob
java.class.path	E:\Appstromcat7\bin\bootstrap.jar;E:\Appstromcat7\bin\lombok-all.jar
java.class.version	51.0
java.endorsed.dirs	E:\Appstromcat7\endorsed
java.ext.dirs	C:\Program Files (x86)\Java\jdk1.7.0_79\jre\lib\ext;C:\Windows\Sun\Java\lib\ext
java.home	C:\Program Files (x86)\Java\jdk1.7.0_79\jre
java.io.tmpdir	E:\Appstromcat7\temp
java.library.path	C:\Program Files (x86)\Java\jdk1.7.0_79\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files\Microsoft SQL Server\120\Tools\Binn\;C:\Program Files\Microsoft SQL Server\130\Tools\Binn\ManagementStudio\;C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\PrivateAssemblies\;C:\Program Files (x86)\Microsoft SQL Server\110\Tools\Binn\;C:\Program Files (x86)\Windows Kits\18\Windows Performance Toolkit\;C:\Program Files (x86)\Microsoft SDKs\TypeScript\1.0\;C:\Program Files\Microsoft SQL Server\120\Tools\Binn\;C:\Program Files\Microsoft\Web Platform Installer\;C:\Program Files (x86)\Java\jdk1.7.0_79\bin;%MAVEN_HOME%\bin;
java.naming.factory.initial	org.apache.naming.java.javaURLContextFactory
java.naming.factory.url.pkgs	org.apache.naming
java.runtime.name	Java(TM) SE Runtime Environment
java.runtime.version	1.7_0_79-b15
java.specification.name	Java Platform API Specification
java.specification.vendor	Oracle Corporation
java.specification.version	1.7
java.util.logging.config.file	E:\Appstromcat7\conf\logging.properties
java.util.logging.manager	org.apache.juli.ClassLoaderLogManager
java.vendor	Oracle Corporation
java.vendor.url	http://java.oracle.com/
java.vendor.url_bug	http://bugreport.sun.com/bugreport/
java.version	1.7_0_79
java.vm.info	mixed mode, sharing

System Log

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

Load Statistics

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

Script Console

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

Manage nodes

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

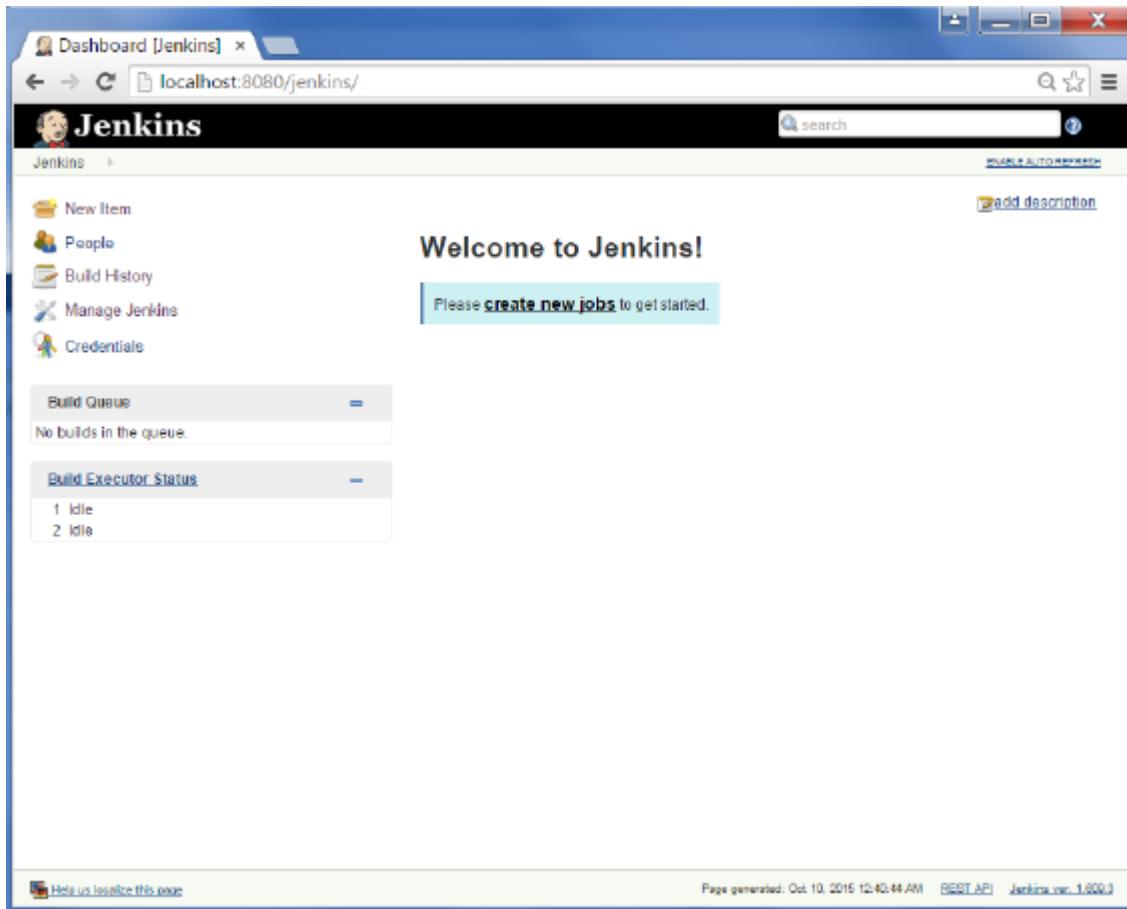
Prepare for Shutdown

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

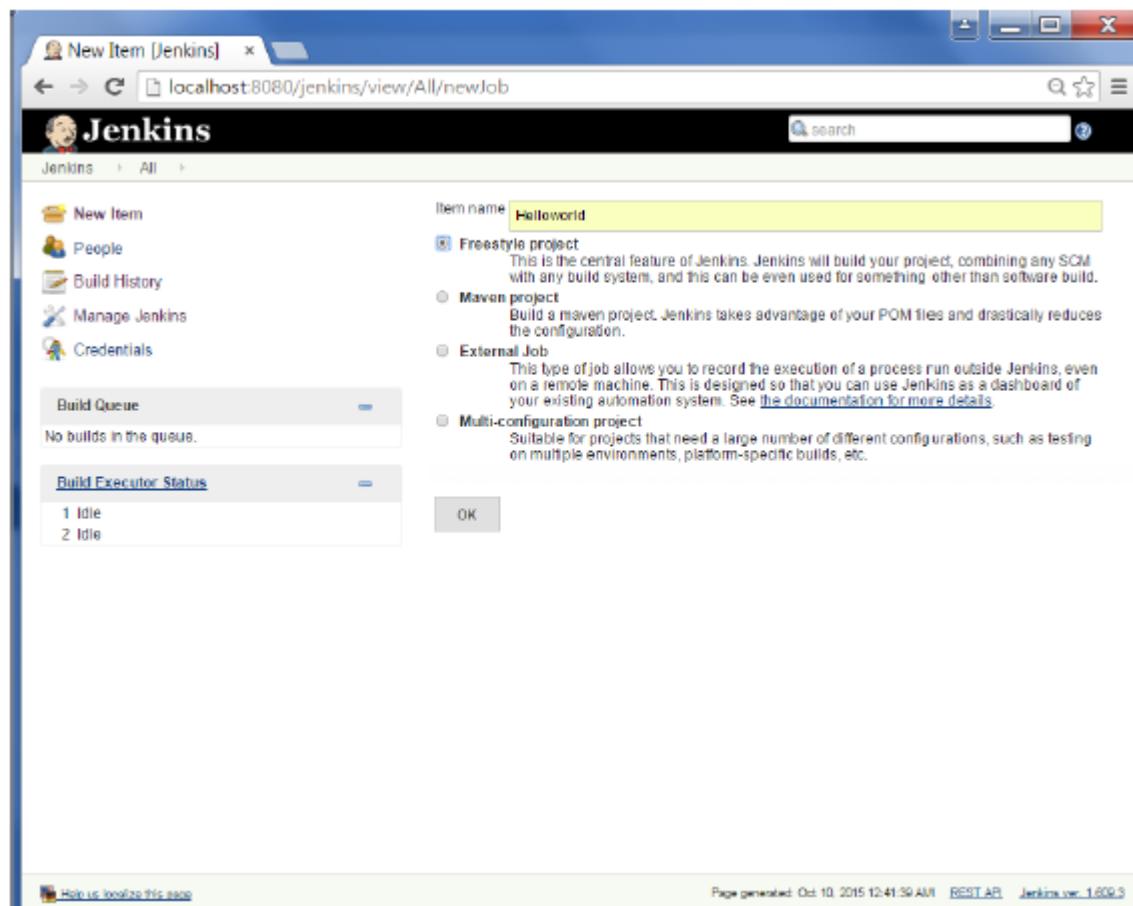
Jenkins - Setup Build Jobs

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

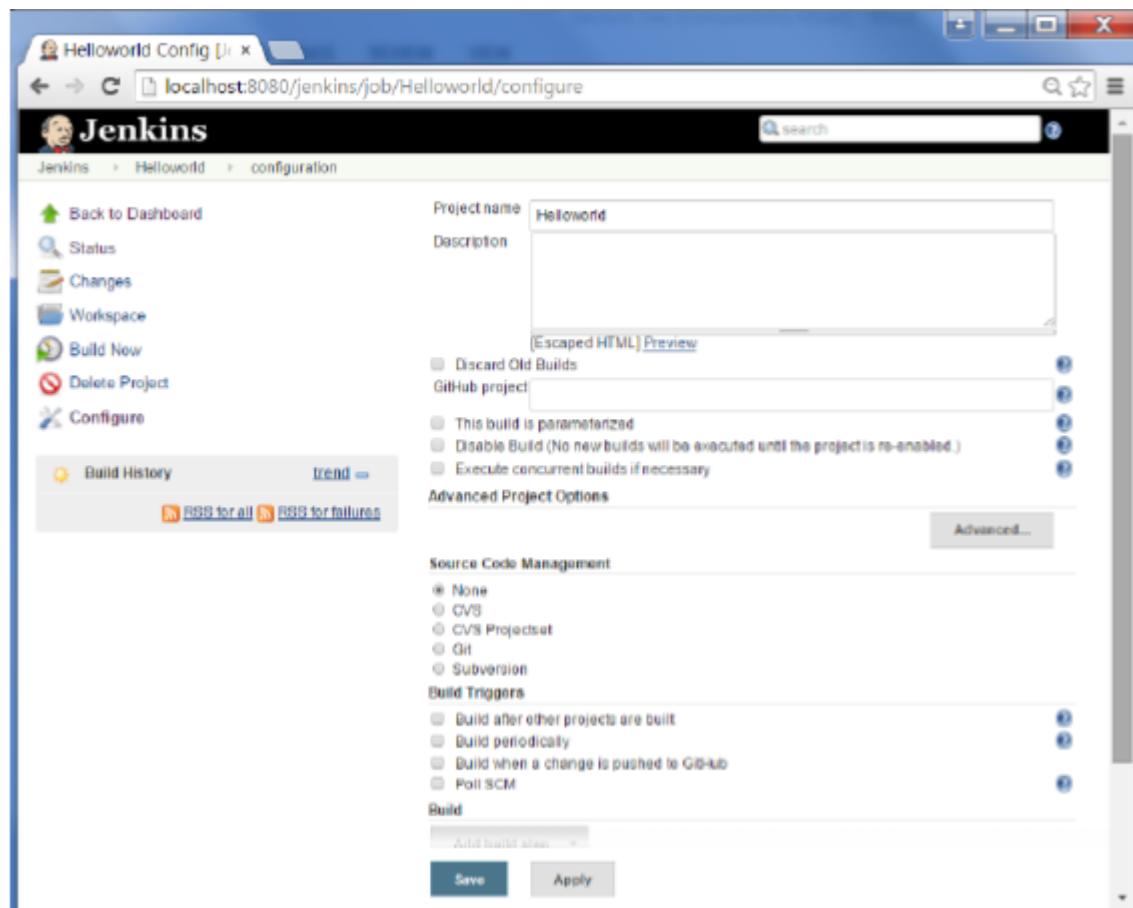
Step 1 – Go to the Jenkins dashboard and Click on New Item



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

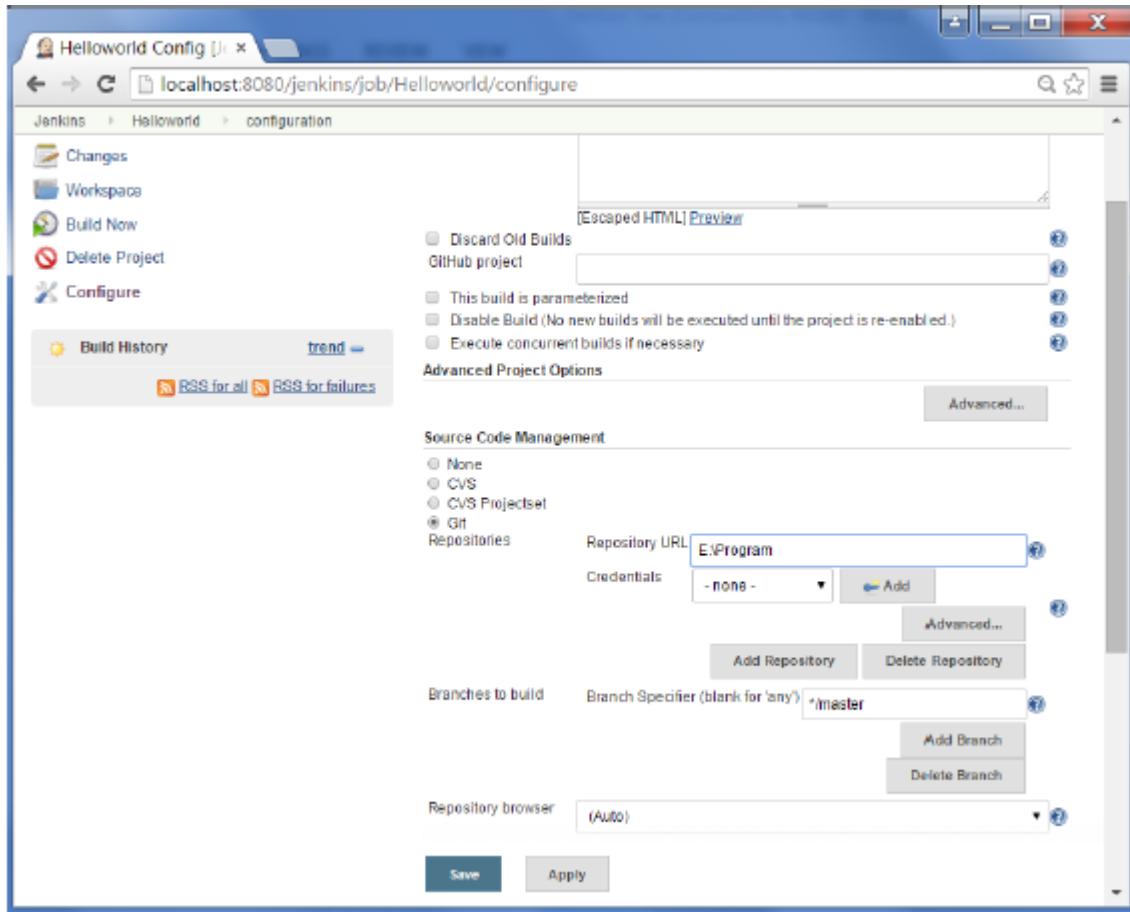


Step 3 – The following screen will come up in which you can specify the details of the job.

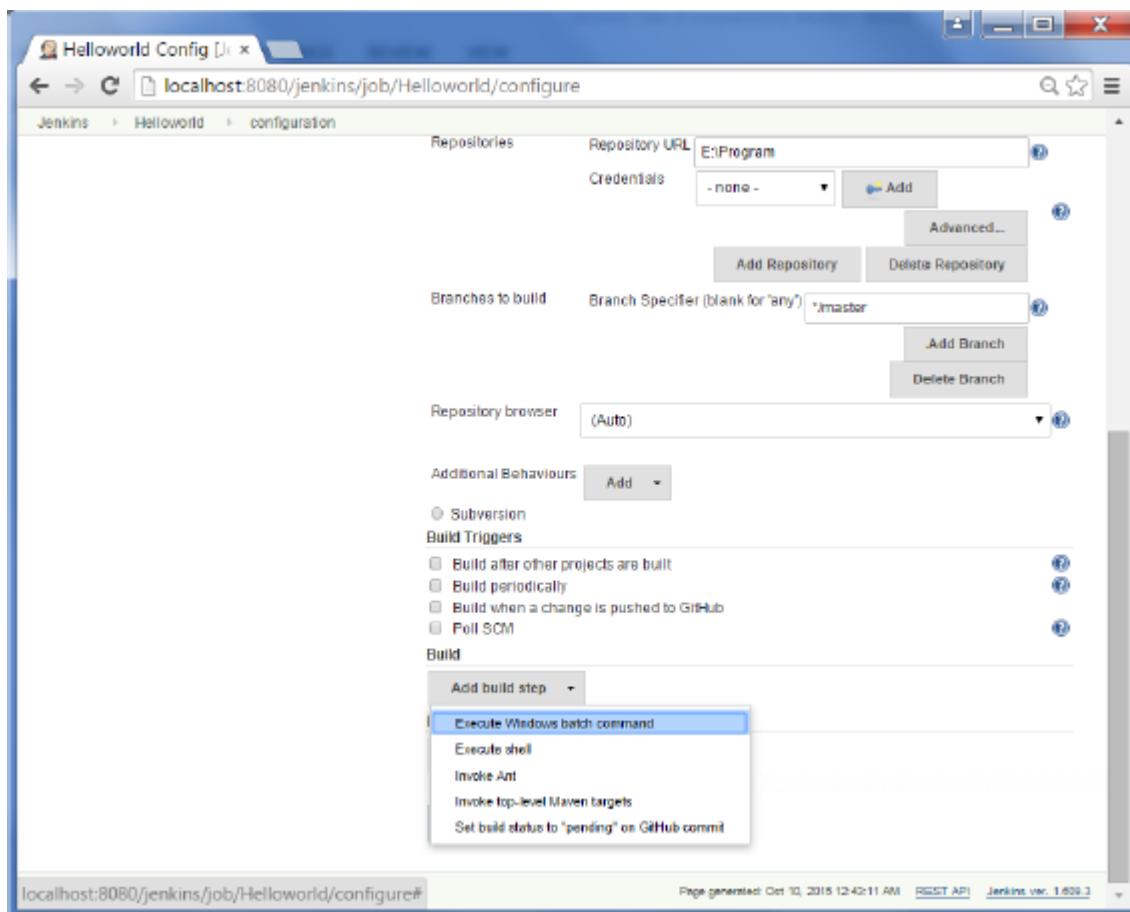


Step 4 – We need to specify the location of files which need to be built. In this example, we will assume that a local git repository(E:\Program) has been setup which contains a ‘HelloWorld.java’ file. Hence scroll down and click on the Git option and enter the URL of the local git repository.

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



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



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

```
Javac HelloWorld.java
```

```
Java HelloWorld
```

The screenshot shows the Jenkins configuration interface for the 'Helloworld' job. At the top, there's a header with the job name and a 'Delete Branch' button. Below it, a 'Repository browser' dropdown is set to '(Auto)'. Under 'Additional Behaviours', there's a 'Build Triggers' section with options like 'Build after other projects are built', 'Build periodically', 'Build when a change is pushed to GitHub', and 'Poll SCM'. The 'Build' section contains a 'Execute Windows batch command' step with the command `javac HelloWorld.java
java HelloWorld`. There are 'Add build step' and 'Post-build Actions' buttons. At the bottom are 'Save' and 'Apply' buttons.

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

The screenshot shows the Jenkins dashboard for the 'Helloworld' project. The left sidebar includes links for Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main area displays the project name 'Project Helloworld' with a 'Workspace' link and a 'Recent Changes' link. A 'Permalinks' section at the bottom provides links for RSS feeds. On the right, there are 'Add description' and 'Disable Project' buttons.

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

The screenshot shows the Jenkins interface for the 'Project Helloworld'. 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 Helloworld' and displays a 'Build History' table with one entry: '#1 Oct 10, 2015 12:52 AM'. To the right of the table are 'Workspace' and 'Recent Changes' links. Below the table is a 'Permalinks' section. At the bottom of the page, there are links for Help, localize this page, Page generated: Oct 10, 2015 12:51:53 AM, REST API, and Jenkins ver. 1.603.3.

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

The screenshot shows the Jenkins interface for the 'Helloworld' project. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', and 'Configure'. To the right, there's a search bar and a 'ENABLE AUTO-SAVE' button. The main title is 'Project Helloworld'. On the left, there's a sidebar with a 'Build History' section showing one build (Oct 10, 2015 12:52 AM) with a status of '21'. Below the sidebar are 'Recent Changes' and 'Permalinks' sections. On the right, there are links for 'Workspace' and 'Recent Changes', along with buttons for 'add description' and 'Disable Project'. At the bottom, there's a footer with links for 'Help us localize this page', 'Page generated: Oct 10, 2015 12:51:53 AM', 'REST API', and 'Jenkins ver. 1.609.3'.

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

The screenshot shows the Jenkins interface for the first build of the 'Helloworld' project, labeled '#1'. The title is 'Build #1 (Oct 10, 2015 12:52:50 AM)'. It shows the build started 4 min 40 sec ago and took 4.7 sec. The sidebar on the left includes links for 'Back to Project', 'Status', 'Changes', 'Console Output' (which is highlighted in blue), 'Edit Build Information', 'Delete Build', 'Git Build Data', and 'No Tags'. The main content area displays build logs: 'No changes.', 'Started by anonymous user', and 'Revision: 4219a82ffadd06fb5c3a9dfe40e731a907f5c8f' with a note '• refs/remotes/origin/master'. The footer at the bottom is identical to the previous screenshot.



The screenshot shows the Jenkins interface for a build named "Helloworld #12 Cons". The left sidebar lists build-related actions: Back to Project, Status, Changes, Console Output (selected), View as plain text, Edit Build Information, Delete Build, Git Build Data, No Tags, and Previous Build. The main content area is titled "Console Output" and displays the command-line log of the build process. The log shows the execution of Git commands to fetch changes from a remote repository, followed by Java compilation and execution of the "HelloWorld.java" file, resulting in the output "Hello World". The build status is marked as "SUCCESS".

```

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

E:\Jenkins\jobs\Helloworld\workspace>javac HelloWorld.java
E:\Jenkins\jobs\Helloworld\workspace>java HelloWorld
Hello World
E:\Jenkins\jobs\Helloworld\workspace>exit 0
Finished: SUCCESS

```

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

Jenkins - Unit Testing

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

xUnit Plugin - Jenkins

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

Dashboard > Jenkins > Plugins > xUnit Plugin

Browse Search

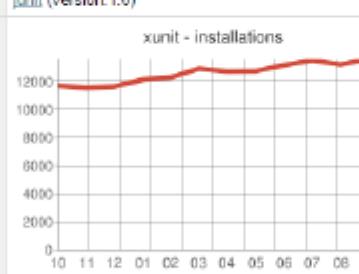
xUnit Plugin

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

Jenkins

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

Plugin Information

Plugin ID	xunit	Changes	In Latest Release Since Latest Release
Latest Release	1.98 (archives)	GitHub	
Latest Release Date	Oct 09, 2015	Open Issues	
Required Core Dependencies	JUnit (version: 1.6)	Pull Requests	
Usage		Installations	2014-Oct 11692 2014-Nov 11557 2014-Dec 11631 2015-Jan 12105 2015-Feb 12262 2015-Mar 12891 2015-Apr 12894 2015-May 12716 2015-Jun 13143 2015-Jul 13470 2015-Aug 13192 2015-Sep 13663

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

CppUnit output

xUnit Plugin - Jenkins

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

Features

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

Supported tools

Embedded tools

- * JUnit itself
- * AllUnit
- * 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 \(Ctest Plugin\)](#)
- * [JSUnit \(JSUnit Plugin\)](#)
- * [JBehave](#)
- * [TestComplete \(TestComplete xUnit Plugin\)](#)

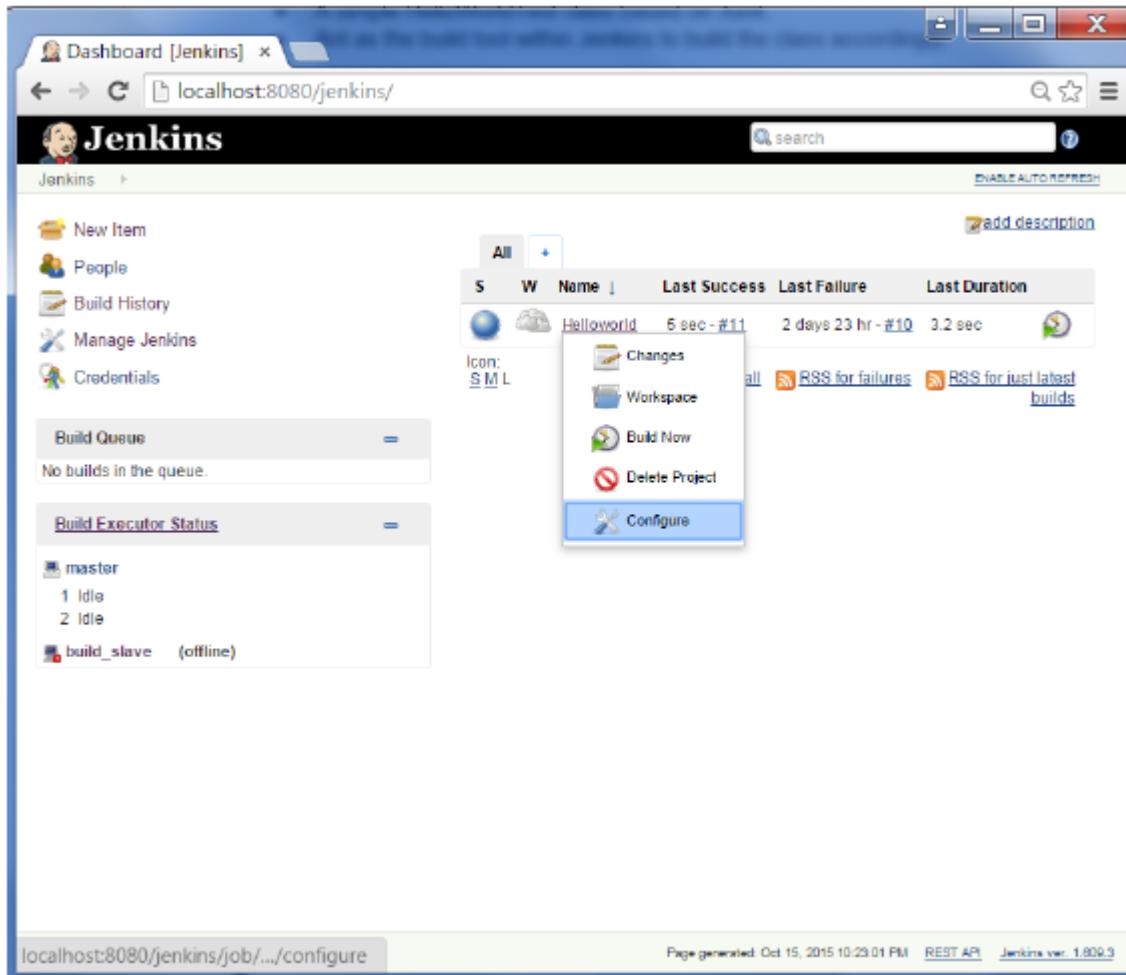
External contributions

Example of a Junit Test in Jenkins

The following example will consider

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

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



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

The screenshot shows the Jenkins configuration page for a job named "Helloworld". The URL is <localhost:8080/jenkins/job/Helloworld/configure>. The page has a header with "Helloworld Config" and navigation links for Jenkins, Helloworld, and configuration. A "Branch" dropdown is set to "(Auto)". Under "Additional Behaviours", there is a radio button for "Subversion" and a "Build Triggers" section with four options: "Build after other projects are built", "Build periodically", "Build when a change is pushed to GitHub", and "Poll SCM". The "Build" section contains an "Execute Windows batch command" step with the command `javac HelloWorld.java
java HelloWorld`. A tooltip for environment variables is visible. A "Delete" button is present. A dropdown menu titled "Add build step" is open, showing options: "Execute Windows batch command", "Execute shell", "Invoke Ant" (which is highlighted), "Invoke top-level Maven targets", and "Set build status to 'pending' on GitHub commit".

Step 3 – Click on the Advanced button.

The screenshot shows the Jenkins configuration page for a job named "Helloworld". The URL is localhost:8080/jenkins/job/Helloworld/configure. The page displays various build triggers and build steps. Under "Build Triggers", "Subversion" is selected. Under "Build", there are two steps: "Execute Windows batch command" with the command `javac HelloWorld.java
java HelloWorld`, and "Invoke Ant" with "Ant Version" set to "Default" and "Targets" empty. At the bottom are "Save" and "Apply" buttons.

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

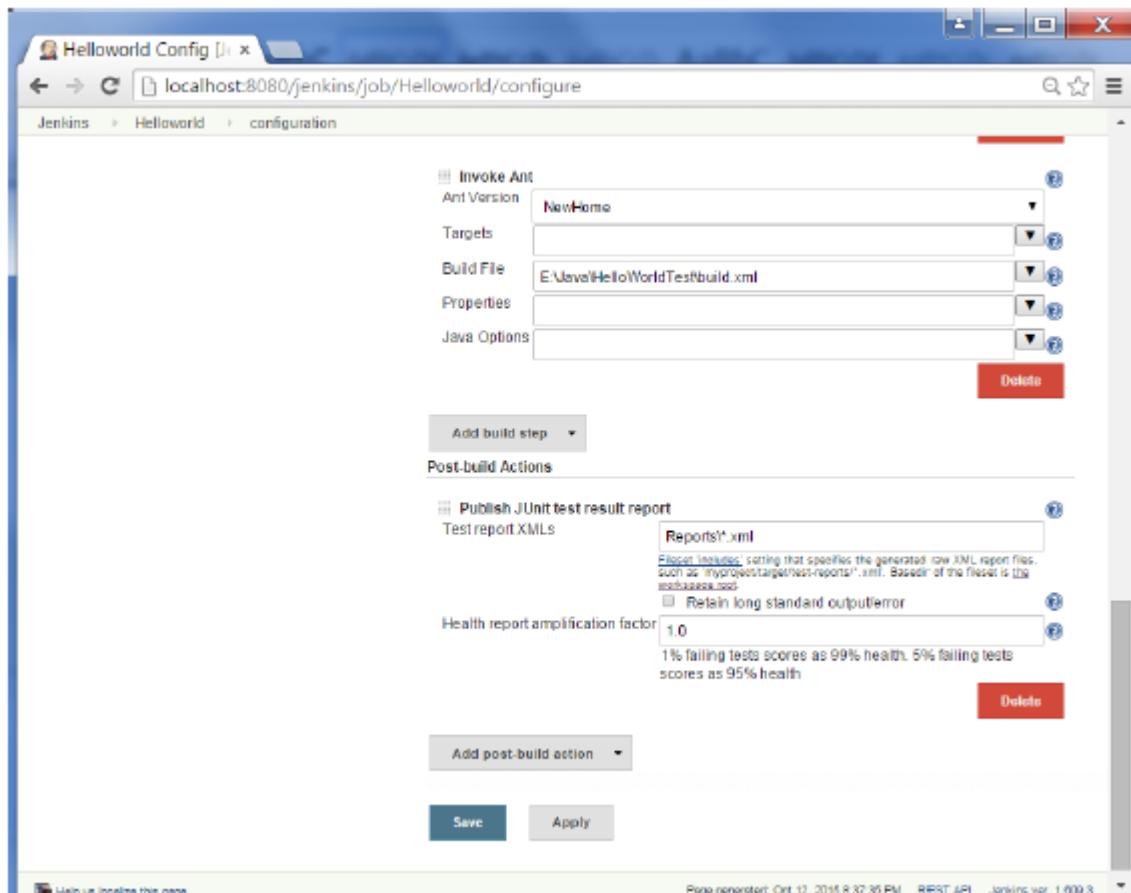
The screenshot shows the Jenkins configuration page for the 'Helloworld' job. Under the 'Build' section, there is a 'Execute Windows batch command' step with the command 'javac HelloWorld.java' and 'java HelloWorld'. Below it is an 'Invoke Ant' step with 'Ant Version' set to 'NewHome', 'Targets' set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

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

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

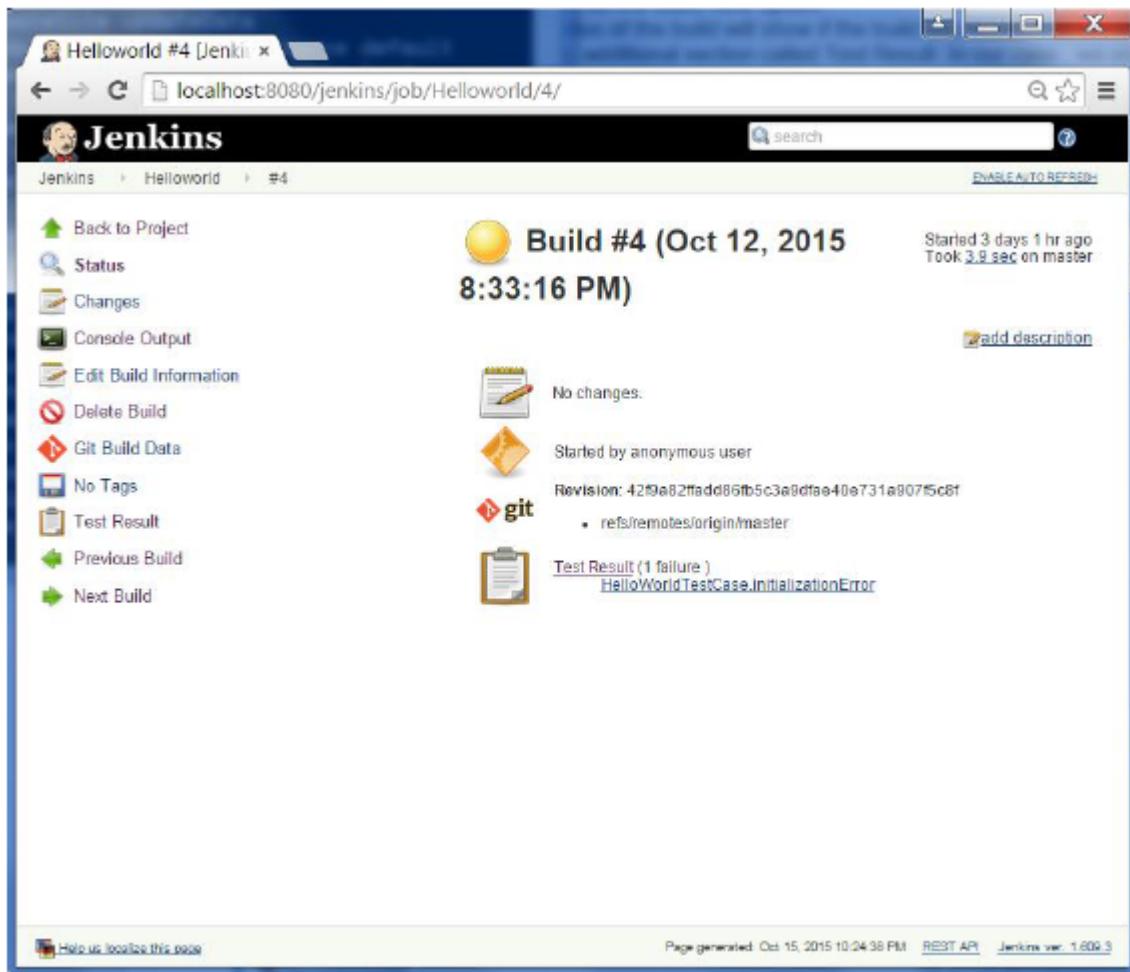
Step 6 – In the Test reports XML's, enter the location as shown below. Ensure that Reports is a folder which is created in the HelloWorld project workspace. The “*.xml” basically tells Jenkins to pick up the result xml files which are produced by the running of the Junit test cases. These xml files which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

Once done, click the Save option at the end.



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

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



The screenshot shows the Jenkins interface for a build named "Helloworld #4". The main title is "Build #4 (Oct 12, 2015) 8:33:16 PM". It indicates the build was started 3 days 1 hr ago and took 3.9 sec on master. A "No changes." message is shown. The build was started by an anonymous user from a git commit (revision: 42f9a82ffadd86fb5c3e0dfa040e731a807f5c8f, refs/remotes/origin/master). A test result section shows one failure: "HelloWorldTestCase.InitializationError". Navigation links on the left include Back to Project, Status, Changes, Console Output, Edit Build Information, Delete Build, Git Build Data, No Tags, Test Result, Previous Build, and Next Build.

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

The screenshot shows the Jenkins Test Result page for a build named 'Helloworld #4'. The main title is 'Test Result' with a status of '1 failures'. A red bar indicates '1 tests' took '10 ms'. Below this, 'All Failed Tests' is listed with one entry: 'HelloWorldTestCase InitializationError' (Duration: 10 ms, Age: 1). Under 'All Tests', a table shows the test results for the package 'root': Duration 10 ms, Fail 1, Skip 0, Pass 0, Total 1. At the bottom, there are links for 'Help us localize this page', 'Page generated: Oct 12, 2015 8:45:49 PM', 'REST API', and 'Jenkins ver. 1.606.3'.

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.

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 [Tomcat11](#) for more details.

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

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

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

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

Update Center [Jen]

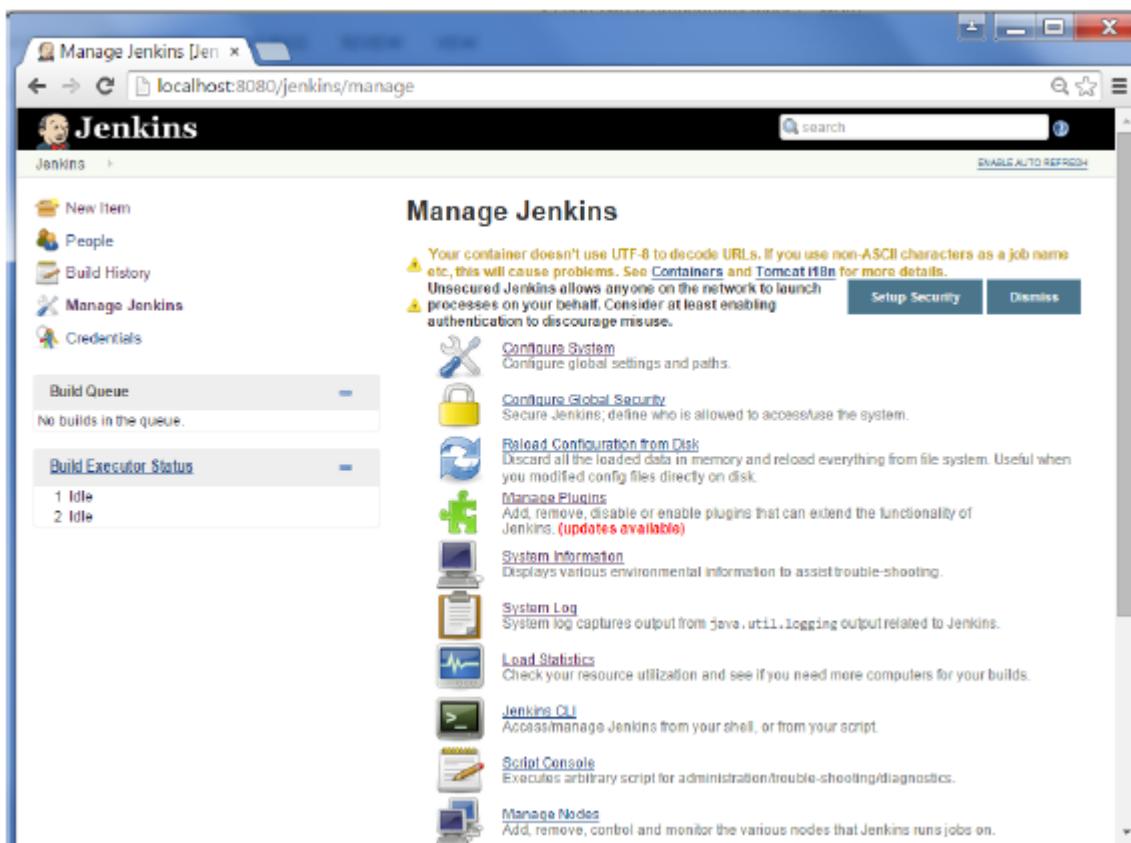
[localhost:8080/jenkins/pluginManager/available](#)

Available

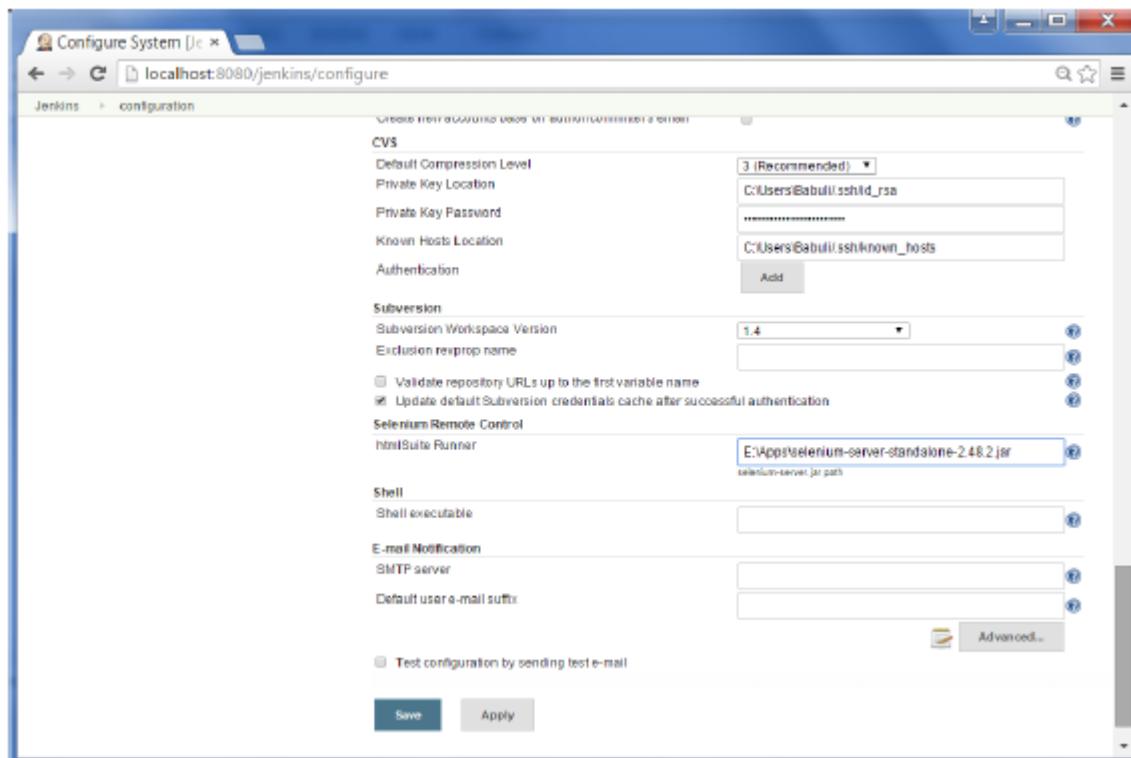
Name	Version
Selenium Auto Exec Server(AES) plugin	0.5
Hudson SeleniumHQ plugin	0.4
Selenium HTML report	0.94
TestingBot plugin	1.11
TestLink Plugin	3.10
Nemvana Plugin for Jenkins	1.02.06
Source OnDemand plugin	1.141
Selenium Builder plugin	1.14
SeleniumRC plugin	1.0

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

Step 3 – Go to Configure system.



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



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

Click on the download for the Selenium standalone server.

The screenshot shows the SeleniumHQ website at www.seleniumhq.org/download/. The 'Download' tab is selected. The 'Selenium Standalone Server' section is highlighted, stating: '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.' Below it, a link to 'Download version 2.48.2' is provided, along with instructions to 'use the Selenium Server in a Grid configuration see the wiki page.'

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

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. The 'HelloWorld' project is listed in the main table. A context menu is open over the project row, with the 'Configure' option highlighted. The URL at the bottom of the browser window is localhost:8080/jenkins/job/HelloWorld/configure.

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

The screenshot shows the Jenkins configuration interface for the 'Helloworld' job. In the 'Build' section, there is one step: 'Execute Windows batch command' with the command 'javac HelloWorld.java' and 'java HelloWorld'. Below this, a dropdown menu is open, showing various build step options, with 'SeleniumHQ htmlSuite Run' selected.

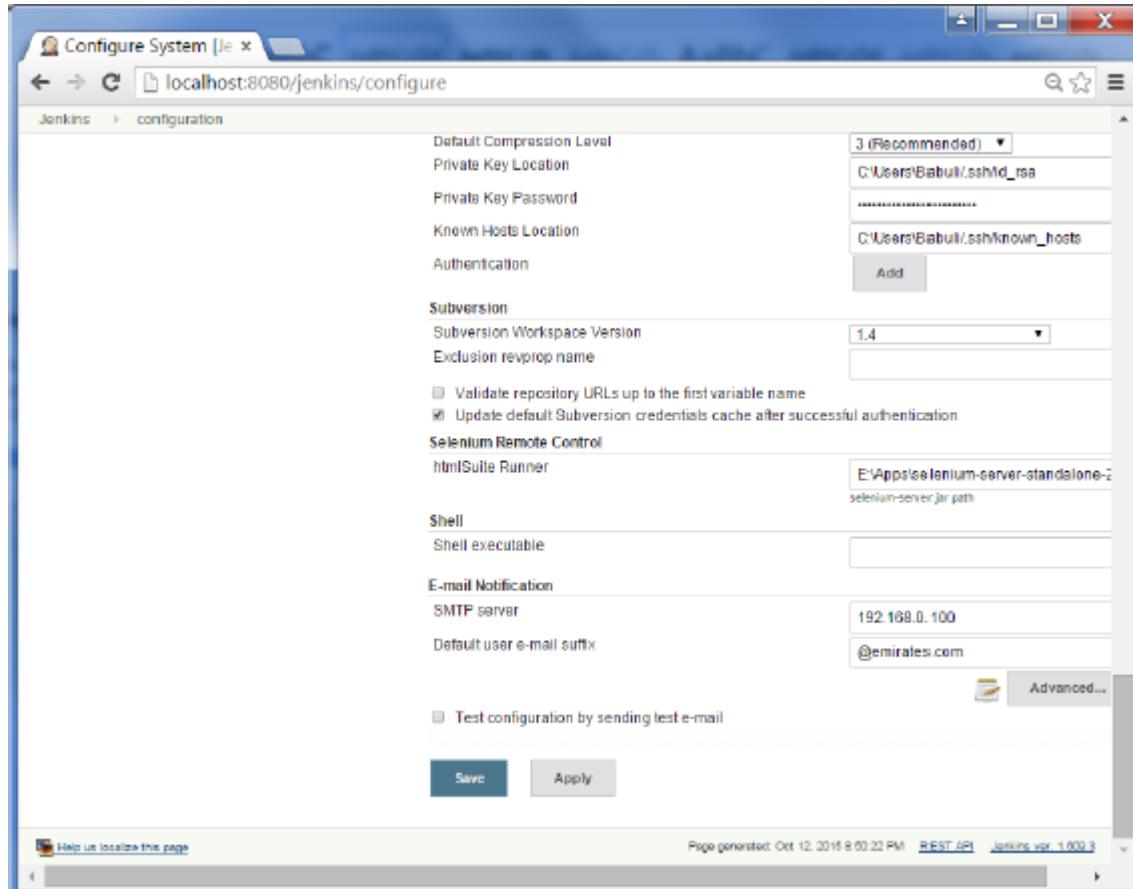
Step 7 – Add the necessary details for the selenium test. Here the suiteFile is the TestSuite generated by using the Selenium IDE. Click on Save and execute a build. Now the post build will launch the selenium driver, and execute the html test.

The screenshot shows the Jenkins configuration interface for the 'Helloworld' job. In the 'Build' section, there is one step: 'Execute Windows batch command' with the command 'javac HelloWorld.java' and 'java HelloWorld'. Below it, there is another step: 'SeleniumHQ htmlSuite Run' with the following configuration: browser set to 'firefox', startURL set to 'https://www.google.ae', suiteFile set to 'E:\Apps\NewSample.html', resultFile set to 'E:\Jenkins\jobs\Helloworld\workspace\Reports\Results.html', and other fields empty. At the bottom, there are 'Save' and 'Apply' buttons.

Jenkins - Notification

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

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



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

The screenshot shows the Jenkins job configuration page for 'Helloworld'. Under 'Build Steps', there is a 'SeleniumHQ html Suite Run' step with the browser set to 'ieExplore', startURL to 'https://www.google.ae', suiteFile to 'E:\Appa\Selenium\Sample.html', resultFile to 'Result.html', and other parameters. Below this is a 'Post-build Actions' section containing an 'E-mail Notification' step with recipient 'S112233@domain.com' and options for sending e-mail for unstable builds or broken builds. Buttons for 'Save' and 'Apply' are at the bottom.

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

The screenshot shows the Jenkins job configuration page for 'Helloworld'. On the left sidebar, 'Configure' is selected. In the main area, under 'Job Notifications', there is a 'Notification Endpoints' section. A new endpoint is being configured with the following details:

- Format:** JSON
- Protocol:** HTTP
- Event:** All Events
- URL:** http://tikalmem301.fxe.4567h
- Timeout:** 30000
- Log:** 0

At the bottom of the configuration, there are checkboxes for 'This build is parameterized', 'Disable Build', and 'Execute concurrent builds if necessary', along with 'Advanced...' and 'Save' buttons.

Here are the details of each option –

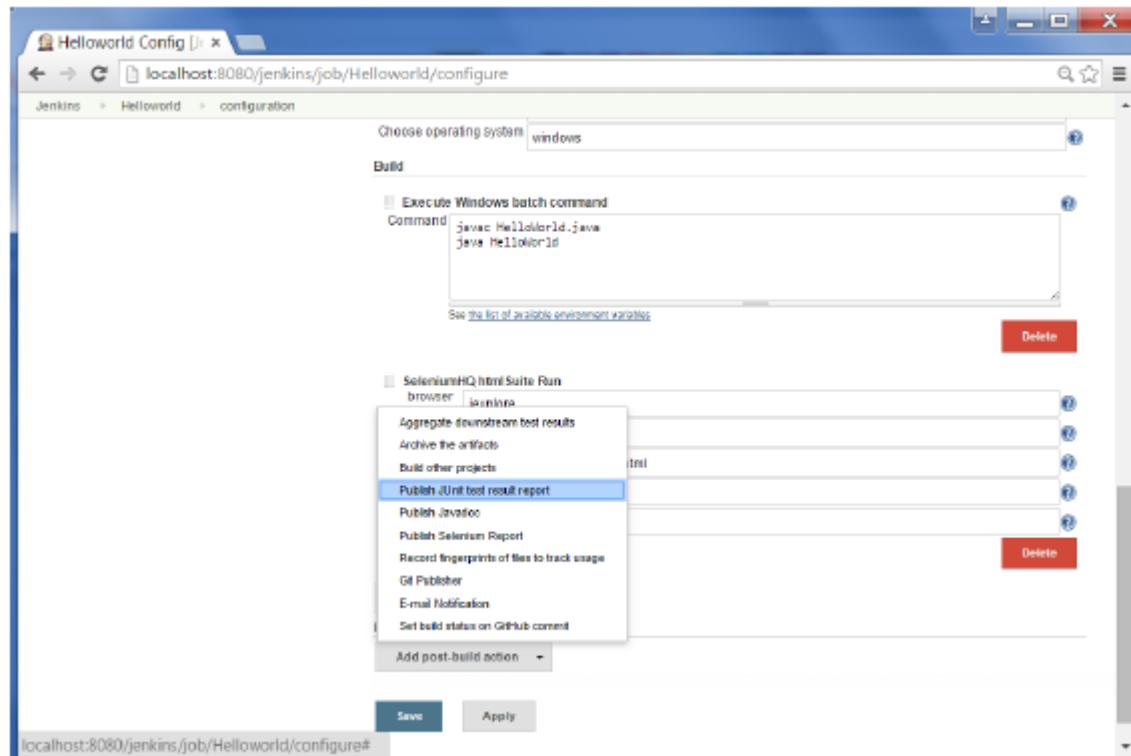
- **"Format"** – This is the notification payload format which can either be JSON or XML.
- **"Protocol"** – protocol to use for sending notification messages, HTTP, TCP or UDP.

- "**Event**" – The job events that trigger notifications: Job Started, Job Completed, Job Finalized or All Events (the default option).
- "**URL**" – URL to send notifications to. It takes the form of "http://host" for HTTP protocol, and "host:port" for TCP and UDP protocols.
- "**Timeout**" – Timeout in milliseconds for sending notification request, 30 seconds by default.

Jenkins - Reporting

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

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



Jenkins - Code Analysis

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

The screenshot shows the Jenkins Static Code Analysis Plugins page. On the left, there's a sidebar with links like Home, Mailing lists, Source code, Bugtracker, Security, Advisories, Events, Donation, Commercial Support, and Wiki Site Map. Under 'Documents', it lists Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, Servlet Container, and Notes. The main content area has a title 'Static Code Analysis Plug-ins' with a sub-section for 'analysis-core'. It shows the plugin ID as 'analysis-core', the latest release as '1.74 (archives)' from 'Sep 07, 2015', and required core dependencies including 'ant', 'token-macro', 'maven-plugin', 'maven-project', and 'dashboard-view'. It also shows GitHub links for source code, issue tracking, and pull requests, and credits 'Ulli Hafner (id: drull)'. A 'Usage' section contains a line graph titled 'analysis-core - installations' showing the number of installations per month from October 2014 to September 2015. The graph shows a steady increase from approximately 25,000 to over 30,000 installations.

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.

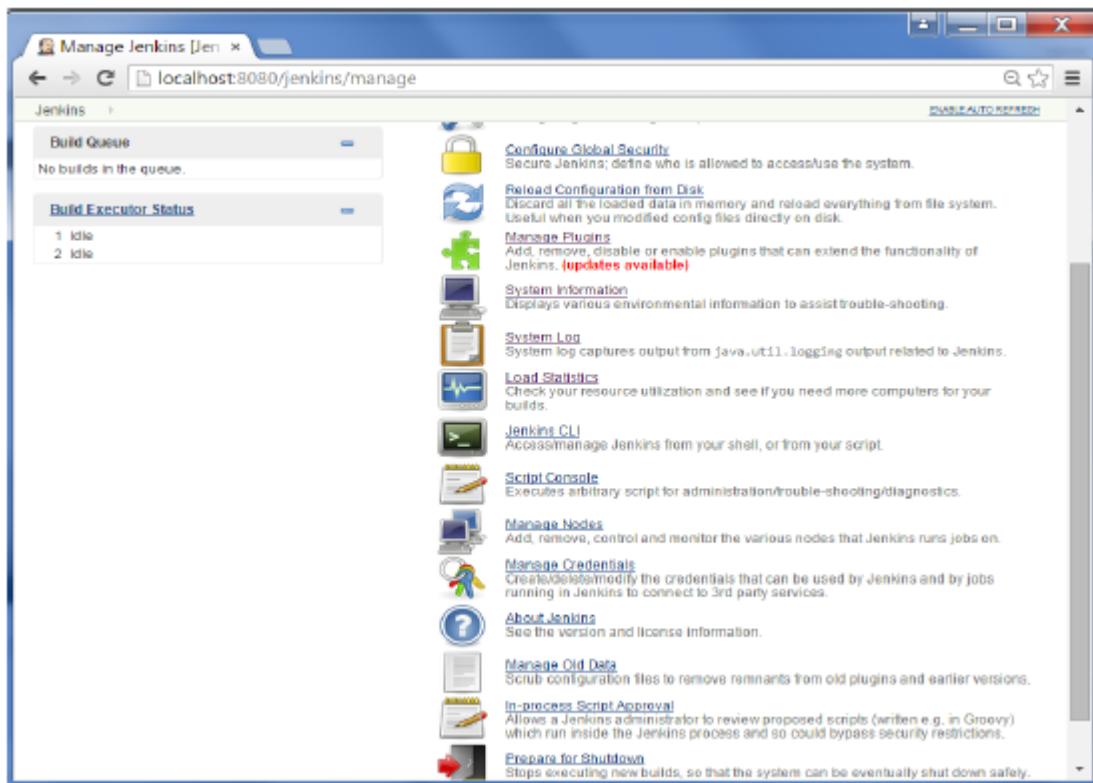
Sometimes you might also need several different environments to test your builds. In this case using a slave to represent each of your required environments is almost a must.

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

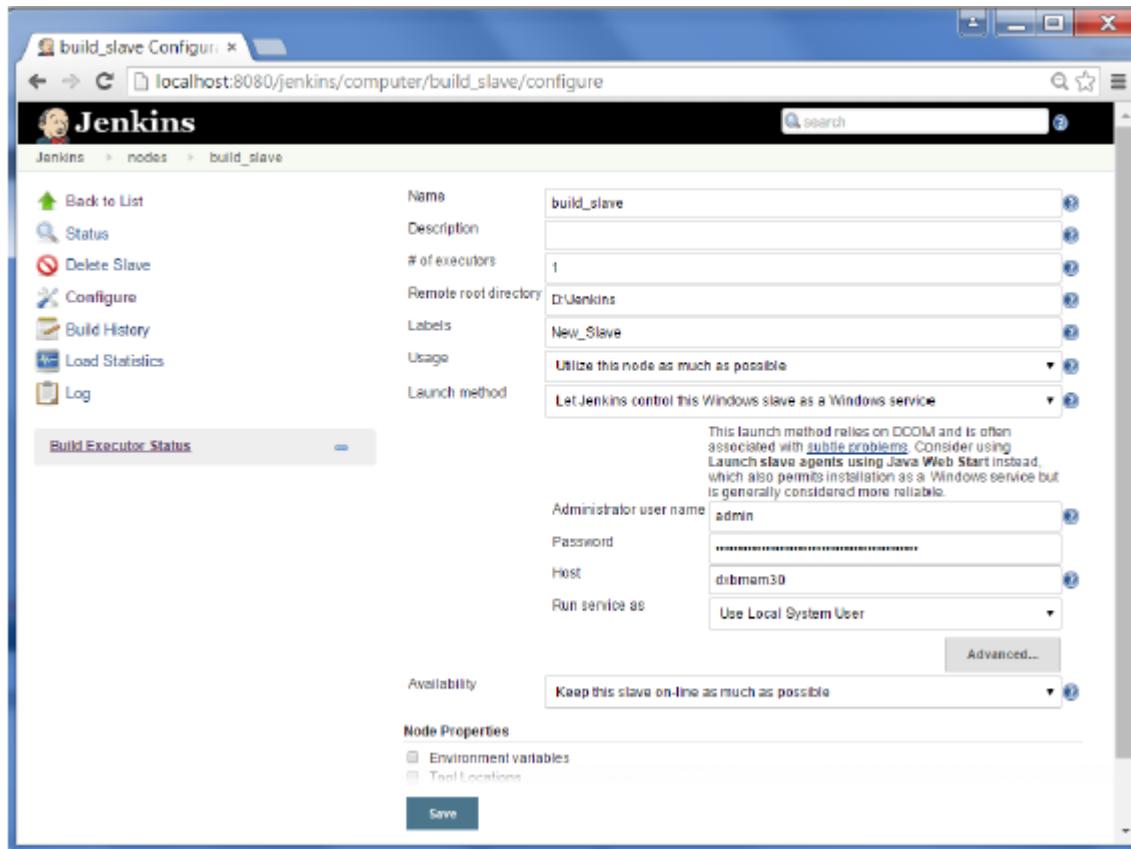
The screenshot shows the Jenkins 'Nodes' page. At the top, there's a navigation bar with links for 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. Below this is a search bar and an 'ENABLE AUTO REFRESH' checkbox. A table lists the nodes: one 'master' node running on Windows 7 (x86), which is 'In sync' with the clock. The table includes columns for Name, Architecture, Clock Difference, Free Disk Space, Free Swap Space, and Frequency. A 'Refresh status' button is at the bottom right of the table.

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

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

The screenshot shows the 'New Node' configuration page. It has fields for 'Node name' (set to 'build_slave') and 'Dumb Slave' (selected). A note explains that this adds a plain, dumb slave to Jenkins. The 'OK' button is visible at the bottom right.

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 interface for managing nodes. On the left, there's a sidebar with links like 'Back to Dashboard', 'Manage Jenkins', 'New Node', and 'Configure'. The main area has sections for 'Build Queue' (which is empty) and 'Build Executor Status' (showing 'master' with 1 idle and 2 idle executors). A table on the right lists nodes: 'build_slave' (Windows 7, offline), 'master' (Windows 7, sync, 229.89 GB free disk space, 12.13 GB swap space, 229.88 temp space), and a summary row 'Data obtained' with metrics 3 ms, 2 ms, 1 ms, and 11 min. A 'Refresh status' button is at the bottom right of the table.

Jenkins - Automated Deployment

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

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

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

Install	Name	Version
<input type="checkbox"/>	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"/>	Capifemoat 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, 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”

The screenshot shows the Jenkins configuration interface for the 'Helloworld' job. In the 'Post-build Actions' section, the 'Deploy war/ear to a container' action is selected. A context menu is open over this action, listing various publishing and deployment options. The 'Deploy war/ear to a container' option is highlighted. At the bottom of the configuration page, there are 'Save' and 'Apply' buttons.

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.

The screenshot shows the Jenkins configuration interface for the 'Demo' job. In the 'Post-build Actions' section, the 'Deploy war/ear to a container' action is selected. The configuration details are as follows:

- WAR/EAR files: target/spare-1.0.war
- Context path: spare
- Containers:
 - Tomcat 7.x:
 - Manager user name: S112233
 - Manager password: (redacted)
 - Tomcat URL: http://dxbmrm10:8080

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

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 at the URL localhost:8080/jenkins/manage. The left sidebar includes links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Under 'Manage Jenkins', there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Manage Jenkins' and contains a list of available Jenkins plugins:

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

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

Jenkins - Quick Guide

localhost:8080/jenkins/pluginManager/available

Available

Install	Name	Version
<input checked="" type="checkbox"/>	Build History Metrics plugin	1.2

Provides build metrics that encompass the history of all the runs

Install without restart Download now and install after restart Update information obtained: 2 mi

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

localhost:8080/jenkins/updateCenter/

Installing Plugins/Upgrades

Preparation

- Checking internal connectivity
- Checking update center connectivity
- Success

Build History Metrics plugin Success

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

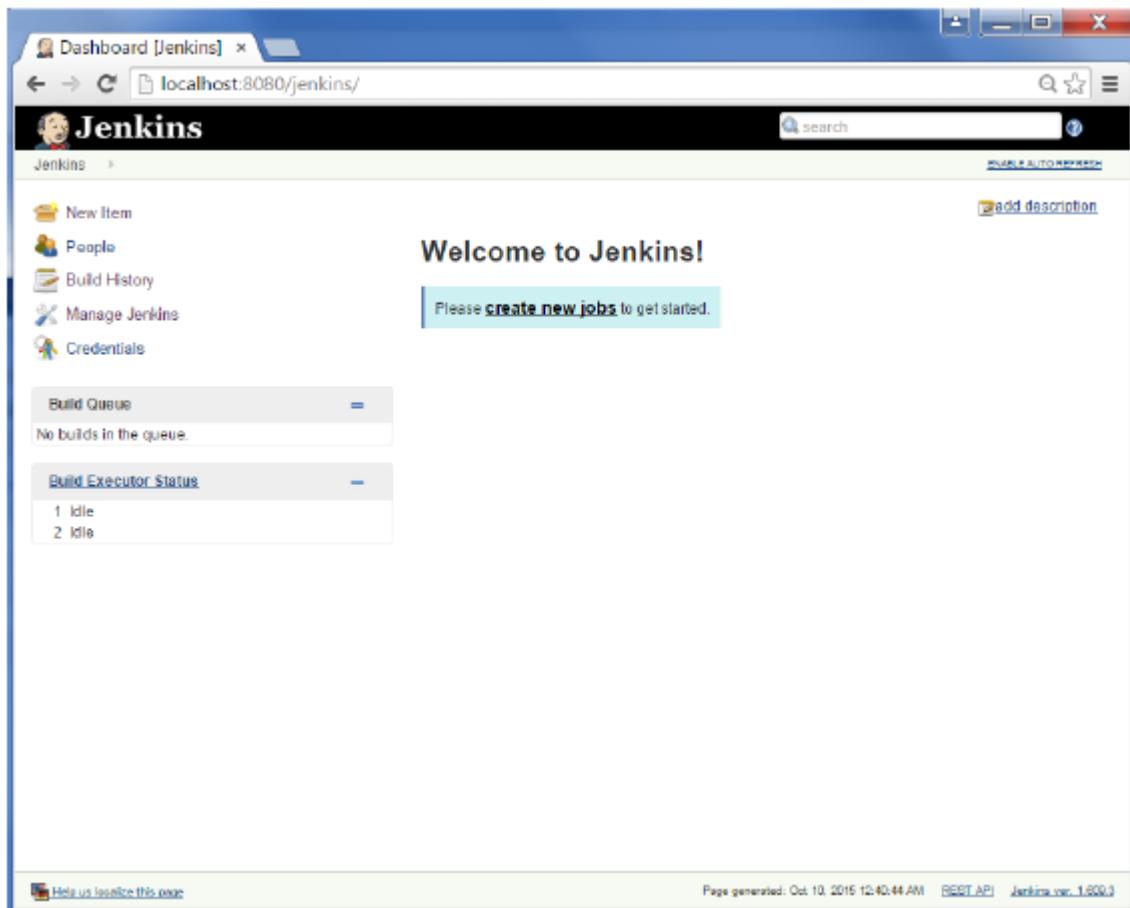
When you go to your Job page, you will see a table with the calculated metrics. Metric's are shown for the last 7 days, last 30 days and all time.

The screenshot shows the Jenkins dashboard 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'. The main area is titled 'Project Helloworld'. It features a 'Workspace' section with a folder icon and a 'Recent Changes' section with a document icon. To the right, there are three tables: 'MTTR', 'MTTF', and 'Standard Deviation', each with three rows: 'Last 7 Days', 'Last 30 Days', and 'All Time'. Below these tables is a 'Permalinks' section with a bulleted list of links to various build logs. At the bottom, there's a footer with links for localization and page generation information.

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

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

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



Step 2 – Go to the Manage Plugins option

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

At the top right, there are 'Setup Security' and 'Dismiss' buttons. A search bar is also present at the top.

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

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [jenki...]" and the address bar shows "localhost:8080/jenkins/pluginManager/available". The main content area is titled "Plugin Manager" with tabs for "Updates", "Available" (which is selected), "Installed", and "Advanced". A search bar at the top right contains the text "global-build-stats". Below the tabs, there's a table with the following data:

Name	Version
Hudson global-build-stats plugin	1.3

The table includes a description: "Global build stats plugin will allow to gather and display global build result statistics. It is a useful tool allowing to display global hudson build trend over time." At the bottom of the table are three buttons: "Install without restart", "Download now and install after restart", and "Update inform".

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

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

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

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



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

The screenshot shows a web browser window titled 'Global Build Stats [ie]' with the URL 'localhost:8080/jenkins/plugin/global-build-stats/'. The page is titled 'Global Build Stats' and features a sidebar with links: 'Back to Dashboard', 'Create new chart', 'Manage retention strategies', and 'Data Initialization'. The main content area includes sections for 'Statistics' (with a note about no charts configured), 'Build Results retention strategies' (with three options: 'Automatically discard results older than 30 days', 'Do not keep build results when they are discarded', and 'Keep existing job results only'), and 'Data initialization' (with a note to click a button to initialize build statistics). A large 'Initialize stats' button is prominently displayed. At the bottom of the page, there is a link to 'Help us localize this page' and footer information: 'Page generated: Oct 24, 2015 4:03:17 PM' and 'jenkins ver. 1.809.3'.

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

The screenshot shows the Jenkins Global Build Stats page at localhost:8080/jenkins/plugin/global-build-stats/#Initialize. The left sidebar has links for Back to Dashboard, Create new chart, Manage retention strategies, and Data Initialization. The main content area is titled "Global Build Stats" and includes a "Statistics" section with a note about no charts being configured. It also includes a "Build Results retention strategies" section with three checkboxes: "Automatically discard results older than [365] days", "Do not keep build results when they are discarded", and "Keep existing job results only". A "Update retention strategies" button is below these checkboxes. Below that is a "Data initialization" section with a note to click the "Initialize stats" button to initialize build statistics. A success message says "Data successfully initialized!" and the "Initialize stats" button is highlighted. At the bottom, there's a "Help us localize this page" link, a timestamp "Page generated: Oct 24, 2015 4:03:17 PM", and a Jenkins version "Jenkins ver. 1.603.3".

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

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

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

Global Build Stats

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

Adding new chart

Title : Demo

Chart Width * Height : 800 * 600

Chart time scale : Daily

Chart time length : 30 days

Filters :

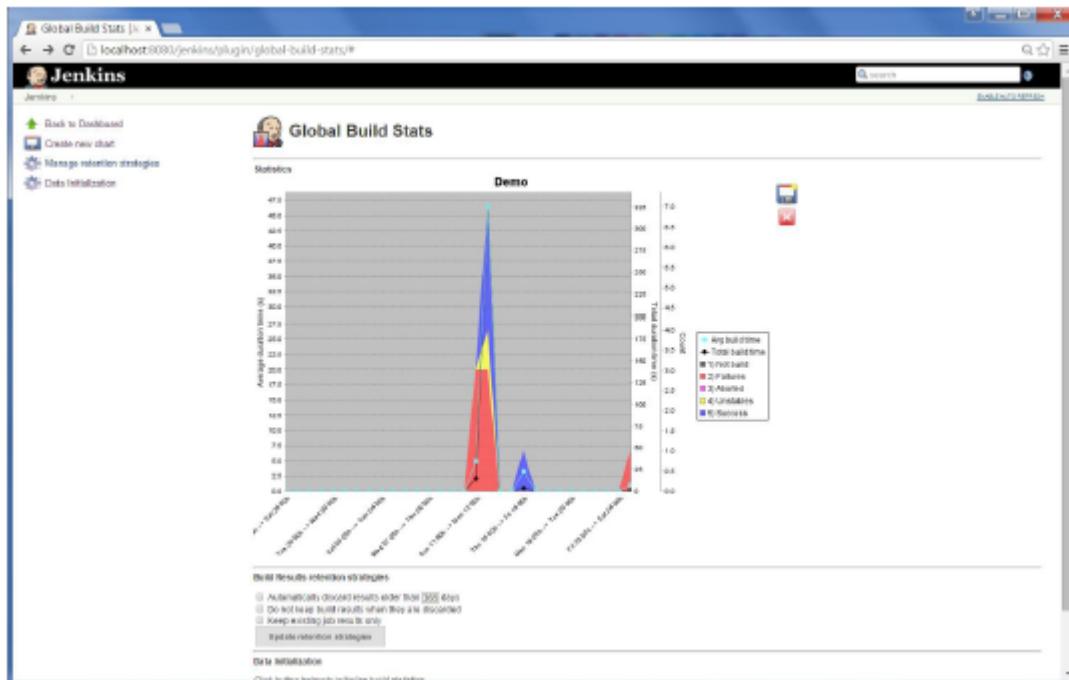
- Job filtering : ALL Jobs Job name regex: []
- Node filtering : ALL Nodes Master only Node name regex: []
- Launcher filtering : ALL Users System only Username regex: []
- Statuses taken into account : Success Failures Unstables Aborted Not Build

Elements displayed on chart:

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

Overview Create new chart Cancel

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



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

The screenshot shows the Jenkins Global Build Search interface. At the top, there are navigation links: 'Back to Dashboard' and 'Back to Global Build Stats'. Below that is a search bar with placeholder text 'Search...'. Underneath the search bar are several filter options: 'Job filtering' (radio buttons for 'All jobs', 'Job name regex' with a text input field containing 'jenkins'), 'Node filtering' (radio buttons for 'All Nodes', 'Master only', 'Node name regex' with a text input field containing 'jenkins'), 'Launcher filtering' (radio buttons for 'All Users', 'System only', 'Username regex' with a text input field containing 'jenkins'), and 'Statuses when triggered' (checkboxes for 'Success', 'Failure', 'Unstable', 'Aborted', and 'Not Started'). A 'Search' button is located below the filters. The main area is titled 'Search results' and contains a table with the following data:

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

At the bottom right of the table, it says 'Page generated: Oct 24, 2015 4:00:40 PM' and 'Version: 1.500.3'.

Jenkins - Server Maintenance

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

URL Options

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

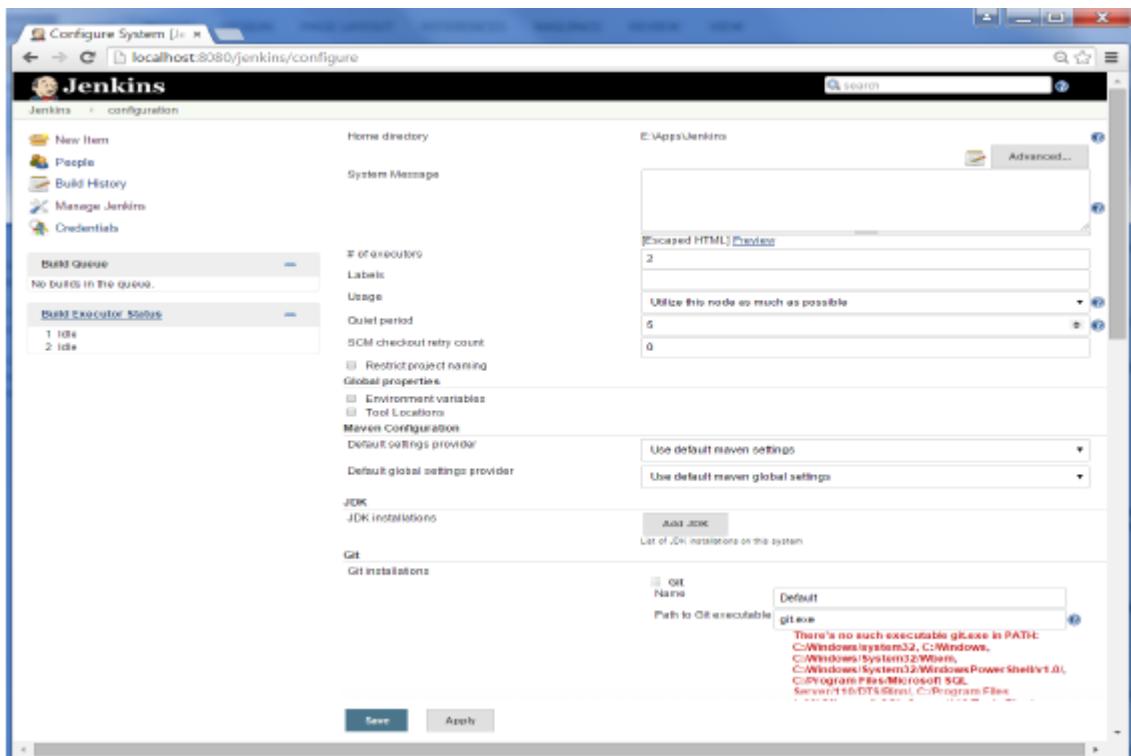
<http://localhost:8080/jenkins/exit> – shutdown jenkins

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

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

Backup Jenkins Home

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

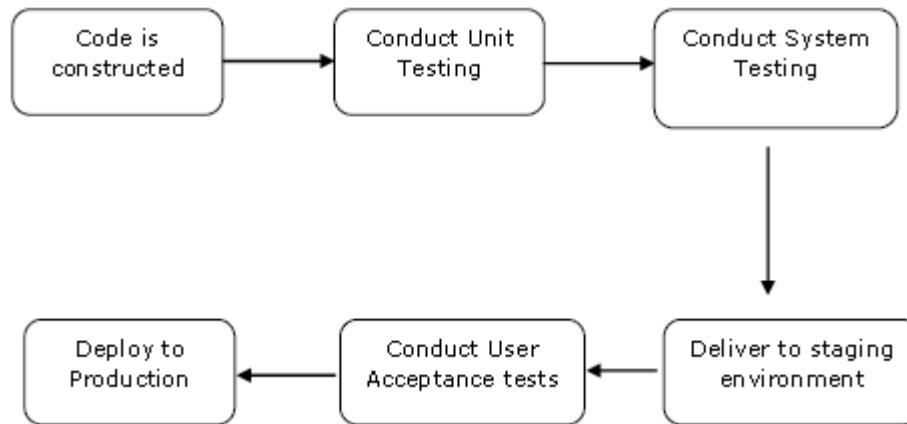


Set up Jenkins on the partition that has the most free disk-space – Since Jenkins would be taking source code for the various jobs defined and doing continuous builds, always ensure that Jenkins is setup on a drive that has enough hard disk space. If you hard disk runs out of space, then all builds on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

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

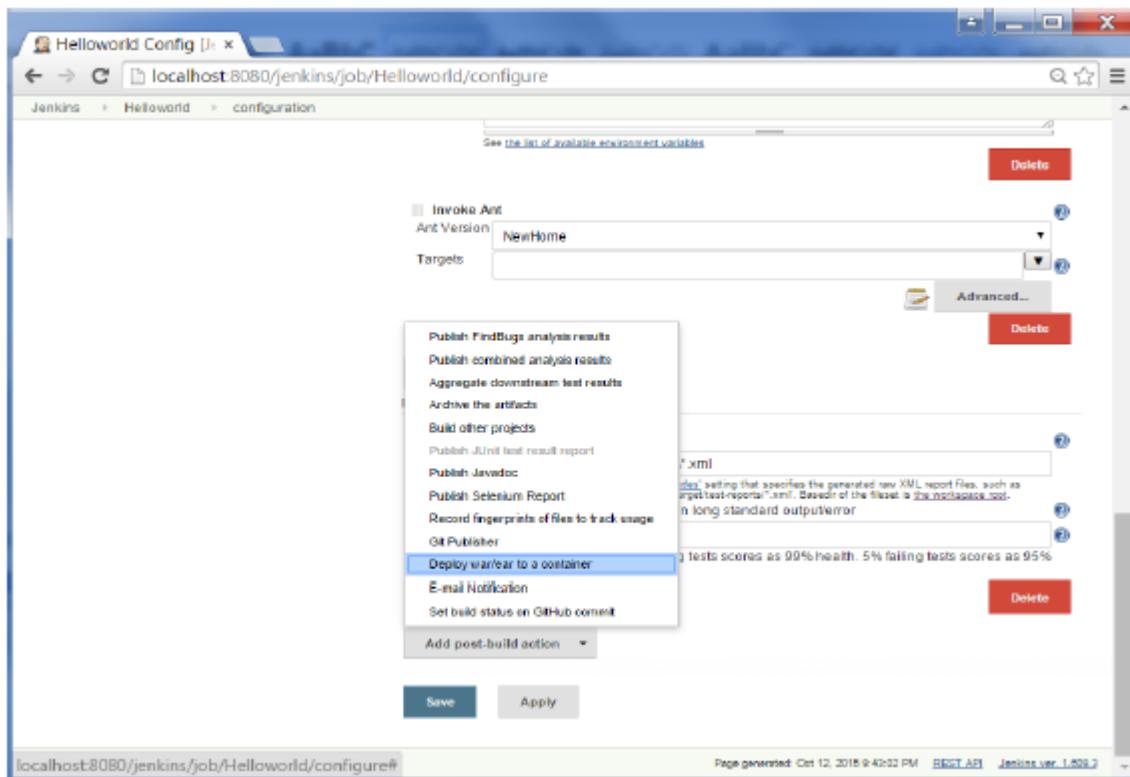
Jenkins - Continuous Deployment

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



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

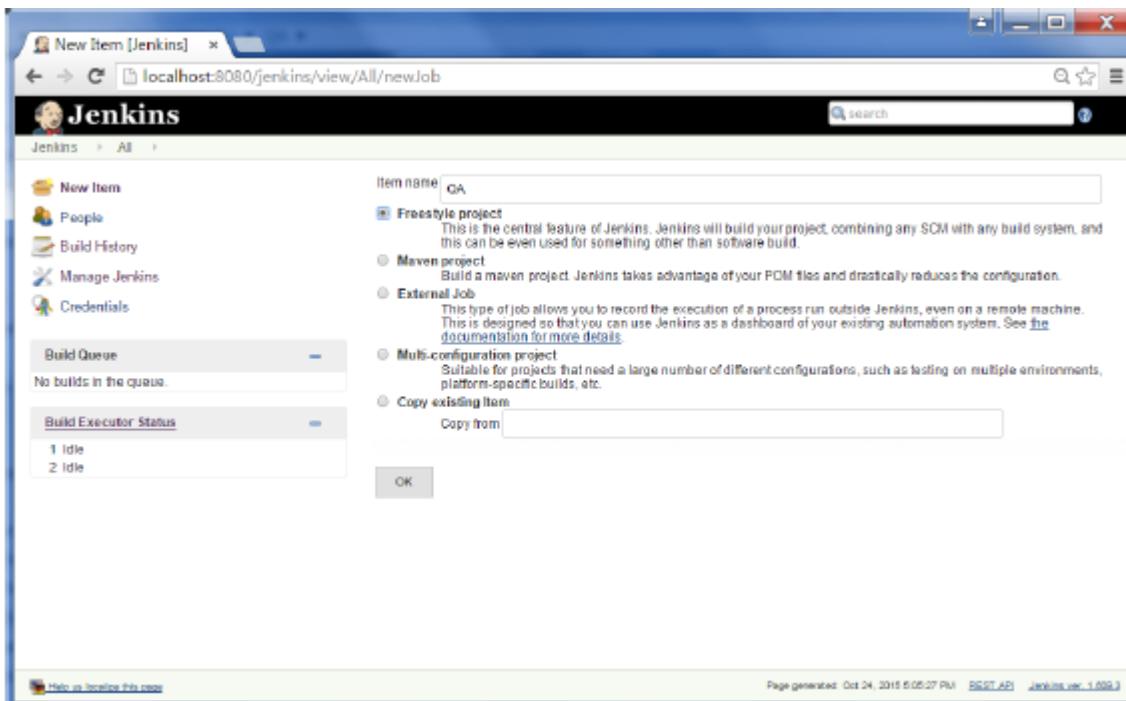
“container Plugin” which was seen in the earlier lessons.



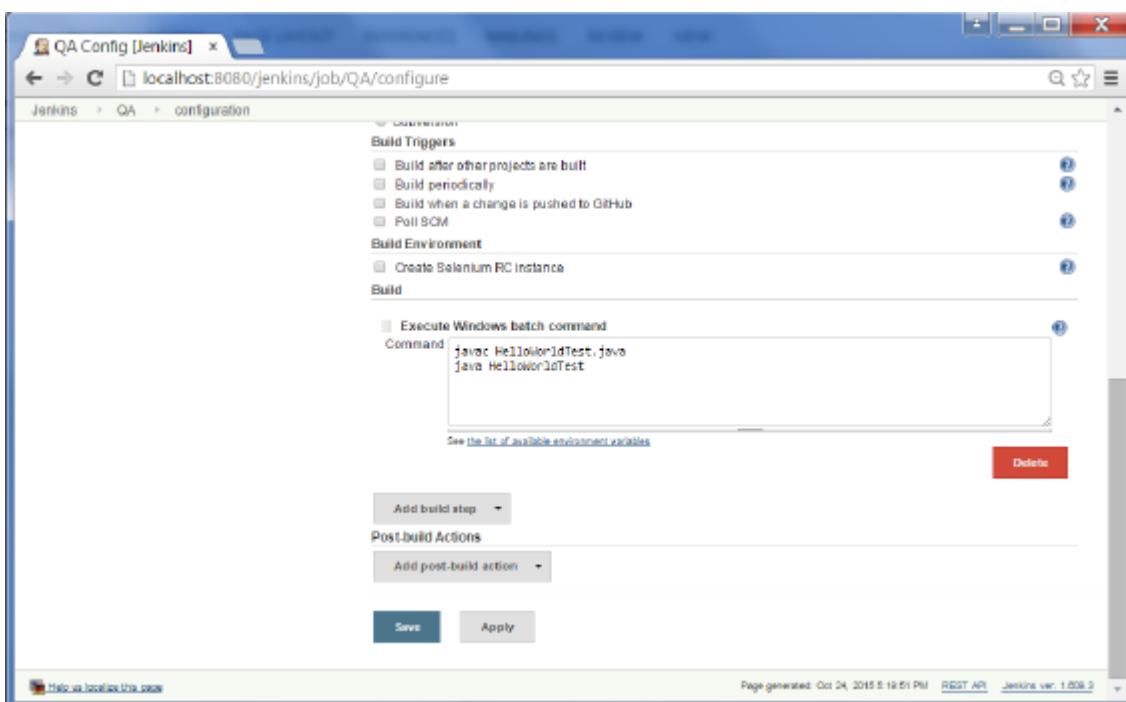
There are plugins available which can actually give you a graphical representation of the Continuous deployment process. But first lets create another project in Jenkins, so that we can see best how this works.

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

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



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



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

The screenshot shows the Jenkins Project QA page for the 'QA [Jenkins]' job. On the left, there's a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, and Configure. The main area is titled 'Project QA' and contains a 'Build History' table with four rows of build logs. To the right is a table showing performance metrics: MTTR, MTTF, and Standard Deviation across three time periods: Last 7 Days, Last 30 Days, and All Time. Below the tables are 'Recent Changes' and 'Permalinks' sections.

	Last 7 Days	Last 30 Days	All Time
MTTR	2 min 28 sec	2 min 28 sec	2 min 28 sec
MTTF	0 ms	0 ms	0 ms
Standard Deviation	75 ms	75 ms	75 ms

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

The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with links: New Item, People, Build History, Manage Jenkins, and Credentials. The main area lists projects: 'Helloworld' (status: 12 days - #15), 'Changes', 'Workspace', 'Build Now', and 'Delete Project'. A 'Configure' button is highlighted in blue at the bottom of the project list. The URL in the browser is 'localhost:8080/jenkins/'.

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

The screenshot shows the Jenkins configuration page for the 'Helloworld' job. A context menu is open over the 'Build other projects' option in the 'Post-build Actions' section. The menu items listed are: Aggregate downstream test results, Archive the artifacts, Build other projects, Publish JUnit test result report, Publish Javadoc, Record fingerprints of files to track usage, Git Publisher, E-mail Notification, and Add post-build action. The 'Build other projects' item is highlighted with a blue selection bar.

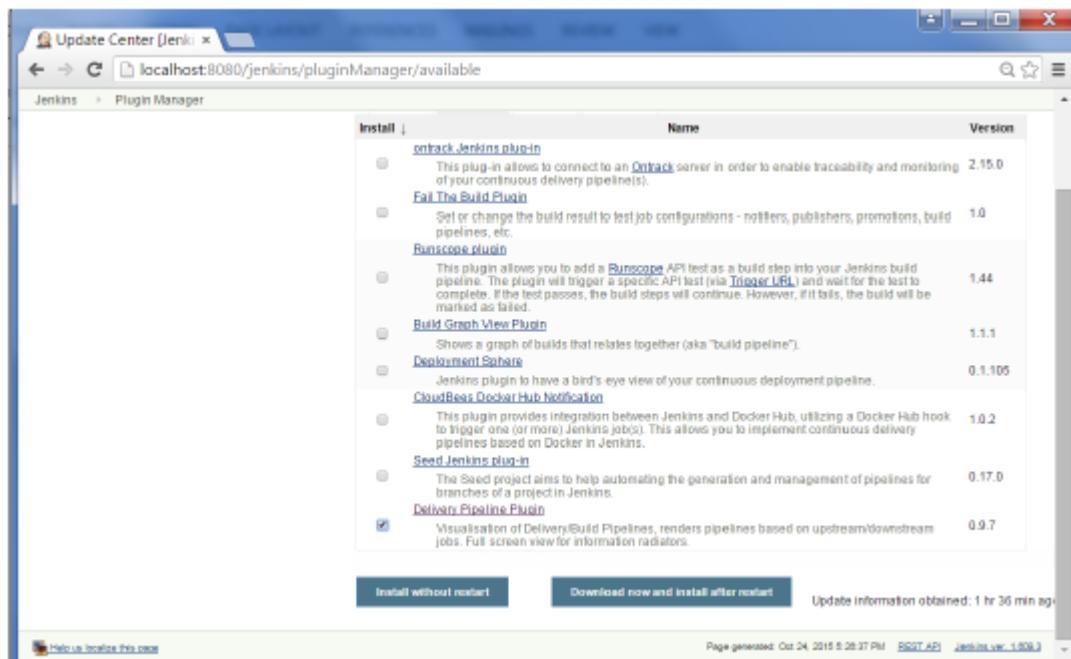
Step 5 – In the ‘Project to build’ section, enter QA as the project name to build. You can leave the option as default of ‘Trigger only if build is stable’. Click on the Save button.

The screenshot shows the Jenkins configuration page for the 'Helloworld' job. The 'Build other projects' post-build action is selected. In the 'Projects to build' field, 'QA' is entered. Below this, there are three radio button options: 'Trigger only if build is stable' (selected), 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. The 'Save' and 'Apply' buttons are visible at the bottom of the configuration panel.

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.

The screenshot shows the Jenkins interface for a build named 'HelloWorld #14'. On the left, there's a sidebar with links like 'Back to Project', 'Status', 'Changes', 'Console Output' (which is currently selected), 'View as plain text', 'Edit Build Information', 'Delete Build', and 'Previous Build'. The main area is titled 'Console Output' and displays the command-line logs for the build. The logs show the build was started by user 'anonymous' in workspace E:\Jenkins\jobs\HelloWorld\workspace. It attempts to run a batch file 'E:\Jenkins\jobs\HelloWorld\workspace>E:\', which fails because 'E\' is not recognized as an internal or external command, operable program or batch file. The build then runs 'javac HelloWord.java' and 'E:\Jenkins\jobs\HelloWorld\workspace>java HelloWord', outputting 'Hello World'. Finally, it exits with code 0. A note at the bottom says 'Warning: you have no plugins providing access control for builds, so falling back to legacy behavior of permitting any downstream builds to be triggered'. The status bar at the bottom right indicates the page was generated on Oct 24, 2015 at 5:22:02 PM.

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.

The screenshot shows the Jenkins dashboard at localhost:8080/jenkins/. On the left sidebar, there are links for New Item, People, Build History, Manage Jenkins, and Credentials. Below the sidebar, there are sections for Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). The main area displays a table of build jobs:

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

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

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

The screenshot shows the 'New View' configuration screen at localhost:8080/jenkins/newView. The 'View name' field contains 'Delivery Pipeline'. The 'Delivery Pipeline View' option is selected, with the description: 'Shows one or more delivery pipeline instances.' There are other options: 'Build Pipeline View' (which shows the jobs in a build pipeline view) and 'List View' (which shows items in a simple list format). Below the configuration area, there are sections for Build Queue (No builds in the queue) and Build Executor Status (1 Idle, 2 Idle). At the bottom right is an 'OK' button.

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

- Ensure the option ‘Show static analysis results’ is checked.
- Ensure the option ‘Show total build time’ is checked.
- For the Initial job – Enter the Helloworld project as the first job which should build.
- Enter any name for the Pipeline
- Click the OK button.

The screenshot shows the Jenkins configuration interface for the 'Delivery Pipeline' view. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Edit View', 'Delete View', 'View Fullscreen', 'Manage Jenkins', and 'Credentials'. Below these are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main panel is titled 'Delivery Pipeline' and contains several configuration sections:

- Name:** Delivery Pipeline
- View settings:**
 - Number of pipeline instances per pipeline: 3
 - Display aggregated pipeline for each pipeline: checked
 - Number of columns: 1
 - Sorting:
 - Update interval: 2
 - Enable start of new pipeline build: checked
 - Enable manual triggers: checked
 - Enable rebuild: checked
 - Show avatars: checked
 - Show commit messages: checked
 - Show job description: checked
 - Show job promotions: checked
 - Show junit results: checked
 - Show static analysis results: checked
 - Show total build time: checked
 - URL for custom CSS file (fullscreen):
- Pipelines:**
 - Components:

Name	Action
FirstJob	Delete
Helloworld	Initial Job
	Final Job (optional)
 - Add
 - Add
- Regular Expression:**

At the bottom are 'OK' and 'Apply' buttons.

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

The screenshot shows the Jenkins interface for a 'Delivery Pipeline' job named 'Firstjob'. The pipeline consists of three sequential steps: 'Helloworld', 'QA', and another 'QA' step. Each step is represented by a box with a green border. The first 'Helloworld' step has a status bar indicating it was triggered 29 minutes ago and took 1 sec. The first 'QA' step also shows a 29-minute duration and 1 sec. The second 'QA' step shows a 1-hour duration and 1 sec. The overall total build time for the pipeline is 2 sec.

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]" and the URL is "localhost:8080/jenkins/pluginManager/available". The main content area has a search bar with "Build pipeline" and tabs for "Updates", "Available", "Installed", and "Advanced". The "Available" tab is selected, showing a list of plugins. One plugin, "Build Pipeline Plugin", is highlighted with a checkmark. The list includes:

Name	Version
Build Pipeline Plugin	1.4.8
Fail The Build Plugin	1.0
Runscope plugin	1.44
Build Graph View Plugin	1.1.1
Delivery Pipeline Plugin	0.9.7

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

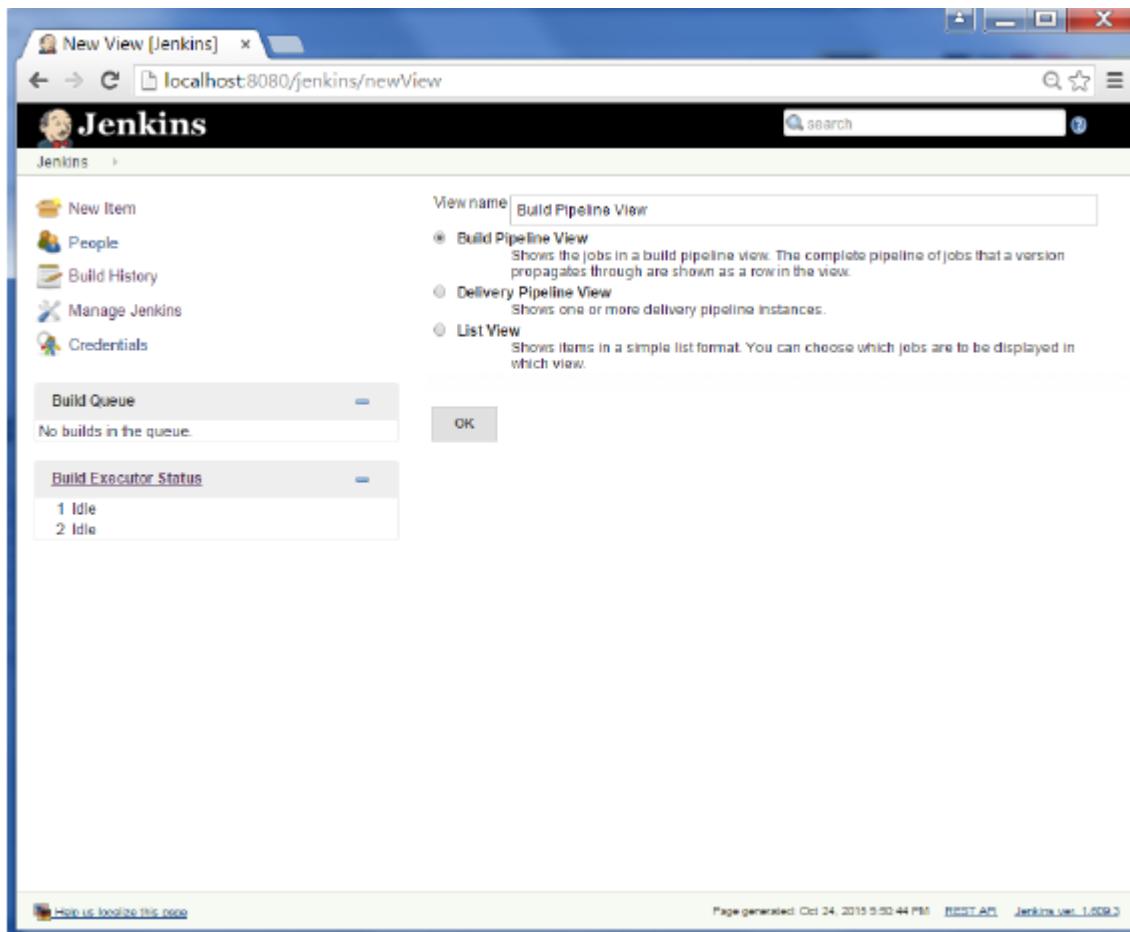
Step 2 – To see the Build pipeline in action, in the Jenkins Dashboard, click on the + symbol in the Tab next to the ‘All’ Tab.

The screenshot shows the Jenkins Dashboard. The title bar says "Dashboard [jenkins]" and the URL is "localhost:8080/jenkins/". The left sidebar has links for "New Item", "People", "Build History", "Manage Jenkins", and "Credentials". The main content area has a "Build Queue" section stating "No builds in the queue." and a "Build Executor Status" section showing "1 Idle" and "2 Idle". On the right, there is a table of builds:

All	S	W	Name	Last Success	Last Failure	Last Duration
Icon:	8	W	HelloWorld	25 min - #14	1 hr 49 min - #12	1.4 sec
Icon:	9	W	QA	25 min - #5	28 min - #2	1.4 sec

Below the table are "Legend" and "RSS" links: "RSS for all", "RSS for failures", and "RSS for just latest builds". At the bottom is a footer with "Help us localize this page", "Page generated: Oct 24, 2015 4:48:09 PM", "REST API", and "Jenkins ver. 1.809.3".

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



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

The screenshot shows the Jenkins 'Edit View [Build Pipeline View]' configuration page. The 'Name' field is set to 'Build Pipeline View'. Under 'Layout', 'Based on upstream/downstream relations' is selected. A dropdown 'Select Initial Job' shows 'HelloWorld'. Other settings include 'No Of Displayed Builds' (1), 'Restrict triggers to most recent successful builds' (Yes), and 'URL for custom CSS files' (Lightbox). Buttons at the bottom are 'OK' and 'Apply'.

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

The screenshot shows the Jenkins 'Build Pipeline View' page. It displays three build steps: 'Pipeline #14' (status: green, duration: 114 sec), '#14 Hellworld' (status: green, duration: 114 sec), and '#5 QA' (status: green, duration: 114 sec).

Jenkins - Managing Plugins

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

The screenshot shows the Jenkins Plugins page at <https://wiki.jenkins-ci.org/display/JENKINS/Plugins>. The left sidebar contains links for Jenkins Home, Mailing lists, Source code, Bugtracker, Security Advisories, Events, Donation, Commercial Support, and Wiki Site Map. Under the 'Documents' section, there are links for Meet Jenkins, Use Jenkins, Extend Jenkins, Plugins, and Servlet Container Notes. The main content area is titled 'Plugins' and includes a note about being last edited by Kohsuke Kawaguchi. It lists several categories: 'How to install plugins' (1.1 Using the interface, 1.1.1 Installing the newest version, 1.1.2 Installing a specific version), 'By hand' (2 Getting notified of plugin releases), 'Developers' (3 Developers), and 'Plugins by topic' (4 Plugins by topic). The '4 Plugins by topic' section is expanded, showing 27 sub-topics such as Source code management, Build triggers, Build tools, Build wrappers, Build notifiers, Slave launchers and controllers, Build reports, Artifact updaters, Other post-build actions, External site/tool integrations, UI plugins, List View column plugins, Page decorators, Authentication and user management, Cluster management and distributed build, CLI extensions, Maven, Parameters, iOS development, .NET development, Android development, and Ruby development.

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

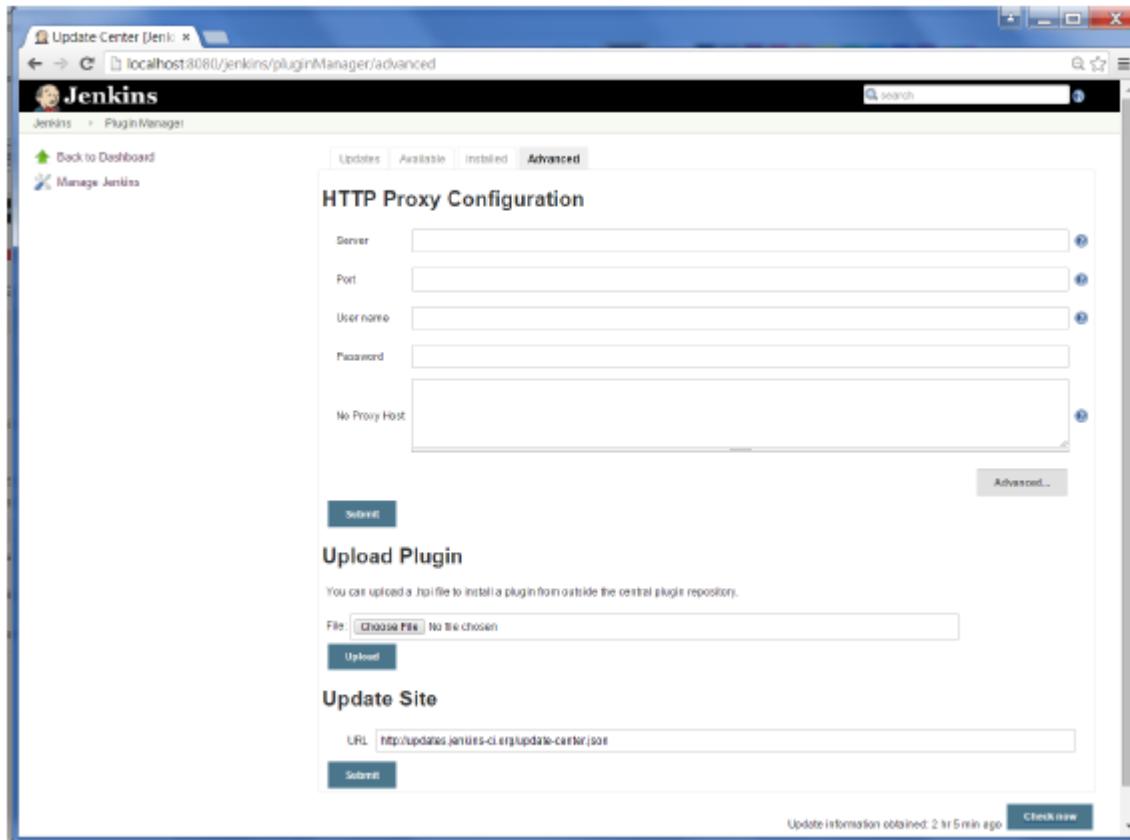
Uninstalling Plugins

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

The screenshot shows the Jenkins Plugin Manager at <http://localhost:8080/jenkins/pluginManager/installed>. The interface has tabs for Updates, Available, Installed (which is selected), and Advanced. The 'Installed' tab displays a list of currently enabled plugins. Each plugin entry includes a checkbox for enabling/disabling, the plugin name, its version, the previously installed version, a 'Pinned' button, and a 'Uninstall' button. The plugins listed include: Ansible Plugin, Built History Metrics Plugin, Built Pipeline Plugin, Credentials Plugin, CVS/EGit Plugin, Delivery Pipeline Plugin, Docker Container Plugin, Deploy to Container Plugin, External Monitor Job Type Plugin, External Notification Plugin, FindBugs Plugin, and FindBugs Analytics Plugin.

Installing another Version of a Plugin

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

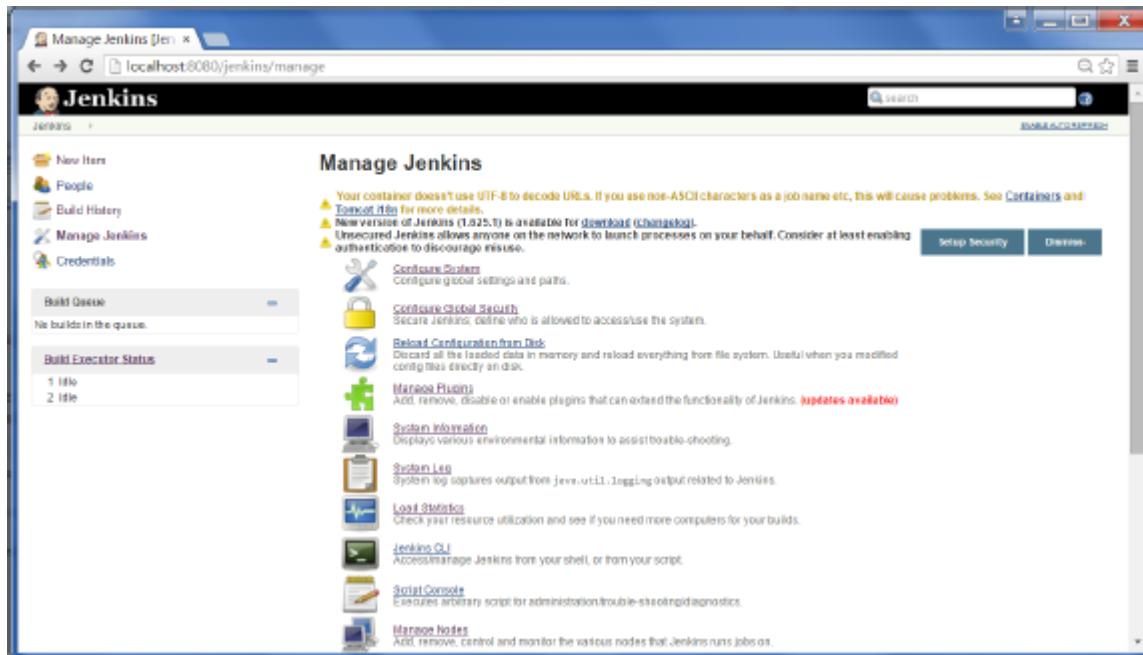


Jenkins - Security

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

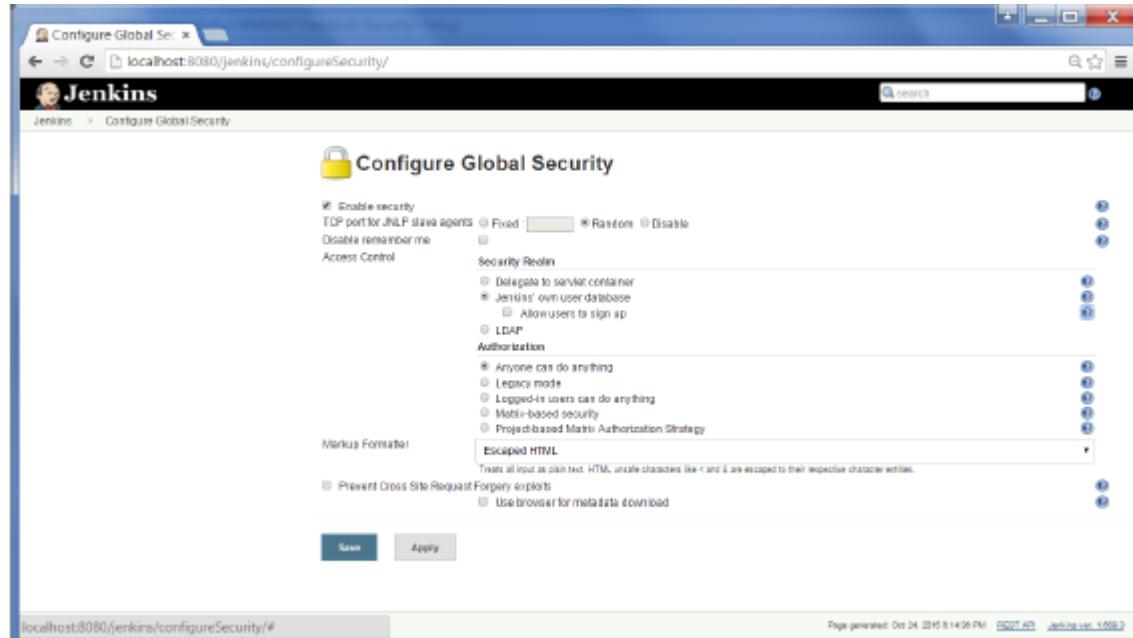
To configure Security in Jenkins, follow the steps given below.

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

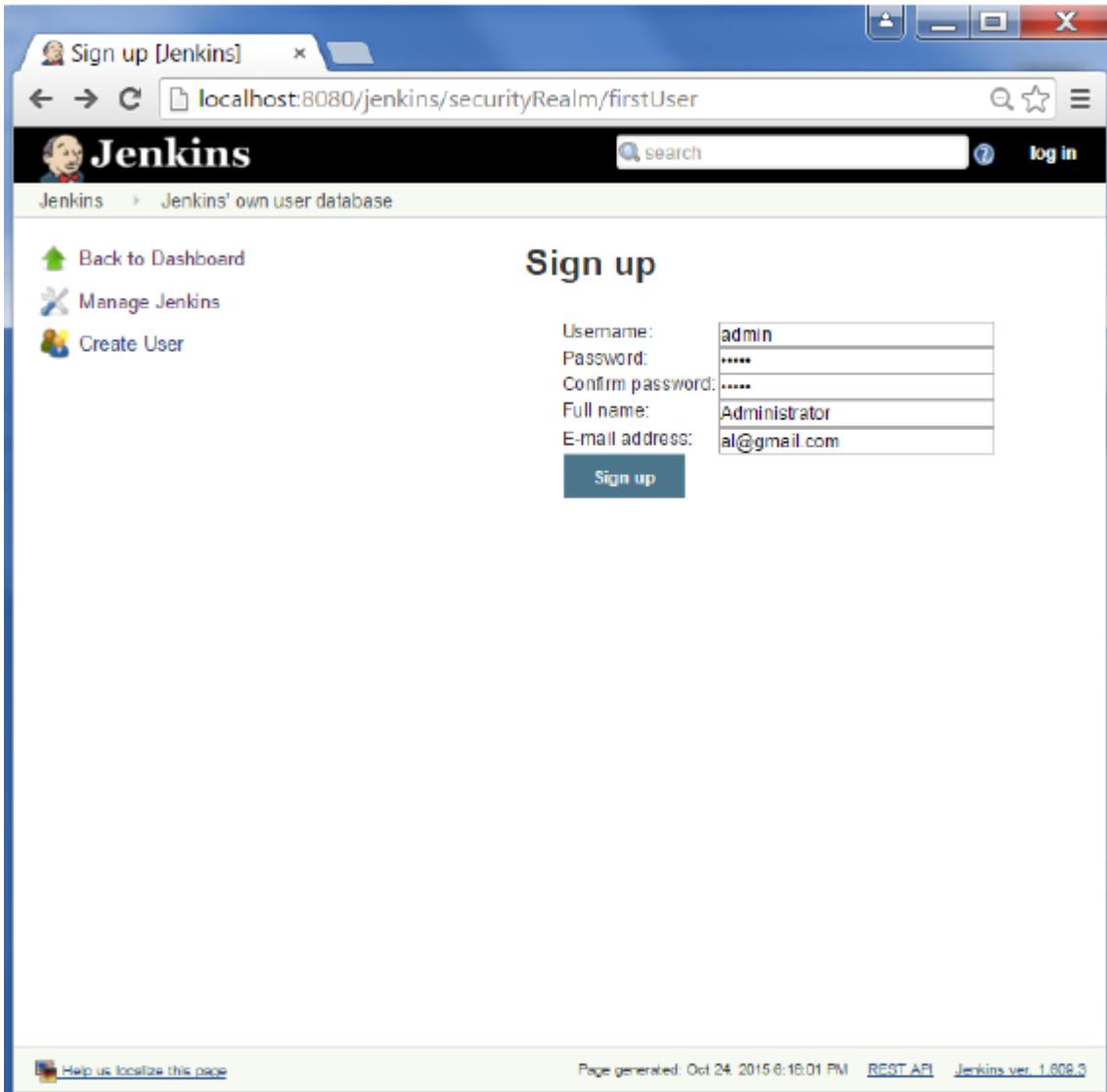


Step 2 – Click on Enable Security option. As an example, let's assume that we want Jenkins to maintain its own database of users, so in the Security Realm, choose the option of ‘Jenkins’ own user database’.

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

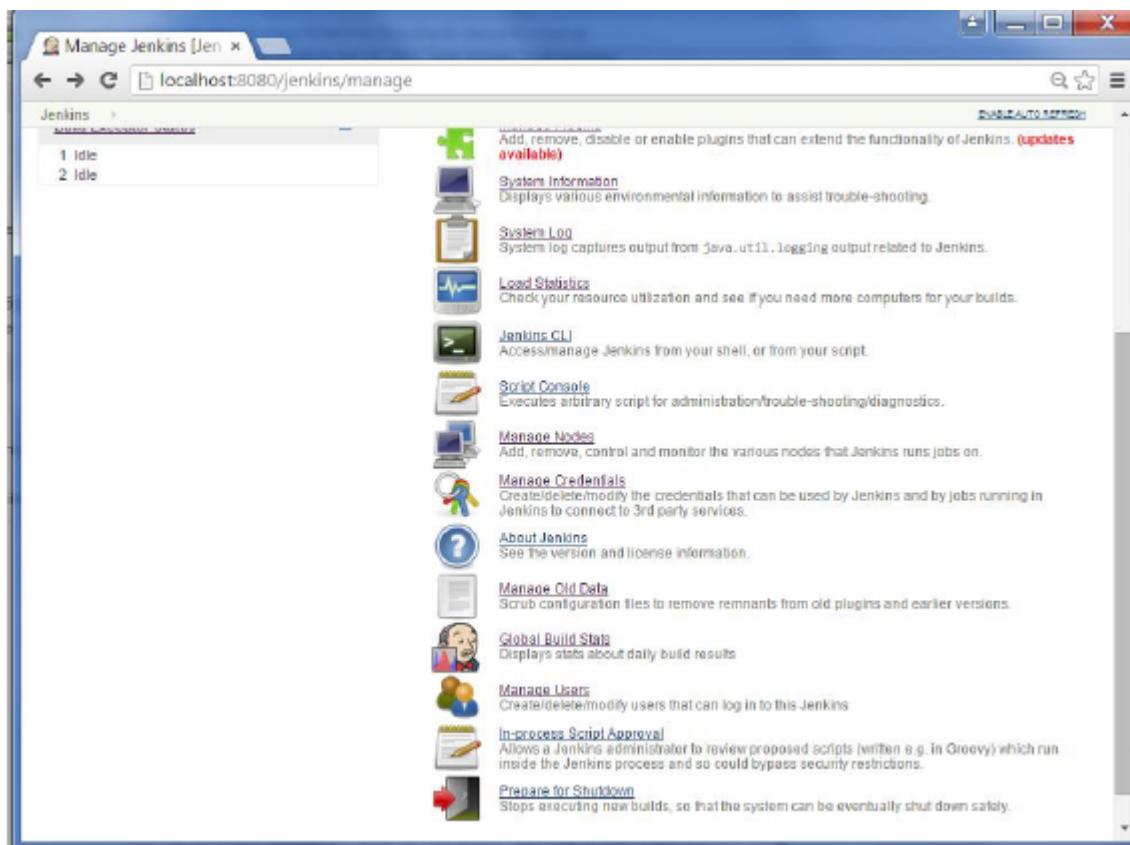


Step 3 – You will be prompted to add your first user. As an example, we are setting up an admin users for the system.

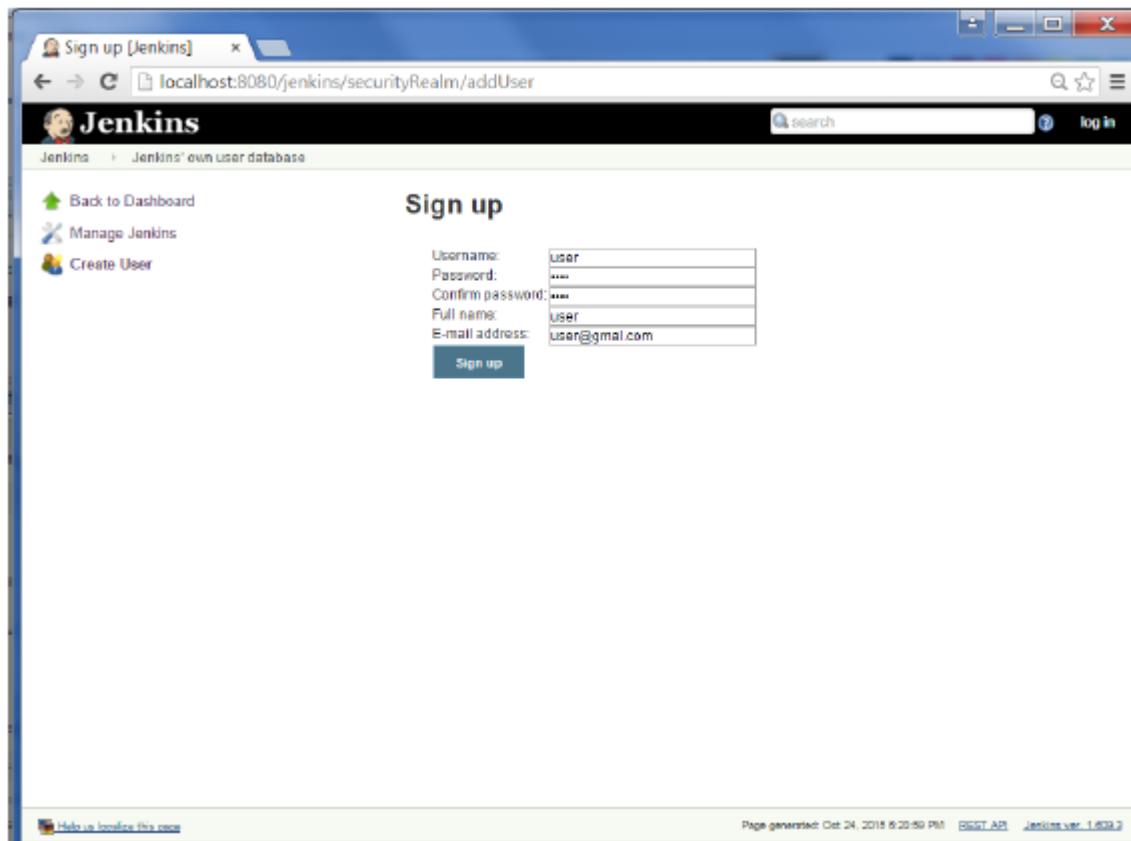


The screenshot shows the Jenkins 'Sign up' page. At the top, there's a navigation bar with links for 'Sign up [Jenkins]', 'localhost:8080/jenkins/securityRealm/firstUser', a search bar, and a 'log in' button. Below the navigation is the Jenkins logo and a 'Jenkins' link. A breadcrumb trail shows 'Jenkins > Jenkins' own user database'. On the left, there's a sidebar with links for 'Back to Dashboard', 'Manage Jenkins', and 'Create User'. The main content area is titled 'Sign up' and contains fields for 'Username' (admin), 'Password' (****), 'Confirm password' (****), 'Full name' (Administrator), and 'E-mail address' (al@gmail.com). A 'Sign up' button is at the bottom of the form. At the bottom of the page, there are links for 'Help us localize this page', 'Page generated: Oct 24, 2015 6:16:01 PM', 'REST API', and 'Jenkins ver. 1.808.3'.

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



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



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

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

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

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

Your Jenkins security is now setup.

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

Jenkins - Backup Plugin

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

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

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

The image consists of two vertically stacked screenshots of the Jenkins Update Center.

Screenshot 1: Available Plugins

This screenshot shows the Jenkins Update Center with the search bar set to "backup". The results table lists several plugins:

Install	Name	Version
<input checked="" type="checkbox"/>	Backup plugin	1.6.1
<input type="checkbox"/>	Backup and interrupt job plugin	1.0
<input type="checkbox"/>	CloudBees Jenkins Enterprise	15.0.1
<input type="checkbox"/>	CloudBees Free Enterprise Plugins	5.0
<input type="checkbox"/>	Periodic Backup	1.3
<input type="checkbox"/>	ThinBackup	1.7.4

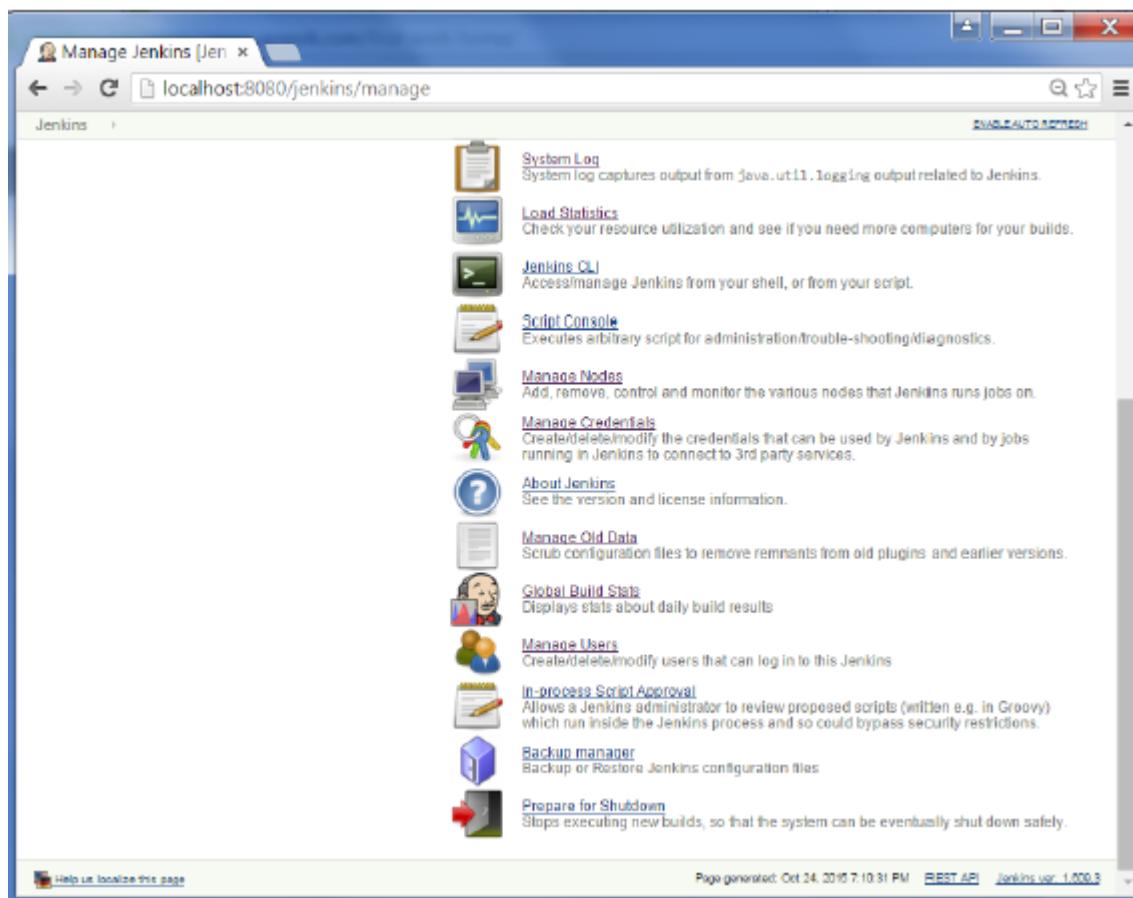
At the bottom, there are three buttons: "Install without restart", "Download now and install after restart", and "Update information of...".

Screenshot 2: Installing Plugins/Upgrades

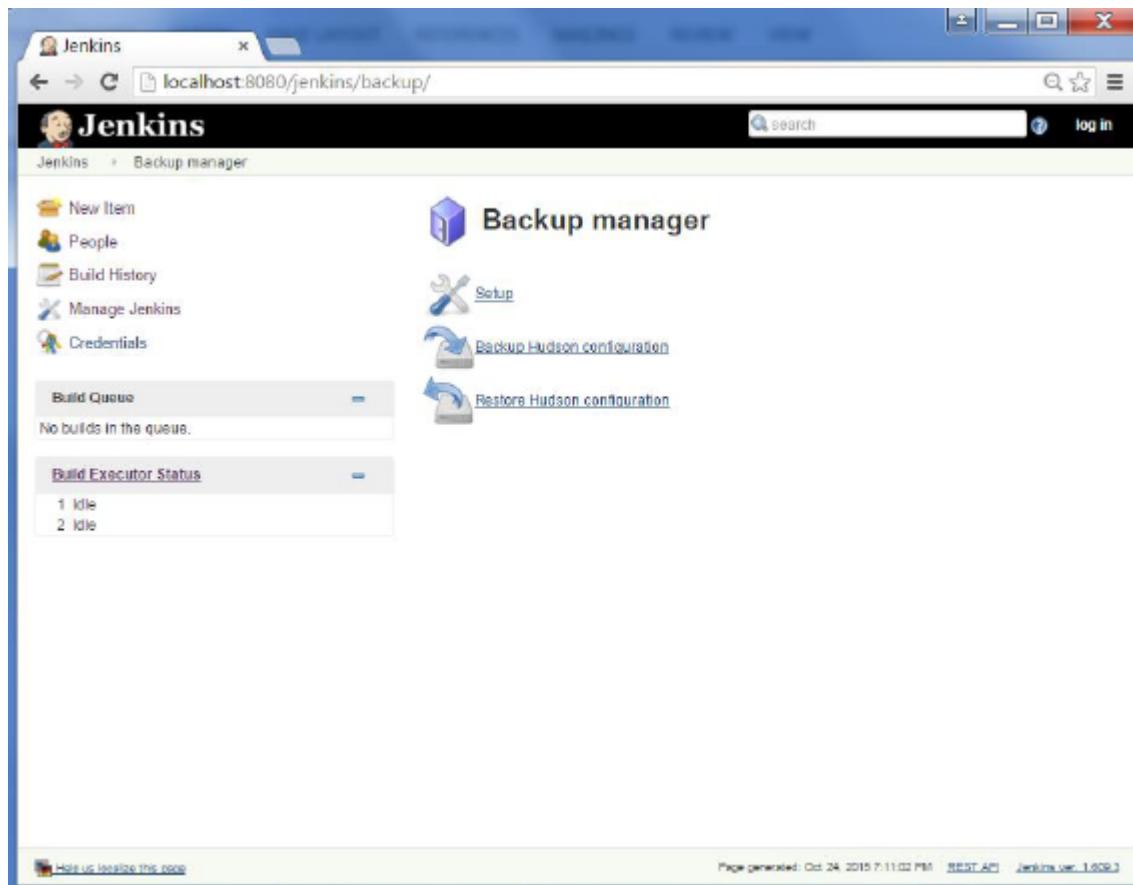
This screenshot shows the confirmation page for the installed "Backup plugin". It indicates a "Success" status. Below the message, there are two links:

- 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.



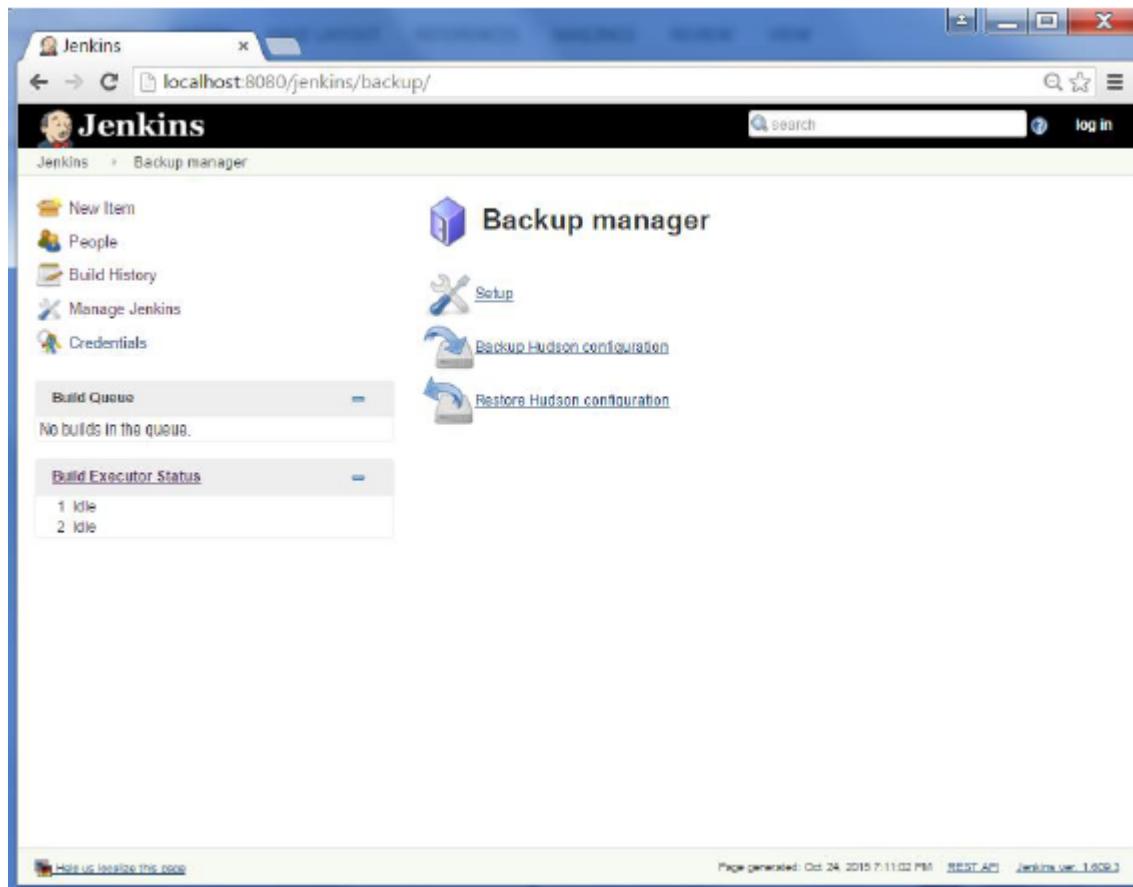
Step 4 – Click on Setup.



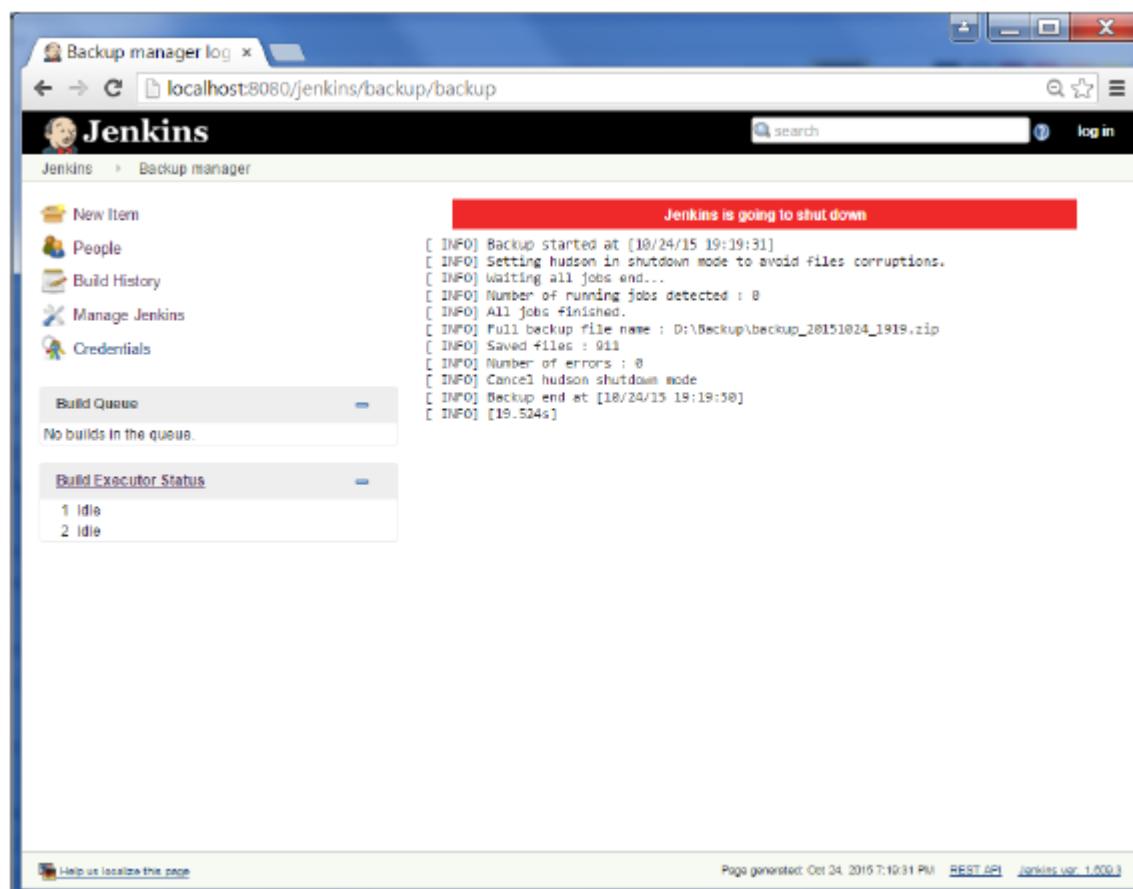
Step 5 – Here, the main field to define is the directory for your backup. Ensure it's on another drive which is different from the drive where your Jenkins instance is setup. Click on the Save button.

The screenshot shows the Jenkins Backup manager interface. On the left, there is a sidebar with links: New Item, People, Build History, Manage Jenkins, and Credentials. Below these are two collapsed sections: Build Queue (No builds in the queue) and Build Executor Status (2 Idle). The main content area is titled "Backup config files". It contains a "Backup configuration" section with fields: Hudson root directory (E:\Jenkins), Backup directory (D:\Backup), Format (zip), File name template (backup_@date@.zip), and Custom exclusions (empty). Below this is a "Backup content" section with checkboxes: Backup job workspace (checked), Backup builds history (unchecked), Backup maven artifacts archives (unchecked), and Backup fingerprints (unchecked). At the bottom right is a "Save" button.

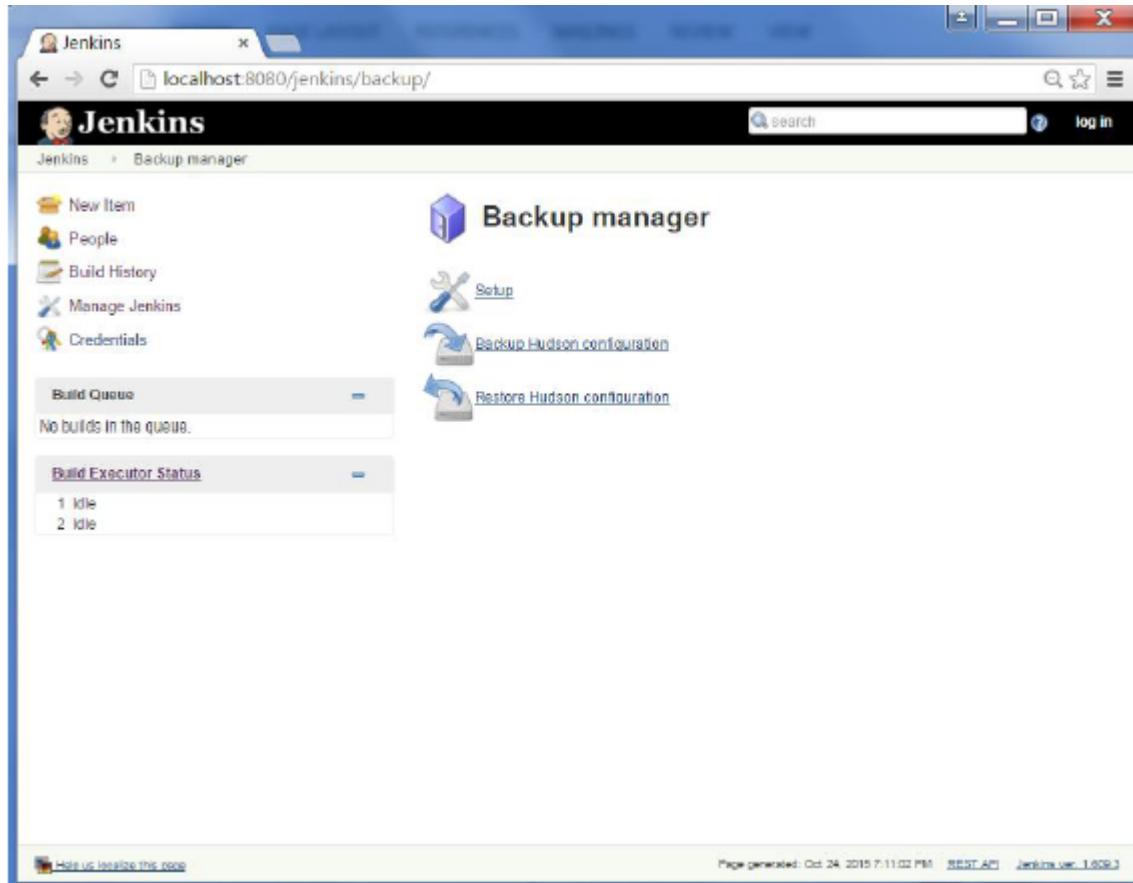
Step 6 – Click on the ‘Backup Hudson configuration’ from the Backup manager screen to initiate the backup.



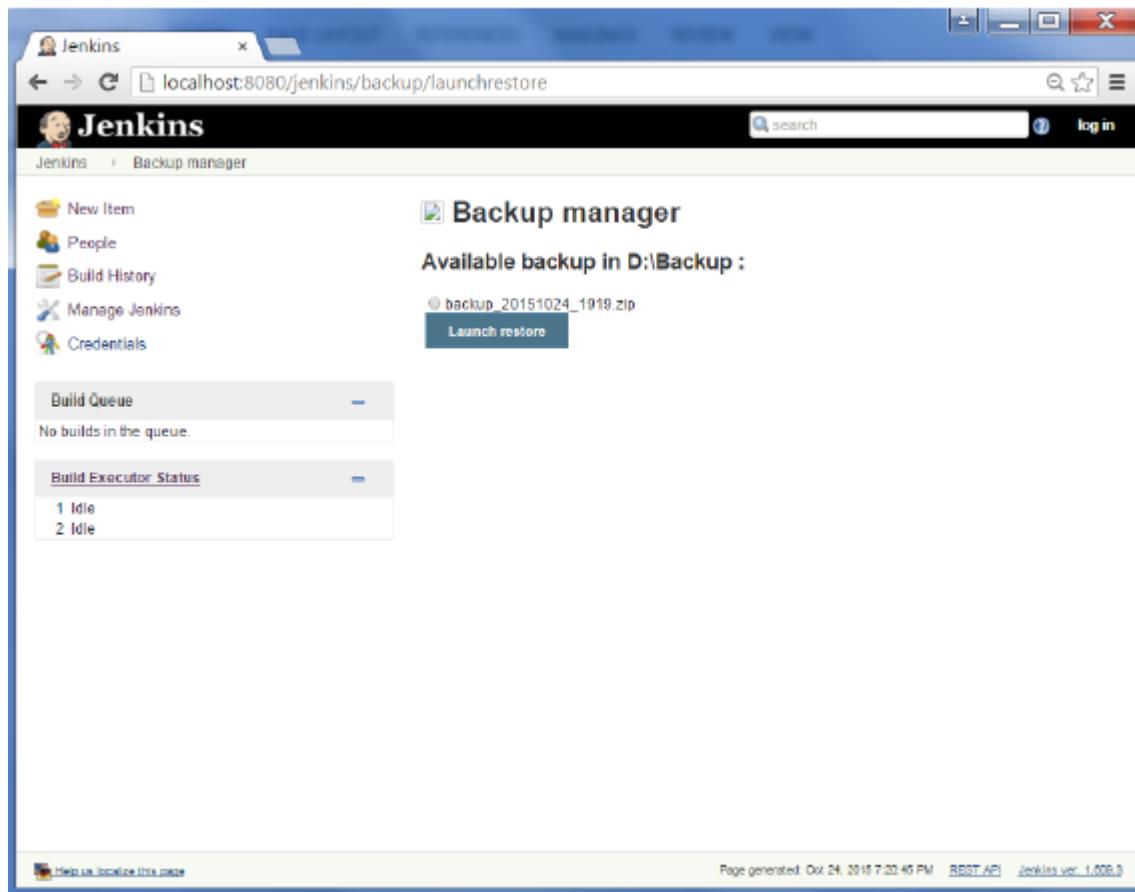
The next screen will show the status of the backup



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



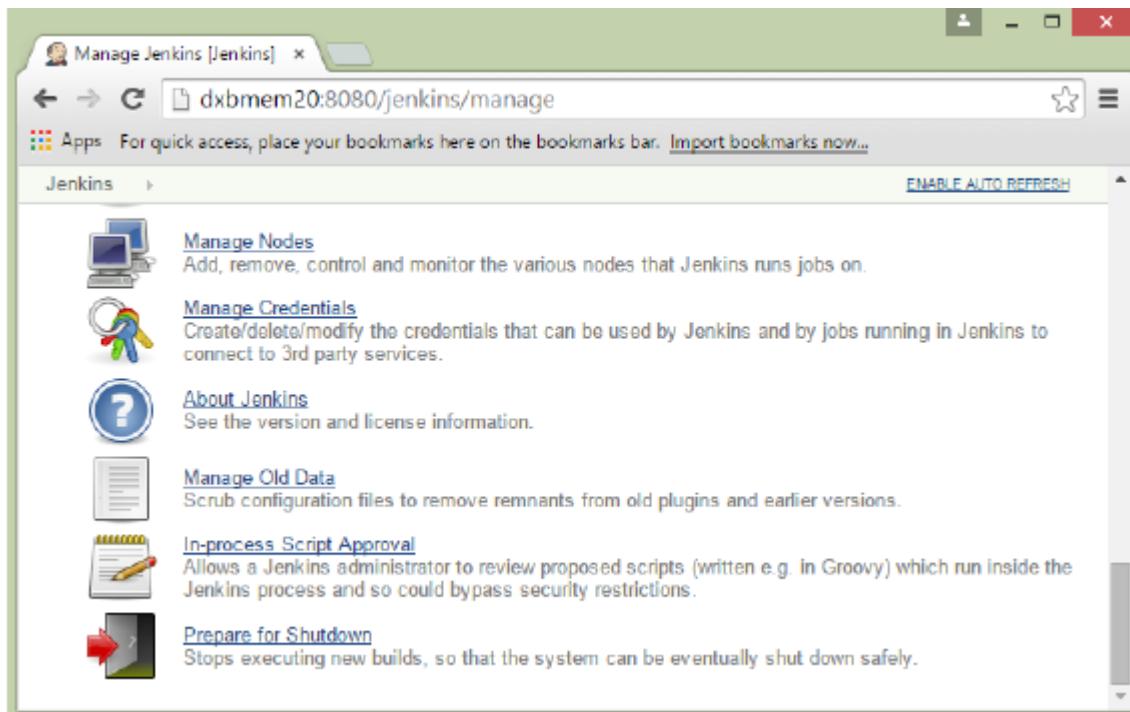
The list of backup's will be shown, click on the appropriate one to click on Launch Restore to begin the restoration of the backup.



Jenkins - Remote Testing

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

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



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

The screenshot shows the Jenkins Nodes page. The left sidebar has a 'Nodes' icon. The main content area shows the following details:

- Total nodes: 1 total, 2 Idle.
- DXBMEM30: 1 Idle.

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

A 'Refresh status' button is at the bottom right. The URL in the address bar is `dxbmem20:8080/jenkins/computer/`.

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

The screenshot shows the Jenkins 'Nodes' page with one slave node listed: 'DXBMEM30'. The node is running 'Windows Server 2012'. A context menu is open over the node, with 'Configure' being the selected option.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Sp
	DXBMEM30	Windows Server 2012	In sync	112.79 GB	4.54 GB	112.79
		Configure				
		Delete Slave				
		Build History				

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'.

Name	DXBMEM30
Description	
# of executors	1
Remote root directory	C:\users\administrator.EMIRATES\jenkins
Labels	
Usage	Utilize this node as much as possible
Launch method	Launch slave agents via Java Web Start

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

The screenshot shows the Jenkins dashboard at <http://dxbmem30:8080/jenkins/>. The main menu includes options like 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below the menu, there's a 'Build Queue' section stating 'No builds in the queue.' and a 'Build Executor Status' section showing '1 Idle' and '2 Idle' executors.

Step 5 – Click on the DXBMEM30 instance.

The screenshot shows the 'Nodes' page at <http://dxbmem20:8080/jenkins/computer/>. It lists two nodes: 'master' (1 Idle, 2 Idle) and 'DXBMEM30' (offline). A table below provides detailed information for each node, including architecture, clock difference, and disk space.

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

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

The screenshot shows a web browser window titled 'DXBMEM30 [Jenkins]'. The URL is 'dxbmem20:8080/jenkins/computer/DXBMEM30/'. The page displays the Jenkins node configuration for 'DXBMEM30'. It includes a stack trace for an error:

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

Instructions for connecting the slave to Jenkins are provided:

- 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

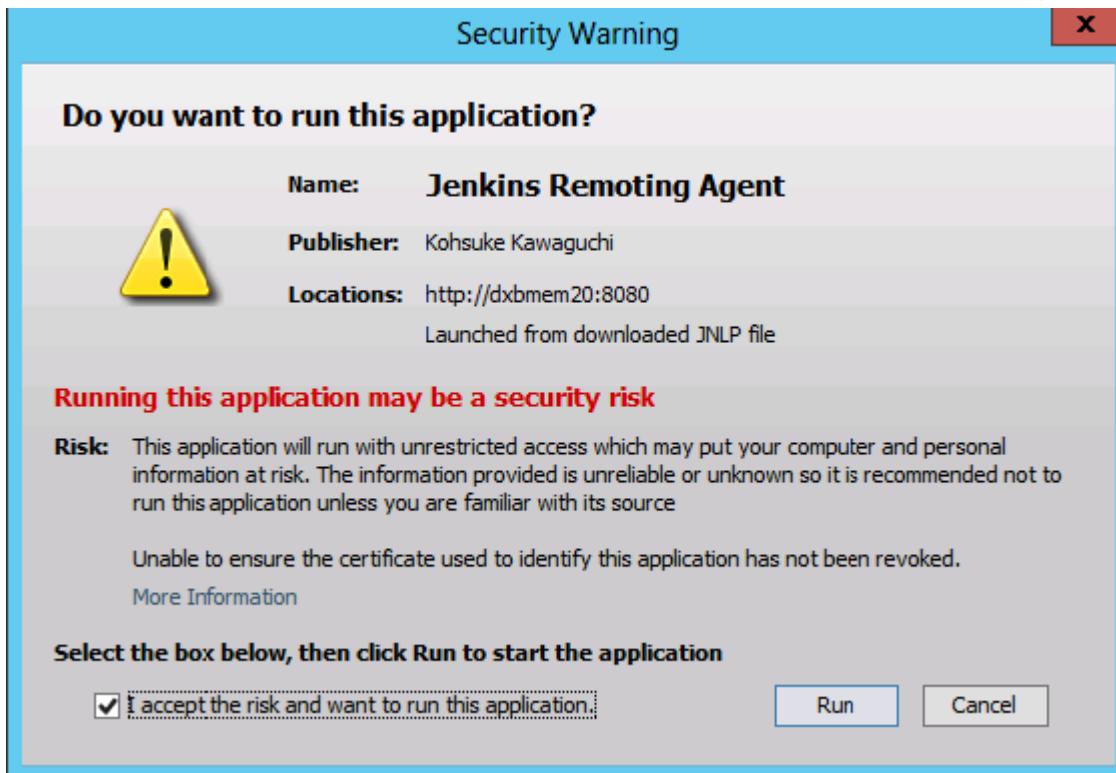
It is noted that the node was created by an anonymous user.

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 `dxbmcm20:8080/jenkins/job/HelloWorld/configure`. The 'configuration' section is open, showing various build options:

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

Label Expression: `DXBMEM30`

Slaves in label: 1

Advanced Project Options button

Source Code Management section (disabled)

Buttons: Save, Apply, Advanced...

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. The URL is `dxbmcm20:8080/jenkins/job/HelloWorld/configure`. The 'configuration' section is open, showing the 'SeleniumHQ htmlSuite Run' step 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:

Delete button

Add build step dropdown

Buttons: 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.